Luiz Henrique dos Santos Cruz

Mapeamento de Modelo Independente de Computação para Modelo Independente de Plataforma, utilizando o modelo de referência ODP

São Paulo

2010

Luiz Henrique dos Santos Cruz

Mapeamento de Modelo Independente de Computação para Modelo Independente de Plataforma, utilizando o modelo de referência ODP

Trabalho apresentado à Escola Politécnica
da Universidade de São Paulo para conclusão
do MBA em Tecnologia da Informação

Área de concentração: Tecnologia da Informação

Orientador: Prof. Dr. Jorge Luís Risco Becerra

São Paulo

2010

Cruz, Luiz Henrique dos Santos

Mapeamento de Modelo Independente de Computação para Modelo Independente de Plataforma, utilizando o modelo de referência ODP / Luiz Henrique dos Santos Cruz. -- São Paulo, 2011.

44p.

Monografia (MBA em Tecnologia da Informação) – Escola Politécnica da Universidade de São Paulo. Programa de Educação Continuada em Engenharia. Área de concentração: Engenharia de software.

Orientador: Prof. Dr. Jorge Risco Becerra

1.Desenvolvimento de software 2.Sistemas de informação. Universidade de São Paulo. Escola Politécnica. Programa de Educação Continuada em Engenharia.

DEDICATÓRIA

Dedico este trabalho a minha querida esposa Karina

pela paciência e apoio durante meus estudos.

AGRADECIMENTOS

Agradeço ao meu orientador Prof. Dr. Jorge Risco pelas orientações, considerações pertinentes e por não desistir de cobrar sempre empenho de todos. Agradeço também a todos os professores do MBA em TI pelos bons ensinamentos.

Ao meu amigo Kleberson dos Santos pela paciência e revisão crítica do meu trabalho.

RESUMO

Este trabalho busca estabelecer premissas para orientar o mapeamento de um Modelo Independente de Computação (CIM) em um Modelo Independente de Plataforma (PIM), seguindo os preceitos da Arquitetura Orientada a Modelos (MDA).

As premissas serão fundamentadas nas visões do modelo de referência para processamento distribuído e aberto (RM-ODP), já os modelos tem como principais elementos, o diagrama de processo de negócio (BPMN) e os diagramas de modelagem (UML). Esta abordagem tem como benefício a criação de uma visão única entre a área de negócio e a área de tecnologia da informação.

Palavras chave: MDA, RM-ODP, UML e BPMN

ABSTRACT

This paper seeks to establish premises to guide the mapping of a computation independent model (CIM) in a platform independent model (PIM), following the concepts of the model-driven architecture (MDA).

The premises are based on views of the reference model for open distributed processing (RM-ODP). The first two MDA models have as main elements, the business process diagram (BPMN) and modeling diagrams (UML). The benefit is to creating a single viewpoint between the business area and the information technology area.

Keywords: MDA, RM-ODP, UML e BPMN

LISTA DE FIGURAS

Figura 3.4 – Diagrama de caso de uso do sistema publicação web

Figura 3.5 – Diagrama de pacotes do sistema publicação web

LISTA DE ABREVIATURAS E SIGLAS

BPM	*Business Process Management*
BPMI	*Business Process Management Initiative*
BPMN	*Business Process Modeling Notation*
CIM	*Computational Independent Model*
ISO	*International Organization for Standardization*
MDA	*Model Driven Architecture*
ODP	*Open Distributed Processing*
OMG	*Object Management Group*
PIM	*Plataform Independent Model*
PSM	*Plataforma Specific Model*
RM-ODP	*Reference Model of Open Distributed Processing*
SI	Sistema de Informação
TI	Tecnologia de Informação
UML	*Unified Modeling Language*

LISTA DE TABELAS

SUMÁRIO

1. Introdução

1.1. Importância do tema

De acordo com o Object Management Group (OMG, 2003), um dos grandes objetivos e desafios arquitetura orientada a modelos MDA (Model Driven Architecture), é a separação do negócio – representado pelo modelo independente de computação (CIM), da lógica da aplicação, – representado pelo modelo independente de plataforma (PIM) e da plataforma – representado pelo modelo específico de plataforma (PSM). Esta separação de domínios, o domínio do problema e o domínio da solução, é uma oportunidade única para entender como é estabelecida a relação entre eles. Da mesma forma, o modelo de referência de processamento distribuído e aberto (RM-ODP) divide o objeto em análise através das visões, com o objetivo melhorar a compreensão e facilitar a interoperabilidade dos elementos de software distribuídos.

No mercado, as empresas são obrigadas a tomarem decisões rápidas, precisam transferir informações com agilidade, adaptar-se a mudanças de demandas, que geralmente ocorrem em ciclos curtos (KOK et al, 2009). E a possibilidade de analisar o problema por camadas de abstração, simplifica a forma de entender o problema que se quer automatizar com um sistema.

Quando a análise envolve processos de negócio, o primeiro nível de abstração, a notação de modelagem de processos de negócio BPMN (Business Process Modeling Notation) é muito importante por ser clara a todos, desde os analistas de negócio que criam os primeiros rascunhos do processo, os técnicos responsáveis por implementar a tecnologia que realizará os processos e quem vai monitorar o negócio (WHITE, 2006). A própria notação tem evoluído no sentido de se aproximar com o processo de desenvolvimento de software.

Além disto, arquitetos de software perceberam que quando o processo começa pela análise dos requisitos de negócio, há menor chance de mudanças e correções (KARDOŠ, 2008).

1.2. Objetivo

Este trabalho busca um método simplificado, baseado na arquitetura orientada por modelos (MDA), para geração do modelo independente de computação (CIM) e um modelo independente de plataforma (PIM). No processo de elaboração dos modelos a transição entre os modelos será feita com base nas visões do modelo de referência ODP (Open Distributed Plataform) e nas melhores práticas de BPMN e UML.

Este método será avaliado com a criação dos modelos para um sistema web de publicação de notícias, sem a pretensão de criar um sistema final e sim uma arquitetura de informação coerente em relação ao negócio.

1.3. Motivação e justificativa acadêmica

Em quase todas as propostas de MDA, há um direcionamento claro para ferramentas e processos de desenvolvimento de software, especificamente a transformação de um modelo PIM em um ou mais PSMs. Mas uma etapa igualmente importante é realizada com prioridade menor, que é a transformação do CIM em PIM, ou seja, tradução do negócio em arquitetura.

A metodologia MDA4WS, emprega o uso de diagramas de processos de negócio no CIM, sugerindo que os processos de negócio sejam gerados a partir de casos de uso e regras de negócio ou os processos existentes sejam refinados, para que seja compreendido o fluxo em um nível melhor de detalhes (ELIZONDO, 2006).

Já o método Maat, propõe a transformação de um PIM em PSM, fazendo uso do escopo e regras de negócio (FERREIRA, 2006). De uma forma sucinta, tenta completar a necessidade de entender o requisito de negócio da empresa com diagramas UML adicionais no modelo PIM e desta forma incorpora ao PIM o CIM.

Ambas as proposições utilizam, mesmo que não claramente, a visão da computação do modelo de referência ODP, orientando a ligação ou junção dos modelos. Quase sempre orientado a tecnologia de serviços web. Vale mencionar que a semelhança entre o MDA e o RM-ODP é notória, principalmente no que se refere a modelagem por pontos de vista.

Para atender um sistema de menor complexidade que o proposto por outros autores (ELIZONDO, 2006) e (FERREIRA, 2006), com o negócio devidamente entendido, a modelagem precisa ter elementos descritivos claros. Nesta ideia que o trabalho se fundamenta.

1.4. Organização do trabalho

Este trabalho orienta-se pelas seguintes etapas:

- Pesquisa bibliográfica: análise seletiva de teses de dissertação e artigos relacionados com MDA, BPMN e UML. Serão levados em conta também as visões do RM-ODP, engenharia de software e atividades do desenvolvimento descritas na ISO 12207;

- Definição do tema: através da análise do material escolhido, o escopo do método englobará o conhecimento do negócio com o conhecimento acadêmico.

- Definição de um método que orienta a criação dos modelos iniciais de uma arquitetura MDA. Para tanto, utiliza-se como exemplo um sistema de publicação de notícias;

- Análise de resultados: Com olhar crítico, os pontos chaves da mapeamento serão questionados e futuras evoluções serão sugeridas.

2. Parte Teórica

2.1. Conceitos básicos

2.1.1. Arquitetura de software

O padrão IEEE 1471-2000, define que a arquitetura é a organização fundamental de um sistema, composta por: componentes, seus relacionamentos, ambiente e os princípios básicos que guiam seu projeto e evolução (IEEE 1471,2000).

Logo, ela é um meio de comunicação entre os participantes. A arquitetura de software representa uma abstração comum do sistema para os participantes do projeto e serve de base para o entendimento, negociação consenso e comunicação. Isto deve ser realizado através de um documento, utilizando uma notação que possa ser entendida facilmente (BASS,2003).

Não é só uma questão política e sim uma abstração de modo que a solução proposta seja compreendida sempre de uma forma aditiva, ou seja, para que cada modelo abstraído sirva para construção de um novo, através de novas conexões.

2.1.2. Model Driven Architecture (MDA)

O MDA, ou mais precisamente o conjunto de conceitos para uma Arquitetura Orientada por Modelos é uma forma de organizar e gerenciar arquiteturalmente um sistema, em um processo de desenvolvimento de software, garantindo que modelos sejam gerados para mapear funcionalidades e comportamentos, sem distorções da plataforma que os implementa. Podemos então enumerar os objetivos do MDA (OMG, 2003):

- Portabilidade, no sentido de um software ser facilmente portável a outra plataforma;
- Interoperabilidade entre sistemas heterogêneos ou camadas de uma mesma aplicação;

- Reuso, com a separação das preocupações de uma plataforma da arquitetura.

