UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

ESTUDO DO PADRÃO PARA INTERCÂMBIO ELETRÔNICO DE DADOS ebXML

FABRICIO JOSÉ REIF

BLUMENAU
2004

2004/2-15

FABRICIO JOSÉ REIF

ESTUDO DO PADRÃO PARA INTERCÂMBIO ELETRÔNICO DE DADOS ebXML

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Sérgio Stringari - Orientador

BLUMENAU
2004

2004/2-15

ESTUDO DO PADRÃO PARA INTERCÂMBIO ELETRÔNICO DE DADOS ebXML

Por

FABRICIO JOSÉ REIF

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Sérgio Stringari – Orientador, FURB
Membro:	Prof. Francisco Adell Péricas, FURB
Membro:	Prof. Alexander R. Valdameri, FURB

AGRADECIMENTOS

Ao meu orientador Prof. Sérgio Stringari e aos amigos da DIATIVA INFORMÁTICA.

A minha namorada pela sua ajuda e compreensão pelos momentos que tive que me ausentar durante o desenvolvimento deste trabalho.

A minha família e em especial para meus pais Edmilson J. Reif e Maria M. F. Reif que possibilitaram que eu realizasse este sonho.

Agradeço a todos aqueles que indiretamente contribuíram para o desenvolvimento deste trabaho.

RESUMO

Este trabalho visa o estudo do padrão para intercâmbio eletrônico de dados ebXML. E para consolidar o estudo é proposto o desenvolvimento de um protótipo de registro/repositório utilizando *web services* para a realização de intercâmbio eletrônico de dados entre parceiros comerciais através da internet, utilizando o padrão ebXML. O desenvolvimento do protótipo segue a especificação do ebXML para o desenvolvimento de aplicações *Registry/Repository*, possibilitando que empresas informem seus perfis comerciais, o descobrimento de seus negócios por outras empresas e assim iniciarem a troca de informações.

Palavras chaves: ebXML; EDI; B2B.

ABSTRACT

This work seeks the study of the pattern for electronic data interchange ebXML. And to consolidate the study the development of a registry/repository prototype it is proposed using web services for the accomplishment of electronic data interchange among commercial partners through the internet, using the pattern ebXML. The development of the prototype follows the specification of the ebXML for the development of applications Registry/Repository, making possible that companies inform their commercial profiles, the discovery of their businesses for other companies and they begin the change of information.

Key-Words: ebXML; EDI; B2B.

LISTA DE ILUSTRAÇÕES

Quadro 1 – Exemplo de uma mensagem EDIFACT 16
Quadro 2 – Exemplo de uma mensagem X12 17
Quadro 3 – Exemplo de uma mensagem XML 18
Quadro 4 – Exemplo da mensagem XML em formato X12 19
Quadro 5 – Exemplo de uma mensagem SOAP 20
Figura 1 – Visão geral do ebXML 24
Figura 2 – Metodologia de modelagem recomendada para ebXML 26
Figura 3 – Representação detalhada do BOV 28
Figura 4 – Visão Funcional de Serviço (FSV) 30
Figura 5 – Fase de Implementação 31
Figura 6 - Fase de Descoberta e Recuperação 32
Figura 7 - Fase de Execução 32
Figura 8 - Visão geral do CPA 34
Figura 9 – Meta modelo ebXML 35
Figura 10 – Visão geral do Registro ebXML 37
Figura 11 – Cadastro de um novo usuário no programa GoXML Registry 38
Figura 12 – Arquitetura do Registro ebXML 39
Figura 13 – Formas de acesso ao registro ebXML 40
Figura 14 - Diagrama de classes do modelo de informações do registro ebXML 41
Figura 15 - Diagrama de classes demostrando a hierarquia entre as classes 42
Figura 16 – Exemplo de associações entre dois objetos do registro 55
Figura 17 – Exemplo de um caso de associação intramural 56
Figura 18 – Exemplo de um caso de associação extramural 57
Quadro 6 – Exemplo de submissão de um esquema de classificação 62
Figura 19 – Estrutra em árvore de um esquema de classificação geográfica 63
Figura 20 – Serviço de mensagens ebXML 64
Figura 21 – Arquitetura do serviço de mensagens 65
Figura 22 – Estrutura de uma mensagem ebXML 66
Figura 23 – Diagrama de casos de uso 69
Figura 24 – Diagrama de classes 71
Figura 25 – Diagrama de seqüência SubmitObjects 73
Figura 26 – Diagrama de seqüência UpdateObjects 74
Figura 27 – Diagrama de seqüência AddSlots 74
Figura 28 – Diagrama de seqüência RemoveSlots 75
Figura 29 – Diagrama de seqüência ApproveObjects 75
Figura 30 – Diagrama de seqüência DeprecateObjects 76
Figura 31 – Diagrama de seqüência RemoveObjects 76
Figura 32 – Diagrama de seqüência SubmitAdhocQuery 77
Figura 33 – Arquitetura do protótipo 77
Quadro 7 – Arquivo asmx web service LifeCycleManager 81
Figura 34 – Visualização do arquivo *asmx* LifeCycleManager 81
Quadro 8 – Arquivo asmx web service QueryManger 82
Figura 35 – Visualização do arquivo *asmx* QueryManager 82
Figura 36 – Aplicação para testes do protótipo ebXML 83
Figura 37 – Tela submissão da requisição de teste 84
Figura 38 – Resposta da requisição SubmitObjectsRequest 85
Quadro 9 – Requisição de teste FilterQuery 85
Figura 39 – Resposta da requisição AdhocQueryRequest 86

LISTA DE TABELAS

Tabela 1 – Atributos da classe RegistryObject 43
Tabela 2 – Valores pré-definidos para o tipo de objeto do registro 44
Tabela 3 – Métodos da classe RegistryObject 44
Tabela 4 – Atributos da classe RegistryEntry 45
Tabela 5 – Valores pré-definidos do atributo stability 45
Tabela 6 – Valores pré-definidos do atributo status 46
Tabela 7 – Atributos da classe Slot 46
Tabela 8 – Atributos da classe ExtrinsicObject 47
Tabela 9 – Método da classe RegistryPackage 47
Tabela 10 – Atributos da classe ExternalIdentifier. 48
Tabela 11 – Atributos da classe ExternalLink 48
Tabela 12 – Método da classe ExternalLink 48
Tabela 13 - Atributos da classe AuditableEvent 49
Tabela 14 – Valores pré-definidos para o atributo eventType 49
Tabela 15 – Atributos da classe User 50
Tabela 16 – Atributos da classe Organization 50
Tabela 17 – Atributos da classe PostalAddress 51
Tabela 18 – Método da classe PostalAddress 51
Tabela 19 – Atributos da classe TelephoneNumber 51
Tabela 20 – Atributos da classe PersonName 52
Tabela 21 – Método da classe Service 52
Tabela 22 – Atributos da classe ServiceBinding 53
Tabela 23 – Método da classe ServiceBinding 53
Tabela 24 – Atributos da classe SpecificationLink 54
Tabela 25 – Atributos da classe Association 58
Tabela 26 – Valores pré-definidos para o atributo associationType 59
Tabela 27 – Atributos da classe ClassificationScheme 60
Tabela 28 – Valores pré-definidos para o atributo nodeType 60
Tabela 29 – Atributos da classe ClassificationNode 60
Tabela 30 – Atributos da classe Classification 61

