UNIVERSIDADE SALVADOR – UNIFACS
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO
MESTRADO EM SISTEMAS E COMPUTAÇÃO

ELISÂNGELA SIMÕES RÊGO

MDAONTO: UM PROCESSO MDA DE DESENVOLVIMENTO DE SOFTWARE COM FOCO EM ONTOLOGIAS

Salvador
2009

ELISÂNGELA SIMÕES RÊGO

MDAONTO: UM PROCESSO MDA DE DESENVOLVIMENTO DE SOFTWARE COM FOCO EM ONTOLOGIAS

Dissertação apresentada ao curso de Mestrado Profissional em Sistemas e Computação, Universidade Salvador – UNIFACS, como requisito parcial para obtenção do grau de Mestre.

Orientadora: Profª. Drª. Laís do Nascimento Salvador

Salvador
2009

FICHA CATALOGRÁFICA
(Elaborada pelo Sistema de Bibliotecas da Universidade Salvador - UNIFACS)

Rego, Elisângela Simões

MDAONTO: um processo MDA de desenvolvimento de software com foco em ontologias / Elisângela Simões Rego. - 2009.
131 f. : il.

Dissertação (Mestrado) - Universidade Salvador – UNIFACS.
Mestrado em Sistemas de Computação, 2008.
Orientador: Prof. Laís do Nascimento Salvador

1. Desenvolvimento de Software. I. Salvador, Laís do Nascimento, orient. II. Título.

CDD: 004

ELISÂNGELA SIMÕES RÊGO

MDAONTO: UM PROCESSO MDA DE DESENVOLVIMENTO DE SOFTWARE COM FOCO EM ONTOLOGIAS

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre em Sistemas e Computação, Universidade Salvador - UNIFACS, pela seguinte banca examinadora:

Laís do Nascimento Salvador - Orientadora _____
Doutora em Engenharia da Computação, Universidade São Paulo (USP)
Universidade Salvador - UNIFACS

Christina Von Flach Garcia Chavez_____
Doutora - Pontifícia Universidade Católica do Rio de Janeiro
Universidade Federal da Bahia - UFBA

Ricardo de Almeida Falbo _____
Doutor em Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro (UFRJ)
Universidade Federal do Espírito Santo - UFES

Salvador, 16 de fevereiro de 2009

RESUMO

O desenvolvimento de software no padrão Arquitetura Dirigida a Modelo (MDA) segue um conjunto de diretrizes, as quais podem promover significativos benefícios na concepção de um novo sistema. Neste sentido, através de ferramentas neste paradigma e com a geração de código, de forma automatizada, em diversas linguagens, é possível promover uma maior produtividade, menor custo e maior qualidade. Apesar destes benefícios, a utilização destes conceitos é pouco explorada e a falta de um processo adequado e conciso dificulta, ainda mais, a utilização deste padrão. Este traz para a indústria e para meio acadêmico a idéia de definir modelos para artefatos de software, definir transformações entre eles e prover ferramentas para automatizar tais transformações. Aliado ao padrão MDA, é proposto, neste trabalho a utilização de ontologias como o objetivo de aquisição de conhecimento de um determinado domínio, sendo a base de sustentação do processo proposto, o qual foi denominado ***MDAONTO***. Assim, a partir de ontologias são desenvolvidos os diversos modelos de software, sob um ponto de vista particular, até a geração do código. Algumas vantagens trazidas pelo processo proposto incluem a especificação de modelos bem formados em todos os níveis do processo de desenvolvimento, automatização de tarefas repetitivas e susceptíveis a erros, aquisição de conhecimento de modo mais produtivo, compartilhamento de conhecimento consensual, disseminação do conhecimento entre os *stakeholders*, apoio à rastreabilidade entre os artefatos, diminuição de riscos, entre outros.

Palavras chaves: Processo de desenvolvimento de software. Ontologia. Padrão MDA.

ABSTRACT

Model Driven Architecture (MDA) software development standard follows a set of guidelines, which can promote substantial benefits in design of a new system. In this sense, using tools of this paradigm and the automated generation code, in several programming languages, it is possible to augment productivity, lower cost and increase quality. Despite these benefits, the use of these concepts is little explored, and the lack of an appropriate and concise process makes it more difficult. This brings to industry and academy the idea of defining models for software artifacts, defining transformations between them and providing tools to automate these transformations.Allied to MDA standard, this work proposes the use of ontologies aiming to acquire knowledge in a particular domain, serving as the basis for upholding the proposed process, which was called **MDAONTO**. Thus, several models of software are developed under a particular point of view using ontologies and this process is up to the code generation.Some advantages offered by the proposed process include the specification of well designed models in all levels of process development, repetitive and susceptible to errors tasks automation, knowledge acquisition in a more productive way, consensual knowledge sharing, spread of knowledge among stakeholders, support for traceability between artifacts, risks reduction, among others.

Keywords*:* Software development process. Ontology. MDA standard.

.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CASE	*Computer-Aided Software Engineering*
CIM	*Computation Independent Model*
CWM	*Common Warehouse Meta-model*
DW	*Data Warehouse*
LINGO	LINguagem Gráfica para descrever Ontologias
MDA	*Model Driven Architecture*
MDA-SDLC	*MDA-based System Development Lifecycle*
MDD	*Model-Driven Development*
MDE	*Model Driven Engeneering*
MOF	*Meta-Object Facility*
OCL	*Object Constraint Language*
OMG	*Object Management Group*
OWL	*Web Ontologoy Language*
PDM	*Platform-Description Model*
PIM	*Platform Independent Model*
PSM	*Platform Specific Model*
QVT	*Query Views Transformations*
RDF	*Resource Description Framework*
RM-ODP	*Reference Model of Open Distributed Processing*
RUP	*Rational Unified Process*

SDP	*Software Development Process*
SPEM	*Software Process Engineering Metamodel*
UML	*Unified Modeling Language*
XP	*eXtreme Programming*

SUMÁRIO

2 ARQUITETURA DIRIGIDA A MODELO (MDA)

O padrão MDA[1] é uma especialização do padrão da engenharia de software dirigida por modelo (MDE), cujo foco é o desenvolvimento de software centrado em modelos. O MDE é uma abordagem concebida com intuito de lidar com a complexidade de plataformas, apresentando visões diferentes de modelos e suas sucessivas transformações (BEZIVIN, 2005).

O MDA é dirigido pela atividade de modelar o software, no qual o ponto chave está na importância de modelos (KLEPPE; WARMER; BAST, 2003; WARMER; KLEPPE, 2003). Na atividade de modelar sistema, existe um elemento básico responsável pela abstração – o modelo – que é uma descrição e/ ou uma especificação do sistema e de seu ambiente para alguma determinada finalidade. Um modelo pode ainda, ser apresentado como uma combinação de desenhos e de texto, o qual pode estar em uma linguagem específica ou natural (OMG, 2003a).

Segundo o padrão MDA, o modelo corresponde a uma representação descrita através de metamodelo e este é descrito através das diretrizes de um meta-metamodelo (OMG, 2001a). No decorrer deste capítulo, será descrita outra consideração sobre a estrutura que define o modelo, assim como, os padrões alinhados a esta estrutura.

Este capítulo está estruturado como se segue: a seção 2.1 apresenta o padrão MDA; a seção 2.2 apresenta o ciclo de vida MDA e uma breve analogia com o processo de desenvolvimento tradicional; a seção 2.3 expõe as transformações deste padrão; a seção 2.4 descreve benefícios desta abordagem e, por fim, a seção 2.5 tece algumas considerações do presente capítulo.

[1] É válido ressaltar que foram encontradas, na literatura técnica, diferentes terminologias que contextualizam MDA, a saber: *framework*, padrão, paradigma, modelo, processo, entre outros. Para este trabalho estas cinco nomenclaturas referem-se a significados análogos acerca deste assunto.

2.1 MDA

O padrão MDA apresenta três camadas de modelo e uma referente ao código gerado (OMG, 2003a), conforme ilustra a Figura 1, a saber: *(i) Modelo Independente de Computador* (CIM); *(ii) Modelo Independente de Plataforma* (PIM); *(iii) Modelo Específico de Plataforma* (PSM) e o *(iv)* Código.

(i) A plataforma CIM foca no entendimento do negócio do sistema a ser desenvolvido. Ou seja, este modelo descreve o ambiente em que o sistema irá operar independentemente de como o sistema será implementado. O CIM, além de ajudar para o entendimento do problema, serve como insumo para o desenvolvimento do PIM e do PSM.

(ii) A plataforma PIM foca a operação do sistema, abstraindo os detalhes necessários de uma plataforma particular. O sistema é modelado para suportar o domínio do negócio, independente de qual seja a tecnologia de implementação (KLEPPE; WARMER; BAST, 2003).

(iii) A plataforma PSM foca na transformação de um PIM para um modelo contendo informações do sistema em uma plataforma específica.

(iv) O Código, por fim, consiste no produto resultante da transformação do PSM em código fonte de uma tecnologia de implementação específica. Após a geração do código, outros produtos podem ser produzidos também, tais como arquivos de configuração, entradas de registro, *scripts*, entre outros (KLEPPE; WARMER; BAST, 2003).

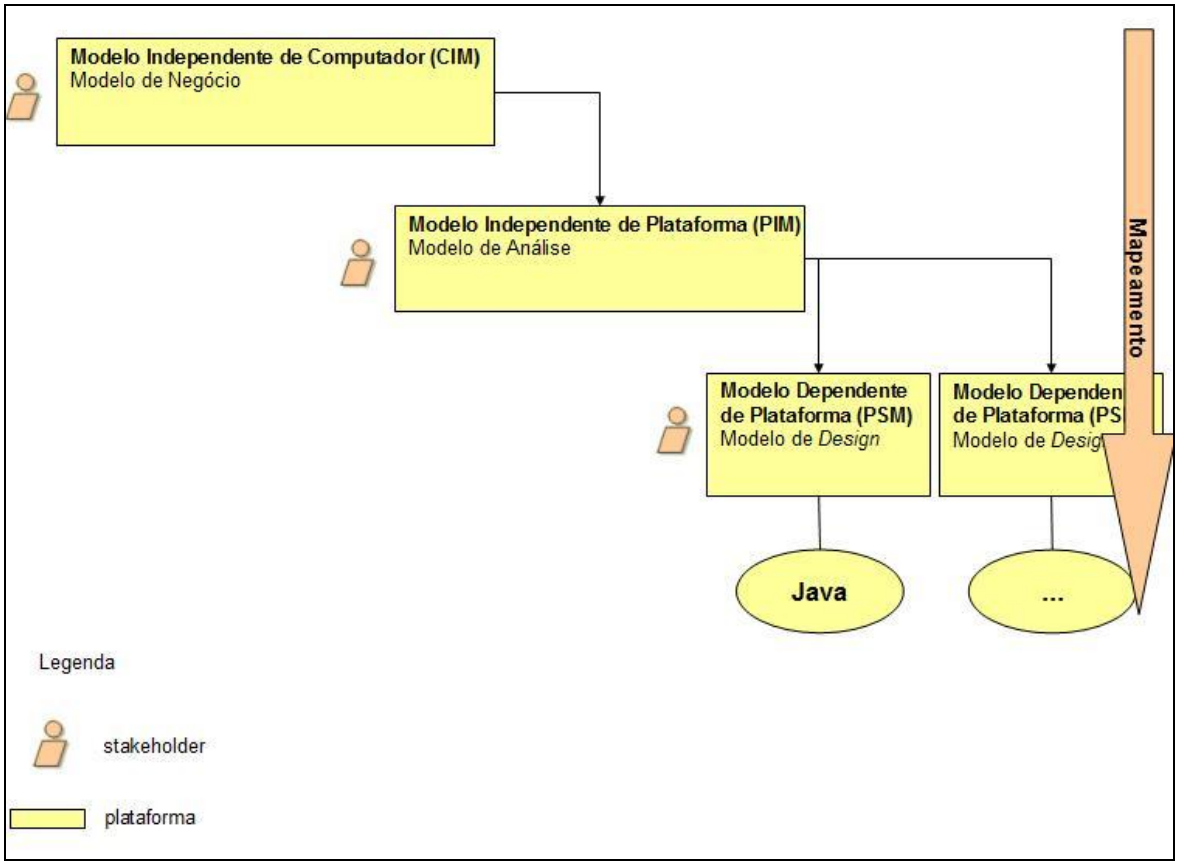


Figura 1 - MDA
Fonte: Adaptado de OMG (2003)

A figura acima exibe as três camadas de modelo e o código. Para ilustrar o código, no exemplo, foi utilizada a linguagem Java, contudo outras linguagens podem ser geradas. A partir, ainda, desta ilustração, é possível perceber que um único PIM originou dois PSM's, contudo, também, é possível a geração de diversos PSM's conforme a necessidade de quem utiliza esta estrutura.

Cada camada é definida conforme uma determinada perspectiva ou um nível de abstração contendo modelos específicos. Assim, a camada mais superior – CIM – apresenta um nível de abstração maior, enquanto, as camadas inferiores apresentam gradativamente um nível de abstração menor, chegando até o nível de código (OMG, 2003a). A divisão em níveis facilita a elaboração do sistema. Neste

sentido, um *stakeholder* que modela no nível PIM, por exemplo, não precisa se preocupar com características específicas de uma plataforma particular.

O MDA está relacionado com vários outros padrões propostos pelo OMG, tais como: *Linguagem de Modelagem Unificada* (UML) e *Common Warehouse Meta-model* (CWM) e *Meta Object Facility* (MOF) (OMG, 2003a).

UML é um metamodelo utilizado para descrever a modelos de sistemas, arquitetura, objetos, interação entre objetos, aspectos de modelagem de dados, bem como, aspectos de projeto dos componentes de desenvolvimento incluindo construção e montagem (OMG, 2004).

CWM é um metamodelo que descreve uma metalinguagem para especificar modelos de domínio de *Data Warehouse* (*DW*), ou seja, o CWM abrange todo o ciclo de vida de produção, construção e gerenciamento de aplicações e suporte (OMG, 2001a).

MOF é uma arquitetura de metamodelagem que está dividida em quatro camadas conforme ilustra a Figura 2, a saber: no nível mais baixo (M0) observam-se as instâncias do mundo real, a qual é representada por um modelo na camada (M1). Este por sua vez, encontra-se em conformidade com seu metamodelo definido no nível (M2) e o próprio metamodelo está em conformidade com o meta-metamodelo da camada (M3). Finalmente, este meta-metamodelo encontra-se em conformidade com ele mesmo. O padrão MDA está em conformidade com MOF.

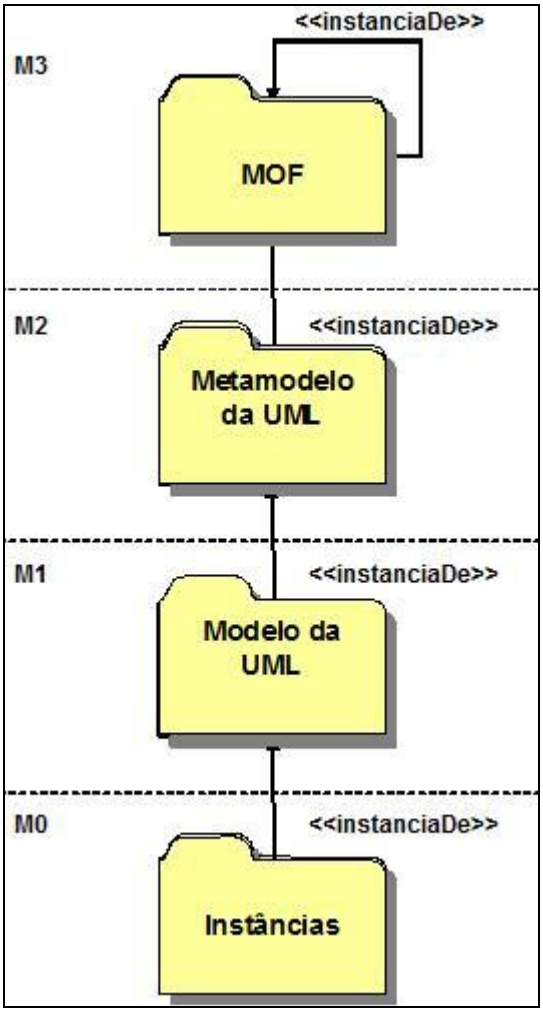


Figura 2 - MOF
Nota: Adaptado DJURIC *et. al*, (2005).

Na figura acima foram utilizados Metamodelo da UML (M2), Modelos da UML (M1) e as suas instâncias para ilustrar um exemplo da aplicabilidade do MOF. Contudo, em outro domínio, como, por exemplo, *Data Warehouse* (DW) e *Business Intelligence* (BI), o Metamodelo do CWM se apresentaria no (M2), um Modelo do CWM comporia o (M1) e (M0) corresponderia às instâncias correspondentes do metamodelo do nível (M2) (POOLE et al, 2003).

2.2 CICLO DE VIDA

O ciclo de vida no padrão MDA é bastante semelhante ao processo tradicional Cascata e suas fases são idênticas a este processo (KLEPPE; WARMER; BAST, 2003). Uma das maiores diferenças está na natureza dos artefatos criados durante o processo de desenvolvimento. No padrão MDA os modelos são formais, isto é, compreensíveis por computadores (KLEPPE; WARMER; BAST, 2003). Conforme ilustra a Figura 3, temos uma percepção do processo de desenvolvimento MDA, complementando as idéias analisadas na Figura 1 do presente capítulo.

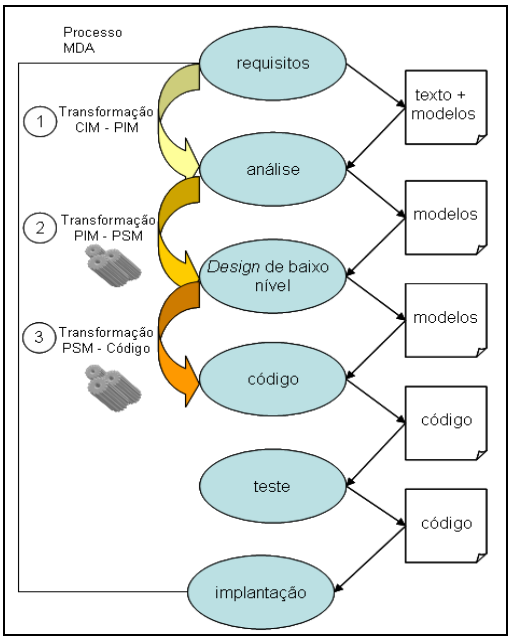


Figura 3 - Processo de desenvolvimento de software utilizando o padrão MDA
Fonte: Adaptado e traduzido de Kleppe, Warmer e Bast (2003).

Neste contexto, para a concepção do processo de desenvolvimento MDA é necessária a ocorrência de transformações entre camadas (ou modelos). Estas correspondem à saída de uma fase com maior nível de abstração para a entrada em outra fase com menor nível de abstração, como discutido anteriormente. Assim, a primeira (1) transformação, entre as camadas CIM-PIM, corresponde à saída da fase de negócio para a entrada na fase de análise. A transformação dois (2), entre as camadas PIM-PSM, corresponde à saída da fase de análise para a entrada na

fase de *design* de baixo nível. E a transformação três (3), transformação PSM para código, ocorre na saída da fase de *design* de baixo nível para a implementação com geração de código.

Analogamente ao processo MDA (Figura 3), é apresentado, na Figura 4 um processo tradicional. Neste contexto, observam-se seis fases deste processo, no qual, a finalização de uma fase fornece, também, subsídios para o início de uma nova fase, contudo, de forma não automatizada.

Neste cenário, caso sejam necessárias alterações após a implantação do sistema, estas terminam sendo feitas na fase de código (processo “atalho”), pois é menos oneroso do que voltar para o início do processo (KLEPPE; WARMER; BAST, 2003; SOMMERVILLE, 2007). Tal fato provoca o distanciamento entre os artefatos concebidos nas três primeiras fases e o código desenvolvido.

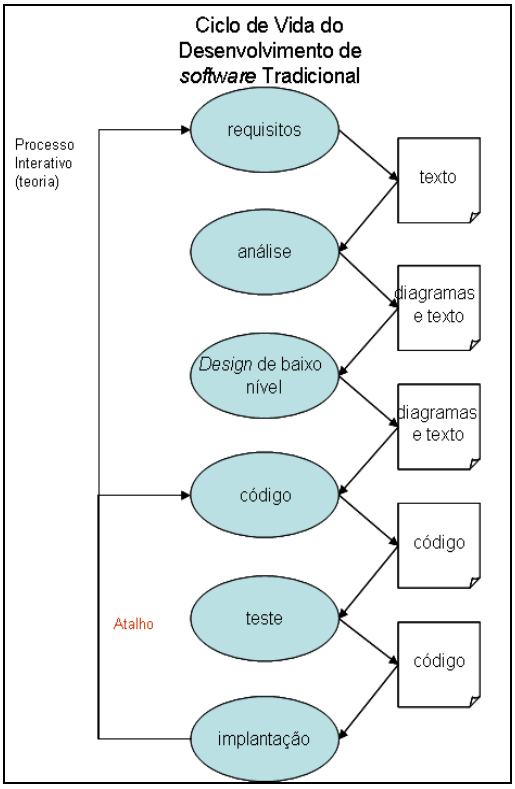


Figura 4 - Processo tradicional Cascata de desenvolvimento de software
Fonte: Adaptado e traduzido de Kleppe, Warmer e Bast (2003).

Uma visão geral do MDA, resumidamente é: (*i*) especificar um sistema independente de plataforma; (*ii*) especificar plataformas; (*iii*) escolher uma plataforma particular para o sistema; e por fim, (*iv*) transformar a especificação elaborada no passo (*i*) em um sistema na plataforma escolhida no passo (*iii*) (OMG, 2003a).

2.3 TRANSFORMAÇÃO

A principal vantagem de se utilizar o padrão MDA em desenvolvimento de sistemas reside na transformação entre modelos PIM, PSM e código, pois estas se dão de forma automatizada, diferentemente dos modelos tradicionais, os quais são realizados de forma manual, como citado anteriormente.

A transformação se produz a partir da aplicação de um conjunto de regras e técnicas que modificam um modelo para gerar outro modelo. A este processo se dá o nome de mapeamento. Assim, são possíveis as seguintes transformações automatizadas: PIM para PIM, PIM para PSM, PSM para PSM e PSM para PIM (OMG, 2001a).

A transformação de PIM para PIM é usada durante o ciclo de desenvolvimento sem precisar de nenhuma informação dependente de plataforma. Esta tem como objetivo o refinamento do modelo, ou seja, o modelo é melhorado e adequado às necessidades do sistema.

A transformação de PIM para PSM é mais interessante, pois um único PIM pode ser transformado em múltiplos PSM's (KLEPPE; WARMER; BAST, 2003; WARMER; KLEPPE, 2003). Neste sentido, a Figura 5 exibe uma visão geral desta transformação. Esta é realizada da seguinte forma: o modelo PIM, adicionado a outro modelo, o qual definirá a plataforma específica, produz o PSM. Este outro modelo corresponde aos dados da linguagem específica e/ ou marcas para a construção do modelo PSM específico (OMG, 2003a).

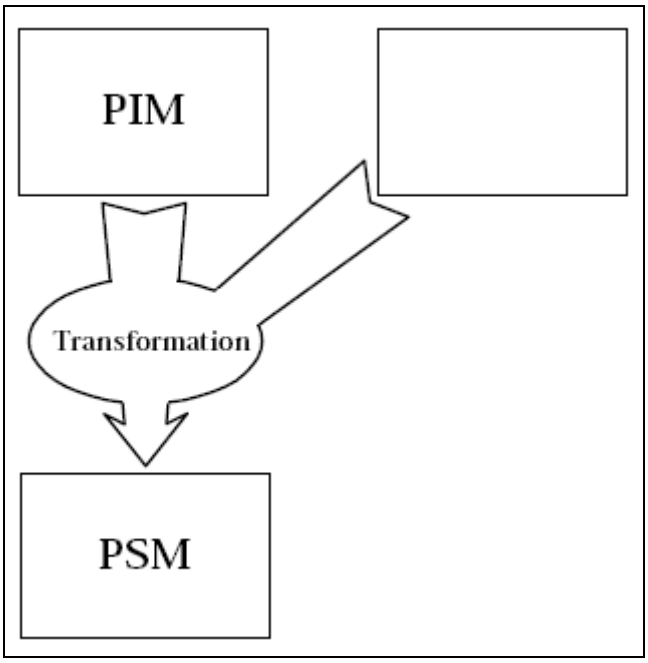


Figura 5 - Visão geral da transformação PIM – PSM
Fonte: Extraído do OMG (2003a).