Em engenharia de software, os modelos servem como uma forma de representar um sistema hipotético, geralmente a ser construído. Além de ser uma forma de testar o sistema sem construir de fato, os modelos servem de base para estimar todos os processos que envolvem a produção de software.

No MDA o modelo geralmente é representado por diagramas e texto descritivo. (OMG, 2003). Um bom modelo deve ter as seguintes características(MELLOR et al, 2005):

- Omite informações para ajudar os visualizadores a verem o problema em mão com muito mais clareza e checar a correção do entendimento desenvolvido;
- Reflete com precisão um pouco da realidade abstrata ou hipotética;
- Deve ser mais econômico do que construir o objeto final;
- Serve de meio de comunicação ilustrando uma ou mais ideias.

Para entender melhor o que esta referência define, é preciso entender os conceitos de: plataforma, pontos de vista, modelos CIM, PIM e PSM e transformação entre modelos.

a) Plataforma

O MDA define plataforma como conjunto de subsistemas e tecnologias que fornecem funcionalidades através de interfaces e padrões de uso especificadas (OMG, 2003).

Vale mencionar que cada plataforma especificada tem suas características específicas de tecnologia. Portanto as funcionalidades são instanciadas ou implementadas de forma coerente apenas quando seguem os padrões impostos pelas tecnologias.

De certa forma é uma das etapas finais, já que implica em caracterizar uma arquitetura em um ambiente específico. Por exemplo, especificar para: J2EE, .NET, XML etc.

b) Pontos de vista do MDA

Com embasamento nos pontos de vista do modelo de referência para sistemas distribuídos e abertos (RM-ODP), da própria OMG, o MDA segue três pontos de vista, ou visões: independente de computação, independente de plataforma e específico de plataforma (OMG, 2003).

Dividir o modelo nestas visões significa abstrair parte da realidade, ou suprimir detalhes estruturais, afim de analisar com mais clareza o problema (e não fugir dele). No primeiro caso a abstração é em linhas gerais uma abstração de tecnologia, voltado para o que o negócio quer solucionar com um sistema de informação. A segunda tem o conceito de abstrair a plataforma, com o intuito de entender os pedaços de software que compõem a arquitetura dos sistema e como é a operação deste sistema. Já no último ponto de vista, a abstração é o contexto, ou seja, através da visão anterior, o nível de detalhe aumenta para instanciar em uma plataforma específica. Os modelos são as visões dos pontos de vista, ou seja, são a realização de um ponto de vista.

c) Modelos CIM, PIM e PSM

Antes de entender individualmente cada modelo é necessário entender o fluxo dos modelos em seus níveis de abstração. A seguinte figura, faz este papel:

Figura 2.1 – Visão geral do MDA

O código obviamente não faz parte dos modelos, mas é onde pretende-se chegar com o processo de desenvolvimento de software.

CIM

No modelo CIM, a abstração é caracterizada como pertencente ao domínio do problema. Algumas regras são implícitas ao modelo:

- Linguagem mais próxima de quem domina o processo;
- Especifica quais objetos do mundo real serão tratados;
- Abstrai solução tecnológica, mas ao mesmo tempo faz alusão a requisitos funcionais e não funcionais;
- Foco em escopo do produto e não em escopo do projeto.

PIM

O modelo independente de plataforma, como o nome diz, reserva uma independência de plataforma no sentido de ser especificado para qualquer plataforma. Uma forma de definir este modelo é pensar em um sistema que rodará independente de tecnologia como uma máquina virtual, com detalhes como a comunicação, o agendamento, o fluxo e nomenclaturas. As características básicas deste modelo são:

- Utilizar uma linguagem específica de cunho tecnológico;
- Especificar elementos básicos do sistema;
- Foco em escopo do projeto e menos em escopo do produto;
- Abstrair plataforma de implementação (genérica o suficiente para ser futuramente implementada em diferentes plataformas).

PSM

Um modelo específico de plataforma é a visão de um sistema para uma plataforma específica. Ele combina detalhes do PIM com as especificações do sistema que a suportará. Um modelo PIM pode gerar múltiplas PSMs. Este é o ganho de portabilidade que estes dois modelos garantem.

d) Transformação de modelos

Transformação é o processo de mapear características de um modelo de forma que a marcação sirva para criar um modelo de outro nível de abstração. A transformação por marcações permite o caminho inverso se necessário (MELLOR,2005).

As entradas deste processo de transformação, são em geral de três tipos:

- Padrões – padrões identificados por um arquiteto de software ou por um sistema que automatiza o processo;
- Escolhas tecnológicas – Entradas manuais do arquiteto de software ou automatizações propostas de acordo com o modelo PIM utilizado;
- Requisitos de qualidade – Conformidade com a plataforma escolhida.

Quando é feita a transição do PIM para o código sem modelos intermediários, a transformação é do tipo direta. Quando há a definição de um ou mais PSMs, a transformação é elaborativa. Há disponível no mercado soluções de ambos os tipos, mas vale ressaltar que vale apenas como avaliação. Para utilização em produção é aconselhável utilização da transformação elaborativa, com revisão do código gerado por planos de teste.

2.1.3. Reference Model – Open Distributed Processing (RM-ODP)

O conceito de processamento distribuído e aberto ODP foi introduzido como padrão em 1995, principalmente pela necessidade de interoperabilidade de sistemas de diferentes companhias ou até ambientes heterogêneos aonde predominavam sistemas monolíticos e hierárquicos (KENT, 1998).

Em um esforço do ISO/IEC e ITU-T, foi criado um modelo de referência, ou guia de arquitetura, para implementação de sistemas ODP (RM-ODP).

O RM-ODP está focado em quatro fundamentos:

- Abordagem da especificação do sistema através de modelagem de objetos;
- Especificação do sistema de forma separada, mas inter-relacionadas através de pontos de vista;
- Definição de uma infraestrutura distribuída transparente ao sistema;
- Um modelo de conformidade.

Estes fundamentos garantem ao sistema distribuído e aberto, que cada parte possa ser adquirida de diferentes fornecedores e mesmo em um sistema heterogêneo e em diferentes infraestruturas, a interoperabilidade seja mantida.

2.1.4. Pontos de vista do RM-ODP

O modelo de referência ODP tem cinco pontos de vista: Corporativo, Informação, Computação, Engenharia e Tecnologia. Uma especificação inclui mais de um ponto de vista, sendo que devem ser mutuamente consistentes (ITU-T/ISO/IEC, 2010). Vale ainda mencionar que cada ponto de vista tem conceitos e regras associadas, relevantes a natureza da especificação.

O RM-ODP não é uma metodologia e sim um framework arquitetural. Portanto, não há uma estratégia formal de utilização, logo recomenda-se que ele se adeque ao sistema que há o desejo de implementação (BECERRA, 98).

Há um consenso em engenharia de software no sentido de utilizar pontos de vista para dividir o problema que se quer analisar.

Os pontos de vista são:

a. Empresa

Foco em escopo, políticas e propósito. Neste ponto de vista dois conceitos são fundamentais: Comunidade, que significa um conjunto de objetos para atingir um objetivo e federação, que é uma comunidade de domínios diferentes. Toda a modelagem segue regras (permissões, obrigações e proibições) praticadas pelo sistema.

A conformidade neste ponto de vista significa que o sistema deve estar de acordo com as regras e políticas do ambiente em que está inserido.

b. Informação

Este ponto de vista define a semântica da informação e do processamento. Analisa a relação dos objetos em seus diferentes estados, a saber:

- Invariante – uma série de predicados em um ou mais objetos informacionais que são sempre verdadeiros;
- Estático – Em um dado tempo de observação, o estado do objeto não sofre variação;
- Dinâmico – Uma especificação de objeto da informação que permite variações de estado ou condição. Esta variação pode criar objetos ou constantes temporárias.

A conformidade é possível através de pontos de referência, relacionados com o ponto de vista da computação e de engenharia. Engenharia para ter acesso ao sistema e computação para saber quais as regras se aplicam. Desta forma, analisa-se como a informação está associada.

c. Computação

Este ponto de vista trata da decomposição do sistema em objetos distribuídos.

Através da decomposição funcional, define como serão as interfaces dos objetos. É responsável por criar templates para os diferentes tipos de interface e métodos de interação (sinal, operação e fluxo).

Uma das características mais importantes deste ponto de vista é a transparência. Em termos de transparência, é estruturado em:

- Organização dos objetos computacionais;
- Ações internas destes objetos;
- Interações destes objetos;
- Contratos de ambiente e suas interfaces.

As próprias interfaces dos objetos podem ser pontos de referência para validação de conformidade.

d. Engenharia

Definição das funções para suportar a interação entre os objetos. Este ponto de vista é estruturado para um sistema distribuído, em duas áreas:

- Responsabilidade dos objetos com o sistema (organização em clusters, cápsulas ou nós);
- Responsabilidade de comunicação entre os sistemas (regras dos canais de comunicação).

e. Tecnologia

Escolha tecnológica. Quais tecnologias de infraestrutura serão utilizadas. Vai desde a linguagem de programação até topologia dos servidores necessários. Mas vale lembrar que trata-se de uma especificação e portanto não são necessários todos os detalhes.

Neste ponto de vista são considerados também como será a implementação e testes de validação.

2.1.5. Business Process Management (BPM)

Em síntese, BPM é definido como uma série de técnicas que auxiliam a melhoria contínua dos processos de negócio. Tem o foco em pessoas e negócios, com referência principalmente em como eles trabalham juntos. Logo trata-se de uma ferramenta estratégica para analisar, entender e documentar os processos de negócio (OMG,2008).