LISTA DE SIGLAS

B2B – *Business to Business*

CPA – *Collaboration Protocol Agreement*

CPP – *Collaboration Protocol Profile*

DTD – *Document Type Definition*

DOM – *Document Object Model*

EDI – *Eletronic Data Interchang*

ebXML – *Eletronic Business XML*

HTTP – *Hipertext Transfer Protocol*

LM – *Life Cycle Management*

MIME – *Multiporpose Internet Mail Extension*

PIP – *Partner Interface Process*

QM – *Query Management*

RC – *Registry Client*

RIM – *Registry Information Model*

RPC – *Remote Procedure Call*

RSS – *Registry Services Specification*

SMTP – *Simple Mail Transport Protocol*

SOAP – *Simple Object Access Protocol*

UDDI – *Universal Description Discovery and Integration*

UML – *Unified Modeling Language*

UMM – *UM/CEFACT Modeling Methodology*

URI – *Uniform Resource Identifier*

URL – *Uniform Resource Locator*

VAN – *Value Added Network*

WSDL – *Web Service Description Language*

XML – *Extensible Marckup Language*

SUMÁRIO

1 INTRODUÇÃO 12
1.1 OBJETIVOS DO TRABALHO 13
1.2 ESTRUTURA DO TRABALHO 13
2 FUNDAMENTAÇÃO TEÓRICA 15
2.1 ELETRONIC DATA INTERCHANGE (EDI) 15
2.2 XML 17
2.3 WEB SERVICES E SOAP 19
2.4 EBXML (ELETRONIC BUSINESS EXTENSIBLE MARKUP LANGUAGE) 20
2.4.1 ARQUITETURA DO ebXML 22
2.4.2 METODOLOGIA DE MODELAGEM RECOMENDADA ebXML 25
2.4.2.1 Visão Operacional do Negócio (BOV) 27
2.4.2.2 Visão Funcional de Serviço (FSV) 29
2.4.3 FASES DO EBXML 30
2.4.3.1 Fase de Implementação 30
2.4.3.2 Fase de Descoberta e Recuperação 31
2.4.3.3 Fase de Execução 32
2.4.4 INFORMAÇÃO DOS PARCEIROS DE NEGÓCIO CPP e CPA's 33
2.4.5 PROCESSO DE NEGÓCIO E MODELAGEM DA INFORMAÇÃO 34
2.4.6 COMPONENTES CENTRAIS E FUNCIONALIDADES DA BIBLIOTECA DE COMPONENTES 36
2.4.7 REGISTRO ebXML 36
2.4.7.1 Classe RegistryObject 42
2.4.7.2 Classe RegistryEntry 44
2.4.7.3 Classe Slot 46
2.4.7.4 Classe ExtrinsicObject 47
2.4.7.5 Classe RegistryPackage 47
2.4.7.6 Classe ExternalIdentifier 47
2.4.7.7 Classe ExternalLink 48
2.4.7.8 Classe AuditableEvent 49
2.4.7.9 Classe User 49
2.4.7.10 Classe Organization 50
2.4.7.11 Classe PostalAddress 51

2.4.7.12 Classe TelephoneNumber 51
2.4.7.13 Classe PersonName 51
2.4.7.14 Classe Service 52
2.4.7.15 Classe ServiceBinding 52
2.4.7.16 Classe SpecificationLink 53
2.4.8 ASSOCIAÇÃO DE OBJETOS DO REGISTRO 54
2.4.8.1 Associação Intramural 55
2.4.8.2 Associação Extramural 56
2.4.8.3 Confirmação de uma associação 57
2.4.8.4 Classe Association 58
2.4.9 CLASSIFICAÇÃO DE OBJETOS DO REGISTRO 59
2.4.9.1 Classe classificationScheme 59
2.4.9.2 Classe ClassificationNode 60
2.4.9.3 Classe Classification 61
2.4.9.4 Exemplo de um esquema de classificação 61
2.4.10 SERVIÇO DE MENSAGENS DO ebXML 63
2.4.11 TRABALHOS CORRELATOS 66
3 DESENVOLVIMENTO DO TRABALHO 68
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO 68
3.2 ESPECIFICAÇÃO 69
3.2.1 CASOS DE USO 69
3.2.2 DIAGRAMA DE CLASSES 70
3.2.3 WEB SERVICE QueryManager (QM) 73
3.2.4 DIAGRAMA DE SEQÜÊNCIA 73
3.2.5 ARQUITETURA DO PROTÓTIPO 77
3.3 IMPLEMENTAÇÃO 78
3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS 78
3.3.2 PROTÓTIPO DE REGISTRO ebXML 79
3.3.3 OPERACIONALIDADE DA IMPLEMENTAÇÃO 82
3.4 RESULTADOS E DISCUSSÃO 86
4 CONCLUSÕES 88
4.1 EXTENSÕES 89
REFERÊNCIAS BIBLIOGRÁFICAS 90

APÊNDICE A – Classe ObjectRequestProtocol 92
APÊNDICE B – Classe SubmitObjectRequest 96
APÊNDICE C – Mensagem de teste do caso de uso SubmitObjects 100
ANEXO A – Exemplo de um CPP 105
ANEXO B – Exemplo de um CPA 108
ANEXO C – Exemplo de um Esquema de Especificação de Processos de Negócio 112
ANEXO D – Esquema XML para a troca de mensagens com o registro ebXML 116

1 INTRODUÇÃO

A crescente necessidade das empresas para uma economia global aberta fez com que utilizassem a computação nos processos de negócios para prover algum tipo de intercâmbio de dados. Com o uso em larga escala da internet as transações comerciais encontraram um novo ambiente para se realizarem, resultando num mercado sem barreiras e fazendo com que as empresas estendessem suas operações, surgindo assim a necessidade de criar padrões para controlar e normatizar tais operações.

O método tradicional utilizado por estas empresas para a troca de informações eletrônicas é uma tecnologia conhecida como *Eletronic Data Interchange* (EDI), em português Intercâmbio Eletrônico de Dados.

A troca de informações de transação comercial por meio eletrônico, entre duas ou mais empresas, é conhecida como comércio eletrônico ou *Electronic Commerce (e-Commerce)*. Os aplicativos de *e-commerce* incluem serviços ao consumidor via internet, sistemas de adminstração de relação com clientes através da web, aplicações de EDI, etc. Não importa o que está sendo vendido, o objetivo básico é obter algum tipo de mensagem/informação de uma empresa por outra empresa, de uma forma que signifique algo para ambas as partes. Esse conjunto de trocas, mais a troca de outras informações empresariais e organizacionais foi denominado de negócio eletrônico ou *Electronic Business (e-Business).* Os aplicativos *e-business* incluem (além dos aplicativos *e-commerce*), sistemas de *Enterprise Resource Planning* (ERP), marketing de relacionamento com os consumidores ou *Customer Relationship Management* (CRM), administração do conhecimento ou *Knowledge Management* (KM) e administração da cadeia de suprimentos ou *Supply Chain Management* (SCM). Tanto em aplicativos e-commerce ou quanto em aplicativos e-business o “e” significa substituir o papel, o trabalho humano pelo trabalho de computadores via redes eletrônicas.