A transformação de PSM para PSM está, geralmente, relacionada ao refinamento do modelo dependente de plataforma. E ainda, é possível observar a transformação de PSM para PIM, a qual é realizada para transformação de um modelo em tecnologia particular para um modelo independente de plataforma.

2.4 BENEFÍCIOS

Entre os benefícios da utilização do padrão MDA, a literatura técnica, normalmente, faz menção a quatro principais: produtividade, portabilidade, interoperabilidade e documentação (KLEPPE; WARMER; BAST, 2003; WARMER; KLEPPE, 2003).

2.4.1 Produtividade

No padrão MDA a produtividade deve-se principalmente à criação do PIM. Assim, os *stakeholders* podem trabalhar independentemente dos detalhes e da especificação da plataforma de destino. Isto melhora a produtividade em dois aspectos: o primeiro é que o desenvolvedor do PIM não terá o trabalho de projetar e escrever os detalhes da plataforma específica e o segundo é que o desenvolvedor com foco no PIM pode ater-se em solucionar os problemas de negócio.

No padrão MDA o modelo de análise não é apenas um artefato que contém uma especificação de um sistema a ser desenvolvido, mas sim, um modelo formal, que corresponde fielmente ao código-fonte gerado. Desta forma, a demanda de completeza deste modelo (nível PIM) deve ser maior que os modelos produzidos no processo de desenvolvimento tradicional, por exemplo. O modelo de análise deve ser descrito cuidadosamente para a produção correta e completa, pois o modelo é a reprodução fiel do código.

2.4.2 Portabilidade

Portabilidade corresponde à facilidade de um software ser transferido de um hardware ou de um ambiente de software para outro. No padrão MDA a portabilidade é alcançada, com o foco no desenvolvimento do PIM. O mesmo PIM pode ser automaticamente transformado em múltiplos PSM’s de diferentes plataformas. Assim, a especificação de um PIM é totalmente portável.

2.4.3 Interoperabilidade

Outro beneficio é a interoperabilidade, onde os múltiplos PSM's gerados pelo mesmo PIM podem se relacionar entre si. Em MDA, este relacionamento é feito através de *bridges,* conforme ilustra a Figura 6. Assim, se for possível transformar a partir de um PIM dois ou mais PSM's em diferentes plataformas, então todas as informações necessárias para ligar os PSM's estarão disponíveis.

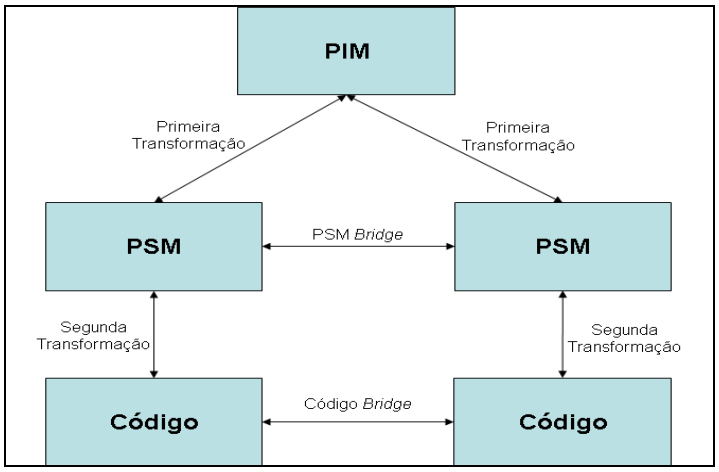


Figura 6 - Interoperabilidade MDA usando *bridges*
Fonte: Extraído e traduzido de Kleppe, Warmer e Bast (2003); Warmer e Kleppe (2003).

Neste contexto, se quisermos gerar um mapeamento entre dois PSM's é necessário se utilizar o PSM *Bridge* conforme ilustra a Figura 6. Assim, um elemento do PSM A pode ser mapeado em um elemento do PSM B. Este mapeamento só é possível por que o MDA endereça os PSM's e o PSM *Bridge* a partir do PIM.

2.4.4 Documentação

A documentação atualizada é um outro beneficio do padrão MDA. O CIM e/ ou PIM é usado para transformar os PSM's, que, por sua vez, são usados para gerar o código-fonte. Desta forma, para que uma alteração seja refletida no código, é necessário que seja alterado, primeiramente, o CIM e/ ou PIM, para, por fim, a

geração do PSM e o código-fonte. De tal modo, a documentação dos modelos não é descartada após a geração do código, como acontece em muitos processos de desenvolvimento de software.

2.5 CONCLUSÃO

Neste capítulo foram apresentados conceitos relativos ao padrão MDA, como, por exemplo, o que é o padrão MDA, quais são as suas camadas, quais os significados delas, entre outros. Foram analisadas, também, outras propostas definidas pelo OMG, tais como: UML, CWM e MOF.

Foi analisado o processo de desenvolvimento MDA e uma analogia ao processo tradicional foi realizada. Assim, foi visto que a diferença principal está nos artefatos gerados no processo de desenvolvimento MDA. Estes são modelos formais (legíveis por computadores), enquanto não existe esta formalização em um processo tradicional.

Outra diferença entre o processo MDA e o tradicional é que no padrão MDA as transformações ocorrem de forma automatizada entre as camadas, enquanto no processo tradicional, depois de concebido um artefato em uma determinada fase, este é encaminhado para uma nova fase de forma não automatizada até findar o processo de desenvolvimento deste ciclo.

Neste capítulo foram analisadas as transformações no processo considerando mais especificamente PIM para PSM’s. E por fim, foram discutidos os benefícios do padrão MDA. Assim, o grande ganho deste padrão, segundo (KLEPPE; WARMER; BAST, 2003; WARMER; KLEPPE, 2003), é a possibilidade de transformação a partir de um único PIM em diferentes PSM’s, permitindo melhor produtividade, menor custo e qualidade na produção de um software. Outro grande benefício desta abordagem é o foco em documentação, mantendo concisão entre a análise e o projeto de software, o qual é de grande valia para as organizações,

devido às dificuldades enfrentadas pelas mesmas na atualização de documentação devido à evolução e/ ou a mudanças de requisitos, por exemplo.

3 ONTOLOGIA

Neste capítulo é discutida uma série de considerações elucidadas na literatura técnica sobre os conceitos relativos a ontologias, em que não se pretende esgotar o assunto, e sim, apresentar, uma visão geral deste tópico, o qual é importante no escopo deste trabalho.

Neste contexto, a palavra ontologia é oriunda do grego, onde *onto* significa *ser* e *logo* significa *palavra*. O termo original advém da palavra aristotélica *categoria*, que pode ser utilizada para categorização de algo (ALMEIDA; BAX, 2003; ALMEIDA, 2006). Nesta perspectiva, uma ontologia visa categorizar algo, como, por exemplo, o domínio de qualidade de software, multimídia, medicina, entre outros.

O termo ontologia pode ser utilizado como uma *especificação formal* e *explícita* de uma *conceituação compartilhada*. Nessa definição, *formal* significa relativo à forma; *especificação explícita* diz respeito a conceitos, propriedades, relações, funções, restrições, axiomas, explicitamente definidos; *compartilhado* quer dizer conhecimento consensual; e *conceituação* diz respeito a um modelo abstrato de algum fenômeno do mundo real (BORST, 1997).

Em Guarino (1998) foi citada uma ontologia como sendo um sistema particular de categorização que fornece uma visão do mundo. Para Medeiros (2004), o conceito acerca de ontologia é uma especificação dos conceitos de um determinado domínio e suas relações, restrições e axiomas, definidos de forma declarativa. Ainda, na W3C (2004), ontologia é um termo emprestado da filosofia que se refere à ciência de descrever os tipos das entidades no mundo e de como são relacionados.

Além disso, uma ontologia indica alternativas para representar um domínio do conhecimento, assim, conforme discorrido em Almeida (2006, p. 106):

> [...] estuda uma série de formalismos capazes de representar os conceitos, as relações entre os conceitos e a semântica de um domínio do

> conhecimento. A semântica, nesse contexto, é parte de um modelo formal em que declarações lógicas representam o conhecimento do domínio, a ser manipulado em um sistema computacional.

Para a Engenharia de Software é possível verificar uma série de benefícios com a inserção de ontologias no processo de desenvolvimento. Assim, a partir de uma ontologia, é possível ter a obtenção do conhecimento de um determinado negócio por todos os envolvidos no âmbito do projeto, como pode ser visto na explanação de Falbo (1998, p. 50):

> [...] principais benefícios do uso de ontologias na especificação de software é a possibilidade de se adotar uma estratégia mais produtiva para a aquisição de conhecimento. [...] Especialistas são recursos escassos, de alto custo, mas essenciais para a aquisição de conhecimento e, portanto, devem ser melhor aproveitados. Assim, é importante reutilizar e compartilhar o conhecimento capturado. [...] Ontologias provêem um eficiente mecanismo para aumentar a produtividade na fase de aquisição de conhecimento. Em uma abordagem baseada em ontologias, a aquisição de conhecimento pode ser realizada em duas etapas. Primeiro, o conhecimento geral do domínio, relevante para uma variedade de sistemas no domínio, é capturado e especificado na forma de ontologias. Estas, por sua vez, são usadas para guiar a segunda etapa da aquisição de conhecimento, onde as particularidades de uma aplicação específica são consideradas. Desta forma, uma mesma ontologia pode ser usada para guiar o desenvolvimento de várias aplicações, diluindo os custos da primeira etapa da aquisição e permitindo a reutilização e compartilhamento de conhecimento [...].

No projeto de software o benefício da utilização de uma ontologia está associado ao desenvolvimento do sistema em um alto nível de abstração para reuso. Além disso, permite desenvolver reuso e compartilhar conhecimento de domínio da aplicação utilizando vocabulário comum (GÓMEZ-PÉREZ; FERNÁNDEZ-LÓPEZ; CORCHO, 2004).

Este capítulo se propõe a realizar uma análise sobre ontologias, explorando seus benefícios. Para tanto, o presente capítulo está dividido em: 3.1 Tipificação; 3.2 Métodos de Construção de Ontologias; 3.3 Linguagens; e, finalmente, a seção 3.4, a Conclusão, a qual apresenta considerações sobre o presente capítulo.

3.1 TIPIFICAÇÃO

Uma ontologia pode ser classificada sob aspectos diferentes, a saber: quanto ao grau de formalismo para especificar o vocabulário de termos e seus significados; quanto ao tipo de estrutura e ao assunto de conceituação da ontologia; quanto à função da ontologia; quanto à estrutura dos tipos de classes presente na ontologia; quanto à sua função no processo de desenvolvimento de sistemas computacionais (ANEXO A – TIPIFICAÇÃO DE ONTOLOGIAS) (ALMEIDA; BAX, 2003; ALMEIDA, 2006).

Uma das mais discutidas, analisadas e difundidas na literatura técnica do assunto é a classificação conforme o nível de generalização, tal como ilustra a Figura 7 desta seção. Esta se divide em: (*i*) Ontologia genérica – apresenta conceitos bastante gerais, que são independentes de um problema ou domínio particular; (*ii*) Ontologia de domínio – apresenta um vocabulário relacionado a um domínio genérico; (*iii*) Ontologia de tarefa – apresenta um vocabulário relacionado a uma tarefa genérica ou atividade; e (*iv*) Ontologia de aplicação – apresenta conceitos que dependem de um domínio e de uma tarefa, que são freqüentemente especializados das ontologias relacionadas (GUARINO, 1998).

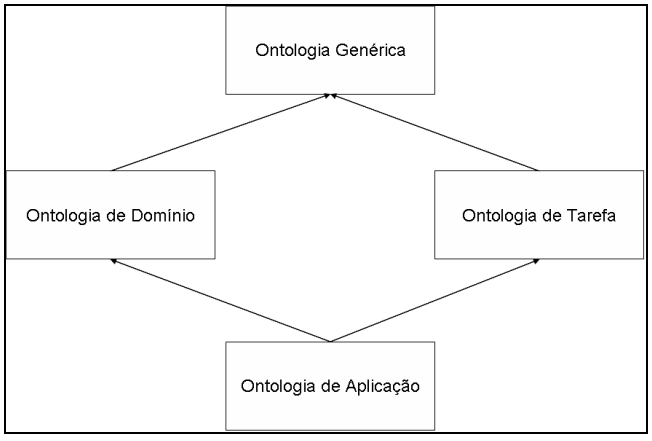


Figura 7 - Classificação da ontologia conforme o nível de generalização
Fonte: Extraído e traduzido de Guarino (1998)

Outra classificação bastante interessante é a em relevância a taxonomia que modelam o domínio, tal como: (*i*) Ontologia *lightweight* – apresenta conceitos,

taxonomias de conceitos, relações entre conceitos e propriedades que descrevem conceitos. (*ii*) Ontologia *heavyweight* – adiciona axiomas e restrições para ontologia *lightweight*. Esta classificação pode ser conferida em (GÓMEZ-PÉREZ; FERNÁNDEZ-LÓPEZ; CORCHO, 2004).

3.2 MÉTODOS DE CONSTRUÇÃO DE ONTOLOGIAS

Em Noy e Guinness (2001) citado por Almeida (2006) ressaltam que não existe uma maneira única ou correta de modelar um domínio. Assim o processo de construção é obrigatoriamente iterativo e os conceitos em uma ontologia podem estar inseridos em objetos ou relacionamentos em domínio específico.

Com a finalidade de ilustrar as possíveis possibilidades para construção de ontologias, esta seção apresenta dois métodos conforme ilustra o Quadro 1 para produção de ontologias, a saber: *Enterprise Ontology* e *Systematic Approach for Building Ontologies* (SABiO), contudo, existe uma série de outros métodos que merecem igual atenção, tais como: *METHONTOLOGY, SENSUS-based method e On-to-knoweledge,* entre outros.

Ontologia – Autor (ano)	**FASES**
Enterprise Ontology – Uschold e King (1995) estendida Uschold e Grüninger (1996)	(*i*) identificação do propósito e o escopo; (*ii*) construção; (*ii.a*) captura; (*ii.b*) codificação; (*ii.c*) integração; (*iii*) avaliação; (*iv*) documentação;
Systematic Approach for Building Ontologies (SABiO) – Falbo (1998)	(*i*) identificação do propósito e especificação de requisitos; (*ii*) captura da ontologia; (*iii*) formalização da ontologia; (*iv*) integração com ontologias existentes; (*v*) avaliação; (*vi*) documentação

Quadro 1 – Análise de métodos para a construção de ontologias
Fonte: Elaboração própria.

3.2.1 Uschold and King's *Method*

O *Uschold and King's Method* apresenta algumas diretrizes baseadas na experiência a partir do desenvolvimento da *Enterprise Ontology* e este método é ilustrado na Figura 8.

Conforme os autores, este método define um pequeno número de estágios que pode vir a servir de base para futuras metodologias mais amplas. Os estágios definidos pelo *Uschold and King's Method* são, a saber: (*i*) identificação do propósito e escopo; (*ii*) construção; (*iii*) avaliação; e (*iv*) documentação (GÓMEZ-PÉREZ; FERNÁNDEZ-LÓPEZ; CORCHO, 2004).

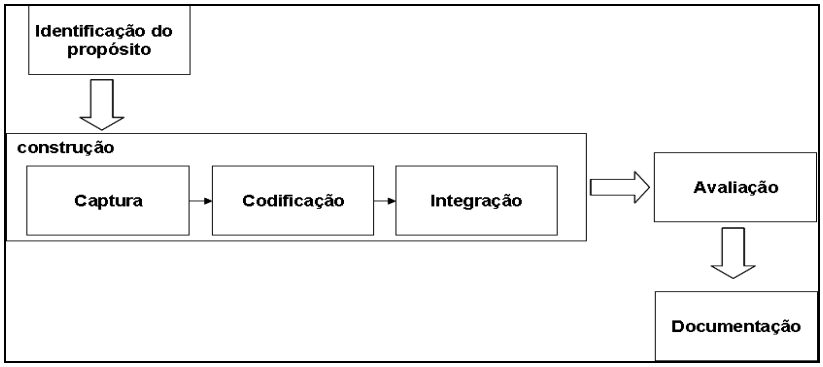


Figura 8 - *Uschold and King's Method*
Fonte: Extraído de Gómez-Pérez, Fernández-López e Corcho (2004).

O primeiro estágio corresponde à *(i) identificação do propósito* e se propõe a identificar o motivo da construção da ontologia e quais são os seus usos projetados. Neste estágio é, também, identificado qual a gama de usuários que a ontologia abrange.

O segundo estágio corresponde à *(ii) construção* da ontologia e está dividido em:

(*ii.a*) *captura* – envolve identificação dos conceitos e relações relevantes no domínio de interesse, a geração de definições textuais precisas para estes elementos (conceitos e relações) e o estabelecimento de termos que referem a tais conceitos.

(*ii.b*) *codificação* – envolve a representação dos conceitos capturados em uma linguagem formal; Comprometimento em seguir uma meta-ontologia, escolha de uma linguagem e a geração de código.

(*ii.c*) *integração com ontologias existentes* – durante a captura e/ ou processo de codificação é possível a integração com ontologias existentes. Segundo os autores é uma tarefa difícil e muitos progressos têm sido realizados nesta área. Uma sugestão viável é tornar explícito todos os pressuposto a cerca da ontologia (USCHOLD; KING, 1995).

O terceiro estágio, *(iii) avaliação*, objetiva a verificação de uma ontologia em termos de especificações de requisitos, questões de competência e/ ou do mundo real. Por fim, quarto estágio, *(iv) documentação,* onde todas as decisões importantes devem ser especificadas, tanto no que tange aos principais conceitos definidos na ontologia, como no que diz respeito às primitivas usadas para expressar definições na ontologia.

3.2.2 *Systematic Approach for Building Ontologies* (SABiO)

O SABiO foi desenvolvido em 1997 e teve sua primeira publicação em 1998 e este, foi baseado em *Uschold and King's Method* (FALBO, 2004). Com intuito de transformar a construção de ontologias em um processo sistemático, segundo o autor, foram propostos um ciclo de vida de desenvolvimento para construção de ontologias, uma notação gráfica para descrever ontologias e uma abordagem de engenharia de conhecimento (FALBO, 1998).

Este processo, conforme ilustra a Figura 9, apresenta as seguintes etapas, a saber: (*i*) identificação de propósito e especificação de requisitos (*ii*) captura da ontologia; (*iii*) formalização da ontologia; (*iv*) integração com ontologias existentes; (*v*) avaliação; e (*vi*) documentação. Nesta figura, as linhas tracejadas indicam que existe interação constante, porém mais fraca, entre as etapas associadas. Ainda, na figura observa-se que as linhas cheias mostram o fluxo principal, e, a linha

envolvendo as etapas de *captura* e *formalização* da ontologia ilustra uma forte interação e, consecutiva iteração, que ocorre entre as etapas (FALBO, 1998).

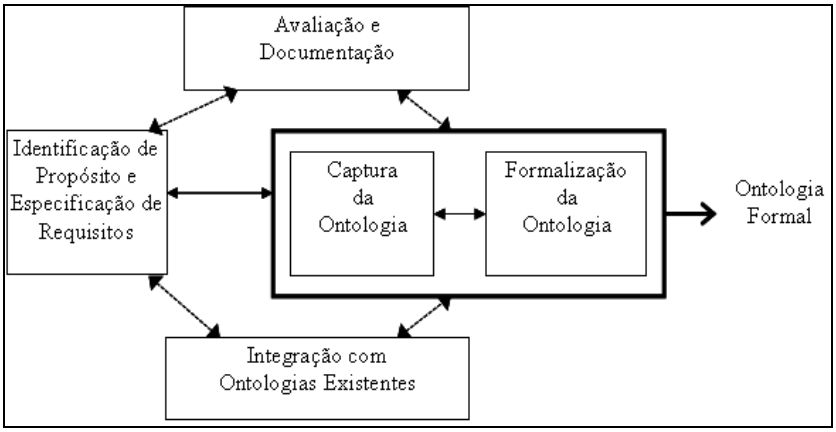


Figura 9 - Etapas do desenvolvimento de uma ontologia e suas interdependências
Fonte: Extraído de Falbo (1998).

A *identificação do propósito e especificação de requisitos* refere-se à primeira atividade e corresponde à cobertura de questões que a ontologia pode responder e seus potenciais usuários, ou seja, a competência da ontologia. Nesta fase é delimitado o escopo da ontologia.

A *captura da ontologia* refere-se à captura da conceituação do universo de discurso, com embasamento na competência da ontologia. Nesta atividade deve-se identificar: conceitos e relações, dicionário de termos, taxionomias (organizadas em categorias e sub-categorias), axiomas. Esta atividade é um processo iterativo e fortemente ligado à avaliação

A *formalização da ontologia* visa representar formalmente as ontologias do universo de discurso capturado na atividade anterior, o que envolve o comprometimento com uma meta-ontologia, a seleção de uma linguagem de representação e a criação da ontologia formal. Assim, segundo o autor, devem-se observar os formalismos disponíveis e caso não exista algum que possa ser utilizado diretamente ou adaptado, é necessário propor um formalismo adequado.

A *integração com ontologias existentes* propõem-se à atividade de integrar a ontologia que está sendo criada com outras ontologias existentes, caso haja

necessidade. Esta atividade pode ser efetuada em paralelo com a *captura* e/ ou *formalização* visando aproveitar as *conceituações* existentes.

A *avaliação*, última atividade, observa-se se a ontologia descrita satisfaz os objetivos estabelecidos na especificação, a qual deve ser avaliada conforme o critério de qualidade para o projeto de ontologias. Esta atividade, conforme ilustrou a Figura 10, é realizada em paralelo com as atividades *captura* e *formalização* da ontologia, em um processo iterativo.

A *documentação*, por fim, determina que todo o desenvolvimento da ontologia deve ser documentado, incluindo propósitos, requisitos e cenários de motivação, as descrições textuais da conceituação, a ontologia formal e os critérios de projetos adotados. Neste contexto, observa-se que esta atividade ocorre em paralelo com as outras atividades.

3.3 LINGUAGENS

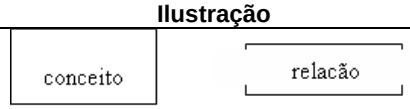
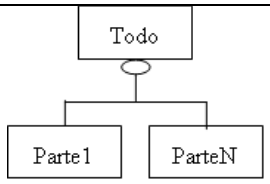
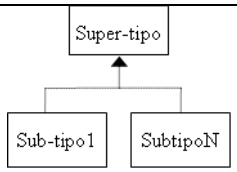
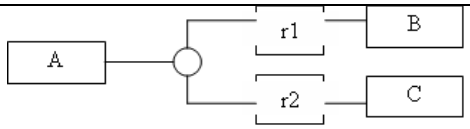
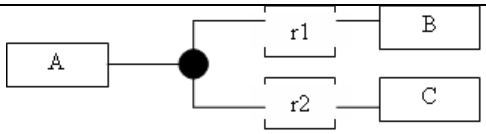
A representação de uma ontologia é feita a partir de uma linguagem, a qual deve expressar seu domínio. Algumas linguagens para representar uma ontologia têm outras funções e são adaptadas para esta tarefa. Também, existem linguagens que foram desenvolvidas especificamente para representação de ontologias (ALMEIDA, 2006).

Nesta seção serão explanados dois exemplos de linguagens: LINGO (LINguagem Gráfica para descrever Ontologias) e OWL (*Web Ontology Language*). Contudo, existe uma série de outras linguagens importantes, as quais merecem igual atenção, tais como: KADS, *Ontolingua*, entre outras.

3.3.1 LINGO (LINguagem Gráfica para descrever Ontologias)

A linguagem LINGO (LINguagem Gráfica para descrever Ontologias) foi criada para modelagem de ontologias e esta, utiliza elementos gráficos para representação, facilitando, desta forma, muito na compreensão de um domínio. Esta linguagem foi criada, por que, segundo o autor, as linguagens de modelagem de especialidade de KADS, linguagens de modelagem de objetos e representações de entidades e relacionamentos embutiam semânticas que eram desnecessárias para descrever ontologias. Alem disto, a semântica da LINGO é capaz de capturar um conjunto de axiomas representado por uma notação gráfica (FALBO, 1998).