Como processo de negócio entende-se uma série de atividades executadas com o propósito de atingir um objetivo para a empresa, como processar uma requisição do cliente, satisfazer uma requisição regulatória, entre outros. A figura a seguir demonstra como é o processo:

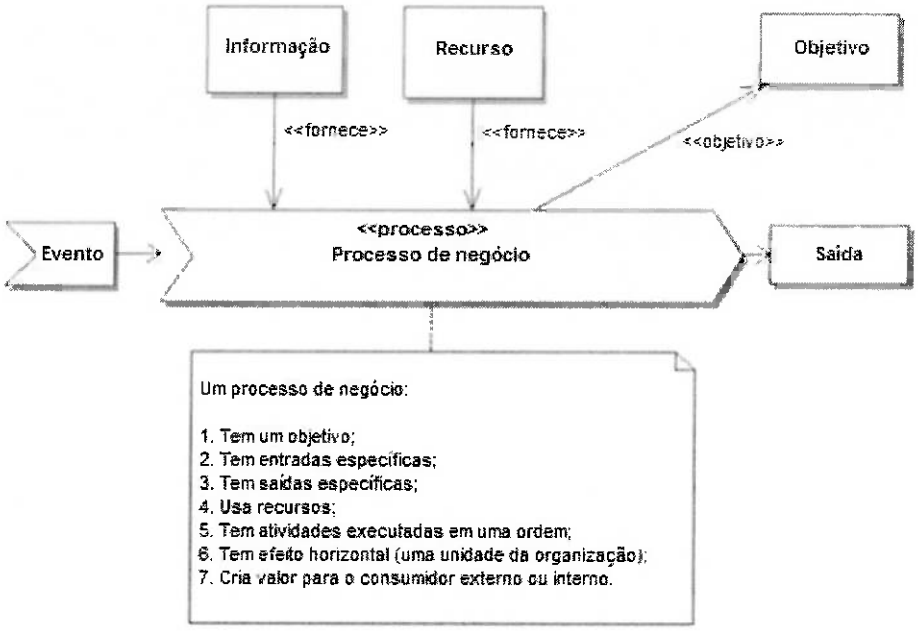

Figura 2.2- Processo de negócio (SPARX, 2004)

2.1.6. Business Process Modeling Notation (BPMN)

BPMN é uma notação e semântica utilizadas na modelagem de processos, do ponto de vista do negócio. De fácil leitura, serve como rascunho inicial de um projeto, podendo até servir como guia de gerência (OMG,2008). Uma das grandes vantagens da notação é a transformação automática dos processos modelados. A linguagem utilizada para esta transformação é o WS-BPEL, que não será abordada neste trabalho.

A notação foi concebida pelo BPMI, que em 2005 se integrou ao OMG para concatenar esforços na melhoria do gerenciamento de processos de negócio. Neste trabalho, a versão 1.2 desta especificação é utilizada (OMG,2009).

Para o correto entendimento da notação é fundamental a compreensão dos elementos básicos. A organização da notação sugere quatro categorias básicas:

a) Objetos de fluxo

Relativos ao comportamento do processo de negócio.

i. Eventos - Representa início ou fim de um evento;

Figura 2.3 – Objeto de fluxo - Evento

ii. Atividades - Atividade do processo;

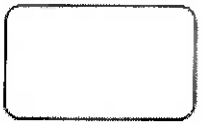

Figura 2.4 – Objeto de fluxo - Atividade

iii. Conectores – Controla a divergência ou convergência de um fluxo.

Figura 2.5 – Objeto de fluxo - Conector

b) Objetos de conexão

Conexão entre os elementos ou entre um elemento e uma informação.

i. Sequência – Demonstra a ordem das atividades;

Figura 2.6 – Objeto conexão - Sequência

ii. Mensagem – Troca de mensagens entre participantes;

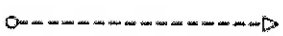

Figura 2.7 – Objeto conexão - Mensagem

iii. Associação – Associa uma informação ao fluxo.

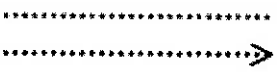

Figura 2.8 – Objeto conexão - Associação

c) Raias

Divisão lógica dos papeis nos processo de negócio.

i. Piscina – Container que representa uma entidade do processo;

Figura 2.9 – Raias - Piscina

ii. Raias – Organiza e categoriza as entidades.

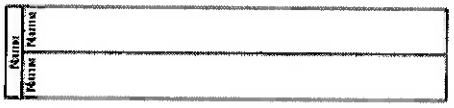

Figura 2.10 – Raias - Raias

d) Artefatos

Utilizados para prover informação adicional ao processo.

i. Objeto – Produto de uma atividade;

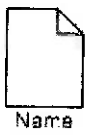

Figura 2.11 – Artefato - Objeto

ii. Grupo – Utilizado para análise e documentação. Não afeta o fluxo;

Figura 2.12 – Artefato - Grupo

iii. Anotação – Informação adicional para entendimento.

Figura 2.13 – Artefato - Anotação

São três os tipos de modelagens propostas pela especificação BPM-N v1.2, a saber:

a) Privada

Este tipo de fluxo ou processo de negócio é utilizado para demonstrar um processo interno de uma organização específica, com especificidades que não interessam o público externo. A característica fundamental é possuir apenas uma piscina.

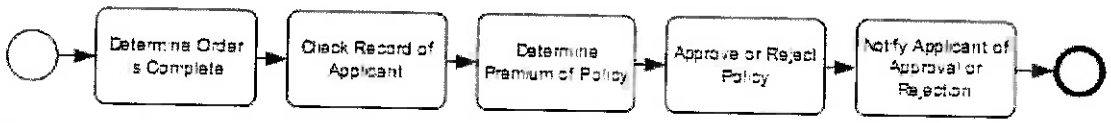

Figura 2.14 – Modelo - Processo de negócio privado (WHITE, 2006)

b) Abstrata

No modelo abstrato há a referência de comunicação de um processo interno com o público externo, mas não há mapeamento das atividades externas (processo abstrato).

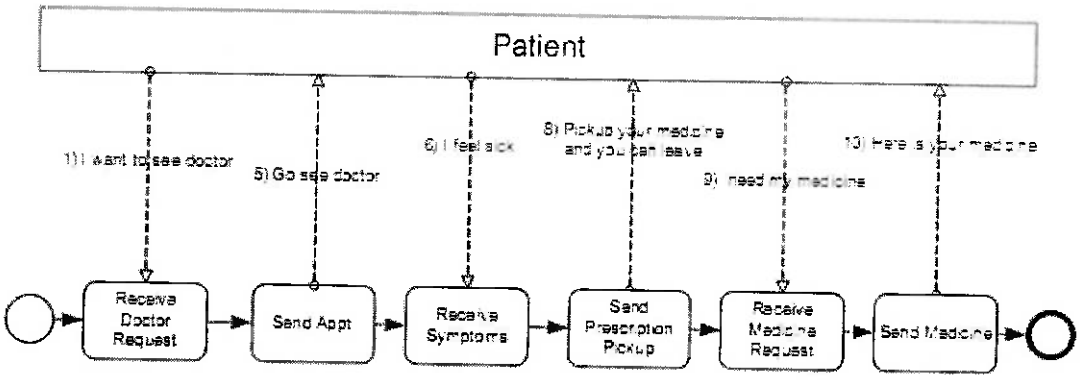

Figura 2.15 – Modelo - Processo de negócio abstrato (WHITE, 2006)

c) Colaborativa ou global

A modelagem global reflete interações entre duas ou mais entidades de negócio. Logo traz um grau de elucidação maior.

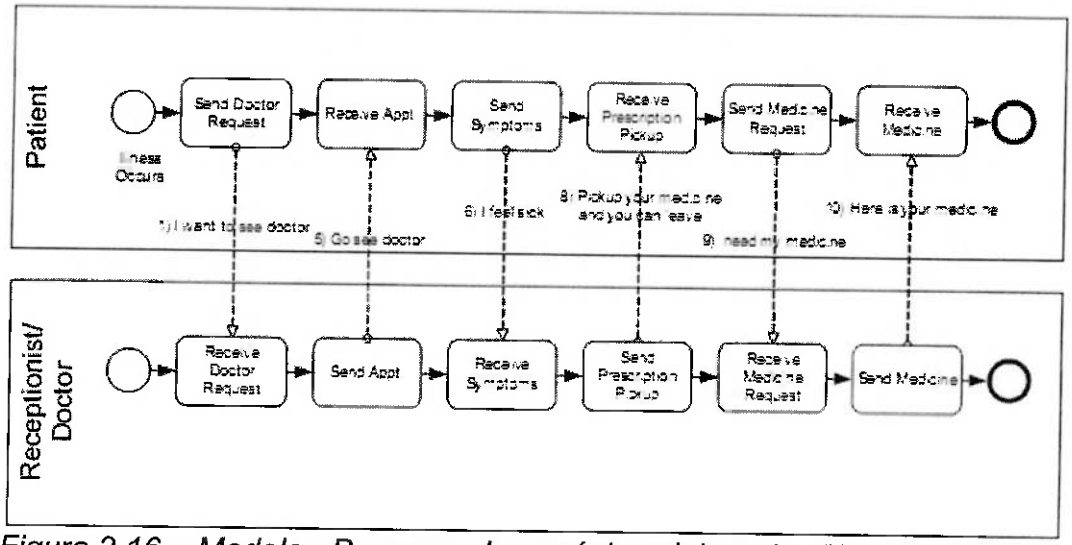

Figura 2.16 – Modelo - Processo de negócio colaborativo (WHITE, 2006)

Para o propósito deste trabalho, o terceiro tipo é o mais indicado, a medida que revela todos os atores envolvidos com o sistema de informação.

2.1.7. MDA4WS

O processo de desenvolvimento de software MDA4WS utiliza como referência de processo de desenvolvimento o modelo cascata (*waterfall*), contemplando as etapas: levantamento de requisitos, análise de requisitos e projeto de software. O objetivo deste método é aplicar o MDA ao desenvolvimento de software orientado a modelos e a web services.