O crescimento no uso da linguagem *Extensible Markup Language* (XML), em português Linguagem de Marcação Estendida, vem afetando todas as áreas da indústria de software, o que também vem ocorrendo com a implementação de aplicações EDI.

De acordo com Anderson et al. (2001), a XML irá cada vez mais ser a base para o intercâmbio, exibição, indexação de dados e assim por diante; ela encontrará seu caminho em quase qualquer aplicação e dispositivo. De telefones móveis a sistemas de entretenimento, de

casas a sistemas de satélites, a marcação baseada em XML será usada para permitir comunicações limpas (mais simples de compreender do que em outros padrões de EDI) e eficientes.

Para pesquisar e identificar técnicas de utilização da XML de forma normatizada e que possa ser compreendida globalmente foi criado o padrão *Electronic Business Extensible Markup language* (ebXML). O ebXML é um novo conceito em intercâmbio eletrônico de dados de negócios, que utiliza *tags* XML para o formato das mensagens entre empresas.

Um novo padrão sempre causa agitação no mercado, e com ebXML não é diferente. Um estudo para verificar sua aplicabilidade e vantagens sobre os modelos de EDI's existentes se torna necessária para as empresas que querem se manter atualizadas em relação ao mercado de intercâmbio eletrônico de dados.

1.1 OBJETIVOS DO TRABALHO

O objetivo principal deste trabalho foi o estudo do padrão para intercâmbio eletrônico de dados ebXML para assim desenvolver um protótipo utilizando *web services* para armazenamento de dados que permitirá duas empresas definirem seu perfil e depois realizarem o intercâmbio eletrônico de dados via internet, utilizando o padrão ebXML.

Os objetivos específicos do trabalho são:

a) o estudo do padrão para intercâmbio eletrônicos de dados ebXML;
b) criar um repositório de dados permitindo que empresas busquem e definam perfis para realizarem o intercâmbio eletrônico de dados (Registro/Repositório ebXML).

1.2 ESTRUTURA DO TRABALHO

O trabalho apresenta quatro capítulos, sendo que o primeiro aborda o assunto e os objetivos do trabalho.

O segundo apresenta os assuntos correlatos ao desenvolvimento do protótipo, sendo eles:

a) EDI – solução muito utilizada pelas empresas para efetuar o intercâmbio eletrônico de dados;

b) XML – linguagem de marcação para delimitar e padronizar dados;
c) *Web Services* e SOAP – tecnologias para o desenvolvimento de sistemas distribuídos;
d) ebXML – padrão baseado em XML para intercâmbio eletrônico de dados.

O terceiro capítulo apresenta a específicação, detalhes da implementação e também demonstra como utilizar o protótipo.

Finalmente o quarto capítulo apresenta as conclusões e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

A seguir são apresentados conceitos e tecnologias que estão relacionadas com o desenvolvimento do protótipo.

2.1 ELETRONIC DATA INTERCHANGE (EDI)

De acordo com Anderson et al. (2001, p. 658) o EDI existe há pelo menos trinta anos. Ele ajudou a preparar o caminho para o *e-Business*. Bem antes da Web e até mesmo da Internet atual, ele existia, empresas trocavam dados eletronicamente usando transações padronizadas para cada indústria. Muitas empresas nas indústrias automotiva, de alimentos e eletrônicos: Por exemplo, alcançaram economias significativas e melhoraram os processos corporativos usando o EDI.

EDI é um padrão estabelecido entre as empresas para a troca eletrônica de informações, por exemplo, financeiras, de fabricação, de estoque, técnicas e de transporte.

Segundo Anderson et al. (2001) o padrão mais conhecido e utilizado na América do Norte é o *American National Standardisation Institute* (ANSI) X12, que atualmente está sendo mantido pela *Data Interchange Standards Association* (DISA), uma associação que não visa o lucro. Já o resto do mundo está envolvido com a iniciativa *United Nations Electronic Data Interchange for Administration, Commerce and Transport* (UN/EDIFACT). Os modelos EDIFACT são supervisionados pelo *Centre for Facilitation of Administration, Commerce and Trade* (CEFACT) na *Economic Commission for Europe* (UNECE), em Genebra, Suíça.

Segundo Müller (1997, p. 13), o padrão aceito pela Associação Brasileira de Normas Técnicas (ABNT) como o padrão oficial para o Brasil é o EDIFACT, sendo o padrão internacional recomendado pela Organização das Nações Unidas (ONU).

No Quadro 1 a seguir, é apresentado um exemplo de mensagem utilizando o padrão EDIFACT. O exemplo se refere a um Pedido de Compra, contendo apenas um item, entre um Comprador identificado pelo Código de Localização EAN 5412345000010 e um Fornecedor identificado pelo Código de Localização EAN 331233345500013. O Código de Localização *International Article Numbering Association* (EAN) identifica os locais, física e funcionalmente de forma única.

O Pedido foi efetuado em 7 de outubro de 1991 às 10:00 hs da manhã horário local e baseia-se em um contrato firmado em 3 de janeiro de 1991 de referência CT1245. A referência do Comprador para o Pedido de Compra é PO112233.

Neste Pedido o Comprador pede 48 unidades do produto identificado pelo Código EAN 33123345501003 ao preço unitário líquido de 550 BF, perfazendo um valor total de 26400 BF. O preço no varejo (incluindo impostos) é 800 BF. É especificada uma taxa de câmbio de 6.28 Francos Belgas para um Franco Francês, válida para o mês de outubro de 1991. O Comprador solicita a entrega em 4 embalagens (cada embalagem irá conter 12 unidades) em 11 de outubro de 1991 às 10:00 hs da manhã.

```
UNH+ME000001+ORDERS:2:901:UM:EAN006
BGM+105+PO112233+940330:1000
RFF+CT+CT12345+940103
NAD+BY+5412345000010:14
NAD+SU+3312345000013:14
DTM+002+940411+1000+054
CUX+BEF:PC
CUX+FRF+BEF+930.05
DTM+151+911001
DTM+152+911031
UNS+D
LIN+1++3312345501003:EN++21:48+550:NT++26400+800::RTP
IMD+C++CU:EM
PAC+4+01+PK
UNS+S
CNT+02:1
UNT+17+ME000001
```

Fonte: EAN BRASIL (1994, p. 48)