As primitivas da linguagem LINGO, de modo geral, está ilustrada no Quadro 2. Esta contém conceitos e relações, relações de composição, notação para hierarquia de conceitos, condicionantes. A partir destes elementos é possível desenvolver uma série de ontologias para os diversos domínios.

Primitivas	Ilustração
Conceitos e Relações	
Relações de Composição	
Notação para Hierarquia de Conceitos	
Condicionante Ou-exclusivo entre Relações	
Condicionante de Obrigatoriedade entre Relações	

Quadro 2 - LINguagem Gráfica para descrever Ontologias
Fonte: Extraído de Falbo (1998).

A Figura 10 ilustra um exemplo extraído de FALBO (1998), onde é possível observar a relação binária entre os conceitos de atividade e produto, sendo que podem também existir relações de ordem superior. As cardinalidades são representadas nas relações e servem para indicar a quantidade de instâncias que um conceito pode ter em uma relação. As cardinalidades do tipo (0..n) não são representadas já que não impõem nenhum axioma.

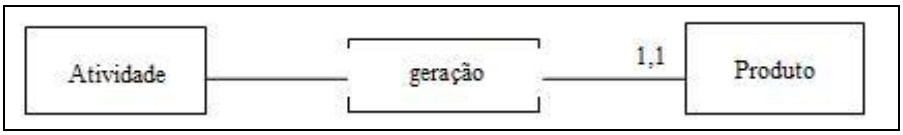


Figura 10 - Exemplo de uma relação binária entre conceitos
Fonte: Extraído de Falbo (1998).

A partir da cardinalidade (1,1), é possível extrair os seguintes axiomas:

- ✓ $(\forall p)\ (produto(p) \rightarrow (\exists a)\ (geração(p,a))$ – este tipo de axioma representa a cardinalidade mínima 1.

- ✓ $(\forall p, a_1, a_2)$ $(geração(p,a_1) \wedge geração(p,a_2) \rightarrow a_1 = a_2)$ – este tipo de axioma representa a cardinalidade máxima 1.

Outros tipos de relações, tais como relações entre instâncias de um mesmo conceito e relações de composição, são melhores explanadas em Falbo (1998).

Atualmente, existe um Perfil UML que apresentam as diretrizes desta linguagem.

3.3.2 OWL (*WEB Ontology Language*)

A OWL (*WEB ONTOLOGY LANGUAGE*) é uma linguagem que visa definir e instanciar ontologias na *WEB*. Neste contexto, a ontologia desenvolvida em OWL pode incluir descrições de classes, de propriedades e suas instâncias. Esta linguagem possui atualmente três sublinguagens – *Lite*, DL (*Description Logics*) e *Full*. Estas apresentam expressividade gradual (do menos expressivo para o mais expressivo) e foram desenvolvidas para serem utilizadas por programadores e usuários (W3C, 2004).

A OWL *Lite* suporta aqueles usuários que necessitam principalmente de uma classificação hierárquica e restrições. Por exemplo, ela só permite restrições de cardinalidade com valores igual zero (0) ou um (1).

A OWL *DL* foi projetada para suportar o segmento de negócio e ampara os usuários que desejam uma maior expressividade mantendo a integridade computacional (legível por computadores). Ainda, OWL *DL* garante a decidibilidade, ou seja, que as computações terminarão em tempo finito. A OWL *DL* inclui todas as construções da linguagem OWL, porém elas somente podem ser usadas com algumas restrições (uma classe não pode igualmente ser um objeto ou a propriedade, uma propriedade não pode igualmente ser um objeto ou uma classe).

A OWL *Full* definida para suportar máxima expressividade e a liberdade sintática do *Resource Description Framework* (RDF) sem nenhuma garantia

computacional. Neste contexto, nesta linguagem, uma classe pode ser tratada simultaneamente como uma coleção de objetos ou como um objeto por si mesmo. A OWL *Full* permite que uma ontologia aumente o vocabulário pré-definido de RDF ou OWL. É improvável que algum software de inferência venha a ser capaz de suportar completamente cada recurso da OWL *Full*.

3.4 CONCLUSÃO

Neste capítulo foram apresentados as definições e os conceitos básicos mais relevantes para contextualizar ontologia. Assim, conforme visto, elas objetivam a compreensão coerente de um determinado assunto, permitindo a universalização dos conceitos. Desta forma, o entendimento consensual de um domínio é um dos benefícios de utilização das ontologias na modelagem de sistemas, permitindo uma melhor comunicação entre todos os indivíduos envolvidos no âmbito de um projeto.

Ainda, foram discutidos outros benefícios como o emprego de ontologias na fase inicial de produção de software, uma vez que a mesma possibilita uma estratégia mais produtiva para a aquisição de conhecimento, em oposição ao modelo tradicional que a cada nova aplicação a ser desenvolvida, uma nova conceituação necessita ser elaborada quase sempre a partir do nada. Neste sentido, é ratificada a importância de utilização da mesma na fase inicial em um ciclo de vida do software.

Neste capítulo foram apresentadas classificações de ontologias com intuito de identificar a classificação que subsidiará o processo descrito no Capítulo 5. Entretanto, foi identificado que não existe um consenso e que o processo de classificação de ontologias não é trivial. Estudiosos deste assunto classificam estas com critérios divergentes, apresentando classificações sobrepostas e ambíguas quando comparadas umas com as outras (ALMEIDA, 2006).

Na seqüência, foram analisados processos de desenvolvimento de ontologias, uma vez que fornecem subsídios para produção de uma nova ontologia. A especificação de suas fases foi, também, área de interesse desta análise.

Ainda, foi apresentado um estudo de possíveis linguagens para produção de ontologias. Neste estudo destacaram-se a linguagem LINGO (FALBO, 1998) e a linguagem OWL (W3C, 2004). Nesta última, verifica-se a existência de três sublinguagens (*Lite*, DL e *Full*), que apresentam expressividade gradual.

4 PROCESSO DE SOFTWARE

O conceito de processo de software pode variar em diferentes situações, portanto, é difícil chegar a um consenso com relação à sua definição. Neste sentido, o que significa processo de software? Segundo Sommerville (2007) um processo de software é um conjunto coerente de atividades para a produção de software.

Ainda, processo de software pode ser visto como um elo que mantém unidos os métodos e as ferramentas. O método**,** por sua vez, proporciona os detalhes para a construção do software, como segue: planejamento e estimativas de projeto, análise de requisito de software e de sistemas, projeto de estrutura de dados, arquitetura de programa e algoritmo de processamento, codificação, teste e manutenção (PRESSMAN, 1995).

O *Rational Unifed Process* (RUP) define processo de software como um conjunto de passos ordenados parcialmente e executados para uma determinada finalidade durante o desenvolvimento de software, enquanto o método é a forma sistemática e regular de realizar algo: planos ou procedimentos detalhados e ordenados de formas lógicas, seguidas de modo a cumprir uma tarefa ou atingir uma meta (RUP, 2001).

Os processos de softwares estão divididos em quatro fases fundamentais, a saber: especificação de software, desenvolvimento de software, validação de software e evolução do software (SOMMERVILLE, 2007). Em Presmman (1995) é possível vislumbrar a constituição do processo de softwares dividida em apenas três fases: definição, desenvolvimento e manutenção. Embora apresentem pontos de vistas diferentes sobre a divisão de um processo de desenvolvimento de software, o conteúdo das fases assegura uma similaridade entre os processos propostos por Sommerville (2007) e Presmman (1995).

A importância do processo de software está fundamentada na premissa em que a qualidade de um produto de software é intimamente dependente da do

processo pelo qual este foi construído. Neste cenário, as conseqüências de um processo maduro são, a saber: definição clara dos papéis e responsabilidades; acompanhamento da qualidade do produto e da satisfação do cliente; expectativas para custos, cronograma, funcionalidades e para a qualidade do produto são usualmente alcançadas (ROCHA; MALDONADO; WEBER, 2001).

Este capítulo visa apresentar um cenário de processos de softwares e suas evoluções, desde o tradicionalmente clássico até os mais atuais. O intuito é ilustrar como eram executados e como evoluíram.

4.1 PARADIGMAS DE PROCESSOS DE DESENVOLVIMENTO DE SOFTWARES

Em 1970, foi publicado o primeiro modelo de processo de desenvolvimento e este se originou de processos mais gerais da Engenharia de Sistema, o qual é chamado de modelo cascata. Neste modelo, cada fase deste processo é realizada depois de finalizada a fase anterior e assim consecutivamente, até findar todo o processo (SOMMERVILLE, 2007; PRESMMAN, 1995).

O modelo cascata fornece-nos um mecanismo de controle em todas as etapas de desenvolvimento com maior rigor. Neste contexto, o mesmo é bastante utilizado em grandes projetos, com equipes distantes geograficamente e com requisitos formais (SOMMERVILLE, 2007).

Esta abordagem apresenta alguns problemas, os quais podem ser encontrados, com mais freqüência, a saber: a existência de falhas nas primeiras atividades torna onerosa a sua correção, sendo necessário, muitas vezes, o re-começo do ciclo de desenvolvimento; dificuldade de declaração de todos os requisitos no início do projeto pelo cliente; e uma versão de desenvolvimento do software só está disponível depois de todas as etapas cumpridas no cronograma (PRESMMAN, 1995).

Sommerville (2007) retrata neste modelo, que devido à existência relacionada aos custos com a produção e a aprovação de documentos, as iterações são

onerosas e envolve um re-trabalho, assim, o autor diz que após um pequeno número de iterações, é normal suspender partes do desenvolvimento, como a especificação de requisitos, e prosseguir com as fases posteriores de desenvolvimento. A conseqüência disto pode significar a produção de um sistema que não atende as necessidades do usuário e, ainda, pode levar a construção de sistemas mal estruturados, uma vez que os problemas de projeto foram contornados por meio de artifícios de programação.

O modelo evolucionário é outro paradigma de processo de desenvolvimento de software. Este é baseado no conceito em que as atividades de especificação, desenvolvimento e validação são realizadas intercaladamente em pequenos incrementos até a entrega final do software. A abordagem incremental fornece meios de gerenciar o problema ocasionado pelas mudanças de requisitos e/ ou de recursos do sistema. Para tanto, os incrementos devem ser pequenos e selecionados cuidadosamente (SOMMERVILLE, 2007).

As atividades deste modelo têm objetivo de realizar a entrega de um conjunto de funcionalidades (pequenos incrementos) ao cliente final. Estes são realizados conforme o grau de entendimento dos *stakeholders* (requisitos melhor compreendidos) e a prioridade do requisito definida pelo cliente. No final, quando todos os incrementos forem terminados, o software estará pronto para ser utilizado (versão final).

Para o desenvolvimento evolucionário têm-se dois tipos principais, a saber: (*i*) desenvolvimento exploratório – neste desenvolvimento tem-se o foco de trabalhar com o cliente, a fim de explorar requisitos e entregar o sistema final, neste, o desenvolvimento começa das partes do sistema que foram compreendidas e tem-se a conclusão do sistema com a inclusão de novas funcionalidades por parte do cliente; e (*ii*) prototipação *throwaway* – neste desenvolvimento são explorados os requisitos que foram mal compreendidos pelo cliente, assim, visa definir melhor os requisitos do sistema.

Em 1990 surgiu os Métodos Ágeis. Este tinha o objetivo de amparar a entrega de sistemas nos quais os requisitos se modificam continuamente, permitindo a entrega do software de maneira mais rápida, com foco no software

(SOMMERVILLE, 2007). Neste cenário, o Manifesto Ágil prega alguns princípios: indivíduos e interações entre eles mais que processos e ferramentas, software em funcionamento mais que documentação abrangente, colaboração do cliente mais que negociação de contratos e respostas a mudanças mais que seguir um plano (BECK, 1999; 2001). Atualmente, existe um grande número de abordagens ágeis, a saber: Crystal; *Adaptative Software Development* (ASD), Scrum, *Feature Driven Development* (FDD); Métodos para Desenvolvimento de Sistemas Dinâmicos (DSDM), entre outras.

O *Rational Unifed Process* (RUP) é mais um exemplo de um processo de desenvolvimento de software. Este pode ser analisado de acordo com três perspectivas: (*i*) uma perspectiva dinâmica, onde são ilustradas as fases; (*ii*) uma perspectiva estática, onde são ilustradas as atividades; (*iii*) uma perspectiva prática, onde há sugestão de boas práticas (SOMMERVILLE, 2007).

No aspecto dinâmico são apresentados as fases, as iterações e os limites do ciclo de vida do processo. O aspecto estático é descrito em termos de componentes, disciplinas, atividades, fluxos de trabalhos, artefatos e papeis de processo (RUP, 2001).

4.2 PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE DIRIGIDOS A MODELOS

Esta seção apresenta processos de desenvolvimento de software dirigidos a modelos. Estes processos foram de fundamental importância para a definição do processo proposto. Estes foram descritos com o objetivo de apresentar ao leitor a sua estrutura e as principais características, com intuito de fornecer uma visão geral.

Tem-se, assim, nesta seção, a saber: ODAC *Methodology*; MODA-TEL *Methodology;* MASTER *Methodology* e *MDA-based System Development Lifecycle* (MDA-SDLC).

4.2.1 Metodologia ODAC

A metodologia ODAC foi construída tendo como base os conceitos do *framework Reference Model of Open Distributed Processing* (RM-ODP). Este *framework* arquitetural define conceitos para o desenvolvimento de sistemas distribuídos e aplicações. Assim, a metodologia ODAC define um conjunto de diretrizes para descrever a arquitetura do sistema, do nível mais abstrato ao nível mais concreto, que é a execução do sistema (GERVAIS, 2002).

O padrão ODP provê um conjunto de conceitos e regras de estrutura. O principal conceito é o *ponto de vista*, que permite estruturar a atividade de modelagem. Um *ponto de vista* é uma subdivisão de uma especificação complexa do sistema. Ele corresponde a uma perspectiva particular, permitindo que o sistema seja visto, a partir de um determinado ângulo, focando em problemas específicos. Os cinco *pontos de vistas* deste padrão são: *entreprise, informational, computacional, engineering* e *technology*.

Toda a especificação do sistema é obtida a partir desses cinco *pontos de vista,* no entanto o ODP é um *framework* arquitetural e não uma metodologia. Neste contexto, ODP não oferece uma seqüência entre os *pontos de vista* e ferramentas, como as de notação.

A metodologia ODAC define, então, os passos que identificam uma correspondência entre as atividades de análise, *design* e implementação com os *pontos e vista* da ODP. Assim, é possível observar na Figura 11 que na metodologia ODAC a análise está associada aos *pontos de vista entreprise, informational* e *computacional*, o *design* está associado ao *ponto de vista engineering* e a implementação está associada ao *ponto de vista technology*.

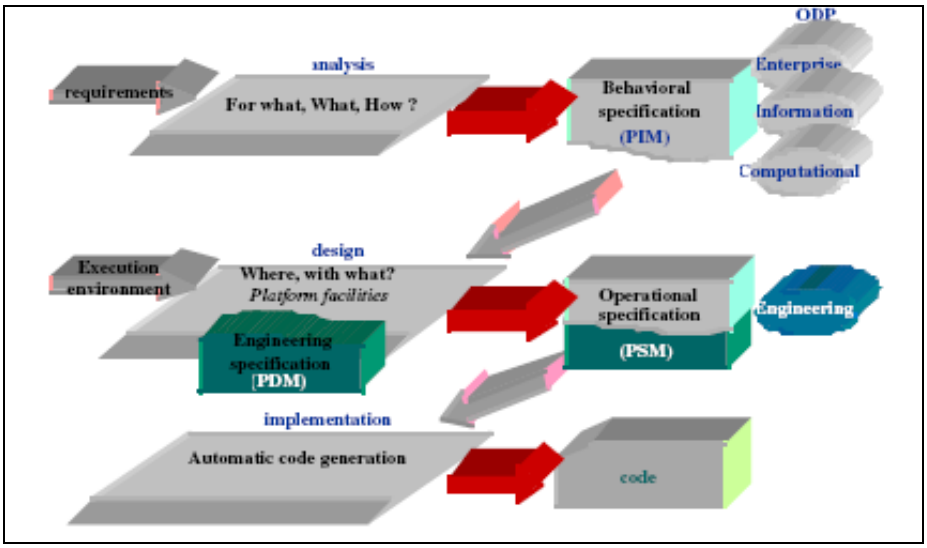


Figura 11 – Metodologia ODAC
Fonte: Extraído de Gervais (2002).

As três categorias de especificação são identificadas, a saber: *Behavioral Specification*, *Engineering Specification* e *Operation Specification*.

Behavioral Specification representa a saída da análise. Ela corresponde às especificações dos *pontos de vista Entreprise, Informational e Computacional* e descrevem o sistema conforme o objetivo, contexto nas organizações, as informações manipuladas e as tarefas que são realizadas. Este é o PIM da plataforma MDA.

Engineering Specification representa o *ponto de vista Engineering* e descreve a característica do ambiente de execução. É possível identificar *Platform-Description Model* (PDM) nesta categoria. Assim para cada tipo de ambiente, é gerado PDM *Guideline* que ajuda a descrever um ambiente específico.

Operation Specification é o resultado da projeção *Behavioral Specification* (PIM) e reflete na execução real do ambiente. Contém a descrição para a geração de código. Corresponde à transformação do PIM em conformidade com PDM. Este é o PSM da plataforma MDA.

4.2.2 Metodologia MODA-TEL

A metodologia MODA-TEL é destinada ao desenvolvimento de aplicações distribuídas baseadas nos princípios e nos conceitos do padrão MDA. Esta identifica fases e atividades de desenvolvimento, como também, os papeis e os produtos de cada atividade de acordo com *Software Process Engineering Metamodel* (SPEM) (GAVRAS et al, 2004a, 2004b).

A metodologia MODA-TEL define diferente papeis em três categorias, tal como segue: (*i*) *knowledge builder* – os *stakeholders* que constroem o conhecimento (repositórios) a ser usado em diferentes projetos MDA. Esta categoria inclui, a saber: arquitetos, peritos em plataforma, engenheiros da qualidade e peritos de metodologia*;* (*ii*) *knowledge facilitators* – os *stakeholders* que montam, combinam, personalizam e distribuem o conhecimento para cada projeto específico MDA. Esta categoria é representada pelos gerentes de projetos e engenheiros de qualidade*; (iii) knowledge users* – os *stakeholders* que desenvolvem o conhecimento e facilitam outras categorias do usuário. Enquadram-se nesta categoria os *designers* e engenheiros de software.

Nesta metodologia observam-se cinco fases de desenvolvimento segundo a Figura 12, a saber:

(1) Gestão de Projetos – visa gerenciar e monitorar o projeto; *apresentam as seguintes atividades*: seleção *Software Development Process* (SDP); organização do projeto; e gerência de qualidade;

(2) Preparação Preliminar – visa identificar necessidades da modelagem e a transformação de requisitos; *apresentam as seguintes atividades*: identificação da plataforma; identificação de linguagem de modelagem; identificação das transformações; e a definição da estratégia de rastreabilidade;

(3) Preparação Detalhada – visa obter as especificações da modelagem e da transformação; *apresentam as seguintes atividades*: especificação de linguagem de modelagem; e especificação das transformações;

(4) Infrastructure setup – visa seleção da ferramenta, a qual oferece suporte e gestão de metadados a serem utilizados na fase de execução; *apresentam as seguintes atividades*: seleção de ferramentas; e *metadata management;*

(5) Execução do Projeto – visa produzir os artefatos necessários do software e produtos finais; *apresentam as seguintes atividades*: analise de requisitos; modelagem; verificação e validação; transformações; codificação e teste; integração e distribuição; e operação e manutenção.

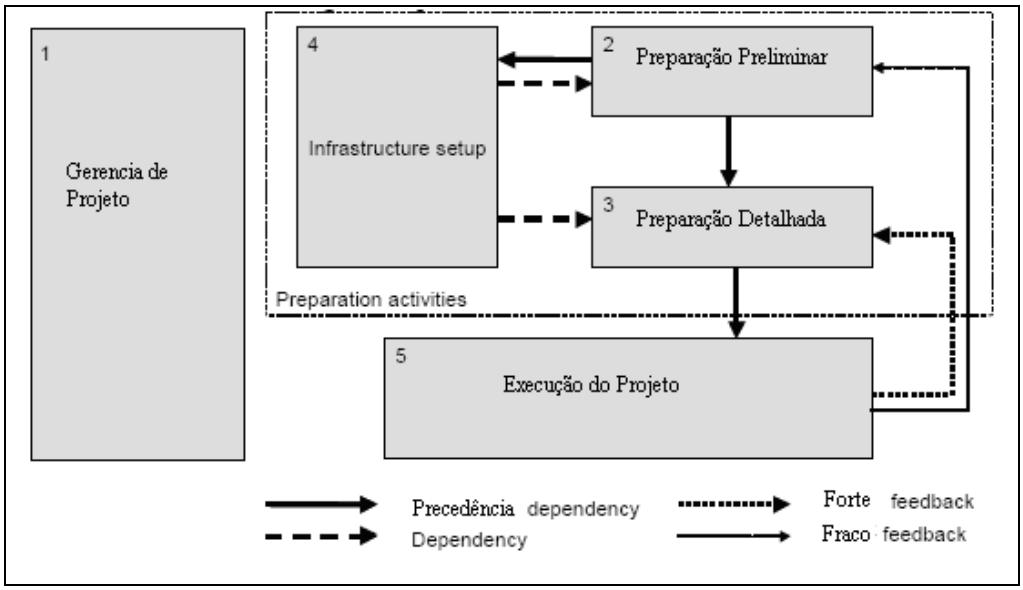


Figura 12 – Fases de desenvolvimento da metodologia MODA-TEL
Fonte: Extraído de Gavras et. al., (2004).

As fases estão associadas a categorias de usuários. Assim, a fase *(1)* é executada principalmente pelos *knowledge facilitators*. As fases *(2)*, *(3)* e *(4)* são executadas principalmente por *knowledge builder* e a fase *(5)* é executada principalmente por *knowledge users*.

4.2.3 *Master Methodology*

O projeto MASTER desenvolveu uma metodologia com o mesmo nome. Esta apresenta um processo *Model-Driven Development* (MDD) e um conjunto de métodos da engenharia (LARRUCEA; DIEZ; MANSELL, 2004). Os papeis e os produtos de cada atividade de acordo com Software *Process Engineering Metamodel* (SPEM).

As fases desta metodologia são:

- ✓ Captura de requisitos do usuário: abrange a elicitação de requisitos e documentação.
- ✓ Definição do contexto PIM: define o domínio do sistema a ser desenvolvido. O resultado desta fase é uma definição clara do sistema, do objetivo e do domínio.
- ✓ Especificação de requisitos PIM: nesta fase são definidos os requisitos do modelo. A atividade principal desta fase inclui a especificação (caso de usos) e agentes (não funcionais) do sistema.
- ✓ Análise PIM: nesta fase observam-se os modelos com suas restrições.
- ✓ *Design*: nesta fase têm-se os modelos detalhados da estrutura e do comportamento do sistema.
- ✓ Codificação e integração: nesta fase é desenvolvido e verificado o código. O código pode ser gerado por um PSM a partir de uma ferramenta MDA.
- ✓ Teste: verifica e valida o sistema final.
- ✓ *Deployment*: nesta fase ocorre a transição do sistema para o ambiente do usuário.

4.2.4 MDA-Ciclo de Vida de Desenvolvimento de Sistema Baseado (MDA-SDLC)

A proposta MDA-SDLC foi desenvolvida utilizando as idéias do ciclo de desenvolvimento de software genérico e a abordagem MDA. Esta abordagem consiste em duas partes principais: (*i*) linguagem de modelagem (sintaxe e semântica) e (*ii*) um processo (ASADI; RAVAKHAH; RAMSIN, 2008).