Como é necessário obter quais serviços o software irá executar para saber se ele consta no catálogo de serviços, ou se é necessário criar, o MDA4WS aplica o modelo de arquitetural MDA, definindo pontos de vista, ou interesses, posteriormente organiza o ponto de vista em uma visão, ou uma realização deste ponto de vista, para finalmente definir os elementos do modelo. A figura a seguir

demonstra como está arquitetado o MDA dentro de um processo de desenvolvimento de software:

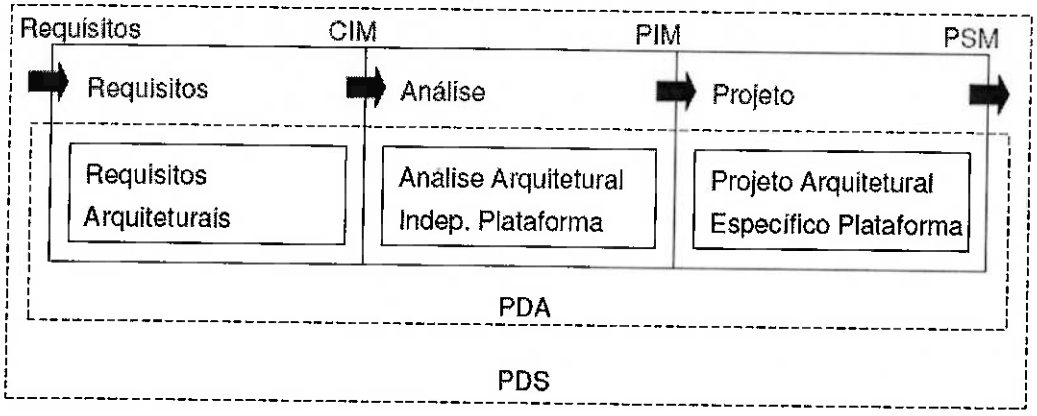

Figura 2.17 – Relação entre as fases do processo de descrição de arquitetura (PDA) MDA e o processo de desenvolvimento de software (PDS) MDA4WS(ELIZONDO, 2006)

O modelo independente de computação (CIM), resultante da fase ou etapa de requisitos do MDA4WS, está resumido na tabela abaixo:

Ponto de vista independente de computação (PIC)	**Visão independente de computação (VIC)**	**CIM**
• Objetivo e escopo do sistema • Atores do sistema • Funcionalidades do sistema • Informações utilizadas pelo sistema	• Descrição dos objetos de escopo Diagrama de caso de uso • Diagramas de objetos do negócio • Diagrama do ciclo de vida dos objetos de negócio • Relação dos requisitos não funcionais	• Todos elementos da visão • Diagramas de fatos e regras de negócio • Diagramas de processo de negócio

Tabela 2.1 – CIM MDA4WS (ELIZONDO, 2006)

Este modelo especifica as funcionalidades e oculta a estrutura de processamento do sistema, visando descrever: objetivos, regras e processos de negócio e a informação utilizada nos processos identificados.

De forma semelhante, o modelo independente de plataforma (PIM), resultante da etapa de análise do MDA4WS, está resumido na tabela abaixo:

Ponto de vista independente de plataforma (PIP)	**Visão independente de plataforma (VIP)**	**PIM**
• Quais são os requisitos não funcionais associado as funcionalidades escolhidas • Qual o estilo arquitetural a ser utilizado • Como as funcionalidades serão decompostas em elementos independentes • Como o sistema realiza operações com os elementos independentes • Como os elementos estão organizados	• Diagrama de estrutura (classes) • Diagrama de comportamento (sequência e interação) • Diagrama de estados • Diagrama de pacotes	• todos os diagramas da visão • diagrama de serviços e mensagens • diagrama de protocolo de serviço • diagrama de entidade

Tabela 2.2 – PIM MDA4WS (ELIZONDO, 2006)

A etapa de análise consiste nas atividades: análise arquitetural independente de plataforma, identificar serviço, identificar realização de serviços e refinar o sistema. A relação do PIM com o CIM, é que nestas atividades, o foco principal é satisfazer os requisitos identificados. Notadamente na etapa de refinamento, é verificado se todas as regras de negócio do CIM são atendidas.

2.1.8. Método Maat

O método de desenvolvimento de software Maat utiliza como referência de processo o ISO/IEC 12207, mais precisamente o processo fundamental de desenvolvimento de software, com as etapas: análise de requisitos, design de software e construção de software detalhadas.

O método vale-se de uma técnica sugerida pela arquitetura orientada a modelos MDA, onde o CIM é encapsulado dentro da abstração de plataforma, o modelo PIM. A figura abaixo representa como o método é definido.

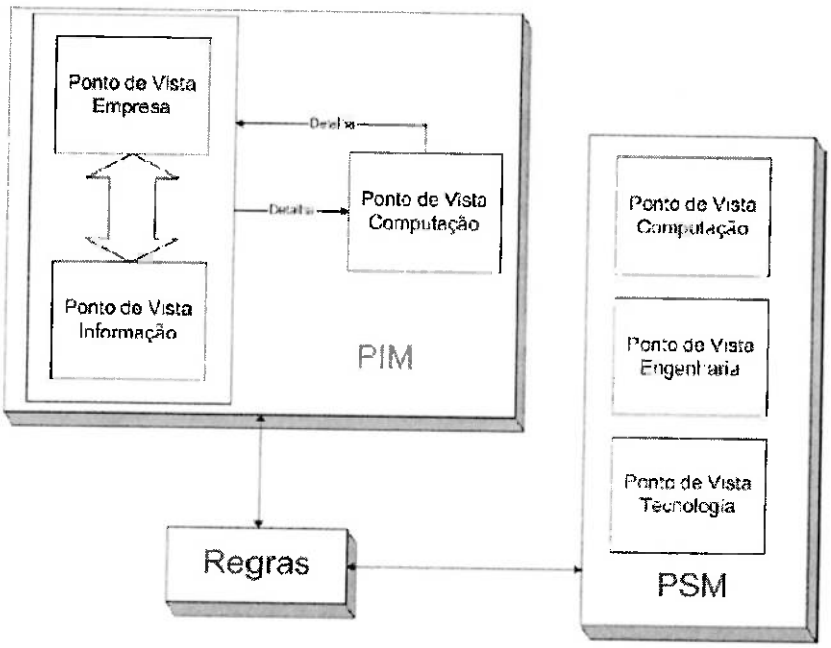

Figura 2.18– Estrutura básica do método Maat (FERREIRA, 2006)

O PIM é definido segundo os pontos de vista, resumidos na tabela abaixo:

Pontos de vista	**PIM**	**Objetivo**
Empresa	• Escopo e regras do sistema • Cenários de caso de uso	Definir abrangência e escopo do sistema
Informação	• Diagrama dos objetos de informação (diagrama de classes)	Definir as informações associadas ao sistema e seus usos

	• Diagrama de estado dos objetos; • Lista de regras e restrições	
Computação	• Realização dos casos de uso (diagrama de sequência, contrato de interação entre objetos e diagrama de estado) • Diagrama de objetos	Definir quais objetos implementam os itens anteriores

Tabela 2.3 – PIM método Maat (FERREIRA, 2006)

O autor através dos pontos de vista de Empresa e Informação, consegue definir as informações independente de computação, relativas ao negócio que ser quer automatizar com software, mas de forma encapsulada. Há um risco de compreensão incorreta do negócio.

A relação do PIM com o PSM é feito através de regras e conformidade dos modelos com estas regras. Esta organização permite que os modelos sejam utilizados em um processo formal de desenvolvimento de software.

Vale entender quais são as regras para o PIM:

- Todo sistema deve ter um caso de uso descrevendo a função que será realizada;
- Todo objeto de informação deve estar relacionado com um caso de uso;
- Os objetos de informação não devem representar classes de programação neste momento;
- Todo caso de uso deve ter pelo menos um objeto computacional responsável pela sua realização;
- Todo caso de uso deve ter um diagrama de sequencia representando o fluxo dos objetos computacionais, seja primário ou alternativo.

3. Proposta principal

Com base nas definições, serão apresentados os dois primeiros modelos sugeridos pelo MDA, o modelo independente de computação (CIM) e modelo independente de plataforma (PIM), em forma de processo e posteriormente será apresentada a experimentação através de um sistema web de publicação de notícias.

3.1. Introdução

A figura 3.1 a seguir, demonstra quais elementos compõem o modelo independente de computação e o modelo independente de plataforma, bem como quais visões do modelo de referência ODP fazem o mapeamento entre os dois modelos.

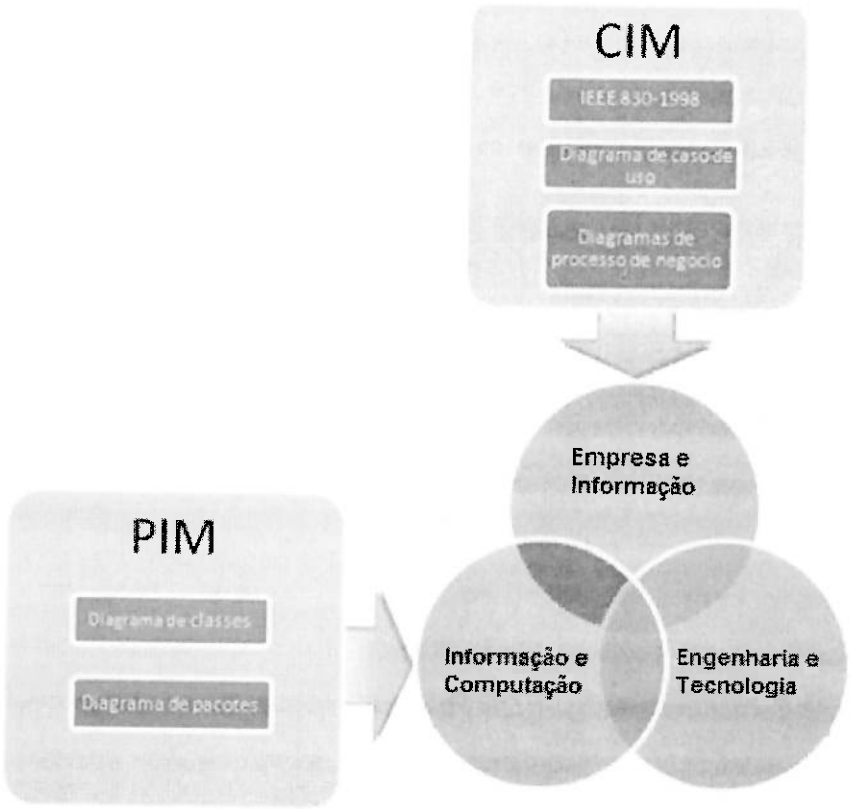

Figura 3.1 – Representação dos modelos CIM e PIM do MDA com visões RM-ODP

Na representação, o CIM tem três elementos, a saber: documento inicial baseado na especificação de exigências de software(ESS) IEEE-830, diagrama(s) de casos de uso e diagrama(s) de processo de negócio.