Quadro 1 – Exemplo de uma mensagem EDIFACT

No Quadro 2 a seguir, é apresentado um exemplo de mensagem com dois pedidos de compra, porém utilizando o formato padrão X12.

```
ISA^00^   ^ZZ^1019000~
^980120^1712^00200^00000000^0^P^P^>~
GS^PO^1019000^COLONIAL^980120^1712^5^X^003030~
ST^850^5001~
BEG^00^SA^JITXX01^980120~
N1^ST^^91^099~
PO1^1^55^CS^10^UK^40605863467557123456~
PO1^2^44^RE^11.51^HI^H444005081~
PO1^3^21^CS^54.43^UK^40605863467755~
PO1^4^1^EA^80^HI^H444007445~
CTT^4~
SE^9^5001~
ST^850^5002~
BEG^00^SA^JITXX02^^980120~
N1^ST^^91^099~
PO1^1^CS^24^VC^1475099619~
PO1^3^66^PG^105.71^HI^H84685765~
PO1^4^77^CS^24.51^^VC^X081314325040~
CTT^4~
SE^9^5002~
GE^2^5~
IEA^1^000000005~
```

Fonte: adaptado de Anderson et al. (2001, p. 662)

Quadro 2 – Exemplo de uma mensagem X12

Pode-se observar que a menos que se conheçam os padrões estas mensagens não fariam sentido. É quase impossível dizer que estas mensagens tratam de um pedido de compra e muito menos o que esses pedidos estejam solicitando.

2.2 XML

O eXtensible Markup Language (XML) é um projeto da *World Wide Web Consortium* (W3C) para criar uma linguagem de marcação simples para delimitar e padronizar dados.

De acordo com Anderson et al. (2001), XML não é um apocalipse que vai erradicar tudo aquilo que apareceu antes dele. Ele irá habilitar programadores a fazer muitas coisas interessantes de maneira simples e flexível, mas o XML não é uma linguagem de programação, nem um sistema com base em objeto, tampouco um sistema operacional. É uma técnica poderosa, elegante para se pensar em dados, trocá-los e apresentá-los independente de plataformas.

Para implementar soluções em gestão de informações utilizando XML, pode-se usar:

a) *Document Type Definition* (DTD) ou *XML Schemas* (Xschema) para especificar o modelo de dados e validar as informações;

b) as API's *Document Object Model* (DOM) ou *Simple API for XML* (SAX) para extrair dados dos documentos, gerar documentos, ler e gravar em banco de dados;

c) *Extensible Style Language* (XSL), *XSL Transformation* (XSTL) e *XML Path Language* (XPath) para transformar os dados em outros formatos;

d) *XML Linking Language* (Xlink), *XML Pointer Language* (XPointer) e *XML Query Language* (XQuery) para criar vínculos lógicos entre os documentos e localizar seus componentes;

e) *XSL Formatting Objects* (XSL-FO) ou *Extensible HyperText Markup Language* (XHTML) para formatar os dados para impressão ou visualização na tela (PDF, Word ou Web);

f) *Validator for XML Schema* (XSV) para gerar informações em forma de gráfico vetorial.

O exemplo do Quadro 3 demosnstra um pedido de compra no formato XML.

```
<?xml version="1.0" encoding="utf-8" ?>
<order>
<order_no>654321</order_no>
     <order_date>20000115</order_date>
     <ship_to>
          <address>
               <address_name>Dave Wilkinson</address_name>
               <address_address>123 W Main St</address_address>
               <address_city>Columbus</address_city>
               <address_state>OH</address_state>
          </address>
     </ship_to>
     <bill_to>
          <DUNS>8580828442</DUNS>
          <address>
               <address_name>Sterling Commerce</address_name>
               <address_address>4600 Lakehurst Ct.</address_address>
               <address_city>Columbus</address_city>
               <address_state>OH</address_state>
               <address_zipcode>43109</address_zipcode>
          </address>
     </bill_to>
     <item>
          <item_identifier>
               <item_SKU>12345678</item_SKU>
               <item_UPC>987654321</item_UPC>
               <item_name>A Real Big Fast Computer</item_name>
          </item_identifier>
          <item_quantity>1</item_quantity>
          <item_unit>EA</item_unit>
          <item_price>3299.99</item_price>
     </item>
</order>
```

Fonte: adaptado de XML Solutions (2004)

Quadro 3 – Exemplo de uma mensagem XML

A mensagem apresentada no Quadro 3 agora é demonstrada no formato X12 do EDI no Quadro 4.

```
BEG*00*NE*654321*200000115~
N1*ST*Dave Wilkinson*1*6147937221~
N3*123 W Main St~
N4*Columbus*OH~
N1*BT*Sterling Commerce*1*8580828442~
N3*4600 Lakehurst Ct~
N4*Columbus*OH*43109~
PO1**1EA*3299.99**UP*987654321*VC*123456789~
PID***A Real Big Fast Computer~
CTT*1~
```

Fonte: adaptado de XML Solutions (2004)

Quadro 4 – Exemplo da mensagem XML em formato X12

Pode-se observar que a mensagem no padrão XML praticamente se descreve a si mesma, de outra forma às mensagens do padrão EDI é necessário conhecer o padrão para que seja possível entender o conteúdo da mensagem.

Mais informações sobre XML pode ser encontrada em W3C (2004).

2.3 WEB SERVICES E SOAP

De acordo com Deitel, Deitel e Nieto (2004, p. 837), um *web service* (ou serviço web) é uma classe armazenada em uma máquina que pode ser acessada em outra máquina em uma rede. A máquina na qual reside o *web service* é referenciada como máquina remota.

Apesar da palavra Web, não significa que seja uma aplicação web específica, mas sim, uma aplicação que utilizada tecnologias web, como servidores e HTTP, possibilitando que os serviços sejam invocados através de outras máquinas na rede.

Conallen (2003, p. 69) define um serviço web como “uma coleção de funções empacotadas e publicadas em uma rede para serem usada por outros programas clientes. [..] um serviço Web é simplesmente um outro tipo de chamada de procedimento remoto *Remote Procedure Call* (RPC)”.

Na plataforma .Net, as chamadas de métodos são implementadas por meio do *Simple Object Access Protocol* (SOAP), baseado em XML, que descreve como marcar as solicitações e respostas de forma que elas possam ser transferidas por protocolos como o HTTP. Um serviço web apresenta duas partes: um arquivo ASMX e um arquivo de código de apoio. O arquivo ASMX pode ser visualizado em qualquer navegador web e contêm informações sobre

o serviço web como as descrições de seus métodos e a maneira de testá-los. O arquivo de código de apoio provê a implementação dos métodos que o serviço web abrange. A descrição do serviço é feita através de um documento XML que está em conformidade com *Web Services Description Language* (WSDL). Um documento WSDL define os métodos que o serviço web disponibiliza e a maneira que os clientes interagem com esses métodos. O Visual Studio .Net gera a descrição do serviço WSDL.