Ela não se propõe a desenvolver uma metodologia concreta, mas um processo geral que define fases e atividades como ilustra a Figura 13. Estas fases possuem atividades associadas, respectivamente, como se segue:

(1) Fase Iniciação do Projeto – atividades: definição CIM; especificação de requisitos; obtenção do financiamento e suporte; definição da equipe; definição do plano geral; e seleção de ferramentas;

(2) Fase de Desenvolvimento PIM – atividades: produzir análise PIM; projeto arquitetural; produzir projeto PIM; verificação e validação; e generalização;

(3) Fase PSM & Desenvolvimento de Código – atividades: transformar PIM para PSM; e produção de código e teste;

(4) Fase Distribuição – Atividades: teste final do sistema; finalizando a documentação do sistema e de usuário; e transição do sistema ao ambiente de usuário;

(5) Fase Manutenção – Atividades: suporte e manutenção.

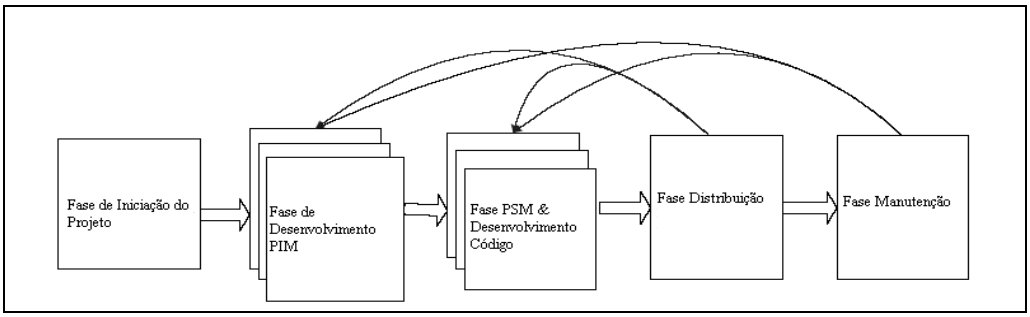


Figura 13 – Proposta MDA-Based SDLC
Fonte: Extraído de Asadi, Ravakhah e Ramsin (2008).

O autor desta proposta ilustra um quadro comparativo contendo os resultados de avaliação de outras metodologias baseadas no padrão MDA. Assim, o autor conclui neste trabalho que sua metodologia apresenta pontos importantes, os quais não são identificados em outras abordagens, tais como: METODOLOGIA MODA-TEL e METODOLOGIA *ODAC*.

4.3 CONCLUSÃO

Nesse capítulo vimos definições acerca da terminologia do processo de software. Ela é definida e explorada de modo diferente entre os grandes especialistas na área de Engenharia de Software. Ainda, foi realizado um estudo na evolução dos processos de software, desde a sua origem até os processos emergentes, como, processos de desenvolvimento de software dirigido a modelo.

Após perpassar por uma série de processos, foi percebido que não existe um processo correto ou errado e sim, um processo mais adequado a projetos específicos. Em projetos menores em que as pessoas não são partes substituíveis e com requisitos instáveis é aceitável que processos ágeis sejam mais adequados, enquanto que para o desenvolvimento de sistemas robustos com equipes grandes e requisitos formais é mandatária a aplicação de modelos rígidos como forma de gerenciamento (SOMMERVILLE, 2007). A escolha de um modelo é fortemente dependente das características do projeto. Assim, é importante conhecer alguns modelos de ciclo de vida de desenvolvimento de software e em que situações são aplicáveis.

Entre os processos emergentes, foram analisadas as abordagens de processos de desenvolvimento de software dirigidos a modelos, o quais são focos deste trabalho. Sendo assim, estas foram apresentadas em ordem cronológica, de modo a facilitar a compreensão do leitor. A partir da apresentação destas, observam-se algumas características:

A metodologia *ODAC* (GERVAIS, 2002) é baseada no *framework* ODP e não apresenta processo associado ao domínio do negócio, o qual corresponde à camada CIM no padrão MDA. Esta metodologia não apresenta uma definição das atividades identificadas por fases, assim como não são identificados artefatos de entrada e saída que compõe cada fase deste processo. Ainda, não apresenta uma definição de *stakeholders* que compõe o processo.

A metodologia *MODA-TEL* (GAVRAS et al, 2004a, 2004b) está decomposta em fases com suas atividades correspondentes. Apresenta definição de categoria de *stakeholders* para concepção das fases do processo. Nesta metodologia, uma categoria dos *stakeholders* está associada a uma fase. Em *MODA-TEL* é possível encontrar artigos com estudo de caso, facilitando o entendimento da mesma. Esta metodologia, também, não apresenta fase associada à plataforma CIM no padrão MDA.

A metodologia MASTER (LARRUCEA; DIEZ; MANSELL, 2004) apresenta um processo *Model-Driven Development* (MDD) e um conjunto de métodos da engenharia e, apresentam os papeis e os produtos de cada atividade de acordo com Software *Process Engineering Metamodel* (SPEM). Ainda, não ocorre uma associação dos *stakeholders* a tarefas específicas, apenas referencia o *SPEM*. A MASTER apresenta, de modo geral, diretrizes que compõe a plataforma CIM no padrão MDA. De tal modo, a MASTER apresenta-se como uma evolução das outras metodologias dirigidas a modelos.

MDA-SDLC (ASADI; RAVAKHAH; RAMSIN, 2008) está estruturado em fases e apresenta atividades agregadas. Neste processo são expostos os artefatos gerados em cada fase, contudo não há associação entre os artefatos e atividades. O processo não apresenta uma definição de *stakeholders*, fazendo referência ao descrito em MODA-TEL (GAVRAS et al, 2004a, 2004b) para definição dos profissionais de tecnologia da informação, no entanto estes usuários estão associados às fases desta metodologia (Metodologia MODA-TEL), e não ao MDA-SDLC.

5 MDAONTO: UM PROCESSO MDA DE DESENVOLVIMENTO DE SOFTWARE COM FOCO EM ONTOLOGIAS

Neste capítulo é apresentado um processo de desenvolvimento de software adequado ao padrão MDA, chamado ***MDAONTO***. Este está em consonância com a aquisição de conhecimento a partir de modelos de negócios baseados em ontologias e, ainda, apresenta uma abordagem dirigida a modelos com suas transformações.

O processo visa o reuso de ontologias, permitindo uma gama de benefícios, os quais serão mais bem esclarecidos no decorrer da apresentação deste processo, a saber: obtenção de conhecimento de forma mais produtiva; desenvolvimento em um alto nível de abstração; compartilhamento de conhecimento consensual; disseminação do conhecimento entre os membros da equipe; apoio a rastreabilidade entre os artefatos; diminuição de riscos, entre outros.

Conforme o cenário exposto, uma ontologia é o ponto de partida para este processo de software. Assim, sugere-se a utilização de ontologias de domínio (GUARINO, 1998) com classificação taxonômica igual a *heavyweight* (GÓMEZ-PÉREZ; FERNÁNDEZ-LÓPEZ; CORCHO, 2004), ainda que outras classificações sejam aceitáveis. Caso não exista uma ontologia inicial para o domínio selecionado, então se sugere a construção de uma ontologia com classificação taxonômica igual a *lightweight*, por ser mais simples sua elaboração.

No contexto de processos existentes, o ciclo de vida no padrão MDA é bastante semelhante ao processo tradicional e suas fases são idênticas à deste processo, como já foi discutido no Capítulo 2 (KLEPPE; WARMER; BAST*, 2003*). A divergência entre os dois ciclos de vida é originada, dentre outros motivos, pela automatização entre as fases no padrão MDA, enquanto que no processo tradicional esta transição entre as fases é realizada de forma manual. Em (KLEPPE; WARMER; BAST*, 2003*) são comparados esses dois processos e é afirmado que, à

medida que os processos tradicionais caminham, o código tende a se distanciar da modelagem, entretanto, no padrão MDA, código e modelagem estão consistentes, pois o código é gerado a partir da documentação.

A definição de *papeis* dos *stakeholders* do ***MDAONTO*** foi baseada no RUP, ainda que não exclusivamente, pois se vale também da proposta descrita em (RÊGO et al, 2007) apresentada no ANEXO B – PAPEL E RESPONSABILIDADE. Os artefatos foram, do mesmo modo, baseados no RUP. No entanto é válido salientar que os *templates* dos documentos podem ser adaptados para a organização onde será aplicado o processo.

A seguir, são explanados os seguintes tópicos: seção 5.1 Visões dos *stakeholders*; seção 5.2 Processo ***MDAONTO***; seção 5.3 Rastreabilidade; seção 5.4 Iterações; seção 5.5 Avaliação comparativa e por fim, na seção 5.6, é apresentada a conclusão deste capítulo.

5.1 VISÕES DOS STAKEHOLDERS

O uso de ontologias facilita a aquisição de informações específicas de um domínio. Esta facilidade se dá pelo fato da ontologia conter informações consensuais de um determinado domínio. Tal benefício tem sido explorado em alguns trabalhos, como Almeida e Bax, (2003), Almeida (2006), Falbo (1998), Guizzardi (1997), dentre outros.

As ontologias do tipo *heavyweight* são interessantes no contexto de engenharia de software para obtenção de conhecimento, pois apresentam conceitos, taxonomias de conceitos, relações entre conceitos, propriedades que descrevem conceitos e adiciona axiomas e restrições representando de forma completa um domínio de negócio. Contudo, estas podem ser complexas e de difícil compreensão para os envolvidos no processo de construção de software.

Uma possível solução é a geração de “visões” com diferentes graus de complexidade (detalhamento), obtidas a partir de ontologias de domínio, que facilitem a compreensão e validação por diferentes tipos de *stakeholders* (RÊGO et al, 2007), mas também passíveis de transformação para outros modelos em um processo MDA típico.

Estas visões são descritas através de modelos que caracterizam níveis diferentes de abstração da ontologia com intuito de facilitar a legibilidade e compreensão para os indivíduos de uma organização. Uma classificação inicial de visões, sugerida em Rêgo e outros (2007) é: (*i*) visão do cliente gerencial, (*ii*) visão do cliente operacional e (*iii*) visão do analista.

A visão do cliente gerencial corresponde à visão macro do negócio, composta por uma ilustração que contempla os principais conceitos e relacionamentos, além da descrição desses conceitos (vocabulário).

A visão do cliente operacional incorpora a anterior, sendo complementada com conceitos e relacionamentos não tratados na visão gerencial. Adicionalmente, um subconjunto de axiomas oriundos da ontologia é transformado em regras de negócio em uma linguagem de fácil entendimento para o cliente operacional. Esta visão pode ser enriquecida com novas regras e requisitos informados pelo próprio cliente operacional.

A visão do analista é uma visão mais técnica. Além de incorporar as visões anteriores, ela pode apresentar artefatos relacionados ao modelo de negócio (RUP, 2001), tais como: glossário de negócio; regras de negócio; caso de uso de negócio; modelo de objetos de negócio; entre outros, cabendo ao analista selecionar os artefatos mais adequados ao sistema a serem produzidos.

5.2 PROCESSO MDAONTO

Nesta seção é apresentado, de modo geral, o processo de desenvolvimento de software adequado ao padrão MDA. As fases do processo proposto estão descritas conforme os critérios definidos no Quadro 3 e as suas respectivas atividades estão apresentadas conforme os critérios definidos no Quadro 4.

Nome da Fase*:** corresponde ao nome da fase do ***MDAONTO.
Aderência ao padrão MDA: corresponde à(s) camada(s) equivalente no padrão MDA.
Entradas e saídas: corresponde a insumos (entradas) e produtos (saídas) da fase.

Quadro 3 - *Template* da fase para caracterização do *MDAONTO*
Nota: Elaboração própria.

Nome da Atividade*:** corresponde ao nome da atividade do ***MDAONTO.
Objetivo: corresponde ao propósito para a realização da atividade.
Artefatos: corresponde à informação que é produzida, modificada ou usada por um processo de desenvolvimento de software (RUP, 2001).
***Stakeholders*:** corresponde a uma sugestão de *papeis* e responsabilidades.
Considerações: apresentação de características e/ ou referencial teórico complementar para entendimento da atividade.

Quadro 4 - *Template* da atividade para caracterização do *MDAONTO*
Nota: Elaboração própria.

As fases do processo ***MDAONTO*** constituem-se em agrupamentos de atividades e estas são, a saber: (*Fase I*) Iniciação; (*Fase II*) Fase de Desenvolvimento PIM; (*Fase III*) Fase de Desenvolvimento PSM e de Código; e (*Fase IV*) Fase de Distribuição e Manutenção.

A Figura 14 ilustra uma visão do ciclo de vida do processo de desenvolvimento de software proposto. Nesta figura são ilustrados os principais elementos deste processo.

Na *Fase I* (iniciação) é possível observar o diferencial do modelo proposto quando se compara com os processos de desenvolvimento de software dirigidos a modelo. Na nova proposta existe a compreensão do domínio de negócio a partir de

ontologias. Neste cenário, o Engenheiro de Ontologias tem a função de entendimento do modelo de negócio a partir da utilização de ontologias. Assim, este *stakeholder* é responsável pelas atividades principais de Selecionar Ontologia e/ ou Criar Ontologia. Para a concepção dos artefatos que compõem o modelo CIM, recortes da ontologia são gerados com base nas visões dos clientes gerencial e operacional e do analista. Para a definição deste modelo é necessária uma equipe mínima, sendo esta formada pelo cliente (gerencial e operacional), analista (responsável), *designer* de negócios, engenheiro de ontologias e analista do processo de negócios. A depender da complexidade do negócio, este é dividido em pacotes que representam um conjunto de funcionalidades que serão implementadas em cada ciclo do processo. Desta forma, este processo conta, também, com uma abordagem iterativa, a qual será abordada na seção 5.4.

A *Fase II* (Fase de Desenvolvimento PIM) corresponde ao desenvolvimento do PIM, a qual se beneficia do modelo de negócio construído na fase anterior e é, assim, desenvolvido o modelo de análise PIM com a utilização da *Linguagem de Modelagem Unificada* (UML) e da *Object Constraint Language* (OCL). Esta é uma transição não automatizada. Deste modo, ela é realizada pelo Especificador de Requisitos e pelo Analista de Requisitos. Contudo, é uma transição mais facilmente realizada, uma vez que a modelagem de negócio já foi concebida.

A Fase *III* (Fase de Desenvolvimento PSM e de Código) corresponde à transformação do modelo PIM para modelo PSM, e por fim, para código, utilizando uma ferramenta no padrão MDA. Na *Fase IV* (Fase de Distribuição e Manutenção) são realizados os testes finais (validação) e a implantação em ambiente de produção.

Vale salientar que neste processo a comunicação é essencial para o bom funcionamento das atividades. Assim, os *papeis* e as responsabilidades dentro do grupo precisam estar muito bem definidos, a fim de facilitar o progresso das atividades. Para tanto, é necessária a definição de um documento que apresente a descrição formal das atividades e das responsabilidades do profissional na organização. Assim, o ANEXO B apresenta todas as definições dos papeis e

responsabilidades utilizadas no processo proposto. Ainda, pode ser visto no ANEXO C as definições dos artefatos utilizados no processo proposto.

Para cada modelo de análise que é processado pelo ***MDAONTO***, é gerada uma entrega parcial, a qual pode ser testada e validada pelo usuário. Essas entregas parciais podem ser processadas rapidamente em função do apoio de ferramentas para transformações automatizadas. Desta forma, apesar do processo ***MDAONTO*** ser baseado em artefatos e na concepção do modelo tradicional, ele apresenta características ágeis como entregas rápidas, incrementos etc.

Algumas das premissas do manifesto ágil estão presentes no ***MDAONTO***, a saber: *colaboração do cliente* – esta se dá através da inserção do cliente como um participante através do processo de avaliação precoce do negócio; *respostas rápidas a mudanças* – através da entrega de pequenos incrementos com validações até o momento em que o sistema é finalizado. As respostas rápidas também são obtidas pela facilidade de compreensão dos modelos por parte de todos os envolvidos no âmbito do projeto, pois estes se apresentam em alto nível de abstração, fator que promove maior legibilidade para a validação do cliente. Vale ressaltar que as alterações ocorrem em alto nível, sendo mais simples e perceptível a sua adequação.

Diferente do método ágil onde *software em funcionamento* mais que documentação abrangente, o processo proposto dá *ênfase à documentação formal*, a qual corresponde ao “espelho” do código gerado. Outra diferença está no fato de que a abordagem ágil foca os *indivíduos e interações* entre eles mais que processos e ferramentas. Por sua vez, o objetivo deste trabalho é definir *um processo formal com auxílio de ferramentas,* tais como: MDA e *Computer-Aided Software Engineering* (CASE’s). Contudo há uma relevante *importância dos indivíduos* presentes nesta abordagem, como, por exemplo, o cliente que participa ativamente desse processo.

Além disso, é possível a execução de forma incremental e iterativa do processo proposto. A atividade especificar análise PIM tem objetivo de realizar a entrega de um conjunto de funcionalidades relativas ao negócio para o cliente final.

No final, quando todos os incrementos forem finalizados, o software estará pronto para ser utilizado.

Uma vez findado o processo, é possível a necessidade de suporte e de manutenção do sistema. Se a manutenção for evolutiva, um novo ciclo pode ser necessário.

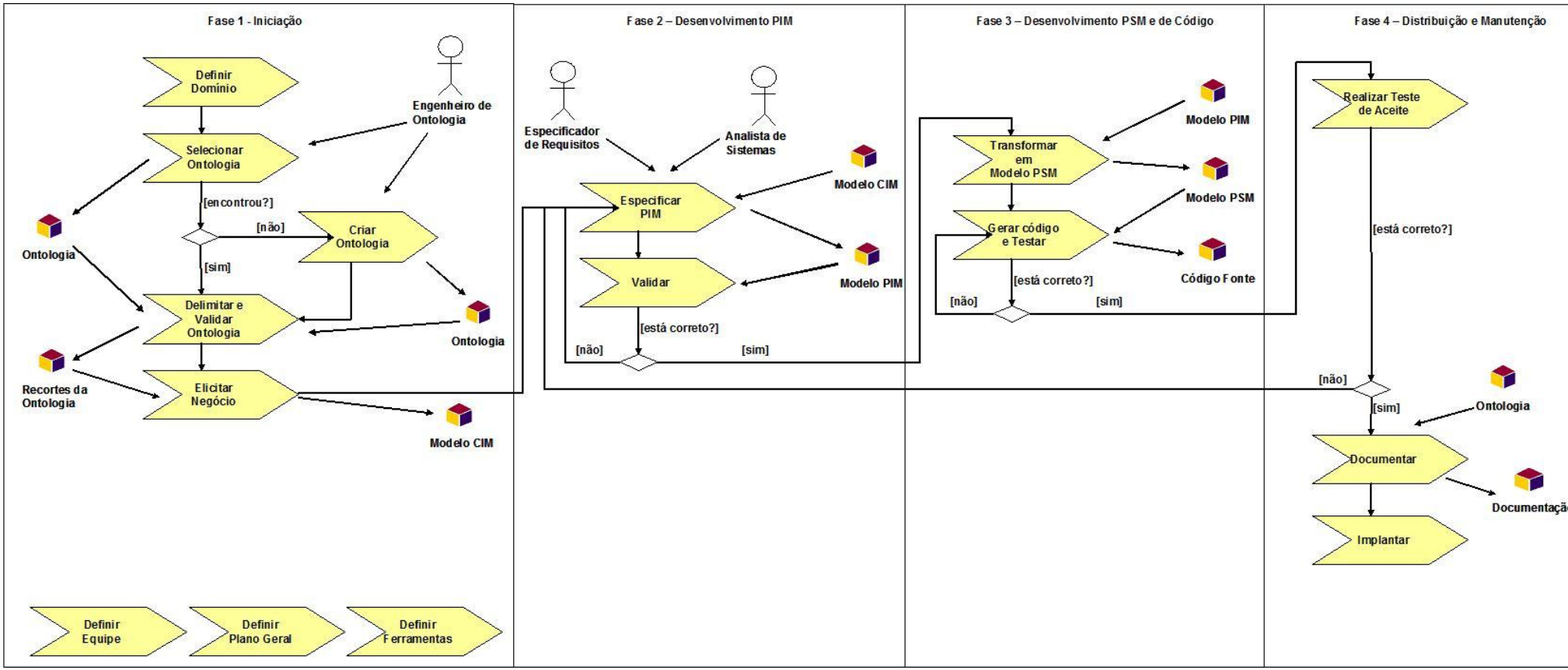


Figura 14 - *MDAONTO*: um processo MDA de desenvolvimento de software com foco em Ontologias
Nota: Elaboração própria.

5.2.1 (FASE I) Iniciação

Na fase Iniciação, conforme ilustrado no Quadro 5, dá-se início ao projeto através das seguintes atividades, como se segue: (*i*) *Definir domínio*, (*ii*) *Selecionar ontologia*, (*iii*) *Delimitar e validar ontologia* e (*iv*) *Elicitar e validar negócio*.

***Nome da Fase*:** Iniciação	
Aderência ao padrão MDA: CIM Nesta fase, espera-se a compreensão de todos os *stakeholders* envolvidos sobre o entendimento do modelo de negócio.	
Entrada	**Saída**
o cliente expressa sua necessidade	documento de visão de negócio
	ontologia
	recortes da ontologia
	glossário de negócio
	regras de negócio
	modelo de caso de uso de negócio
	modelo de objetos de negócio
	documento de papel e responsabilidade

Quadro 5 - (Fase I) Iniciação
Nota: Elaboração própria.

(i) Definir domínio

Objetivo: nesta atividade é identificado o domínio de negócio contendo as principais definições e objetivo.

Artefato: documento de visão do negócio.

***Stakeholders*:** cliente gerencial e analista (responsável).

Considerações: A definição do domínio satisfaz às necessidades expressadas pelo cliente para o desenvolvimento de um sistema específico. Nesta atividade o cliente e o analista discutem o escopo do negócio para a produção da aplicação em questão. A partir deste

momento são designados um *designer* de negócios e um engenheiro de ontologias.

(ii) Selecionar ontologia

Objetivo: nesta atividade é selecionada uma ontologia para suprir o domínio solicitado pelo cliente.

Artefato: ontologia.

***Stakeholders*:** analista (responsável), *designer* de negócios e engenheiro de ontologias.

Considerações: Nesta atividade são realizadas pesquisas e análises dentre uma gama de ontologias, a fim de eleger a que melhor se adapte ao escopo pretendido e possa, assim, servir de alicerce para todo o processo. Caso não exista uma ontologia para o domínio empreendido, deverá ser executada a sub-atividade ((*ii*.*a*) *Criar uma ontologia*).

A escolha da ontologia não é um processo trivial: é pertinente uma série de discussões para identificação de quais características são necessárias para a triagem de ontologias. Algumas perguntas são válidas, a saber: “Como identificar uma ontologia correta e com qualidade para um determinado domínio de negócio?” Uma sugestão para a solução deste problema é requerer ontologias bem referenciadas e validadas por um especialista de domínio no âmbito da organização, por exemplo. Outra abordagem é estabelecer questões de competência relacionadas ao domínio do negócio e verificar que ontologias respondem a essas questões.

(ii.a) Criar uma ontologia

Objetivo: nesta atividade é criada uma ontologia para suprir o domínio solicitado pelo cliente.

Artefato: ontologia.

***Stakeholders*:** analista (responsável), *designer* de negócios e engenheiro de ontologias.

Considerações: Para a produção de uma ontologia pode-se verificar o Capítulo 3. Esta é uma atividade alternativa caso não seja encontrada uma ontologia que atenda ao domínio do negócio.

As seguintes perguntas deverão ser questionadas nesta fase: “É viável a construção de uma ontologia para subsidiar um modelo de negócio?”, “Quais os ganhos na construção de uma ontologia?”, “A ontologia pode ser reutilizada em outros projetos de software?”.

Deve-se levar em consideração que esta é uma atividade alternativa e a sua execução ocasiona alguns impactos tais como: riscos de recursos – necessidade de *stakeholders* especializados na criação de ontologias; tempo – no sentido desta atividade ser um passo a mais neste processo; custos – acarretados pela alocação de recursos e tempo, com a finalidade da produção da ontologia; entre outros.