O objetivo do documento inicial é auxiliar a formalizar a ideia que se tem do negócio e ambiente no qual o software será inserido. Além disto, lista as funcionalidades do futuro sistema, especificadas em seguida com um grau maior de detalhes nos casos de uso. O diagrama de processo de negócio completa o entendimento do negócio, a medida que retrata de uma forma ampla o funcionamento do negócio. Tal modelo está intimamente relacionado com as visões de Empresa e de Informação, tal qual a proposta MDA4WS e método Maat.

O PIM tem dois elementos, a saber: o diagrama de classes e o diagrama de pacotes. O diagrama de classes está focado em identificar quais são os elementos que compõem o sistema e como estes elementos são utilizados. Já o diagrama de pacotes decompõe os elementos do diagrama de classe em estruturas independentes.

A relação entre os dois modelos é feita através da interseção no diagrama de Venn, cujos conjuntos de elementos representam os pontos de vista empresa, informação e computação. Esta interseção é representada neste trabalho pelas premissas, definidas a seguir. As premissas são pontos de observação que por inferência geram funções, ou seja, relacionam elementos do CIM e PIM.

3.2. Detalhamento dos modelos e premissas

Com os elementos dos dois primeiros modelos do MDA apresentados, o próximo passo é a definição das características de cada elemento e como eles devem ser produzidos, bem como quais premissas deve-se seguir para produzir o PIM a partir dos elementos do CIM.

O nível de detalhamento fica a critério do entendimento do negócio e do sistema, isto é, caso a complexidade seja maior, um refinamento maior pode ser dado ao modelo. O roteiro a seguir, tem elementos que podem ser mais detalhados.

3.2.1. Características do CIM

Será apresentado neste modelo um diagrama de processo de negócio e uma documentação leve do sistema que pretende-se fazer.

a) Especificação dos requisitos

Como mencionado anteriormente, por uma questão prática, o documento segue alguns elementos do padrão IEEE 830-1998, já que trata-se de um guia de práticas para especificação de requisitos de software ou Especificação de Exigências de Software (EES). Na sua essência, este documento deve ser capaz de responder os questionamentos:

- Qual é o objetivo do software que atenderá o negócio em questão;
- Metas, objetivos e benefícios relevantes para o domínio do problema.

Representando o ponto de vista da empresa (RM-ODP), os elementos sugeridos para este documento são:

i. Itens:

- Propósito (Seção 1.1 do documento de EES)
    - Delinear o propósito do documento.
- Âmbito (Seção 1.1 do documento de EES)
    - Identificar o produto de software pelo nome;
    - Explicar o que o produto fará e o que não fará;
    - Definir a aplicação do software, incluindo os benefícios relevantes.
- Funções do produto (Secção 2.2 do documento de EES)

- Fornecer um resumo das principais funções que o software vai desempenhar.

ii. Participantes sugeridos: Arquiteto de software, analista de negócio e cliente.

iii. Formato sugerido: Documento texto.

b) Definição das regras de negócio

Os requisitos especificam as condições exigidas para se alcançar o propósito de um SI e portanto tem o mesmo valor semântico de regras de negócio (GUILARDUCCI, 2007). Entendendo que são fundamentais para o sistema, vale separar em um documento à parte.

i. Itens:

Deve contemplar os elementos:

- Políticas regulatórias;
- Criticidade;
- Obrigações contratuais;
- Decisões estratégicas;
- Cálculos específicos;
- Leis e regulamentações.

ii. Participantes sugeridos: Arquiteto de software, analista de negócio e cliente.

iii. Formato sugerido: Documento texto, em formato de tabela. Como no exemplo:

Regra de negócio	Descrição	Criticidade
[Rn01]	Data deve obedecer	Alta

	padrão dd/mm/yyyy	

Tabela 3.1 – Exemplo de regras de negócio

c) Diagrama de caso de uso

O objetivo do diagrama é descrever o cenário estático, mostrando as funcionalidades do sistema do ponto de vista do usuário. Representa o ponto de vista da empresa.

i. Itens:

Os seguintes elementos descritivos devem aparecer no diagrama:

- atores;
- casos de uso;
- relacionamentos entre estes elementos.

Estes relacionamentos podem ser:

- associações entre atores e casos de uso;
- generalizações entre os atores;
- generalizações, extends e includes entre os casos de uso.

ii. Participantes sugeridos: Analista de negócio e cliente

iii. Formato sugerido: Diagrama UML geral e dependendo da complexidade (se o analista julgar necessário), um documento em forma de texto para detalhar, com o formato:

Atores:	Atores envolvidos no caso de uso
Pré-condição:	Estado do sistema antes da execução do fluxo
Fluxo de eventos:	Fluxo primário de eventos, incluindo as regras de negócio quando necessário.

Pós-condição:	Estado do sistema após a execução do fluxo

Tabela 3.2 – Exemplo de detalhamento de caso de uso

d) Modelo de processos de negócio

Os modelos de processo de negócio, dizem em linhas gerais como é a sequência de atividades cuja meta é atingir o objetivo do negócio. Como o entendimento precisa ser claro para todos, a notação BPMN deve ser utilizada. Para simplificar, a técnica de separar em processos menores distintos, para depois serem agrupadas em um modelo colaborativo, pode ser adotada. Este diagrama de fluxo é a mais importante ferramenta deste modelo por representar a semântica do sistema, ou o ponto de vista da informação. Quando um item não for claro o suficiente para os envolvidos o diagrama de fluxo deve ser refeito.

i. Participantes sugeridos: Analista de negócio e cliente
ii. Formato sugerido: Diagrama BPMN.

Ferramentas como o BizAgi Process Modeler (BizAgi, 2010) ou Intalio BPMS (Intalio, 2010), são úteis por serem compatíveis com a notação e por auxiliarem a criar o diagrama. Em um primeiro momento não é necessário, mas estas ferramentas podem servir para instanciar o modelo PSM diretamente em forma de webservices para avaliar a consistência do modelo.

Com as características definidas o modelo, resumido é:

Ponto de vista RM-ODP	**CIM**
Empresa	• Especificação dos requisitos • Documento regras de negócio
Empresa	• Diagrama de caso de uso
Informação	• Documento do processo de negócio

Tabela 3.3 – Resumo do CIM

3.2.2. Premissas

As premissas são proposições, baseadas nos conceitos dos pontos de vista do RM-ODP, que permitem chegar a conclusões. Isto é, através da observação de uma característica de um elemento do modelo independente de computação, permita definir como deve ser uma característica de um elemento do modelo independente de plataforma. Além das visões, as melhores práticas de UML e BPMN fazem parte da criação das premissas.

No caso do CIM, o elemento para análise é o diagrama BPMN, por conter os conceitos dos elementos anteriores – deve obedecer as regras de negócio, deve executar os casos de uso e deve respeitar a descrição inicial. No caso do PIM, o elemento mais representativo escolhido é o diagrama de classes, já que o diagrama de pacotes é o próximo passo. Vale ressaltar que apesar dos modelos escolhidos, os outros não devem ser desconsiderados. São inclusive parte da análise.

1ª Premissa

Definição: Se uma raia do diagrama de processos de negócio (organização de uma entidade de negócio) for representada no diagrama de caso de uso por um ator (o ator executa todas as atividades de uma organização de negócio) pode ser gerada uma classe para representar (objeto computacional) a interação com o sistema. Se houverem mais de uma raia, é possível utilizar mais de uma classe ligadas por generalização.

Correspondência: Ponto de vista da empresa e informação tem ligação com ponto de vista da computação.

Restrições: A premissa é válida apenas quando a organização de uma entidade do negócio fizer sentido como papel no sistema de informação.

2ª Premissa

Definição: Se um artefato for atualizado por uma atividade e esta for repetitiva (informação com estado dinâmico), por conseguinte, é possível criar uma classe para representar a informação (objeto computacional, com métodos que alteram estes valores).

Correspondência: Ponto de vista da informação tem ligação com ponto de vista da computação.

Restrições: Esta premissa não é válida para diagramas com realimentação.

3ª Premissa

Definição: Da mesma forma que a segunda premissa, quando houver um artefato gerado e inalterado no diagrama de negócio (informação estática), é possível representá-lo com uma classe (objeto computacional). Contudo, sua relação com as demais classes depende da análise do diagrama de casos de uso e das regras de negócio.

Correspondência: Ponto de vista da informação tem ligação com ponto de vista da computação.

Restrições: Se o artefato for apenas uma informação de controle do processo, não há necessidade de representação no diagrama de classes. Mas esta informação estática de controle, deve ser levada em conta em um documento a parte.