O SOAP é um protocolo independente de plataforma e fácil de entender. Semelhante, o HTTP foi escolhido para transmitir mensagens SOAP porque é um protocolo padrão para envio de informações pela internet. A combinação XML – HTTP habilita diferentes sistemas operacionais a enviar e receber mensagens SOAP. Um exemplo de uma mensagem SOAP pode ser visto no Quadro 5.

```
POST /Hugeinteger/HugeInteger.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset="utf-8"
Content-Length: length
SOAPAction: "http://www.deitel.com/Bigger"

<?xml version="1.0" encoding="utf-8"?>

<SOAP:Envelope
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP:Body>
       <Bigger xmlns="http://www.deitel.com">
          <first>3412342345552345</first>
          <second>234555234666345</second>
        </Bigger>
   </SOAP:Body>
</SOAP:Envelope>
```

Fonte: adaptado de Deitel, Deitel e Nieto (2004, p. 840)

Quadro 5 – Exemplo de uma mensagem SOAP

2.4 ebXML (ELETRONIC BUSINESS EXTENSIBLE MARKUP LANGUAGE)

De acordo com ebREQ (2001), no início de dezembro de 1999 um esforço conjunto de 18 meses do comitê das Nações Unidas para Facilitação do Comércio e Negócio Eletrônico (UN/CEFACT) e a Organização para o Avanço dos Padrões de Informações Estruturadas (OASIS) foi lançado para iniciar um projeto mundial para padronizar as especificações de negócios de XML. O UN/CEFACT e OASIS estabeleceram o Grupo de Trabalho de XML de Negócios Eletrônicos para desenvolver uma estrutura técnica que permitirá que o XML seja

utilizado de uma maneira coerente para a troca de todos os dados de negócios eletrônicos. Estas especificações podem ser encontradas em ebXML (2004).

O EDI tem sido usado para conduzir negócios eletronicamente, mas o alto custo de implementar estes padrões e a rigidez das estruturas por detrás deles, limitam seu uso para poucas organizações maiores que podem dispor da integração e manutenção deles. O ebXML, além de fornecer a funcionalidade requerida pelas grandes empresas, também facilita a implementação e a administração para pequenas e médias empresas. Para isso, foi desenvolvido o ebXML, um padrão para intercâmbio de informações empresariais que reduz os custos de implementação e que seja de rápido desenvolvimento.

A idéia geral do ebXML é criar um único lugar de comércio eletrônico global onde as empresas de qualquer tamanho e de qualquer lugar do mundo possam se encontrar e conduzir negócios baseados em XML para a troca de mensagens através da internet. Para facilitar esta troca de mensagens, o ebXML fornece uma estrutura para a interoperabilidade comercial e um mecanismo que permite que as empresas se encontrem, estabeleçam um relacionamento, e assim, conduzam negócios entre si.

Para o desenvolvimento do ebXML, o grupo de trabalho (UN/CEFACT e OASIS) contou com a participação de centenas de pessoas, organizações, companhias e de consórcios ao redor do mundo. As mensagens ebXML são consistentes e baseadas em processos empresariais bem desenvolvidos, trabalha com semântica empresarial clara e permite que os negócios sejam conduzidos de acordo com um padrão ou um padrão mutuamente concordado pelos parceiros comerciais. De acordo com Anderson et al. (2001) o grupo de trabalho está divido atualmete em 8 equipes.

Além de Marketing, Conhecimento e Educação, existem as seguintes equipes:

a) requisitos ebXML: objetivos a curto e longo prazo;
b) metodologia do processo de negócios: estrutura e interoperabilidade, intercâmbio de modelos (esquemas UML para XML e vice-versa);
c) transporte/roteamento e empacotamento: envelopamento para roteamento de conteúdo de mensagens, segurança, entrega de mensagens, processamento de batch, qualidade de serviço. Trata das regras técnicas e diretrizes para o *e-Business*;
d) Componentes centrais: usam semânticas existentes de uma maneira uniforme, reutilizam/compõem elementos/componentes existentes, extensões de mensagens

existentes para cobrir novas aplicações, identificam semânticas de dados horizontais ou centrais;

e) arquitetura técnica: a semântica equivalente entre elementos de dados, transformação de semânticas e internacionalização. Trata das regras de design de mapeamento do grupo de trabalho EDIFACT e ebXML, diretriz para transformar as mensagens EDIFACT em mensagens XML;
f) registro e repositório: criação de versões, ferramenta para ferramenta, repositório para repositório, ferramenta para repositório e vice-versa. Trata dos assuntos de interoperabilidade, repositórios físicos e arquitetura do registro;
g) coordenação técnica e suporte: mantém o site web da ebXML, certificação, bancos de teste, kits iniciadores de programadores e publica critérios em conformidade com software.

O padrão ebXML é aberto e de fácil implementação e não compete com padrões existentes como UN/EDIFACT e X12, etc., assim preserva muito do investimento existente nestas aplicações.

2.4.1 ARQUITETURA DO ebXML

De acordo com ebTA (2001), durante 25 anos o EDI deu às companhias o prospecto de eliminar documento de papéis, reduzindo custos e melhorando a eficiência da troca eletrônica comercial. A idéia era que empresas de todos os tamanhos pudessem administrar *e-Business* sem acordo anterior de ambas as partes, mas esta visão não foi percebida com o EDI; só companhias de grande porte podem dispor desta implementação e o EDI é centralizado ao redor de um empreendimento dominante que impõe uma integração proprietária para seus parceiros de negócios.

De acordo com Anderson et al. (2001), nos últimos anos, o XML se tornou rapidamente a primeira escolha para definir formatos de trocas de mensagens em aplicações de *e-Business* na internet. Muitas pessoas interpretam que com o surgimento da XML o EDI estará fora de uso, mas esta é uma visão equivocada no ponto de vista técnico e de negócios.

Implementações de EDI apresentam experiência significativa em Processos de Negócios, e companhias com investimentos grandes em integração de EDI não os abandonarão sem uma boa razão. O XML pode habilitar modelos empresariais mais flexíveis