Apesar dos impactos a esta atividade, existe uma série de trabalhos que inserem a utilização de ontologias no processo de software atualmente e/ ou abordagens dirigidas à arquitetura baseadas em ontologias que discutem benefícios da inserção da mesma no processo de desenvolvimento de software. Assim, por exemplo, é possível vislumbrar uma proposta de integração de ontologias ao Processo Unificado, permitindo alguns benefícios, entre eles, a rastreabilidade ontológica entre os artefatos (NOLL, 2007). Naquela proposta é realizada a inserção de uma nova funcionalidade na ferramenta ARGOUML (disponível em http://argouml.tigris.org/) que transforma o modelo de análise em uma ontologia. Ainda, outras propostas são sugeridas visando à integração de ontologias com o processo de

desenvolvimento de software desenvolvidos na comunidade científica e na indústria.

Após a criação da ontologia, segue-se o fluxo normal do processo ***MDAONTO*** na íntegra. Nas atividades ((*iii*) *Delimitar e validar ontologia*) e ((*iv*) *Elicitar e validar negócio*) foca-se o refinamento e adequação da ontologia no processo proposto.

(iii) Delimitar e validar ontologia

Objetivo: nesta atividade são realizadas a análise e seleção de partes necessárias da ontologia para suprir o domínio de negócio. Assim, o mesmo pode ser validado precocemente, com apoio dos modelos obtidos a partir de ontologias de domínio.

Artefato: ontologia, recortes da ontologia que representam as visões dos *stakeholders* (visão do cliente gerencial, visão do cliente operacional e visão do analista).

***Stakeholders*:** clientes (gerencial e operacional) e analista (responsável).

Considerações: Nesta atividade o analista extrai recortes da ontologia com intuito de facilitar a compreensão dos diversos *stakeholders* (vide seção 5.1). Os elementos da ontologia possibilitam a geração de modelos de negócio e um conjunto inicial de regras de negócios.

(iv) Elicitar negócio

Objetivo: nesta atividade são elicitados novos elementos do sistema a fim de complementar o domínio de negócio.

Artefato: ontologia, glossário do negócio, regras de negócio, modelo de caso de usos de negócio, modelo de objetos de negócio.

***Stakeholders*:** clientes (gerencial e operacional), analista (responsável), *designer* de negócio, analista do processo de negócios e engenheiro de ontologias.

Considerações: nesta atividade os recortes da ontologia são utilizados para a compreensão de todos os *stakeholders* envolvidos. Assim, ocorre o levantamento de novas informações do negócio proposto através de reuniões, *brainstorms,* documentos da empresa, sistemas similares, entre outros. Ao final desta atividade, uma especificação do domínio de negócio com suas regras é detalhada, sendo esta composta pelos recortes da ontologia e dos demais artefatos de negócio. O conjunto destes artefatos corresponde ao modelo CIM.

Muitos artefatos no modelo CIM advêm da utilização da ontologia. Assim, por exemplo, o glossário de negócio pode ser desenvolvido a partir do vocabulário da ontologia e esse vocabulário pode auxiliar no processo de elicitação de novas terminologias e, ainda, auxiliar o entendimento consensual dos conceitos do negócio.

Outro beneficio trazido nesta atividade com a introdução da ontologia é o entendimento das regras de negócios da ontologia pelo engenheiro de ontologias e estas são apresentadas ao cliente em uma linguagem em alto nível, para fim de validação. A partir desta discussão, novas regras poderão aparecer de modo mais natural.

5.2.2 (FASE II) Fase de Desenvolvimento PIM

A FASE DE DESENVOLVIMENTO PIM (conforme ilustrado no Quadro 6) apresenta análise do sistema que será concebido. Ao final desta fase é obtido um modelo formal (legível pelas ferramentas MDA) do sistema e independente de plataforma. Esta fase apresenta duas atividades, a saber: (*viii*) *Especificar análise PIM* e (*ix*) *Validar*.

***Nome da Fase*:** Fase de desenvolvimento PM	
Aderência ao padrão MDA: PIM Ao final desta fase, o modelo de análise estará apto para servir de insumo para a fase seguinte, onde será transformado em modelo de projeto por uma ferramenta MDA.	
Entrada	**Saída**
documento de visão de negócio	modelo de análise (PIM)
Ontologia	
recortes da ontologia	
glossário de negócio	
regras de negócio	
modelo de caso de uso de negócio	
modelo de objetos de negócio	
Ferramentas	

Quadro 6 - (Fase II) Elaboração
Nota: Elaboração própria.

(v) Especificar PIM

Objetivo: nesta atividade os elementos que compõe o modelo de negócio selecionado a partir das atividades ((*iii*) *Delimitar e validar ontologia*) e ((*iv*) *Elicitar e validar negócio*) são transformados em modelo independente de plataforma. Ou seja, o modelo de negócio é transformado em modelo de análise. Ao final desta atividade uma especificação de análise desejada deve estar completa, a qual é formada por requisitos funcionais e não funcionais. Recomenda-se a utilização da linguagem inteligível pela ferramenta. Contudo no exemplo proposto no próximo capítulo, será utilizado a *Linguagem de Modelagem Unificada* (UML) (OMG, 2004) e da *Object Constraint Language* (OCL) (WARMER; KLEPPE, 2003; OMG, 2006).

Artefato: modelo de análise (PIM).

***Stakeholders*:** analista de sistemas e especificador de requisitos.

Considerações: os requisitos funcionais utilizados nesta atividade são provenientes do entendimento dos modelos de negócio (ontologia, recortes da ontologia, glossário do negócio, regras de negócio, modelo

de caso de usos de negócio, modelo de objetos de negócio) para a produção do modelo de análise (PIM).

Neste contexto, observa-se que as ontologias são mais expressivas, então, no mapeamento ontologia para UML e OCL é possível a perda de semântica. A idéia de inserir OCL nesta atividade é uma tentativa de armazenar os axiomas (NOLL, 2007). Contudo, é descrito que na transformação de uma ontologia para um modelo de análise, alguns elementos da ontologia são utilizados somente para melhor compreensão do negócio proposto, uma vez que as representações orientadas a objetos apenas capturam a taxonomia dos conceitos e seus relacionamentos e não as regras lógicas que uma ontologia define.

Além disso, a utilização de uma ontologia facilita o processo de elicitação de requisitos, permitindo que o tempo gasto possa ser reduzido, já que partes dos requisitos e das regras de negócio estão contempladas na ontologia (COTA; MENEZES; FALBO, 2004).

Esta atividade tem objetivo de mapear um conjunto de funcionalidades do modelo de negócio para a análise. Cada mapeamento corresponde a um incremento que será entregue ao cliente.

(vi) Validar

Objetivo: nesta atividade é realizada a revisão formal do modelo de análise, ou seja, o revisor de requisitos confronta os modelos referentes ao negócio com o modelo de análise, verificando se existem inconsistências.

Artefato: modelo de análise (PIM).

***Stakeholders*:** revisor de requisitos e cliente operacional.

Considerações: Caso seja encontrada alguma inconsistência durante a validação, o fluxo é desviado para a atividade ((*viii*) *Especificar análise PIM*).

Nesta atividade o cliente pode ajudar a validação do escopo requerido do sistema que será elaborado, contendo as principais funcionalidades do mesmo.

5.2.3 (FASE III) Fase de Desenvolvimento PSM e de Código

Esta fase é dependente da ferramenta MDA selecionada e corresponde à fase de projeto. Essa é composta por duas atividades, como segue: (*x*) *Transformar em modelo PSM* e (*xi*) *Gerar código e testar*. O Quadro 7 ilustra esta fase.

***Nome da Fase*:** Fase de desenvolvimento PSM e de CÓDIGO	
Aderência ao padrão MDA: PSM e Código Durante as atividades desta fase, o PIM produzido na fase anterior é transformado no PSM e em código em uma linguagem específica. A maior parte destas atividades é executada por ferramentas de MDA.	
Entrada	**Saída**
modelo de análise (PIM)	modelo de projeto (PSM)
	código em uma linguagem específica
	script de Teste

Quadro 7 - (Fase III) Fase de desenvolvimento PSM e de código
Nota: Elaboração própria.

(vii) Transformar em modelo PSM

Objetivo: nesta atividade são transformados os artefatos da fase desenvolvimento PIM em um modelo de uma plataforma particular a partir da ferramenta MDA selecionada na atividade ((*vii*) Definir ferramentas).

Artefato: modelo de projeto (PSM).

***Stakeholders*:** arquiteto de software e implementador.

Considerações: Nesta atividade, os passos para a transformação de PIM para PSM dependem da ferramenta MDA selecionada. Assim, os *stakeholders* envolvidos nesta atividade, devem seguir as diretrizes definidas a partir da ferramenta selecionada (ASADI; RAVAKHAH; RAMSIN, 2008).

(viii) Gerar código e testar

Objetivo: nesta atividade é gerado código em uma linguagem específica com auxílio de uma ferramenta MDA, a partir da atividade ((*x*) Transformar em modelo PSM) e, ainda, é realizado teste do sistema.

Artefato: código em uma linguagem específica e *scripts* de teste.

***Stakeholders*:** implementador e testador.

Considerações: Nesta atividade, os passos para a transformação de PSM para código dependem da ferramenta MDA. Assim, os *stakeholders* envolvidos devem seguir as diretrizes definidas a partir da ferramenta para realizar a produção de código (ASADI; RAVAKHAH; RAMSIN, 2008).

O testador tem, dentre outras, a responsabilidade de identificar e definir cada s*cript* de teste. Este artefato é composto por orientações da realização de um teste. Os *scripts* de teste podem assumir a forma de instruções de texto documentadas e executadas manualmente ou de instruções que podem ser lidas pelo computador para ativar a execução automática do teste (RUP, 2001).

Testes automáticos por ferramentas MDA e CASE's estão fora do escopo deste trabalho.

5.2.4 (FASE IV) Fase de Distribuição

Este fase corresponde à entrega do sistema no ambiente do usuário e observam-se as seguintes atividades, a saber: (*xii*) *Testar*, (*xiii*) *Documentar* e (*xiv*) *Implantar*. O Quadro 8 ilustra esta fase.

***Nome da Fase*:** Fase de distribuição	
Aderência ao padrão MDA: O objetivo desta fase é a entrega do sistema desenvolvido ao usuário final e fornecer, posterior manutenção.	
Entrada	**Saída**
modelo de projeto (PSM)	aplicação desenvolvida (ou conjunto de funcionalidades)
código em uma linguagem específica	manual do usuário
script de teste	

Quadro 8 – (Fase IV) Fase distribuição e manutenção
Nota: Elaboração própria.

(ix) Realizar teste de aceite

Objetivo: nesta atividade é realizado o teste do sistema final e de aceitação.

Artefato: aplicação desenvolvida (ou conjunto de funcionalidades).

***Stakeholders*:** analista (responsável) e cliente operacional.

Considerações: nesta atividade o teste é realizado pelo cliente operacional, com o apoio do analista, com o intuito de aceitação do escopo do sistema que está sendo entregue. Caso haja inconsistências e/ ou erros, estas serão apontadas e o processo retornará para a fase especificar PIM com o objetivo de correção dos defeitos encontrados. É interessante ressaltar que erros nesta atividade serão mínimos, uma vez que já houveram validações realizadas pelos clientes (gerencial e operacional).

(x) Documentar

Objetivo: nesta atividade é desenvolvida a documentação final do software.

Artefato: manual do usuário.

***Stakeholders*:** analista (responsável) e redator técnico.

Considerações: Nesta atividade, o *stakeholder* responsável gera a documentação final do escopo do sistema a ser entregue. Neste contexto, o mesmo consulta a ontologia com a finalidade de geração destes artefatos. Nesse sentido, esta atividade pode ser considerada simples e perpassa por algumas atividades anteriores, uma vez que parte da documentação já está pronta e coesa, bastando somente colocá-la em linguagem de fácil entendimento para os usuários do sistema.

(xi) Implantar

Objetivo: nesta atividade o sistema recém desenvolvido é instalado em ambiente do usuário.

***Stakeholders*:** analista (responsável) e implementador.

Considerações: Durante a realização desta atividade, o sistema é instalado no ambiente de usuário, e algumas as tarefas podem ser necessárias, tal como, migração de dados (ASADI; RAVAKHAH; RAMSIN, 2008).

5.2.5 Processos de Apoio do MDAONTO

Na fase Iniciação, dá-se início as atividades de apoio, como se segue: (*a*) Definir equipe, (*b*) Definir plano geral e (*c*) Definir ferramentas.

(a) Definir equipe

Objetivo: nesta atividade é definida a equipe responsável pelo projeto conforme os papeis e responsabilidade descritos no ANEXO B.

Artefato: documento de papel e responsabilidade.

***Stakeholders*:** analista (responsável).

Considerações: A definição dos *stakeholders* é uma sugestão para a aplicação do processo proposto. Contudo, é possível outra abordagem em conformidade à estrutura organizacional.

(b) Definir plano geral

Objetivo: nesta atividade são discriminadas informações sobre o projeto e uma análise de risco e estimativa de esforço é realizada.

Artefato: plano de desenvolvimento de software e lista de riscos.

***Stakeholders*:** analista (responsável) e analista de requisitos.

Considerações: os documentos gerados nesta fase serão revisados no decorrer de todo o processo.

O plano de desenvolvimento de software é um artefato composto que abrange as informações necessárias ao gerenciamento do projeto. Este contém artefatos separados, desenvolvidos durante a fase *Iniciação* e é mantido durante todo o projeto (RUP, 2001).

A lista de riscos é de suma importância. Os riscos devem ser identificados para que possam ser previstos e diminuídos. Os riscos

podem ser caracterizados em quatro grupos, a saber: (*a*) *riscos de recursos*: que estão associados à organização, aos fundos (investimento), ao pessoal e ao tempo; (*b*) *riscos de negócios*: que estão associados à própria característica do negócio e a concorrência com outras organizações (se, por exemplo, um concorrente obtiver a liderança do mercado do mesmo produto); (*c*) *riscos técnicos*: que estão associados aos riscos de escopo, riscos de tecnologia, riscos de dependência externa; e, por fim, os (*d*) *riscos de programação*: que estão associados à complexidade, restrições de tempo real, restrições de armazenamento, experiência, disponibilidade de ferramentas apropriadas e pressão de programação (RUP, 2001).

Os riscos descritos em (*c*) e (*d*) podem ser diminuídos neste processo da seguinte forma: os *riscos técnicos relacionados ao escopo* são minimizados nas fases ((*ii*) *Selecionar ontologia*), ((*iii*) *Delimitar e validar ontologia)* e ((*iv*) *Elicitar e validar negócio),* onde o escopo é analisado e definido antecipadamente de forma consensual com ajuda do cliente. Os *riscos técnicos relacionados à tecnologia* são diminuídos no sentido que a tecnologia a ser utilizada deve ser antecipadamente conhecida por uma ferramenta MDA, para, assim, realizar as transformações entre modelos (PIM-PSM e PSM-Código) não havendo surpresas no final do projeto. Os *riscos técnicos relacionados à dependência externa* são, também, minimizados com a utilização da ontologia, pois não há dependência de produtos e/ ou componentes externos, uma vez que a ontologia contém subsídios para suprir as necessidades relacionadas ao domínio do negócio empreendido. Os *riscos relacionados à programação* são minimizados com a utilização de programação em alto-nível e a utilização do padrão MDA que atenua os seguintes problemas: produtividade, interoperabilidade, portabilidade, integração, manutenção e documentação consistente.

(c) Definir ferramentas

Objetivo: nesta atividade são definidas uma ou mais ferramentas para subsidiar as atividades do processo de software.

Artefato:

***Stakeholders*:** analista (responsável) e cliente gerencial.

Considerações: Para esta atividade é indicado o ***PROTÓTIPO MDAONTO***[2] que apresenta uma abordagem dirigida por (meta) modelos com foco em regras de negócio. Contudo outras ferramentas no padrão MDA podem ser utilizadas. Vale ressaltar, que nessa atividade o analista e o cliente gerencial em comum acordo determinam a tecnologia a ser utilizada para produção do sistema. Contudo, há limitação das tecnologias que estão disponíveis em uma ferramenta MDA. Assim, a transformação só é possível para as linguagens já mapeadas pela ferramenta.

Ferramenta *Computer-Aided Software Engineering* (CASE) podem apoiar a construção de modelos. Ainda, as ferramentas que compreendem ontologias podem ser necessárias neste processo de desenvolvimento de software, tal como, o *Protégé* (disponível em http://protege.stanford.edu/).

5.3 RASTREABILIDADE

Esta seção apresenta a rastreabilidade entre os artefatos deste processo conforme (IEEE std 610.12, 1990 apud MPS.BR, 2007, p. 16-17).

> Rastreabilidade é o grau em que o relacionamento pode ser estabelecido entre dois ou mais produtos de desenvolvimento de software, especialmente produtos que tenham uma relação de predecessor-sucessor ou de mestre-subordinado com outro; por exemplo, o grau em que requisitos e projeto (design) de um determinado componente de software combinam.

[2] Resultado do projeto "Solução de Transformação Bidirecional de Modelos UML e código baseada na Metodologia MDA" em desenvolvimento, financiado pela Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB)

A Figura 15 ilustra a rastreabilidade entre a ontologia e as visões. Esta é apresentada de forma horizontal, uma vez que existe uma dependência entre esses artefatos em um mesmo nível. Um dos benefícios dessa rastreabilidade é o mapeamento da ontologia em visões, onde cada uma dessas visões expressa as necessidades específicas dos *stakeholders*. Atenuando o problema clássico de comunicação, as visões da ontologia têm um papel fundamental de “ponte” neste panorama discuto.

A rastreabilidade vertical estabelece uma rastreabilidade bidirecional desde um requisito fonte, passando pelos seus requisitos de mais baixo nível, até o produto. Esse mecanismo de rastreabilidade vertical é essencial para a realização da análise de impacto de mudanças de requisitos (MPS.BR, 2007). No contexto do processo ***MDAONTO***, o documento de visão do negócio, que expressa a necessidade do cliente, auxilia na seleção da ontologia e, posteriormente, dá origem aos modelos de negócio, modelo PIM, modelo PSM até o código. Ou seja, a partir deste documento é possível realizar a rastreabilidade vertical. Deste modo, é possível fazer a análise de impacto, custo, entre outros, causados pelas mudanças que possam vir a ter no projeto.

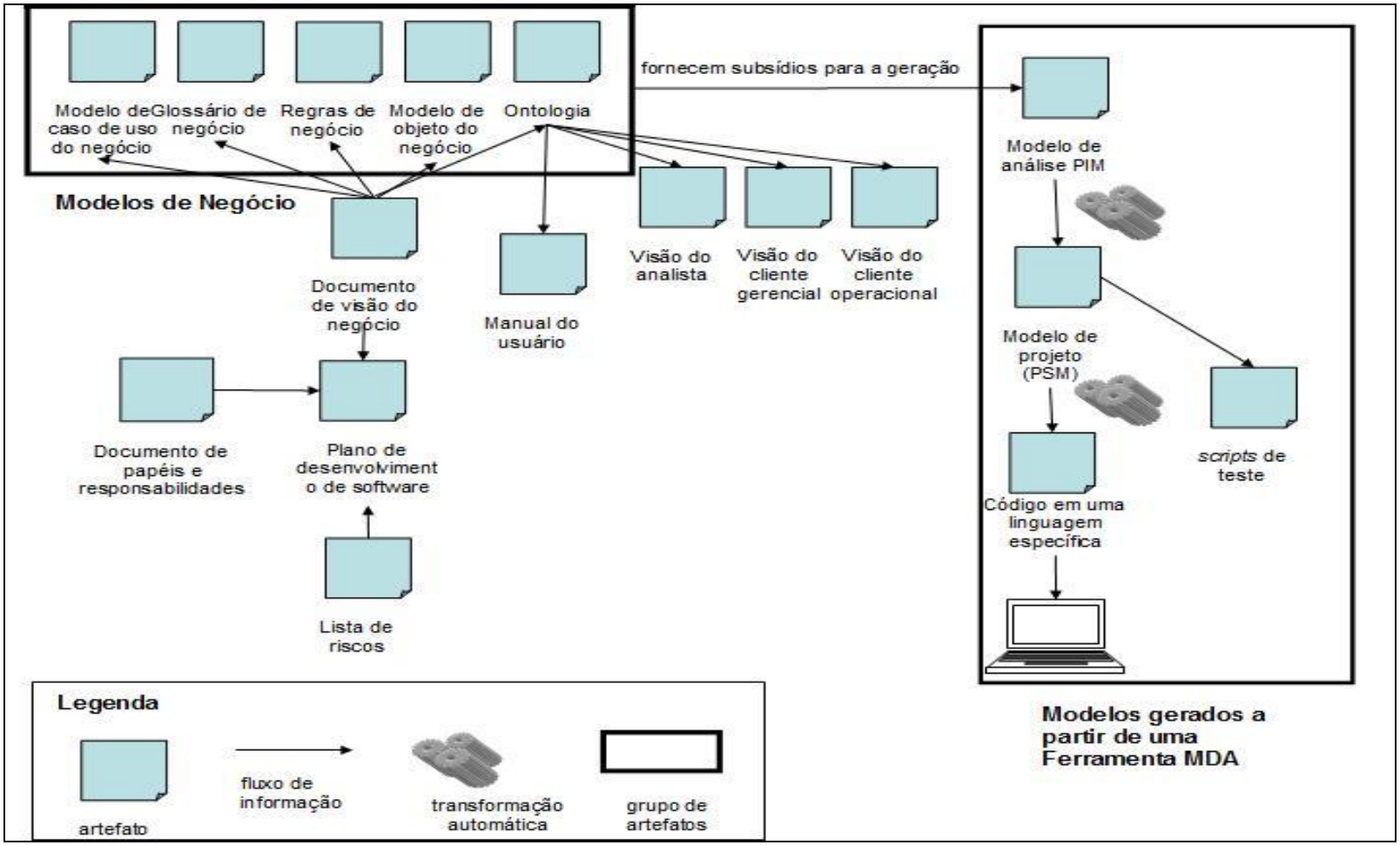


Figura 15 - Rastreabilidade
Nota: Elaboração própria.

5.4 AVALIAÇÃO COMPARATIVA

Esta seção visa oferecer uma análise comparativa dentre as abordagens dirigidas a modelos descritas no Capítulo 4 e o ***MDAONTO***. Permitindo, assim, uma análise dos pontos fortes e fracos de cada processo visto neste trabalho.

Em (ASADI; RAMSIN, 2008) é realizado um estudo visando à avaliação de metodologias de software dirigidas a modelos, no qual os autores utilizam um método similar ao proposto em *Feature Analysis* (KITCHENHAM; LINKMAN; LAW, 1997). A abordagem *Feature Analysis* apresenta um método para avaliar métodos e ferramentas, e foi desenvolvida em 1996 em um projeto entre o meio acadêmico e a indústria. Este método provê duas formas de avaliar qualquer produto em termos de resultados: forma simples e a forma de escala. Na forma simples, a abordagem apresenta uma lista com respostas “sim ou não”, a fim de responder uma determinada característica. Na forma em escala, é apresentada uma lista contendo respostas com valores de 1 até 5, a depender do grau de conformidade do produto com uma determinada característica.

Assim, em (ASADI; RAMSIN, 2008) é descrito que a partir da seleção de uma metodologia baseada em MDA são avaliados um Critério de Avaliação ou *Evaluation Criteria* (EC). Esses critérios são selecionados através de duas formas, a saber: escala e narrativo. Na escala é apresentado o grau de presença do critério de avaliação na metodologia e, o tipo narrativo apresenta o grau de aplicação do critério de avaliação de modo narrativo. O autor descreve que a concepção dos critérios foi obtida através de inúmeros estudos e, a partir destes, foram definidos meta-critérios com o propósito de avaliação de metodologias, a saber:

- ✓ Existência de critérios relacionados com a ferramenta – este critério avalia se a metodologia descreve como será a realização de uma determinada tarefa ou se deixa essa tarefa a cargo de uma ferramenta; este critério é representado pelo Quadro 9.

- ✓ Existência de critérios relacionados com o padrão MDA – utilizados para avaliar o critério relacionado com o padrão MDA. São aplicados às metodologias no contexto MDA; este critério é representado pelo Quadro 10.

- ✓ Existência de critérios gerais – utilizados para avaliar a conformidade do critério a partir de aspectos gerais, os quais podem ser aplicados a todos os tipos de metodologia; este critério é representado pelo Quadro 11.