4ª Premissa

Definição: Troca de mensagens entre piscinas diferentes (entidades com diferentes objetivos) do processo de negócio, podem ser representadas por interfaces entre classes (interfaces requeridas e fornecidas), no diagrama de classes. Para facilitar o entendimento, estes elementos podem dividir conjuntos de classes no diagrama de pacotes.

Correspondência: Ponto de vista da informação tem ligação com ponto de vista da computação.

Restrições: A premissa não é válida para realimentações.

5ª Premissa

Definição: Atividades do diagrama de processo de negócio que remetam a ideia de interface de comando e que foram descritas no caso de uso desta forma, sugere a criação de classes abstratas com estereótipo <<boundary>> e <<control>>, no diagrama de classes.

Correspondência: Ponto de vista da empresa e informação tem ligação com ponto.

Restrições: Premissa válida para casos específicos em que é necessário uma interface com o usuário ou com outro sistema. É necessário análise das regras de negócio e do objetivo do sistema de informação.

3.2.3. Características do PIM

Neste modelo serão apresentados diagramas UML que representam a arquitetura. As premissas do ítem anterior são o ponto de análise para transformar o CIM em PIM.

a) Diagrama de classe

Tem foco nas principais interfaces da arquitetura, nos principais métodos, e não como eles irão ser implementados. É uma representação fiel dos conceitos do domínio em estudo. Representa o ponto de vista da informação e computação.

Além disto, representa o ponto de vista independente de plataforma, à medida que reflete a estrutura do sistema, como bem referencia o MDA4WS.

Os seguintes itens são necessários:

i. Participantes sugeridos: Arquiteto de software e analista de sistema

ii. Formato sugerido: Diagrama UML, de preferência na versão 2.0, com a possibilidade de representar interfaces de uma forma mais abstrata.

b) Diagrama de pacotes

O diagrama de pacotes descreve como os elementos estão organizados em pacotes e demonstra de certa forma a dependência entre eles. Pode ser utilizado tanto para descrever um estilo de estrutura arquitetural, quanto divisões do sistema em questão. Ele representa o ponto de vista da computação.

Ele deve ser implementado seguindo:

i. Participantes sugeridos: Arquiteto de software e analista de sistema

ii. Formato sugerido: Diagrama UML.

Com os elementos sugeridos para o PIM, tem-se o resumo:

Ponto de vista RM-ODP	**PIM**
Informação/Computação	• Diagrama de classes
Computação	• Diagrama de pacotes

Tabela 3.6 – Resumo do PIM

A análise do CIM, através das premissas, auxilia na criação dos elementos do diagrama de classes do PIM. Se constatado que as restrições não impeçam este

caminho de modelagem (do CIM ao PIM), a simplicidade torna a modelagem rápida, facilitando a implementação. Se necessário, documentos alternativos podem ser gerados para especificar melhor o estado da informação, levando em conta as regras de negócio do CIM.

4. Experimentação

4.1. Contexto

Para demonstrar a criação do modelo independente de computação e do modelo independente de plataforma, com o propósito de experimentar as premissas de mapeamento, o contexto utilizado é um sistema web de publicação de notícias. Este tipo de sistema atende o requisito de negócio de uma empresa que atua no ramo de editoração, cuja principal atividade, é veicular notícias. Cada segmento da empresa atende um nicho específico de publicação e portanto requer sistemas pequenos, simples, mas com alto grau de especificidades (detalhes de interface, nomenclaturas e regras de negócio).

4.2. Modelo CIM do publicador de notícias

a) Especificação dos requisitos

1. Propósito

O objetivo deste documento é prover uma ideia inicial do sistema de informação que será desenvolvido para publicação de notícias, em ambiente web.

2. Âmbito

O sistema de informação de publicação web, recebe o nome de SPN (Sistema de Publicação de Notícias). Um sistema web de publicação de notícias é um sistema de informação para a publicação de textos (notícias, análises e informações) em diversas meios. O conceito de meio, neste caso, não está relacionado as tecnologias de implementação e sim as formas como o leitor visualiza esta informação, podendo ser um site de internet, um dispositivo móvel, ou mesmo aplicações de empresas associadas, que coletam estas informações (agência de notícias, por exemplo).

O SPN é relevante a medida que não há publicadores disponíveis no mercado que possuam a exibição de informações específicas antes de exibir o editor de textos propriamente dito.

3. Funções do produto

O sistema de publicação deve executar as seguintes funções:

- Coletar informações de calendário, notícias, anúncios e métricas e exibir na forma de texto;
- Prover área de criação de texto, com editor do tipo RichText com campos: autor, título, subtítulo, texto e data
- Prover possibilidade de inclusão de conteúdo multimídia associado ao texto (foto, vídeo e arquivo), com campos: autor e descrição;
- Classificação dos textos criados por categorias;
- Texto pode ser revisado por editor, no mesmo ambiente, antes de ser publicado;
- A publicação pode ser imediata ou agendada para data futura, com o layout/template desejado(meio de leitura);
- Controle de acesso por senha. Somente o editor pode cadastrar usuários novos.

A parte de publicação e coleta de informações não terá acesso direto. O SPN será acessível via web pelos jornalistas e editores.

b) Regras de negócio

A tabela abaixo apresenta algumas regras de negócio. Durante a criação do PSM, pode ser necessário uma revisão e refinamento, mas não faz parte do objeto em análise neste trabalho.

Regra de negócio	Descrição	Criticidade
[Rn01]	Data deve obedecer padrão dd/mm/yyyy	Alta
[Rn02]	Campos título, autor, data e categoria são obrigatórias	Alta

[Rn03]	Título não deve exceder 140 caracteres	Média
[Rn04]	Campo autor deve permitir mais de um autor	Média
[Rn05]	Uma nota autorizada pelo editor, não deve aparecer para o jornalista	Média
[Rn06]	Notícia deve ter pelo menos uma categoria associada	Alta
[Rn07]	As categorias são fixas e imutáveis	Alta
[Rn08]	As notícias devem ser publicadas de acordo com a chegada na fila de publicação	Média

c) Diagrama de caso de uso

Abaixo todas as funções necessárias para executar as atividades mapeadas. É importante ressaltar, que os atores Agente coletor e Agente publicador, são agentes internos que seguem uma programação pré-estabelecida, a não ser que alguma função faça uma solicitação, portanto estão definidos dentro da fronteira do sistema.

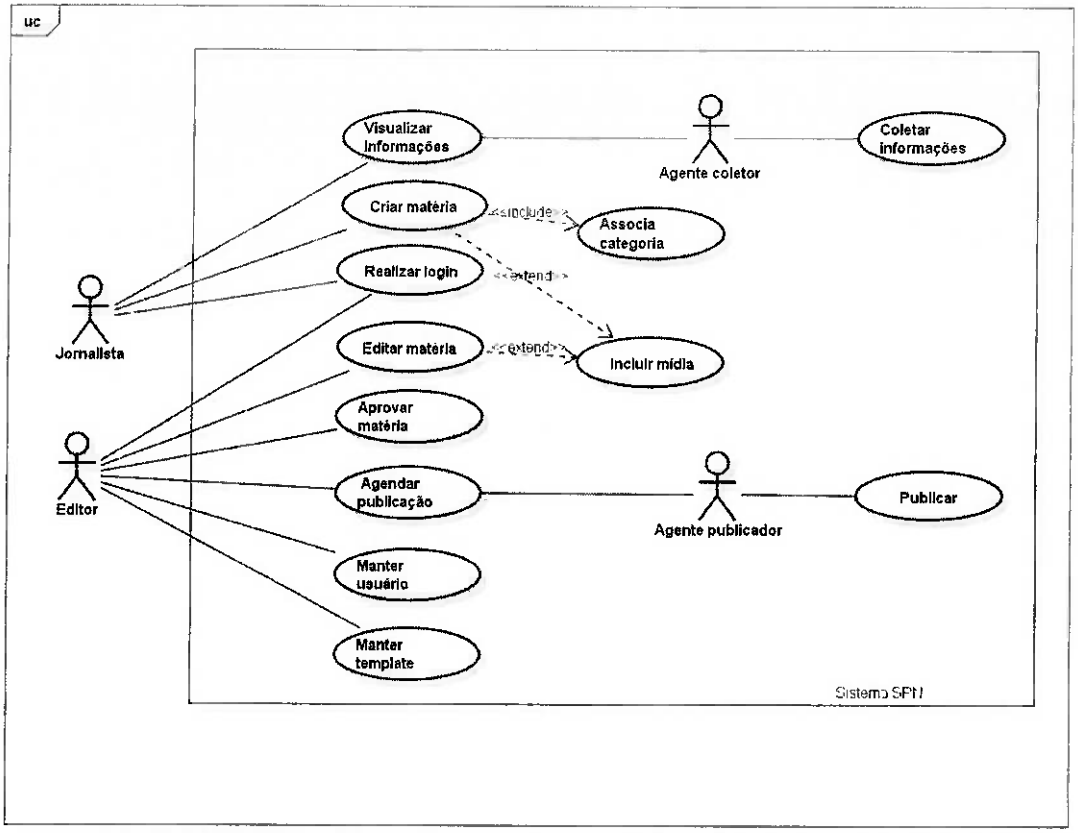

Figura 3.4 – Diagrama de caso de uso do sistema publicação web

d) Modelo de processos de negócio

Com os elementos mapeados anteriormente e na presença de todos os envolvidos, o modelo é construído para representar o negócio que o sistema de informação quer atender.