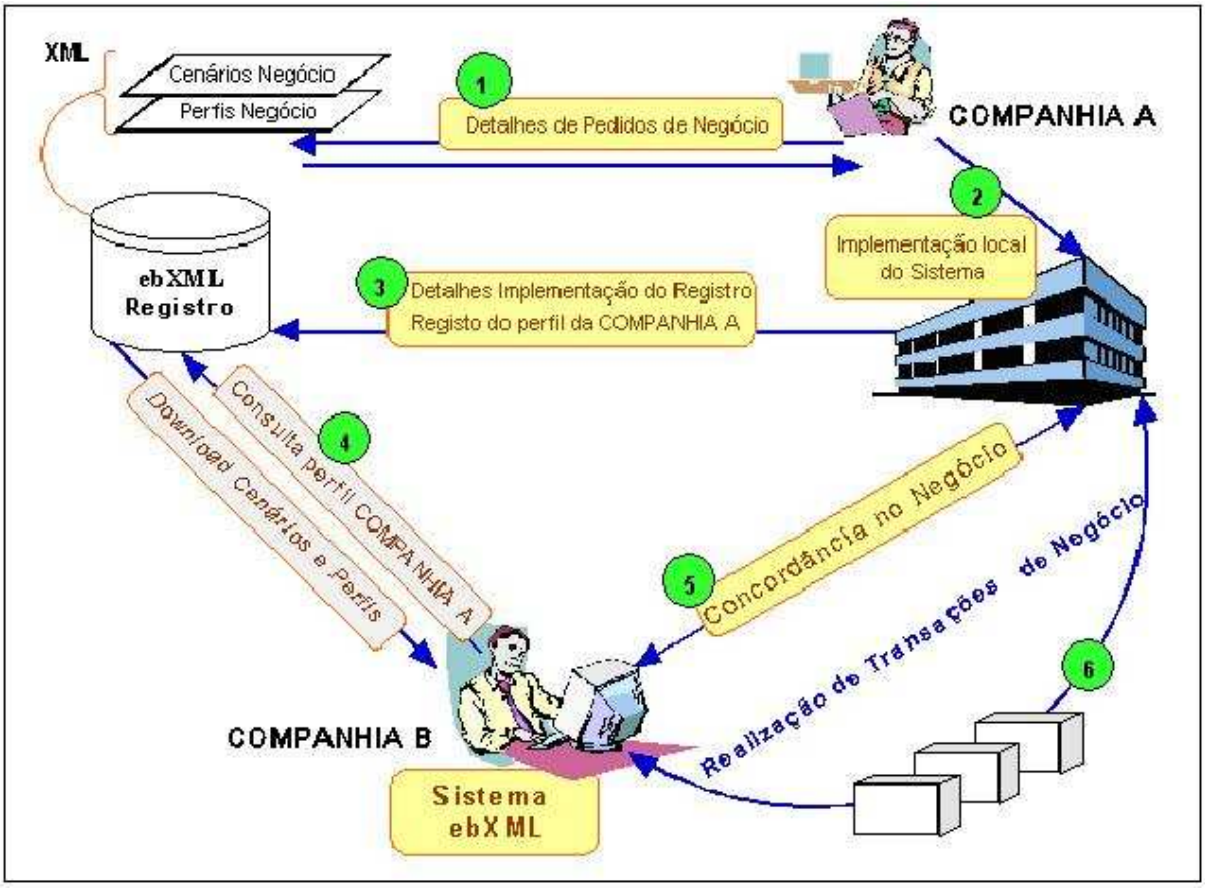
e inovadores que EDI. Mas as mensagens que satisfazem esses processos empresariais são independentes da sintaxe na qual as mensagens são codificadas.

As especificações de ebXML provêem um *framework* no qual podem ser preservados os investimentos significativos de EDI em Processos de Negócio em uma arquitetura que explora as novas capacidades técnicas da XML.

Os passos a seguir introduzem os conceitos relacionados à arquitetura:

a) um mecanismo padrão para descrever um Processo Empresarial e seu modelo de informação associado;
b) um mecanismo para registrar e armazenar o Processo Empresarial e os Meta Modelos de Informação, assim eles podem ser compartilhados e utilizados;
c) descoberta de informação sobre cada participante: Processos Empresariais, Interfaces de Serviços Empresarias que eles oferecem, mensagens empresariais que são trocadas nas interfaces de serviços empresarias respectivas e a configuração técnica para o transporte, segurança e protocolos de codificação;
d) um mecanismo para registrar as informações acima mencionada de forma que possa ser descoberta e recuperada;
e) um mecanismo para descrever a execução de um acordo empresarial que pode ser derivado da informação obtida por cada parceiro comercial participante vista no item c;
f) um *framework* de serviço de mensagens (*Messaging Service*) de negócios unificado que habilite interoperabilidade, segurança e a troca segura de mensagens entre os parceiros comerciais;
g) um mecanismo para configuração do respectivo serviço de mensagens para empregar dentro do Processo de Negócio que esteja em conformidade com as restrições definidas pelo arranjo empresarial.

A figura 1 demonstra, de forma geral, um caso de uso para dois parceiros de negócio, que primeiro configuram seu perfil e depois realizam a troca de informação. Este modelo provê um exemplo do processo e passos que podem ser exigidos para configurar e implementar aplicações ebXML e componentes relacionados com a arquitetura. Estes componentes podem ser implementados de uma maneira incremental.



Fonte: fonte adaptado de ebTA (2001, p. 8)

Figura 1 – Visão geral do ebXML

A seguir são apresentados os 6 passos do modelo do processo:

Passo 1: a Companhia A acessa um Registro de ebXML localizado na internet. Neste Registro a Companhia A encontra os processos de negócios que podem ser associados à empresa. Caso não encontre, poderá definir novos processos de negócios que devem estar de acordo com a especificação de processos de negócios (ebBPSS, 2001).

Passo 2: após revisar os conteúdos do Registro, a companhia A decide organizar e construir sua própria aplicação ebXML (que deve estar de acordo com a interface de serviço do Registro). A prática do desenvolvimento da aplicação não é uma condição prévia necessária para a participação em aplicações ebXML. Aplicações ebXML podem estar disponíveis comercialmente.

Passo 3: a companhia A então submete sua própria informação de Perfil Empresarial (incluindo detalhes da implementação e links de referência) para o registro ebXML. Este documento deve estar de acordo com a especificação (ebCPP, 2001). O perfil empresarial

submetido ao Registro descreve as capacidades e restrições da companhia, como também seus negócios empresariais suportados.

Passo 4: a companhia B descobre os negócios empresariais suportados pela companhia A no registro ebXML.

Passo 5: a companhia B envia uma notificação para a companhia A declarando que deseja realizar negócios empresarias utilizando ebXML. A companhia B adquire uma aplicação comercial de ebXML. Antes de iniciarem a troca de informações empresariais a companhia B envia diretamente para a companhia A um acordo empresarial. O acordo empresarial proposto esboça um cenário empresarial mutuamente acordado e acordos específicos. O acordo empresarial também contém informação pertencente às exigências relacionadas às mensagens para as transações empresarias acontecerem e exigências relacionadas à segurança. A companhia A então aceita o acordo empresarial.

Passo 6: agora a companhia A e B estão prontas para realizar *e-Business* utilizando ebXML.