Segundo o autor desta proposta, os critérios de avaliação de um processo de desenvolvimento de software devem validar os meta-critérios. Para que os critérios de avaliação sejam válidos, estes devem apresentar quatro características: (*i*) ser abrangente, de modo que seja aplicável em todas as metodologias MDA; (*ii*) ser preciso, a fim de discernir as semelhanças e diferenças entre metodologias baseadas em MDA; (*iii*) ser completo, a fim de cobrir todas as características importantes de metodologias baseadas no padrão MDA; e (*iv*) ser equilibrado, a fim de avaliar de forma igual os tipos de recursos em uma metodologia: técnico, de gestão e utilização.

O Quadro 9 apresenta os critérios relacionados ao meta-critério “Existência de critérios relacionados com a ferramenta”.

Critério	Tipo de Critério	Descrição dos Níveis
Transformação PIM para PSM	Narrativa	**Envolve**: A metodologia explicitamente participa da atividade e fornece técnicas precisas e/ ou orientações. **Transfere**: A atividade é transferida para as ferramentas e a metodologia não descreve os passos que devem ser realizados pela ferramenta.
Transformação de PSM para código	Narrativa	
Teste automático	Narrativa	
Rastreabilidade entre os modelos	Narrativa	

Quadro 9 – Meta-critério “Existência de critérios relacionados com a ferramenta”
Fonte: Adaptado de Asadi e Ramsin (2008).

O Quadro 10 apresenta os critérios relacionados ao meta-critério “Existência de critérios relacionados com o padrão MDA”.

Critério	Tipo de Critério	Descrição dos Níveis

Seleção de ferramenta / implementação	Escala	**A**: a metodologia não fornece uma ferramenta específica e não existem orientações de como selecionar uma ferramenta adequada. **B**: a metodologia não fornece um conjunto de ferramentas. Ela, apenas, orienta, de maneira geral, como selecionar uma ferramenta adequada. **C**: a metodologia fornece um conjunto completo de ferramentas, ou fornece orientações precisas para a seleção ferramentas apropriadas.
Criação CIM	Escala	**A**: a produção do modelo não é abordada pela metodologia. **B**: a metodologia fornece orientações gerais para a criação do modelo; os passos de criação não são determinados com precisão. **C**: a metodologia explicitamente descreve passos e técnicas para criar o modelo.
Criação PIM	Escala	
Criação PSM	Escala	
Utilização de perfil UML	Narrativa	

Quadro 10 - Meta-critério “Existência de critérios relacionados com o padrão MDA”
Fonte: Extraído de Asadi e Ramsin (2008).

O Quadro 11 apresenta os critérios relacionados ao meta-critério “Existência de critérios gerais”.

Critério			**Tipo de Critério**	**Descrição dos Níveis**
Corbetura	Ciclo de vida genérico	Engenharia de requisitos	Escala	**A**: a metodologia não fornece cobertura para a fase. **B**: a metodologia fornece orientações gerais para a fase. **C**: a metodologia fornece diretrizes detalhadas para a fase.
		Análise	Escala	
		Design	Escala	
		Implementação	Escala	
		Teste	Escala	
		Implantação	Escala	
		Manutenção	Escala	
	Atividades de apoio	Gerência de projetos	Escala	**A**: a metodologia não fornece cobertura para a atividade. **B**: a metodologia fornece orientações gerais para a atividade. **C**: a metodologia fornece diretrizes detalhadas para a atividade.
		Garantia da qualidade	Escala	

		Gestão de riscos	Escala	
Análise de domínio do problema			Escala	**A**: análise de domínio do problema não é abordada. **B**: análise de domínio do problema é implícita. **C**: análise de domínio do problema é explicitamente abordada pela metodologia e rastreabilidades são mantidas.
Reusabilidade			Escala	**A**: a tarefa é transferida para os *stakeholders*, a metodologia não fornece técnicas / orientações. **B**: a metodologia fornece técnicas explícitas para criação de artefatos reutilizáveis. **C**: além do B, o método descreve técnicas para registrar as características sintáticas/ semânticas dos aspectos reutilizáveis para futura reutilização.
Adaptação			Escala	**A**: não existem técnicas para a adaptação da metodologia. **B**: a metodologia fornece notações estendidas. **C**: além do B, o método fornece técnicas explícitas para adaptação do processo e / ou linguagem de modelagem.
Completude da Definição das Fases			Escala	**A**: algumas fases da metodologia não estão especificadas completamente. **B**: todas as fases são completamente especificadas (em volume), mas algumas fases não são detalhadas. **C**: todas as fases são completamente especificadas e detalhadas.
Tipo de metodologia			Escala	**Extensão**: A metodologia é o resultado de uma extensão de uma metodologia existente baseada no padrão MDA. **Baseada no padrão MDA (Verdadeira)**: A metodologia foi criada a partir do zero, baseando-se no padrão MDA.
Escopo da aplicação			Narrativa	

Quadro 11 – Meta-critério “Existência de critérios gerais”
Fonte: Adaptado de Asadi e Ramsin (2008).

Foram analisadas em (ASADI; RAMSIN, 2008) as seguintes metodologias: MASTER, MODA-TEL, ODAC, entre outras. Este trabalho acrescenta, também, a avaliação do MDA-SDLC, pois essa metodologia foi utilizada como base para a criação do processo proposto neste trabalho e, também, por demonstrar pontos importantes com relação a todas as outras metodologias encontradas. Assim, os Quadros 12, 13 e 14 apresentam os resultados aplicados.

Critério	*MDAONTO*	MDA-SDLC	MASTER	MODA-TEL	ODAC
Transformação PIM para PSM	envolve	envolve	envolve	envolve	transfere
Transformação de PSM para código	envolve	envolve	envolve	envolve	transfere
Teste automático	transfere	transfere	envolve	transfere	transfere
Rastreabilidade entre os modelos	envolve	transfere	transfere	envolve	transfere

Quadro 12 – Resultado aplicado ao meta-critério "Existência de critérios relacionados com a ferramenta"
Nota: Elaboração própria.

Com relação aos critérios relacionados com a ferramenta, é possível observar que as metodologias MDA-SDLC e ***MDAONTO*** se preocupam em descrever este conjunto de características com mais detalhes, mesmo que estas sejam implementadas pelas ferramentas MDA, não sendo observado este grau de completude nas outras metodologias avaliadas, conforme descrito no Quadro 12. Na ***MDAONTO*** existe a realização de testes, embora esses sejam feitos de modo não automatizado.

O forte da ***MDAONTO*** é a transformação CIM-PIM, a qual, não está contemplada nos critérios relacionados com o meta-critério analisado.

Critério	*MDAONTO*	MDA-SDLC	MASTER	MODA-TEL	ODAC
Seleção de ferramenta / Implementação	C	B	C	B	A
Criação CIM	C	B	B	A	A
Criação PIM	C	C	C	B	C
Criação PSM	C	C	C	B	B
Utilização de perfil UML	Usado no PIM para representar os requisitos	Usado no PIM para representar os requisitos	Usado no PIM com gerenciamento de informação	Usado para representar os requisitos	Usado para descrever o desenvolvimento dos passos

Quadro 13 – Resultado relacionado ao Meta-critério "Existência de critérios relacionados com o padrão MDA"

Fonte: Elaboração própria.

Com relação aos critérios relacionados com o padrão MDA, dentre os critérios avaliados, é importante destacar a criação do CIM, como pode ser observado no Quadro 13, pois esse modelo corresponde à base principal para a geração dos outros modelos. Neste contexto, é possível observar que as metodologias avaliadas, não abordam de forma satisfatória a produção deste modelo, com exceção do ***MDAONTO***, pois este, se beneficia da utilização de ontologias para a criação do modelo CIM. Assim, destaca-se no processo ***MDAONTO***, alguns benefícios, como a obtenção de conhecimento de forma mais produtiva, compartilhamento de conhecimento consensual, disseminação do conhecimento entre os membros da equipe, dentre outros.

Critério			**MADONTO**	**MDA-SDLC**	**MASTER**	**MODA-TEL**	**ODAC**
Corbetura	Ciclo de vida genérico	Engenharia de requisitos	C	C	C	B	A
		Análise	C	C	C	B	C
		Design	C	C	C	B	C
		Implementação	C	C	B	B	B
		Teste	C	C	C	B	A
		Implantação	C	C	B	B	A
		Manutenção	C	C	A	B	A
	Atividades de apoio	Gerência de projetos	C	C	C	B	A
		Garantia da qualidade	B	A	B	B	A
		Gestão de riscos	C	B	A	A	A
Análise do domínio de problema			C	B	B	A	A

Reusabilidade	B	B	B	B	A
Adaptação	A	A	B	B	A
Completude da Definição das Fases	B	B	C	B	B
Tipo de metodologia	Extensão	Baseada no padrão MDA	Baseada no padrão MDA	Baseada no padrão MDA	Extensão
Escopo da aplicação	Sistemas de informação	Sistemas de informação	Sistemas de informação	Aplicações distribuidas	Sistemas orientados a agente

Quadro 14 – Resultado relacionado ao Meta-critério “Existência de critérios gerais”
Fonte: adaptado de Asadi e Ramsin (2008).

Com relação aos critérios relacionados de modo geral (Quando 14), é possível observar que a ***MDAONTO*** e a MDA-SDLC abordam a maioria dos critérios de maneira completa. Apesar da semelhança entre as duas abordagens, é importante ressaltar que, em alguns critérios, a abordagem realizada pela ***MDAONTO*** pode ser considerada mais completa, pois esta utiliza os benefícios da ontologia. Como por exemplo, a gestão de riscos, que o ***MDAONTO*** aborda definindo as diretrizes de como fazer a avaliação dos riscos a partir de recortes da ontologia.

O Quadro 15 apresenta uma comparação entre o MDA-SDLC e o ***MDAONTO***. A partir deste é possível identificar algumas divergências:

- ✓ Na primeira fase do ***MDAONTO*** são apresentadas atividades para subsidiar a seleção de uma ontologia e a extração de conhecimento a partir da mesma.

- ✓ Na segunda fase do ***MDAONTO*** – Desenvolvimento do PIM - observa-se a não existência de atividades de projeto arquitetural e produzir projeto PIM. Contudo, é importante sublinhar que a atividade produzir projeto PIM é realizada no ***MDAONTO*** através da atividade (*x*) Transformar em modelo PSM. Neste sentido, o ***MDAONTO*** segue de forma precisa as diretrizes determinadas no padrão MDA.

- ✓ Na terceira fase do ***MDAONTO***, há uma junção de duas atividades presentes no MDA-SDLC (Geração de código e Testando). Neste sentido, os *stakeholders* que realizam a geração de código são responsáveis pela realização do teste.

- ✓ Na quarta fase do ***MDAONTO***, há junção de duas fases do MDA-SDLC. Contudo, o conteúdo das fases não é alterado. Vale ressaltar que no ***MDAONTO*** a atividade destinada à documentação é beneficiada pela utilização de ontologias, tornado esta atividade mais fácil de ser realizada.

Fase	***MDAONTO***	***Fase MDA-SDLC***	***MDA-SDLC (Tradução)***
Iniciação	(*i*) Definir domínio	Fase Iniciação do Projeto	Definição CIM
	(*ii*) Selecionar ontologia		X
	(*iii*) Delimitar e validar ontologia		Especificação de requisitos
	(*iv*) Elicitar e validar negócio		
	X		Obter financiamento e suporte
	(*a*) Definir equipe		Iniciar a equipe
	(*b*) Definir plano geral		Definir plano geral
	(*c*) Definir ferramentas		Seleção de ferramenta
Desenvolvimento do PIM	(*viii*) Especificar PIM	Fase de Desenvolvimento PIM	Produzir análise PIM
	X		Projeto arquitetural
			Produzir projeto PIM
	(*ix*) Validar		Verificação / validação
	X		Generalização
Desenvolvimento PSM e de código	(*x*) Transformar em modelo PSM	Fase PSM & Desenvolvimento de Código	Transformar PIM em PSM
	(*xi*) Gerar código e testar		Geração de código
			Testando
Distribuição e manutenção	(*xii*) Testar	Fase Distribuição	Teste final do sistema
	(*xiii*) Documentar		Finalizando a documentação do sistema e de usuário
	(*xiv*) Implantar		Transição do sistema ao ambiente de usuário
	X	Fase Manutenção	X

Quadro 15 - Comparação entre processos
Fonte: Elaboração própria.

5.5 CONCLUSÃO

Neste capítulo foi analisada a geração de “visões” de *stakeholders*, as quais apresentam diferentes graus de complexidade (detalhamento) de uma ontologia. Estas permitem a validação em diferentes níveis de detalhes e abstração, com foco em uma perspectiva particular.

O processo ***MDAONTO*** é um processo de desenvolvimento de software, o qual visa à utilização de uma ontologia associada ao padrão MDA. Para a construção desse processo observou o estudo e a análise de diversos processos de software, até em paradigmas diferentes. Contudo, dentre os trabalhos citados no capítulo anterior, o que mais influenciou na produção do processo proposto foi a MDA-SDLC, a qual foi publicado no mesmo ano de concepção deste trabalho. Um ponto que distingue a proposta apresentada neste trabalho é aquisição de conhecimento a partir de modelos de negócios baseados em ontologias e as atividades para subsidiá-las. Outros diferenciais são as definições dos *stakeholders*, a rastreabilidade dos artefatos, a inclusão de iterações, sugestão de artefatos, dentre outros.

Este processo utilizou os benefícios de uma ontologia para produção de artefatos durante o ciclo de desenvolvimento. A utilização de ontologias no processo de software visa, dentre outros objetivos, permitir a cooperação entre os grupos de engenharia de software e engenharia de conhecimento, favorecendo o reuso de conhecimento e, principalmente, promovendo o entendimento consensual entre todos os membros da equipe de um projeto de software.

É importante sublinhar que a abordagem apresentada se beneficia da existência de repositórios de ontologias existentes em áreas distintas de conhecimento, representando um ganho substancial nas fases iniciais do processo de

desenvolvimento de software, oferecendo uma visão do domínio a ser modelado para os *stakeholders*.

A utilização do ***MDAONTO*** permite que o software seja concebido com melhor produtividade, menor custo e maior qualidade. Melhor produtividade é desencadeada pelo desenvolvimento do sistema independentemente dos detalhes de especificação da plataforma (PIM) destino, possibilitando o desenvolvimento focado no modelo de negócio. E também, pela automatização das atividades. Menor custo em conseqüência da melhor produtividade e diminuição das taxas de riscos na produção do sistema. E por fim, maior qualidade devida à consistência da documentação e da rastreabilidade dos artefatos e, ainda, padronização do código.

Outro objetivo do processo ***MDAONTO*** é facilitar o desenvolvimento de novas aplicações em tecnologias diferentes com pouco esforço e, ainda, aproximar o cliente do processo de desenvolvimento de software.

A definição clara de papéis e responsabilidade e dos artefatos é um outro diferencial do processo proposto, uma vez que, nos processos analisado que tinham o objetivo de transformação entre modelos não havia esta acepção. Normalmente, referenciavam a outros processos, sem mesmo identificar em quais atividades atuaria um determinado *stakeholder*, com exceção, de MODA-TEL. Mesmo em MODA-TEL, esta definição é apresentada pela fase e não pela atividade, não oferecendo ao leitor um entendimento claro das responsabilidades de cada *stakeholder*.

A proposta deste trabalho ilustra, de modo claro, a rastreabilidade entre os artefatos. De modo geral, os processos analisados não entram em detalhes de como são compreendidos os artefatos, nem qual é sua importância e relevância para a concretização de uma tarefa específica.

A proposta apresentada mostra características ágeis, como entregas rápidas ao cliente com esforço pequeno, pois, uma vez selecionada a ontologia adequada, o processo para concepção do novo sistema é conduzido de forma simples. Assim, um

conjunto de funcionalidades é entregue ao cliente final, que as validam, e um novo incremento acontece até a produção do sistema específico.

6 APLICAÇÃO E AVALIAÇÃO DO MDAONTO

Este capítulo apresenta a aplicação do processo através de um exemplo ilustrativo e a avaliação do mesmo através de um estudo de opinião.

A partir de processo proposto, foi elaborado um exemplo ilustrativo, que apresenta uma aplicação do processo de software proposto ao desenvolvimento de um sistema particular a partir de uma ontologia (no domínio de multimídia). Nesse exemplo ilustrativo, a autora deste trabalho é quem realiza todas as atividades inerentes ao processo. Assim, a definição de papel e responsabilidade não foi analisada. Um outro aspecto que não foi analisado nesse estudo foi a quarta fase do processo proposto, a qual está relacionada à entrega do sistema.

Neste capítulo, ainda, é apresentado um estudo de opinião. Este visa julgar a compreensão, aplicação e confiabilidade do mesmo, como também, os benefícios da utilização da ontologia.

A seguir, são explanados os seguintes tópicos: seção 6.1 Caso Ilustrativo; seção 6.2 Estudo de Opinião; e por fim, na seção 6.3, é apresentada a conclusão deste capítulo.

6.1 CASO ILUSTRATIVO

Para este estudo ilustrativo, foi escolhido o domínio de multimídia como uma necessidade expressa (hipotética) pelo cliente e tendo como alvo desse estudo o domínio de vídeo sob demanda (VOD), descrito em Guizzardi (1997). É importante

ressaltar que nesse exemplo ilustrativo não será abordada de forma completa a ontologia, apenas serão realizadas referências a essa ontologia.

Para o entendimento do domínio de multimídia segue a seguinte definição desse mesmo autor (GUIZZARDI, 1997, p. 116-117):

> A multimídia constitui uma tecnologia interdisciplinar que orienta aplicações que visam atender às necessidades multi-sensoriais da natureza humana. Sistemas multimídia tornam isso possível devido às novas alternativas de integração, manipulação, armazenamento, transmissão, exibição e interação de diferentes formatos de mídia (texto, gráfico, animação, áudio e vídeo).

Em VOD, o telespectador é agente ativo, assim, este pode utilizar diferentes níveis de interação para a seleção de programação dos vídeos, deste modo, os telespectadores têm a funcionalidade de acesso a um servidor remoto de vídeos (por exemplo, a partir de um terminal presente em casa ou escritório). As conexões de vídeo são formadas sob demanda, através de uma rede de comunicação que provê a união entre o cliente e o servidor de vídeo de comunicação. A Figura 16 apresenta uma visão simplificada do cliente gerencial extraída de (GUIZZARDI, 1997), a qual ajuda no entendimento do domínio de VOD.

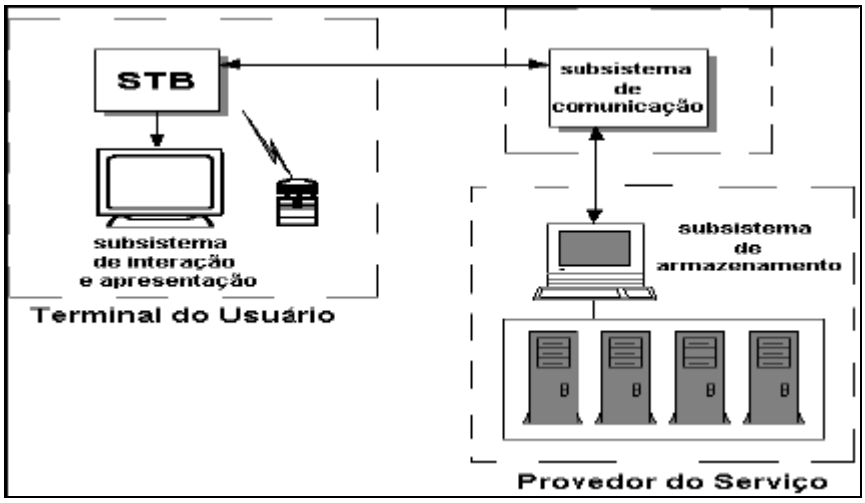


Figura 16 – Domínio de VOD.

Fonte: Extraído de Guizzardi (1997).

Após a análise do contexto da aplicação do sistema a ser desenvolvido, dá-se início à primeira fase, conforme se segue:

(i) Definir domínio

A empresa FICTICIA deseja uma solução para armazenamento de vídeos, uma vez que esta fará a locação de filmes a partir de uma página disponível na Internet. Em reunião com o cliente, esta expôs suas necessidades e ficou acordado que será desenvolvido um sistema para o armazenamento de vídeos. Posteriormente, outros módulos serão desenvolvidos com intuito de deixar o sistema completo. Após esta definição, foi desenvolvido o documento de visão.

(ii) Selecionar ontologia

A partir da análise do documento de visão, deu-se início a uma pesquisa de uma ontologia que atendesse ao domínio do sistema. Deste modo, foi selecionada uma ontologia expressa na linguagem LINGO no domínio de VOD. Esta engloba o domínio de negócio de armazenamento de vídeos, que foi requerido pelo cliente. A Figura 17 ilustra a ontologia selecionada na linguagem LINGO.

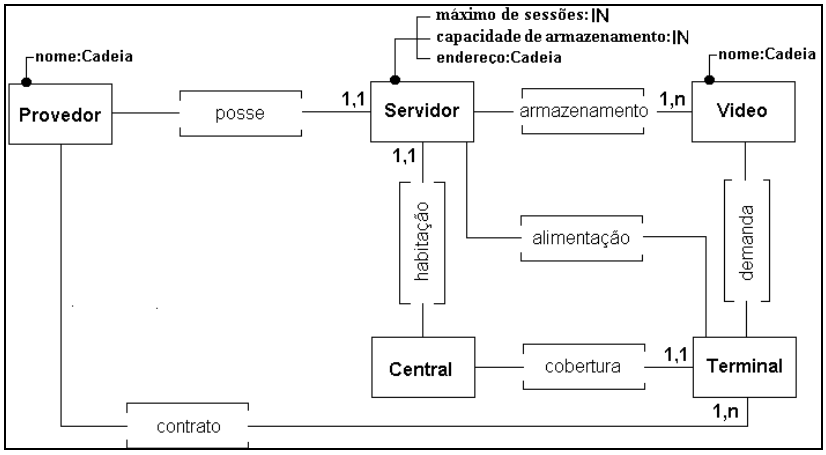


Figura 17 – Diagrama de conceitos e relações do domínio.
Fonte: Extraído de Guizzardi (1997).

Abaixo seguem as descrições dos conceitos e das relações presentes na ontologia. (GUIZZARDI, 1997, p. 139-143)

> Terminal: O Terminal representa o equipamento através do qual o usuário final tem acesso ao sistema [..] Servidor: Armazena os Vídeos e os transmite aos Terminais, em um formato suportado por qualquer Terminal participante [..] Vídeo: É, provavelmente, o principal conceito neste domínio. Um Vídeo diz respeito a uma entidade que identifica univocamente um título que compõe o catálogo de um Provedor [..] Provedor: o Provedor é uma entidade que possui Servidores e é contratada por Terminais, sendo responsável por manter os catálogos de Vídeo em seus Servidores e por gerenciar os usuários que os contratam [..] Central: Essa é a única entidade que aparece no modelo devido aos requisitos tecnológicos necessários para o oferecimento do serviço [..]
>
> [..] cobertura: Enfatiza a necessidade que deve existir, em um sistema de comunicação, entre Terminal e Central para que esse possa assistir a Vídeos transmitidos por um Servidor [..] habitação: Um Servidor deve possuir uma conexão com uma Central para que possa transmitir Vídeos para os Terminais cobertos por ela [..] alimentação: Esta relação encapsula um conjunto de pré-condições que devem ser satisfeitas para que um Terminal possa ser alimentado por um Servidor, ou seja, para que uma sessão de exibição de Vídeo possa ser firmada entre duas instâncias desses conceitos [..] demanda: Essa relação é definida em função de outras relações e é responsável por responder à principal questão de competência da ontologia: Que Vídeos podem ser assistidos por um dado Terminal?

(iii) Delimitar e validar a ontologia

A partir da ontologia de domínio de VOD, foi extraído, apenas, o domínio necessário para subsidiar a concepção do sistema. A Figura 18 expressa um recorte da ontologia original e esta representa, também, a visão do cliente operacional.

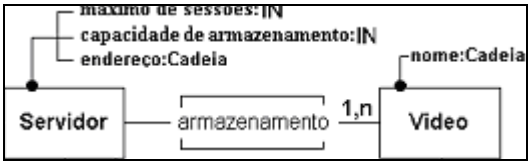


Figura 18 – Recorte da ontologia original e visão do cliente operacional.