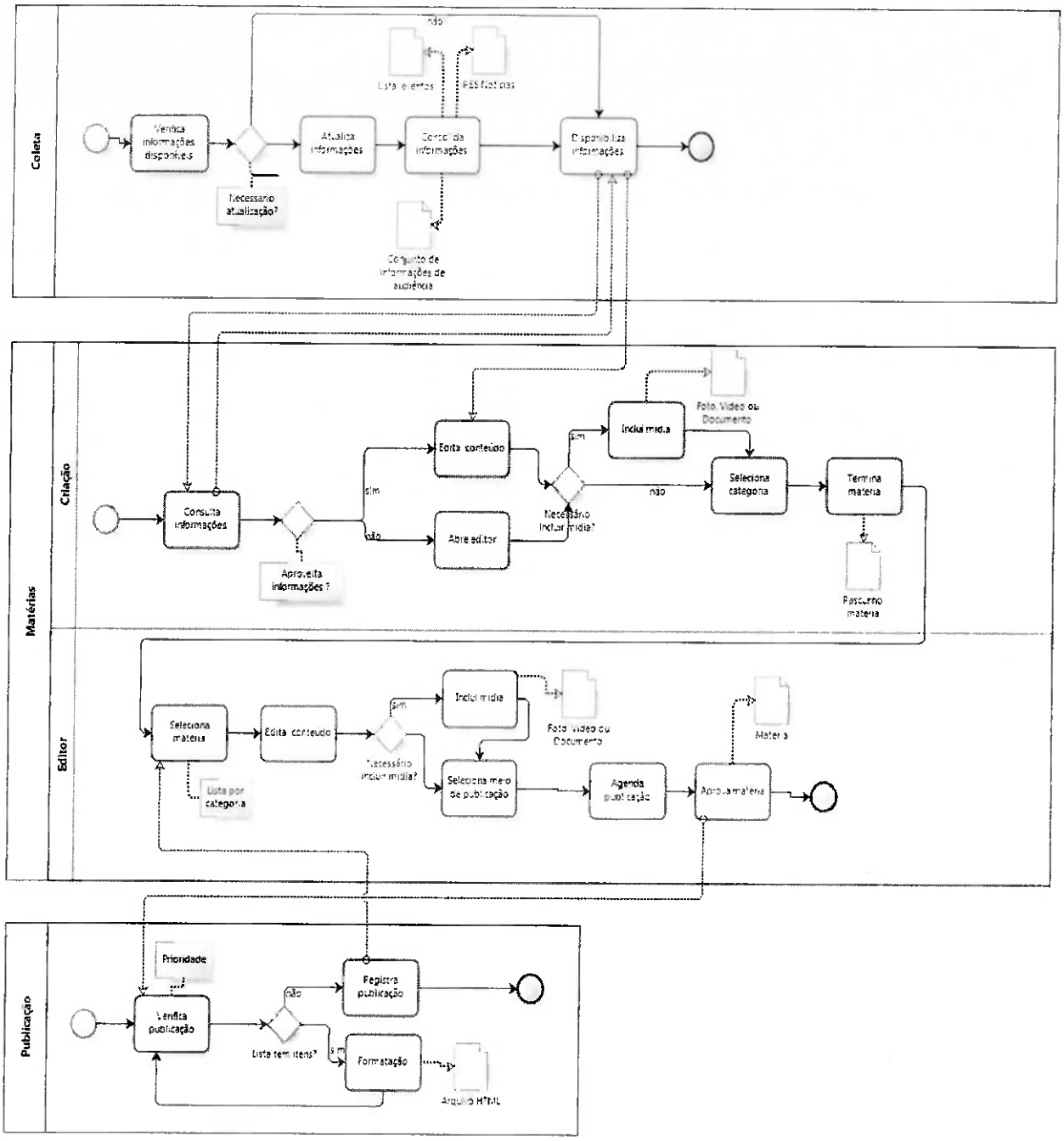

Figura 3.2 – Processo de negócio sistema publicação web

A piscina coleta define qual é o fluxo de coleta de informação de terceiros e como ela interage com a criação de uma matéria a ser publicada. De forma semelhante, a piscina matérias define o fluxo correto para produção e armazenamento de conteúdo, com processo de revisão e aprovação. E a última

piscina define o fluxo para disponibilização do conteúdo em diversos meios, de acordo com o público alvo.

Uma obsercação pertinente: No diagrama de processos de negócio, os textos descritivos da raia Editor (regra de negócio 6, do ítem 4.2 b) e raia Publicação (regra de negócio 8, do item 4.2b) são restrições.

4.3. Aplicação das premissas para gerar o PIM

A aplicação das premissas, conforme descrito anteriormente, está na observação dos elementos e possível conclusão.

Da premissa 1, observando no modelo de processo de negócio, as raias da piscina matérias manipulam notícias ou matérias (informação). Também há representação relacionada no diagrama de caso de uso. Por inferência criamos a classe usuário e por generalização as classes jornalista e editor.

Pela premissa 2, a atividade de incluir mídia e editar matéria são repetitivas e a informação (notícia ou matéria, documento, foto e vídeo) é um atributo mutável durante o processo. Logo, criamos duas classe, a matéria e mídia, para representar o objeto que vai manipular esta informação.

Pela premissa 3, analisamos informações com atributos que não mudam durante o processo de negócio, verifica-se que as categorias de uma notícia são informações estáticas. Por inferência, podemos criar a classe categoria.

A premissa 4, sugere uma divisão lógica entre três processos: coleta, edição e publicação. Neste caso como o sistema irá automatizar todo o processo de negócio, vale a criação de classes com o estereótipo interface.

Pela premissa 5, analisam-se as atividades de disponibilizar informações a consultar as informações. Isto remete a ideia de uma interface de interação. Como neste caso trata-se de um sistema web, cujos atores utilizarão para exercer as atividades, criamos duas classes com estereótipo <<boundary>> e <<control>>.

a) Diagrama de classes

Com as classes sugeridas pelas aplicação das premissas (item 4.3), é possível obter correspondência entre os elementos, cujo diagrama é representado pela figura 3.3, a seguir:

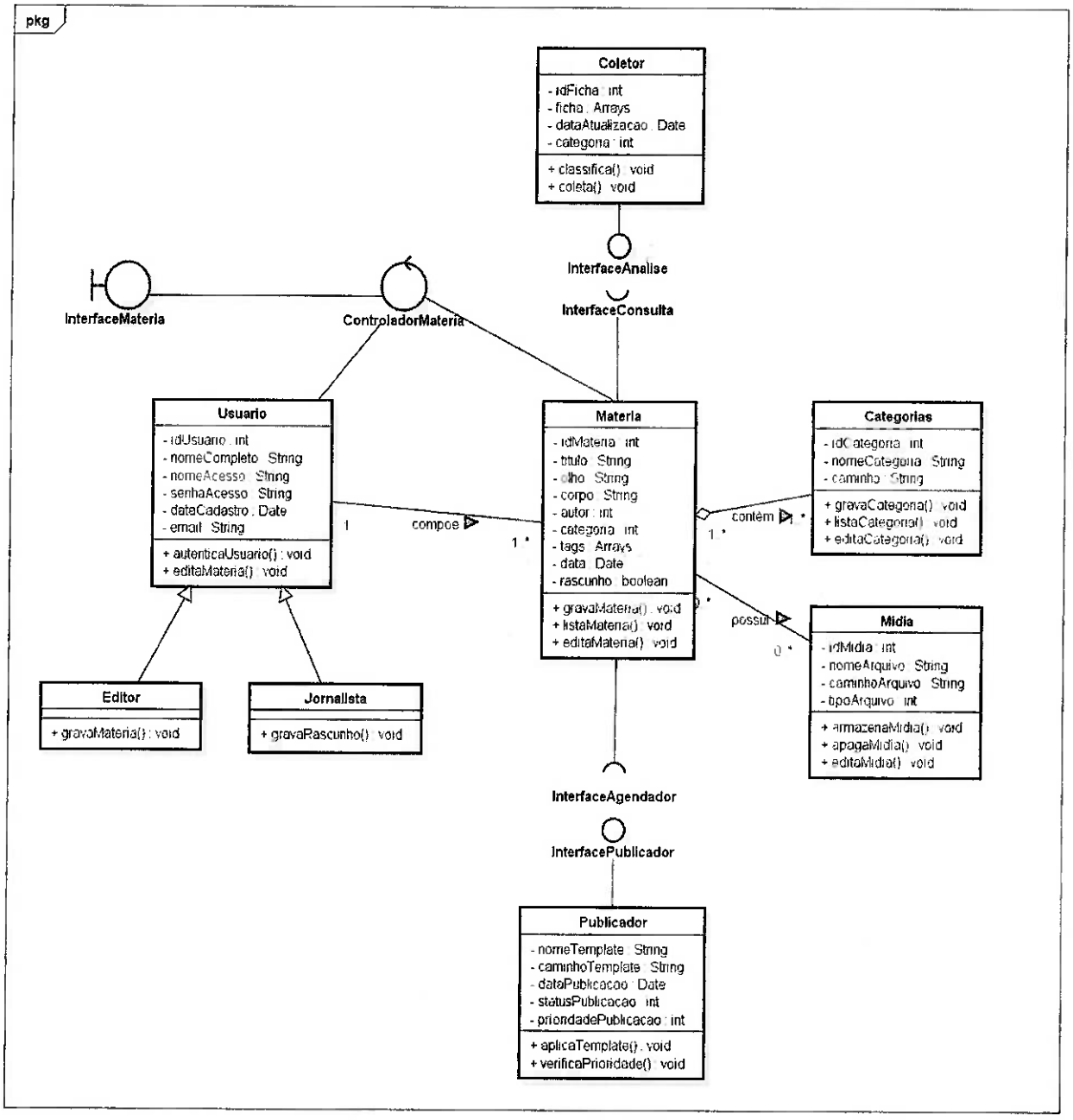

Figura 3.3 – Diagrama de classes do sistema publicação web

b) Diagrama de pacotes

Respeitando a premissa 4, a troca de mensagens entre piscinas distintas no diagrama de processos de negócio, definem pacotes de classes, no diagrama de pacotes.