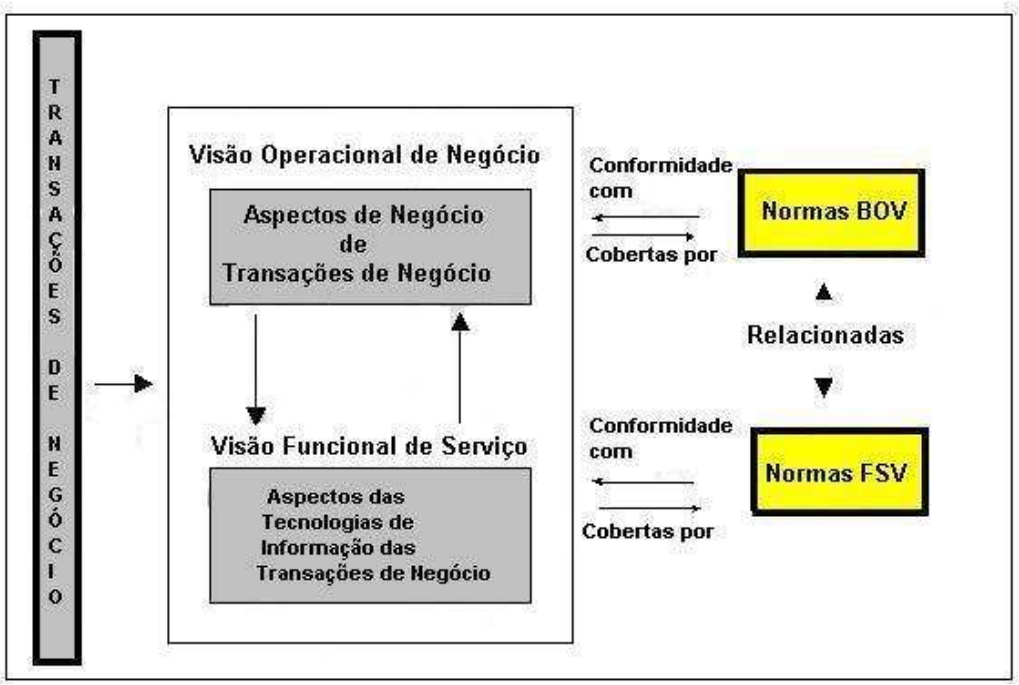
2.4.2 METODOLOGIA DE MODELAGEM RECOMENDADA ebXML

O processo empresarial e a modelagem da informação não são obrigatórios. Porém, se os desenvolvedores e usuários decidirem modelar os processos empresarias e informações, então a metodologia recomendada é a UN/CEFACT *Modeling Methodology* (UMM) que utiliza a *Unified Modeling Language* (UML).

Enquanto a prática empresarial de uma organização a outra é altamente variável, a maioria das atividades pode ser decomposta em processos de negócio que são mais genéricas a um tipo específico de negócio. Esta análise pelo processo de modelagem identificará o processo empresarial e o meta modelo de informação que são os candidatos prováveis para a padronização. O ebXML considera a construção de uma estrutura de componentes reutilizáveis para prover interoperabilidade de componentes.

A UMM está dividida em duas visões para descrever os aspectos pertinentes às transações de *e-Business.* Este modelo é baseado no *Open-edi Reference Model*, ISO/IEC 14662.

A Figura 2 a seguir demonstra a metodologia de modelagem recomendada com suas respectivas visões.



Fonte: fonte adaptado de ebTA (2001, p. 10)

Figura 2 – Metodologia de modelagem recomendada para ebXML

A UMM está dividida em visão de negócios *Business Operational View* (BOV) ou Visão Operacional do Negócio, e na visão de suporte *Functional Service View* (FSV) ou Visão Funcional de Serviço, descritas acima. A pretensão para ebXML é que o FSV sirva como um modelo de referência que pode ser usado por vendedores de software comerciais para guiar e ajudar durante o processo de desenvolvimento. A meta fundamental da UMM é prover uma distinção entre as visões operacionais e funcionais, para assegurar o nível máximo de interoperabilidade e compatibilidade.

O BOV está focalizado nos negócios como:

a) a semântica dos dados de negócio, transações e dados associados no intercâmbio;
b) processos de negócios e meta modelos de informação;
c) a arquitetura para transações de negócios:
   - convenções operacionais;
   - acordos e contratos;
   - obrigações mútuas e exigências.

Estas especificações aplicam às necessidades empresariais dos parceiros de negócios ebXML.

O FSV está direcionado para suportar os serviços e mecanismos necessários do ebXML. Focalizado nos aspectos de informática como:

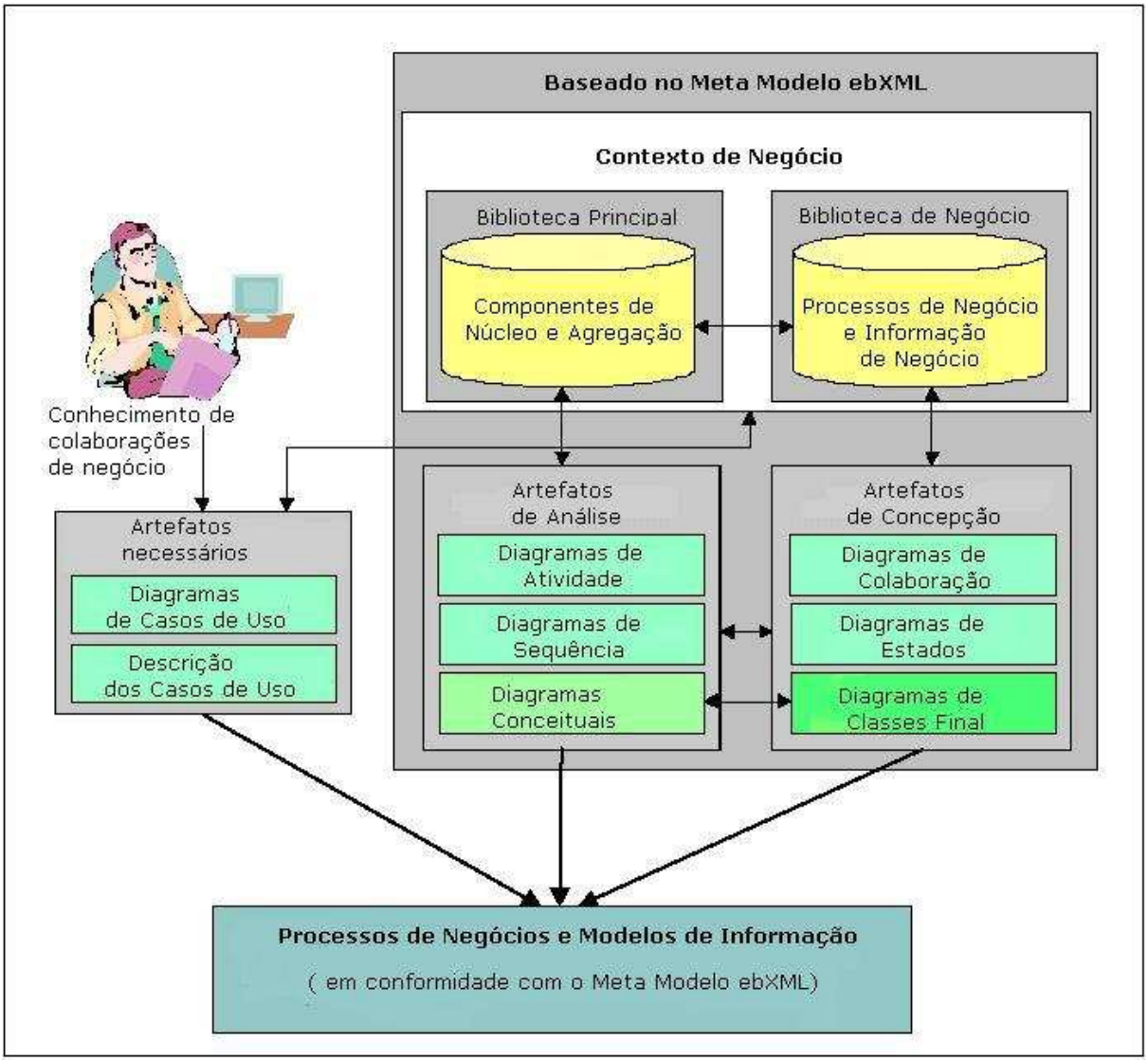
a) capacidades funcionais;
b) interface de serviços empresariais;
c) protocolos e serviços de mensagens.