Fonte: Extraído de Guizzardi (1997).

A Figura 19 apresenta um diagrama de classe em UML e este mostra uma possível visão do analista. Outros elementos da UML poderiam ser utilizados.

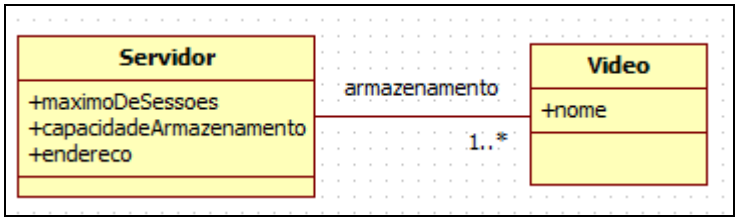


Figura 19 – Visão do analista.

Fonte: Elaboração própria.

Com a ajuda das visões pode-se validar o negócio do novo sistema junto aos clientes gerencial e operacional.

Nesta fase, o analista compreende as regras de negócio descritas a partir da ontologia e as discute com o cliente com o objetivo de validação.

(iv) ***Elicitar negócio***

As visões anteriores ajudaram para a melhor compreensão do negócio. Deste modo, a partir destas visões, ocorreram reuniões, *brainstorms*, avaliação de documentos da empresa e de sistemas similares entre outros, com o intuito de levantar mais informações, além das expressadas pela ontologia, a fim de completar o domínio expresso pelo novo sistema.

Logo, foi identificada a necessidade de separar os vídeos por servidores através da classificação de gênero. Outro requisito identificado é o tratamento diferenciado aos vídeos que são considerados como lançamentos, pois estes têm uma maior demanda de acessibilidade.

Durante esta atividade foi gerado o *glossário de negócio* (APÊNDICE A – ARTEFATOS CASO ILUSTRATIVO), que além dos termos identificados na ontologia, foi acrescido com outros identificados na elicitação do negócio, como, por exemplo: gênero de Comédia, que corresponde ao humor nas artes cênicas.

No documento *regras de negócios* são declaradas as condições que foram extraídas a partir da ontologia e as condições que foram identificadas.

Para continuidade da execução, foi identificado que os artefatos descritos anteriormente são suficientes. Portanto não houve necessidade da elaboração dos demais artefatos que são sugeridos pelo processo ***MDAONTO***, para esta atividade.

a. Definir equipe

A equipe, conforme descrito anteriormente, foi constituída somente pela autora deste trabalho que descreveu, de forma hipotética, todo o fluxo deste processo.

b. Definir plano geral

Não houve a aplicação desta atividade.

c. Definir ferramentas

Foram selecionados o ***PROTÓTIPO MDAONTO*** (atualmente em desenvolvimento) para a realização das transformações no paradigma MDA e a ferramenta a CASE, *open source*, STARUML.

Após a análise do modelo de negócio ao qual corresponde a camada CIM no padrão MDA, dá-se início a segunda fase (DESENVOLVIMENTO PIM), conforme se segue:

(v) Especificar PIM

Nesta atividade os elementos que compõem o modelo de negócio selecionado são transformados em um modelo independente de plataforma. Assim, a partir da solicitação dos clientes e da ontologia selecionada, foi gerado o modelo de análise, representado pelo diagrama de classes da Figura 20. Como a ferramenta selecionada, atualmente,

conhece o diagrama de classe, neste sentido, foi desenvolvido este diagrama, o qual contém dados da ontologia e dados das solicitações dos clientes.

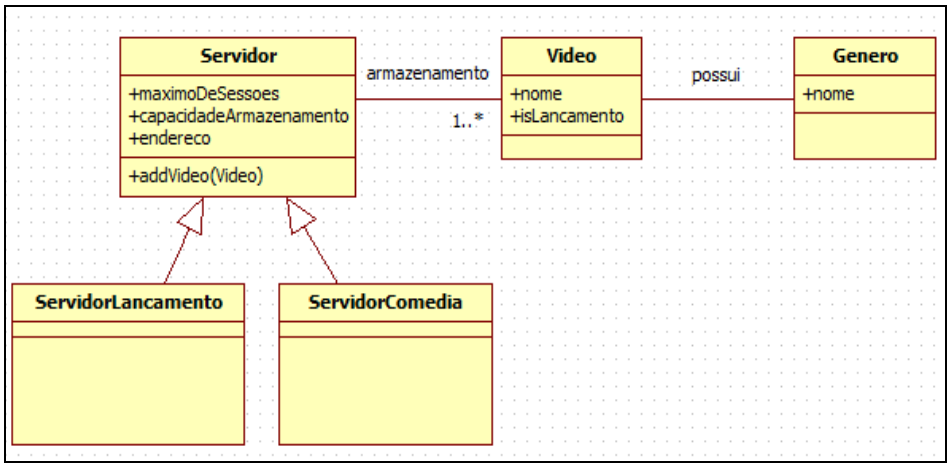


Figura 20 – Diagrama de classes compostos pelos os requisitos da ontologia e pelos requisitos do cliente.

Fonte: Elaboração própria.

A Figura 21 ilustra a geração do modelo PIM a partir do diagrama de classe descrito na Figura 24, este passo depende da ferramenta MDA selecionada, para este estudo foi utilizado o PROTÓTIPO ***MDAONTO***.

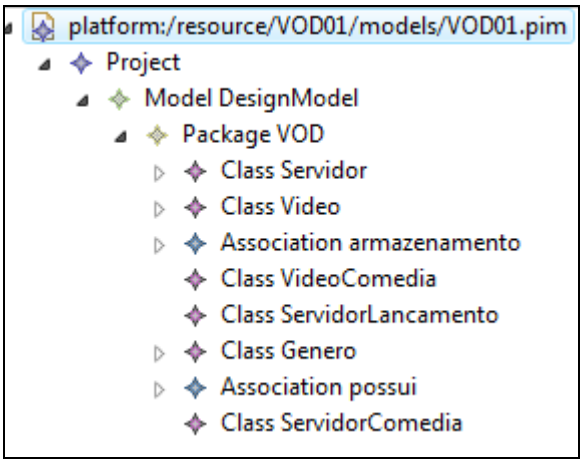


Figura 21 – Representação do PIM no PROTÓTIPO ***MDAONTO***

Fonte: Elaboração própria.

(vi) Validar

Nesta atividade é realizada a revisão formal do modelo de análise, ou seja, é confrontado o modelo referente ao negócio com o modelo de análise, verificando se existem inconsistências. Neste contexto, não houve inconsistências entre os artefatos gerados na fase anterior e nesta fase.

Após a análise do modelo PIM, dá-se início à terceira fase (DESENVOLVIMENTO PSM E DE CÓDIGO), conforme se segue:

(vii) Transformar em modelo PSM

Nesta atividade são transformados os artefatos da fase desenvolvimento PIM em um modelo de uma plataforma particular (Figura 22).

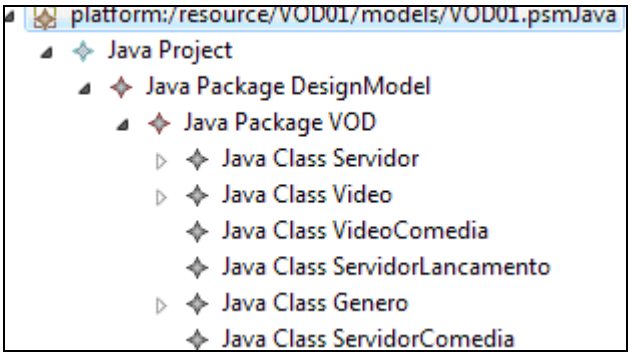


Figura 22 – Tradução entre PIM e PSM

Fonte: Elaboração própria.

Os passos para a transformação de PIM para PSM, também, dependem da ferramenta MDA selecionada.

(viii) Gerar código e testar

Nesta atividade é gerado código em uma linguagem específica, a empresa FICTICIA desejava o seu sistema em JAVA. Neste sentido, o código foi gerado nessa linguagem. Os passos para a transformação de PSM para código dependem da ferramenta MDA.

Após a geração de código, dá-se início à última fase. Para este exemplo, não foram analisadas as atividades desta fase, devido à dificuldade de simulação.

6.2 ESTUDO DE OPINIÃO

Para este estudo seguiram-se três fases, a saber: (*i*) definição dos objetivos: identificação e análise de pontos importantes; (*ii*) planejamento e operação: elaboração do processo de avaliação e execução da pesquisa; e, por fim, (*iii*) a análise e interpretação dos dados: interpretação dos dados obtidos na pesquisa.

O objetivo principal deste estudo é verificar se o processo ***MDAONTO*** é conciso e se atende aos propósitos por ele definidos. Para isto foram analisados quatro pontos: utilização da ontologia para facilitar a compreensão do negócio na elaboração de cada artefato; utilidade dos artefatos na condução do processo; adequação dos *stakeholders* ao processo; e adequação das atividades ao processo.

Para cada ponto exposto acima, foi concebida uma questão a fim de realizar esta avaliação, a saber: (*Q1*) A utilização da ontologia facilita na compreensão do negócio para a elaboração de cada artefato? (*Q2*) Quais os artefatos que são úteis ao processo? (*Q3*) Quais os *stakeholders* que estão adequados ao processo? e, por fim, (*Q4*) Quais as atividades que estão adequadas ao processo?

Para a realização desta pesquisa foi confeccionado um questionário dividido em duas partes. A primeira parte do questionário consiste na identificação do perfil dos participantes, visando aferir a formação e a experiência profissional de cada entrevistado. A Figura 23 ilustra o questionário de avaliação do perfil do entrevistado.

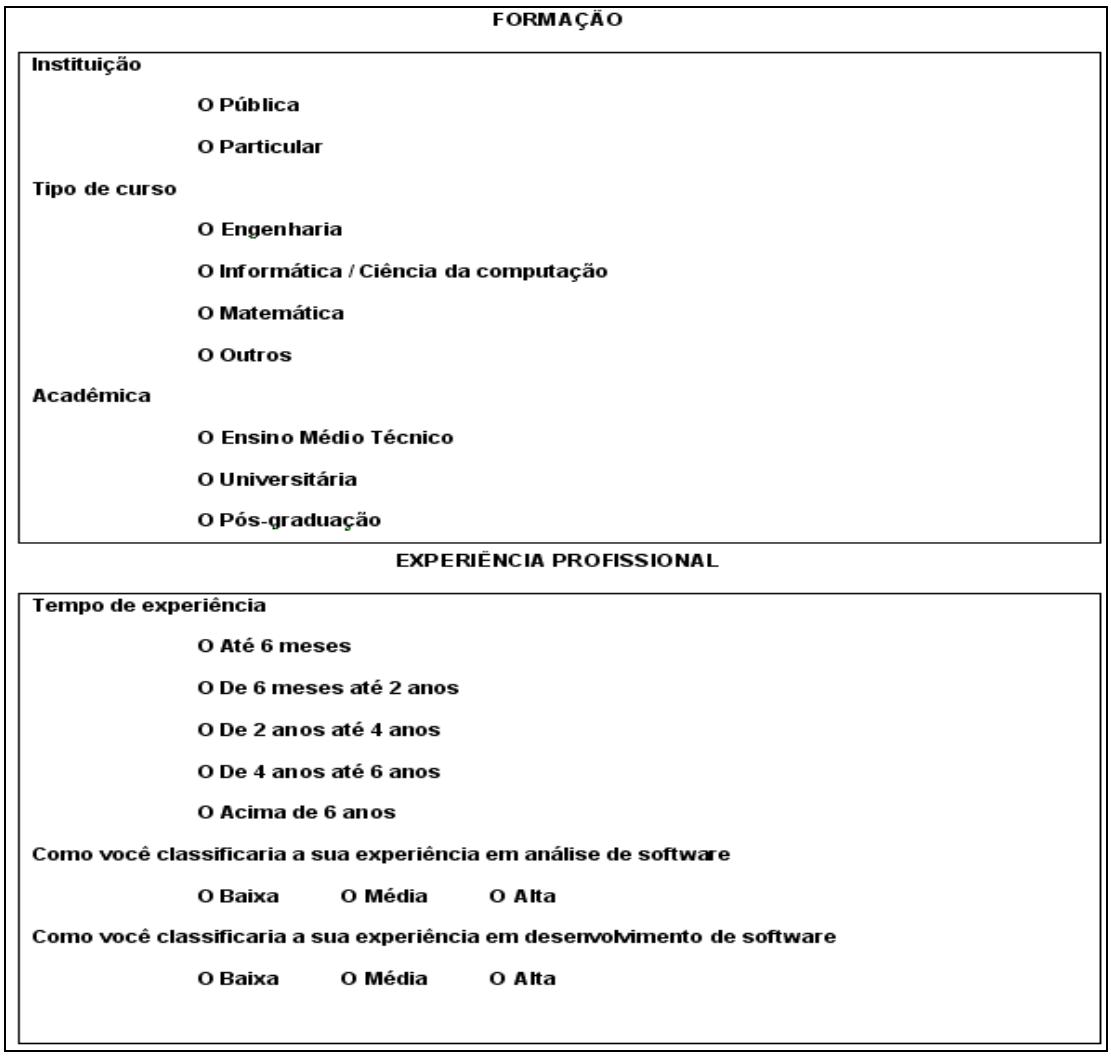


Figura 23 - Questionário de avaliação de perfil do entrevistado
Fonte: Elaborado pela autora.

A segunda parte do questionário apresenta as quatro questões, onde cada questão visa avaliar a aderência de um elemento dentro dos objetivos propostos pelo processo:

- Questão 01 (*Q1*) ***A utilização da ontologia facilita na compreensão do negócio para a elaboração de cada artefato?*** Esta questão visa medir se, com a utilização da ontologia, é possível facilitar a compreensão do negócio para a construção de cada artefato proposto no processo. O Quadro 16 ilustra o questionário referente à primeira questão, onde os entrevistados respondem *sim* ou *não*, ou seja, ajuda ou não ajuda.

Artefatos	Questão 01 *(Q1)*	
	Sim	**Não**
documento de visão do negócio		
Visão do cliente gerencial		
Visão do cliente operacional		
Visão do analista		
glossário do negócio		
regras de negócio		
modelo de caso de uso de negócio		
modelo de objeto de negócio		
documento de papel e responsabilidade		
Plano de desenvolvimento de software		
lista de riscos		
modelo de análise (PIM)		
modelo de projeto (PSM)		
código em linguagem específica		
scripts de		
aplicação desenvolvida (ou conjunto de funcionalidades)		
manual do usuário		

Quadro 16 - Questionário de avaliação do processo *MDAONTO*, primeira questão
Fonte: Elaboração própria.

- Questão 02 (*Q2*) ***Quais os artefatos que são úteis ao processo?*** Esta questão visa avaliar se cada artefato proposto está adequado ao processo ***MDAONTO***. O Quadro 17 ilustra o questionário referente à segunda questão, onde os entrevistados respondem *sim* ou *não*, ou seja, são úteis ou não são úteis.

Artefatos	Questão 02 *(Q2)*	
	Sim	**Não**
documento de visão do negócio		
visão do cliente gerencial		
visão do cliente operacional		
visão do analista		
glossário do negócio		
regras de negócio		
modelo de caso de uso de negócio		
modelo de objeto de negócio		
documento de papel e responsabilidade		
plano de desenvolvimento de software		
lista de riscos		
modelo de análise (PIM)		
modelo de projeto (PSM)		
código em linguagem específica		

scripts de teste		
aplicação desenvolvida (ou conjunto de funcionalidades)		
manual do usuário		

Quadro 17 - Questionário de avaliação do processo *MDAONTO*, segunda questão
Fonte: Elaboração própria.

- Questão 03 (*Q3*) ***Quais os stakeholders que estão adequados ao processo?*** Esta questão visa avaliar se cada *stakeholder* proposto está adequado ao processo ***MDAONTO***. O Quadro 18 ilustra o questionário referente à terceira questão, onde os entrevistados respondem *sim* ou *não*, ou seja, está adequado ou não está adequado.

Stakeholders	**Questão 03 *(Q3)***	
	Sim	**Não**
cliente gerencial		
cliente operacional		
analista (responsável)		
designer de negócio		
analista do processo de negócios		
analista de requisitos		
analista de sistemas		
revisor de requisitos		
especificador de requisitos		
implementador		
arquiteto de software		
testador		
redator técnico		

Quadro 18 - Questionário de avaliação do processo *MDAONTO*, terceira questão
Fonte: Elaboração própria.

- Questão 04 (*Q4*) ***Quais as atividades que estão adequadas ao processo?*** Esta questão visa avaliar se cada atividade proposta está adequada ao processo ***MDAONTO***. O Quadro 19 ilustra o questionário referente à quarta questão, onde os entrevistados respondem *sim* ou *não*, ou seja, está adequada ou não está adequada.

Atividades	**Questão 04 *(Q4)***	
	Sim	**Não**

(i) Definir domínio		
(ii) Selecionar ontologia		
(ii.a) Criar uma ontologia		
(iii) Delimitar e validar ontologia		
(iv) Elicitar e validar negócio		
(v) Definir equipe		
(vi) Definir plano geral		
(vii) Definir ferramenta		
(viii) Especificar análise PIM		
(ix) Validar		
(x) Transformar em modelo PSM		
(xi) Gerar código e testar		
(xii) Testar		
(xiii) Documentar		
(xiv) Implantar		
(xv) Suporte e Manutenção.		

Quadro 19 - Questionário de avaliação do processo *MDAONTO*, quarta questão
Fonte: Elaboração própria.

Após definição dos objetivos foi dado início à operação. Assim, foi conduzido junto aos alunos do mestrado de Sistemas e Computação da Universidade Salvador (UNIFACS) uma apresentação do processo ***MDAONTO*** e resolução de dúvidas para garantir o entendimento do processo. No segundo momento, foram entreguess os questionários citados anteriormente e uma ontologia expressada na linguagem LINGO.

A última fase, por fim, é a análise e interpretação dos dados. O Quadro 20 ilustra os perfis dos participantes da pesquisa. É possível observar que todos têm uma média experiência em análise e desenvolvimento de software, deste modo todos os participantes estão aptos a responder ao questionário.

Participante	**Instituição**	**Tipo de curso**	**Acadêmica**	**Tempo de experiência**	**Experiência em análise**	**Experiência em desenvolvimento**
Participante 1	Pública	Informática / Ciência da computação	Pós-graduação	De 6 meses até 2 anos	Média	Média
Participante 2	Particular	Informática / Computação	Universitária	De 4 anos até 6 anos	Alta	Média
Participante 3	Particular	Informática / Computação	Universitária	Acima de 6 anos	Média	Alta
Participante 4	Particular	Informática / Computação	Universitária	Acima de 6 anos	Média	Média

Participante 5	Particular	Informática / Computação	Pós-graduação	Acima de 6 anos	Alta	Alta
Participante 6	Particular	Informática / Computação	Universitária	De 4 anos até 6 anos	Média	Alta
Participante 7	Particular	Informática / Computação	Universitária	De 4 anos até 6 anos	Média	Alta
Participante 8	Particular	Informática / Computação	Universitária	De 2 anos até 4 anos	Alta	Baixa
Participante 9	Pública	Informática / Computação	Universitária	De 2 anos até 4 anos	Média	Alta
Participante 10	Particular	Informática / Computação	Pós-graduação	De 2 anos até 4 anos	Média	Média
Participante 11	Pública	Informática / Computação	Universitária	Acima de 6 anos	Alta	Alta

Quadro 20 - Perfis dos participantes do estudo
Fonte: Elaboração própria.

A primeira questão avaliou a utilidade de uma ontologia na construção de artefatos associados ao processo ***MDAONTO*** – os resultados encontram-se no gráfico da Figura 24. Analisando o gráfico é possível concluir que a ontologia facilita na produção dos artefatos (vide a lista de artefatos no ANEXO C). É possível observar, ainda, que o plano de desenvolvimento de software se beneficia pouco da ontologia. De certo, esse produto depende mais das decisões de projeto do que do contexto de negócio.

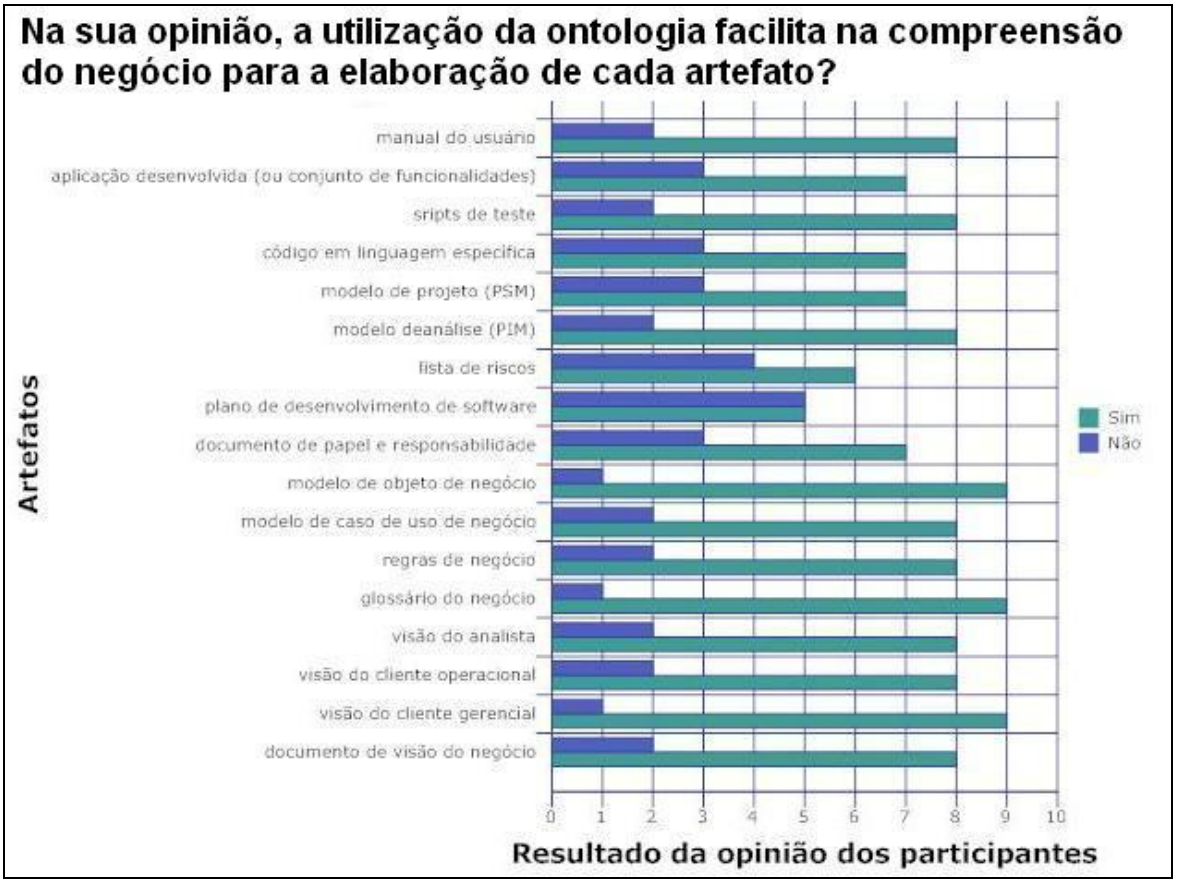


Figura 24 - Resultado relacionado à primeira pergunta do questionário
Fonte: Elaboração própria.

A segunda questão buscou analisar se o artefato apresentado é útil ao processo proposto, ou seja, esta medição buscou avaliar se um determinado artefato atende às expectativas do processo e auxilia na execução de uma atividade.

Analisando o gráfico da Figura 25, observa-se que, no entendimento dos participantes, a maior parte dos artefatos é útil ao processo ***MDAONTO***.