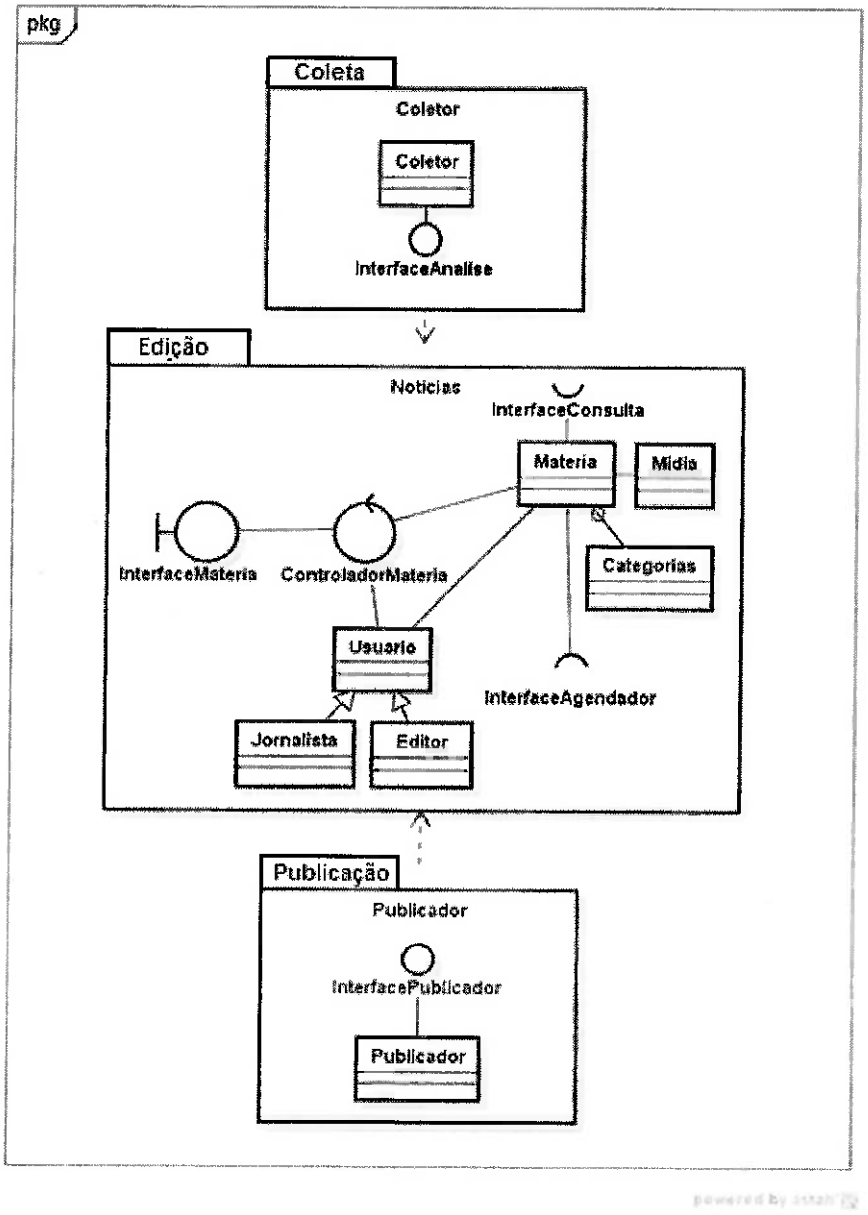

Figura 3.5– Diagrama de pacotes do sistema publicação web

4.4. Análise de resultados

O mapeamento baseado em visões facilita a transição principalmente entre os elementos principais dos dois modelos, o independente de computação e o independente de plataforma. As premissas são um bom indicativo do que analisar, mas nem sempre podem ser adotadas. É preciso ainda analisar o diagrama de casos de uso e principalmente as regras de negócio.

Vale mencionar que um refinamento, baseado ainda na descrição do processo de negócio e do sistema que se quer criar, é de fundamental importância

O diagrama de pacotes, poderia ser facilmente dividido em apresentação, aplicação e dados. Sendo que no último estariam as classes que fariam a persistência dos dados das classes com o estereótipo de entidade, subentendidas no diagrama de classes. Tal fato acarretaria em um estilo arquitetural de três camadas. Mas é bom salientar que nem sempre a divisão das piscinas do processo de negócio sugere uma divisão lógica do sistema.

Em termos de simplicidade o modelo tem poucos elementos e são fáceis de interpretar. Logo, a abordagem pode ser implementada rapidamente. Fato que trás a agilidade necessária para o negócio.

5. Considerações finais

5.1. Conclusão

É possível fazer o mapeamento entre dois domínios de abstração diferentes, uma no domínio do problema e outra no domínio da solução, utilizando os pontos de vista do RM-ODP, por dividir o objeto em análise, além da divisão de domínios proposta.

O nível de detalhamento proposto nos modelos que representam estes dois domínios vale apenas para um sistema de menor complexidade. Em um sistema maior, com mais componentes, o nível de detalhe proposto serviria apenas como etapa inicial, devendo ser refinado para atingir o modelo. A complexidade dos diagramas de negócio poderiam ser também um impedimento no uso deste modelo simplificado.

O experimento possibilita o entendimento de como é a aplicação do método simplificado, com a transformação baseada nas premissas apresentadas. Contudo, pelo caracter simplista, as restrições das premissas não são tratadas e inviabiliza a aplicação em sistemas e negócios mais complexos.

Ao contrário do observado em alguns trabalhos, aonde o CIM é incorporado ao PIM (FERREIRA,2006)(GERVAIS,2002), este trabalho articula uma visão única entre processo de negócio e o modelo do sistema, mitigando de certa forma a possibilidade de erros de interpretação, durante o processo de desenvolvimento de software.

A evolução natural da modelagem simplificada pode ser feita com a utilização das visões do RM-ODP restantes, a da Engenharia e a de Tecnologia, para definir o modelo específico de plataforma. Isto envolveria a escolha das linguagens, hardware necessário, fases da implementação e principalmente detalhamento melhor dos elementos do PIM. Estes elementos na sua concepção envolvem elementos criados

com as outras visões, logo o modelo PSM também incluiria alguns destes elementos.

Estender para o PSM e refinar os modelos, também no sentido de completar o processo formal de desenvolvimento de software, garantindo rastreabilidade e reuso de software.

6. Referências bibliográficas

1. OMG, **Business Process Maturity Model (BPMM), version 1.0**, <http://www.omg.org/spec/BPMM/1.0/PDF>. Junho. 2008.

2. OMG, **Using BPM and SOA to achieve the Agile Enterprise**, <http://www.bpm-consortium.org/120407_BPM_SOA_Agile_Enterprise.pdf>. Acessado em Setembro de 2010.

3. OMG, **BPMN v1.2**, <http://www.omg.org/spec/BPMN/1.2/PDF/>, Janeiro. 2009.

4. MELLOR, S. J. et al. **MDA destilada: Princípios de Arquitetura Orientada por Modelos,** Editora Ciência Moderna. 2005.

5. JACOBSON, I.; BOOCH, G.; RUMBAUGH,J. **The Unified Software Development Process**. Addison Wesley. 1998.

6. FERREIRA, A. P., **Utilização do MDA em conjunto com ODP em soluções distribuídas.** 2006. 105p. Dissertação (MBA) – Escola Politécnica, Universidade de São Paulo. 2006

7. ELIZONDO, J. D.C., **Aplicabilidade de MDA em Projeto de Web Services**, 2006, 98p. Dissertação (Mestre Engenharia Computação), Instituto de Pesquisas Tecnológicas do Estado de São Paulo, Universidade de São Paulo. 2006.

8. KARDOS, MARTIN e DROZDOVÁ, MATILDA, **Analytical Method of CIM to PIM Transformation in Model Driven Architecture (MDA)**, Apresentado em ASWEC. <http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4483222>. 2008.

9. RECKER, JAN, **BPMN Modeling – Who, Where, How and Why,** BPTRENDS, <www.bptrends.com>. 2008.

10. BASS, Len; Clements, Paul; Kazman, Rick; **Software Architecture in Practice**, Second Edition. 2. ed. Boston, Addison Wesley. 2003.

11. OMG, Object Management Group, **MDA Guide v1.01,** 2003. <http://www.omg.org/cgi-bin/doc?omg/03-06-01>, Acessado em Setembro de 2010.

12. Institute of Electrical and Electronics Engineers, Inc. **IEEE Recommended Practice for Architectural Description of Software-Intensive Systems**, 2000.

13. KENT , ALLEN, **Encyclopedia of library and information science**, Volume 63, Marcel Dekker, Inc., 1998.

14. GUILARDUCCI DE A., GEOFLÁVIA, **Uma Abordagem para Tratamento de Regras de Negócio em Sistemas de Informação**, Instituto de Informática da Universidade Federal de Goiás, 2007.

15. WHITE, STEPHEN A., **Introduction to BPMN**, IBM Corporation, 2006

16. Intalio BPMS, <http://www.intalio.com/bpms>, Acessado em Novembro 2010.

17. BizAgi Process Modeler, <http://www.bizagi.com>, Acessado em Novembro 2010.

18. GERVAIS,M-P. **Towards an MDA-Oriented Methodology**. Laboratory d'Informatique de Paris. COMPSAC'02, 2002.

19. KOK, R. L. K. et al, **Business Process Management (BPM) Standards: A Survey**, Business Process Management Journal, Emerald Publishing, Vol. 15 No. 5, Pp. 744-791, 2009.

20. Sparx System, **Whitepaper: The Business Process Model**, <http://www.sparxsystems.com.au/resources/>, 2007

21. ITU-T X.903, ISO/IEC 10746-3, **Information Technology – Open Distributed Processing – Reference Model – Architecture v01.03**, 2010

22. BECERRA, J. L. R. ***Aplicabilidade do Padrão de Processamento Distribuído e Aberto nos Projetos de Sistemas Abertos de Automação.*** Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo, 1998.