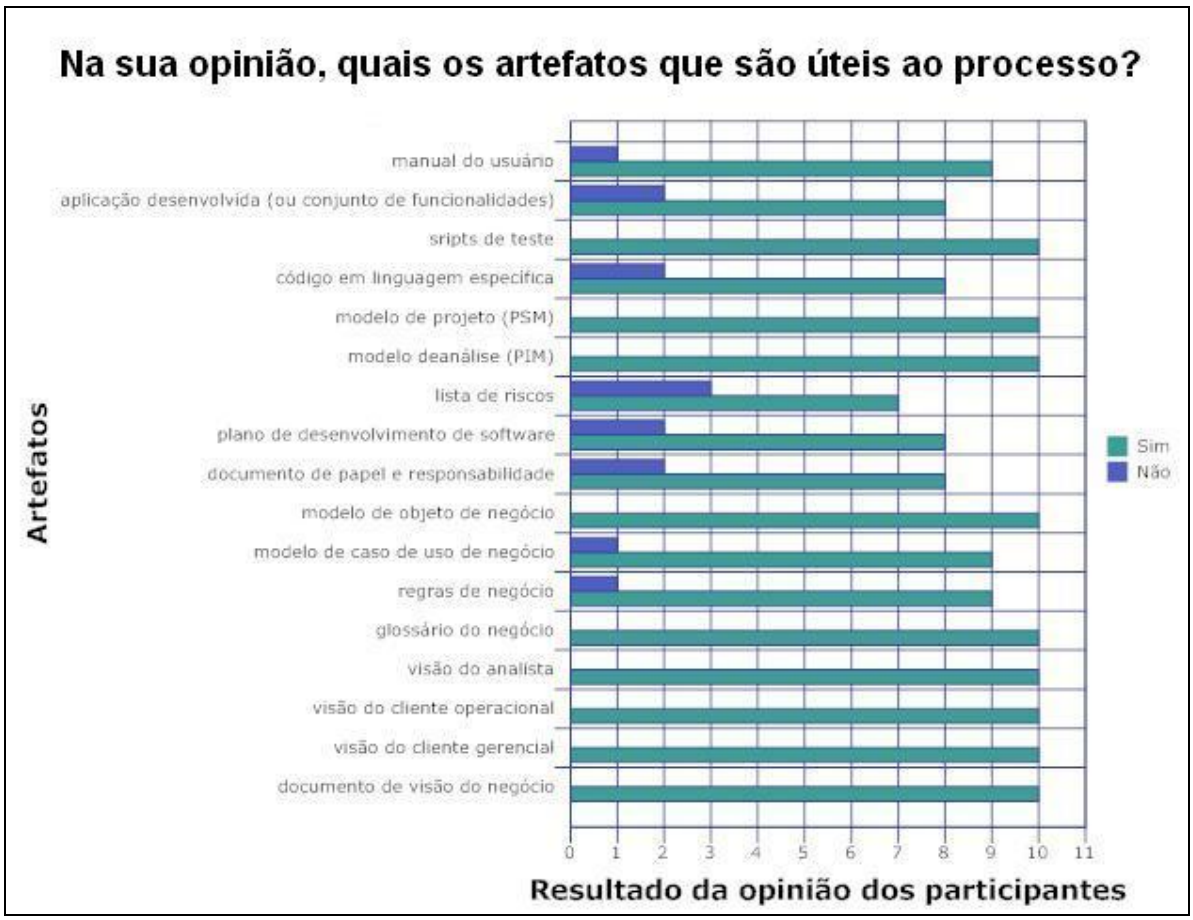


Figura 25 - Resultado relacionado à segunda pergunta do questionário
Fonte: Elaboração própria.

A terceira questão buscou analisar se os *stakeholders* propostos estão adequados ao processo ***MDAONTO***, ou seja, esta medição buscou avaliar se um determinado *stakeholder* está adequado à execução das atividades propostas no processo.

Analisando o gráfico da Figura 26, é possível verificar que os *stakeholders* atendem as responsabilidades impostas pelo processo proposto.

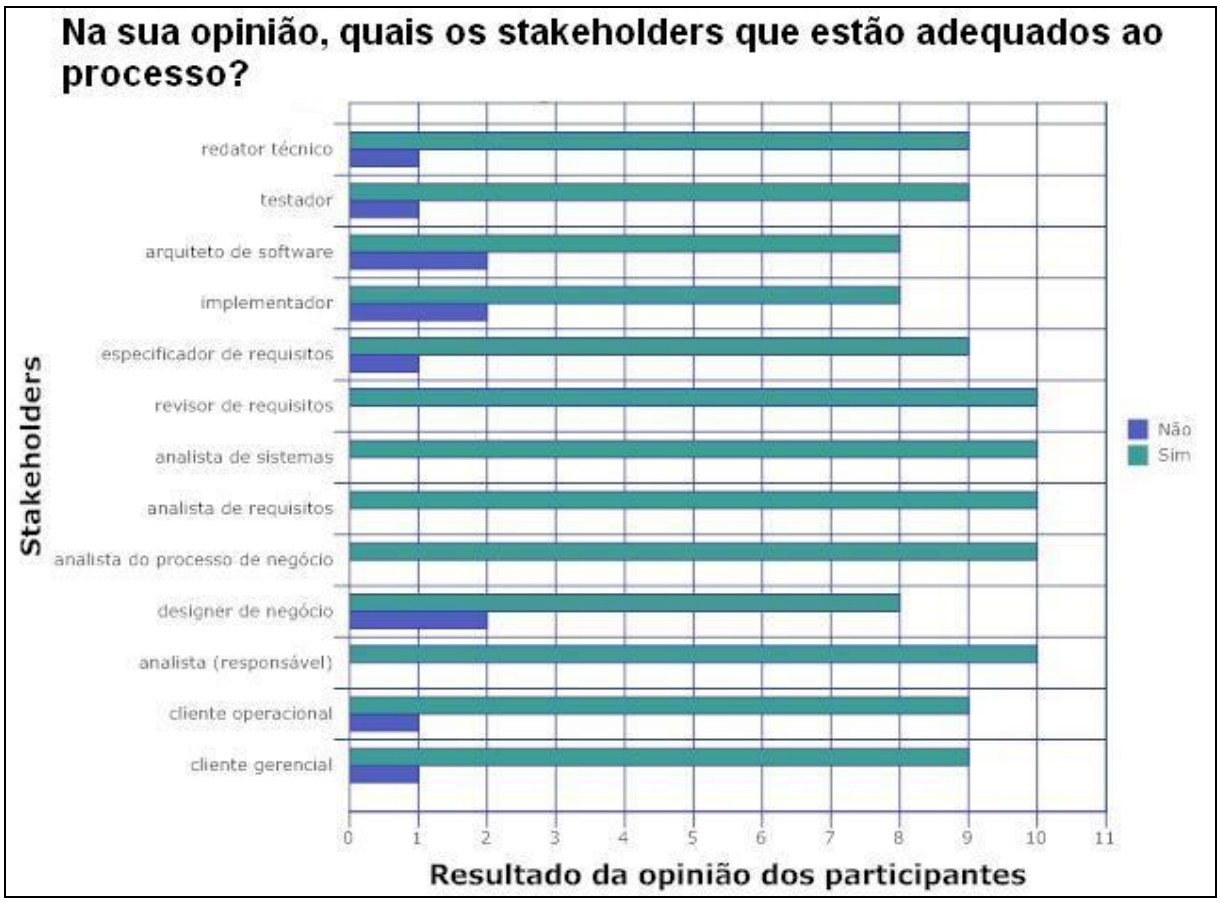


Figura 26 - Resultado relacionado à terceira pergunta do questionário
Fonte: Elaboração própria.

A quarta questão buscou analisar se as atividades propostas estão adequadas ao processo ***MDAONTO***, ou seja, esta medição buscou avaliar se uma determinada atividade está adequada ao processo proposto. Analisando o gráfico apresentado na Figura 27, é possível concluir que todas as atividades definidas no processo estão aderentes.

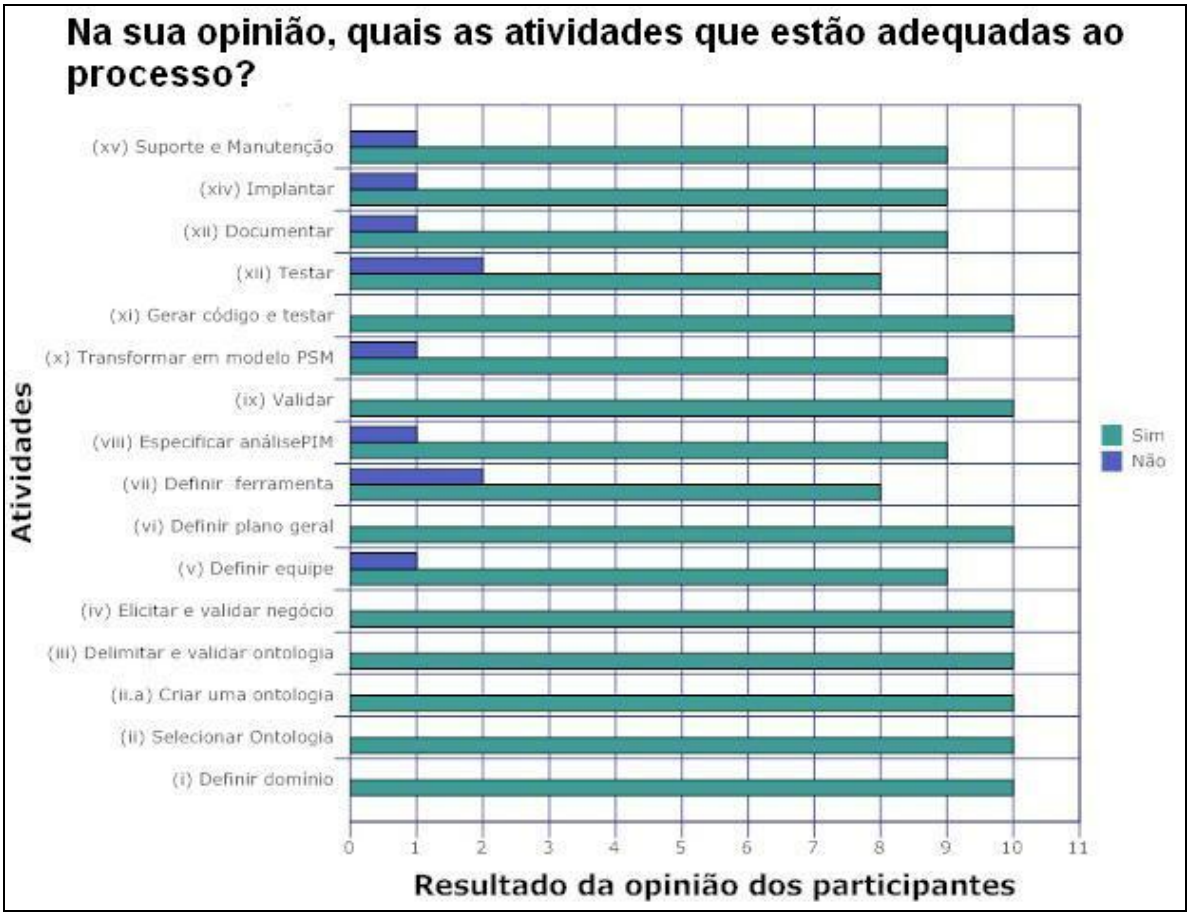


Figura 27 - Resultado relacionado à terceira pergunta do questionário
Fonte: Elaboração própria.

6.3 CONCLUSÃO

Este capítulo apresentou a aplicação do processo ***MDAONTO*** em um exemplo ilustrativo e sua avaliação através de um estudo de opinião.

O exemplo ilustrativo apresentou uma aplicação do processo no desenvolvimento em um sistema de armazenamento de vídeos a partir de uma ontologia (no domínio de multimídia).

O estudo, por sua vez, visou à verificação do processo proposto. Nele, como resultado da pesquisa conduzida, tem-se que a maior parte das fases, atividades e *stakeholders* está concisa e adequada aos objetivos empreendidos pelo processo proposto.

Vale ainda salientar que a utilização da ontologia no processo foi bem aceita por todos os participantes da pesquisa de opinião. Denotando, então, que os benefícios discutidos anteriormente são de grande valia na utilização de uma ontologia aliada ao processo de desenvolvimento de software.

Vale lembrar que os participantes (do estudo de opinião) não tinham conhecimento prévio do processo e, mesmo assim, este foi bem aceito por todos os presentes. É importante ressaltar que as questões empreendidas servirão de base para estudos mais aprimorados, como, por exemplo, uma experimentação. Algumas discussões poderão conduzir a hipóteses relevantes em trabalhos futuros. É necessário, então, um estudo mais detalhado a fim de melhorar e aplicar, de fato, o processo descrito neste trabalho.

REFERÊNCIAS

ALMEIDA, M.; BAX, M. **Uma visão geral sobre ontologias:** pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. Ciência da Informação, v.32, n. 3, set/dez. 2003. Disponível em: <http://www.scielo.br/pdf/ci/v32n3/19019.pdf >. Acesso em: 1 abr. 2008.

ALMEIDA, M. **Um modelo baseado em ontologias para representação da memória organizacional.** 2006. 345 f. Tese (Doutorado) - Escola de Ciência da Informação da Universidade Federal de Minas Gerais – UFMG, Belo Horizonte, 2006.

ASADI, M.; RAMSIN, R. MDA-based methodologies: an analytical survey. In: PROCEEDINGS OF THE 4TH EUROPEAN CONFERENCE ON MODEL DRIVEN ARCHITECTURE FOUNDATIONS AND APPLICATIONS, 4, 2008, Berlin. **Anais**... Berlin: Schieferdecker and A. Hartman, 2008. p. 419–431.

ASADI, M.; RAVAKHAH, M.; RAMSIN, R. An MDA-based system development lifecycle, In: ASIA INTERNATIONAL CONFERENCE ON MODELING & SIMULATION, 2., 2008, Kuala Lumpur, Malásia. **Anais eletrônico**… Kuala Lumpur, Malásia: IEEE, 2008. Disponível em: <http://www2.computer.org/portal/web/csdl/doi/10.1109/AMS.2008.19>. Acesso em: 20 dez. 2008.

BECK, Kent. et al. **Manifesto for agile software development**, 2001. Disponível em: <http://agilemanifesto.org/>. Acesso em: 6 ago. 2007.

BECK, Kent. **Extreme programming explained:** embrace change. [S.l.]: Addison-Wesley Professional, 1999. 167 p.

BEZIVIN, J. On the unification power of models, **Software and System Modeling. Journal**, v.4, n. 2, 2005. Disponível em: < http://www.sciences.univ-nantes.fr/lina/atl/www/papers/OnTheUnificationPowerOfModels.pdf >. Acesso em: 20 jan. 2009.

BORST, W. **Construction of engineering ontologies, graduate school for information and knowledge systems**. 1997. Disponível em: <http://www.ub.utwente.nl/webdocs/inf/1/t0000004.pdf>. Acesso em: 1 abr. 2008.

COTA,R.I.; MENEZES, C.S.; FALBO, R. A. Modelagem organizacional utilizando ontologias e padrões de análise, In: MEMORIAS DE VII WORKSHOP IBEROAMERICANO DE INGENIRÍA Y DESARROLLO DE AMBIENTES DE SOFTWARE, 7., 2004, Portugal. **Anais eletrônico**... Disponível em:

<http://www.inf.ufes.br/~falbo/download/pub/2004-IDEAS-2.pdf>. Acesso em: 20 jan. 2009.

DJURIC, D. et al. Ontology modeling and MDA**, Journal of Object Technology**, v. 4, n. 1, 2005. Disponível em: <http://www.jot.fm/issues/issue_2005_01/article3/article3.pdf>. Acesso em: 1 abr. 2008.

FALBO, R. **Integração de conhecimento em um ambiente de desenvolvimento de software**, 1998. 203 f. Tese (Doutorado) – Universidade Federal do Rio de Janeiro / COPPE.

FALBO, R. Experiences in using a method for building domain ontologies, In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING, 16., 2004. Canada. **Anais eletrônicos**... Disponível em: <http://www.inf.ufes.br/~falbo/download/pub/2004-OIA.pdf.>. Acesso em: 22 jan. 2008.

GAVRAS, Anastasius; BELAUNDE, Marino; PIRES, L. Ferreira; ALMEIDA, J. Paulo A. Towards an MDA-based development methodology, In: EUROPEAN WORKSHOP ON SOFTWARE ARCHITECTURE, 1., 2004a, Netherlands**, Anais eletrônicos**… Disponível em: <http://arch.cs.utwente.nl/pubs/pires/2004/2004-GaBeFeAl-confpaper.pdf>. Acesso em: 1 jul. 2008.

GAVRAS, Anastasius; BELAUNDE, Marino; PIRES, Luís Ferreira; ALMEIDA, J. Paulo A. Towards an MDA-based development methodology for distributed applications, In: EUROPEAN WORKSHOP ON MODEL-DRIVEN ARCHITECTURE WITH EMPHASIS ON INDUSTRIAL APPLICATIONS, CTIT TECHNICAL REPORT TR-CTIT-04-12, 1., 2004b, **Anais**… Netherlands: M. van Sinderen, L. Ferreira Pires, 2004. p. 71-81.

GERVAIS, Marie-Pierre. Towards an MDA-oriented metodology, In: PROCEEDINGS OF THE 26TH ANNUAL INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, 26., 2002, **Anais**… Paris: IEEE Press Xplore Digital Library, 2002, p. 265-270.

GÓMEZ-PÉREZ, A.; FERNÁNDEZ-LÓPEZ, M.; CORCHO, O. **Ontological engineering,** 2nd., British: Springer-Verlag London Limited, 2004. 403p.

GUARINO, N. Formal ontology and information system, In: INTERNATIONAL CONFERENCE ON FORMAL ONTOLOGY AND INFORMATION SYSTEMS, 1., 1998, Trento, Italy. **Anais eletrônicos**... Disponível em: <http://www.loa-cnr.it/Papers/FOIS98.pdf>. Acesso em: 1 jun. 2008.

GUIZZARDI, G.; **Hipervisao:** a logical architecture for video on demand systems. 1997. Dissertação (Mestrado) – Departamento de ciência da computação. Universidade Federal do Espírito Santo.

HAAV, H.M.; LUBI, T.L. A survey of concept-based information retrieval tools on the WEB, In. EAST-EUROPEAN CONFERENCE ADBIS, 5., 2001. **Anais eletrônicos**… Disponível em: <http://www.science.mii.lt/ADBIS/local2/haav.pdf>. Acesso em: 1 maio 2008.

JASPER, R.; USCHOLD, M. A framework for understanding and classifying ontology applications, In. WORKSHOP ON ONTOLOGIES AND PROBLEM-SOLVING METHODS, 1999. **Anais eletrônicos**… Disponível em: <http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18/11-uschold.pdf>. Acesso em: 1 maio 2008.

KLEPPE, Anneke; WARMER, Jos; BAST, Win. **MDA Explained the model driven architecture:** practice and promise. [S.l.]: Part of the Addison-Wesley Object Technology Series, 2003. 170 p.

KITCHENHAM, B.; LINKMAN, S.; LAW, D. DesmeT: a methodology for evaluating software engineering methods and tools. **Computing and Contrological Engineering Journal**, v. 8, n. 3, p.120-126. , jun. 1997

LARRUCEA, X.; DIEZ, A.B.G.; MANSELL, J.X. Practical model driven development process. In. WORKSHOP ON MODEL DRIVEN ARCHITECTURE (MDA), 2004. **Anais eletrônicos**... Disponível em: <http://www.cs.kent.ac.uk/projects/kmf/mdaworkshop/submissions/Larrucea.pdf>. Acesso em: 22 jan. 2009.

MEDEIROS, C. Gerenciamento de dados em base de dados heterogêneas, In. IV ERBASE - ESCOLA REGIONAL DE COMPUTAÇÃO BAHIA E SERGIPE, 2004. **Anais eletrônicos**... Disponível em: <http://www.uefs.br/erbase2004/documentos/erbase/erbase4_p09_claudia_medeiros.pdf>. Acesso em: 22 jan. 2009.

MIZOGUCHI, R.; VAN WELKENHUYSEN, J.; IKEDA, M.; Task ontology for reuse of problem solving knowledge In: ECAI'94 TOWARDS VERY LARGE KNOWLEDGE BASES, **Anais Eletrônico**… Amsterdam, Amsterdam: IOS Press, p. 46,.

ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO – SOFTEX. **Guia de Implementação**: Parte 1: Nível G (Versão 1.1). 2007. 26 p. Disponível em: < http://www.softex.br/mpsBr/_guias/MPS.BR_Guia_de_Implementacao_Parte_1_V1.1.pdf>. Acesso em: 22 jan. 2009.

NOLL, Rodrigo P. **Rastreabilidade ontológica sobre o processo unificado.** 2007. 132 f. Dissertação (Mestrado) - Pontifícia Universidade Católica do Rio Grande do Sul. Disponível em: <http://tede.pucrs.br/tde_busca/arquivo.php?codArquivo=791 >. Acesso em: 22 jan. 2009.

NOY, F.N.; GUINNESS, D. L. Ontology development 101: a guide to create your first ontology, 2001. **Anais eletrônicos**... Disponível em: <http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html>. Acesso em: 22 jan. 2008.

OMG, OBJECT MODELING GROUP. Model driven architecture (MDA). 2001. **Anais eletrônicos**... Disponível em: <http://www.omg.org/docs/ormsc/01-07-01.pdf>. Acesso em: 01 janeiro de 2008.

OMG, OBJECT MODELING GROUP. MDA guide version 1.0.1. 2003. **Anais eletrônicos**... Disponível em: <http://www.omg.org/docs/omg/03-06-01.pdf>. Acesso em: 01 janeiro de 2008.

OMG OBJECT MODELING GROUP. UML 2.0 superstructure specification, 2004. **Anais eletrônicos**... Disponível em: <http://www.omg.org/docs/formal/05-07-04.pdf>. Acesso em: 10 abril de 2008.

OMG, OBJECT MODELING GROUP – OMG. Object Constraint Language, 2006. **Anais eletrônicos**... Disponível em: <http://www.omg.org/docs/formal/06-05-01.pdf> Acesso em: 10 abril de 2008.

POOLE, J.; CHANG, D.; TOLBERT, D.; MELLOR, D. **Common warehouse metamodel:** developer's guide. Indianapolis: Willey Publishing & Sons Inc, 2003, 716 p.

PRESSMAN, Roger S. **Engenharia de software**. [S.l.]: Makron Books, 1995.

RATIONAL Unified Process **–** RUP, 2001**.** Apresentação de um processo de desenvolvimento de software. Disponível em: <http://www.wthreex.com/rup/>. Acesso em: 01 junho de 2008.

ROCHA, A. R. C.; MALDONADO, J. C.; WEBER, K. C. **Qualidade de software:** teoria e prática, São Paulo: Makron books 2001. 303 p.

RÊGO, Elisângela; PINTO, George; SALVADOR, Lais; CHAVEZ, Christina; Santos, Weber. Extração de visões centradas em stakeholders a partir de uma ontologia: abordagem MDA. In. SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 21, 2007, João Pessoa. **Anais**... João Pessoa, 2007.

SOMMERVILLE, Ian. **Engenharia de software**. São Paulo: Pearson Addison-Wesley, 2007.

UNCHOLD, M. GRUNINGER, M.; Ontologies: principles, methods and applications. In. KNOWLEDGE ENGINEERING REVIEW, 1996. **Anais eletrônicos**... Disponível em: <https://eprints.kfupm.edu.sa/55793/1/55793.pdf>. Acesso em: 01 junho de 2008.

USCHOLD, M.: KING, M. Towards a methodology for building ontologies. In. WORKSHOP ON BASIC ONTOLOGICAL ISSUES IN KNOWLEDGE SHARING, 1995. **Anais eletrônicos**… Disponível em: <http://www.cs.unicam.it/insegnamenti/reti_2008/Readings/Uschold95.pdf>. Acesso em: 01 junho de 2008.

VAN-HEIJIST, G.; SCHREIBER, A. T.; WIELINGA, B. J. Using explicit ontologies in KBS development. **International Journal of Human-Computer Studies**. v. 46, n. 2-3, p. 183 - 292 , m. fev./mar. 1997.

WARMER, Jos; KLEPPE, Anneke. **The object constraint language:** getting your models ready for MDA. 2 nd ed. [S.l.]: Addison Wesley, 2003.

W3C. OWL WEB ontology language guide. 2004. Disponível em: <http://www.w3.org/TR/owl-guide/>. Acesso em: 10 abr. 2008.