Inclui, mas não limita:

a) aspectos de implementação, recuperação, distribuição e cenários de tempo de execução;
b) interfaces de usuário, de infra-estrutura e de transferência de dados;
c) protocolos para interoperabilidade.

2.4.2.1 Visão Operacional do Negócio (BOV)

De acordo com ebTA (2001) a técnica de modelagem descrita nesta seção não é um requisito obrigatório em transações de negócios concordadas de ebXML. A Figura 3 a seguir, mostra de forma detalhada a visão operacial do negócio.



Fonte: fonte adaptado de ebTA (2001, p. 11)

Figura 3 – Representação detalhada do BOV

Na Figura 3, Conhecimento de Colaboração do Negócio *(Business Collaboration Knowledge)* é capturado em uma Biblioteca Principal ou Central *(Core Library*). A biblioteca central contém dados e definições de processos e é a ponte entre o negócio específico ou idioma industrial e os conhecimentos expressados pelos modelos em um contexto mais generalizado em um idioma neutro.

A primeira fase define os requisitos ou artefatos necessários que descrevem o problema usando diagramas de Casos de Uso. Se as entradas da biblioteca central que estão disponíveis no registro ebXML estiverem de acordo serão utilizadas, caso contrário serão criadas entradas novas para a biblioteca central que serão registradas em um registro ebXML.

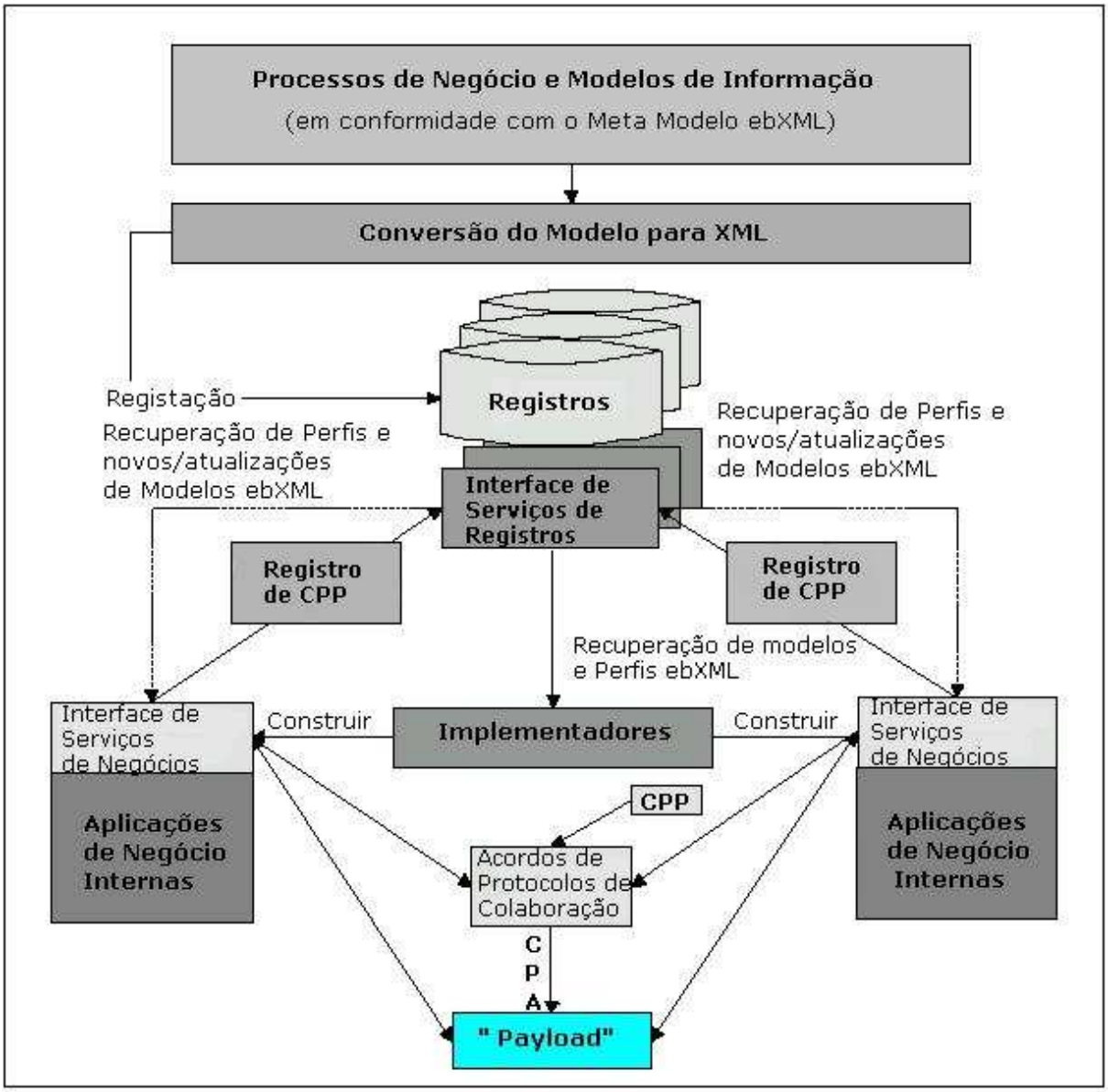
Na segunda fase (análise) serão criados Diagramas de Atividade e Seqüência descrevendo os processos empresariais. Diagramas de Classes capturarão a informação associada (documentos empresariais). A fase de análise reflete o conhecimento empresarial contido na biblioteca central. Nenhum esforço é feito para forçar a aplicação aos conceitos da orientação a objetos. Processos de negócios comuns na Biblioteca de Negócio *(Business Library)* podem ser referenciados durante o processo de análise e criação de artefatos de projeto.

A fase de projeto é o último passo de padronização que pode ser realizada aplicando princípios da orientação a objetos baseado no UMM. Além dos Diagramas de Colaboração pode ser criado também um Diagrama de Estados. O Diagrama de Classes da fase de análise sofrerá harmonização para alinhá-lo com outros modelos na mesma indústria e por outras.

Em ebXML, interoperabilidade é alcançada aplicando Objetos de Informação do Negócio *(Business Information Objects)* por todos os modelos de classe.

2.4.2.2 Visão Funcional de Serviço (FSV)

Como ilustrado na Figura 4 a seguir, o Serviço de Registro *(Registry Service)* serve como facilidade de armazenamento pelo processo de negócio e modelos de informação, as representações baseadas no XML desses modelos, Componentes Centrais *(Core Components)* e Perfis de Protocolo de Colaboração *(Collaboration Protocol Profiles).*



Fonte: fonte adaptado de ebTA (2001, p. 13)

Figura 4 – Visão Funcional de Serviço (FSV)

2.4.3 FASES DO EBXML

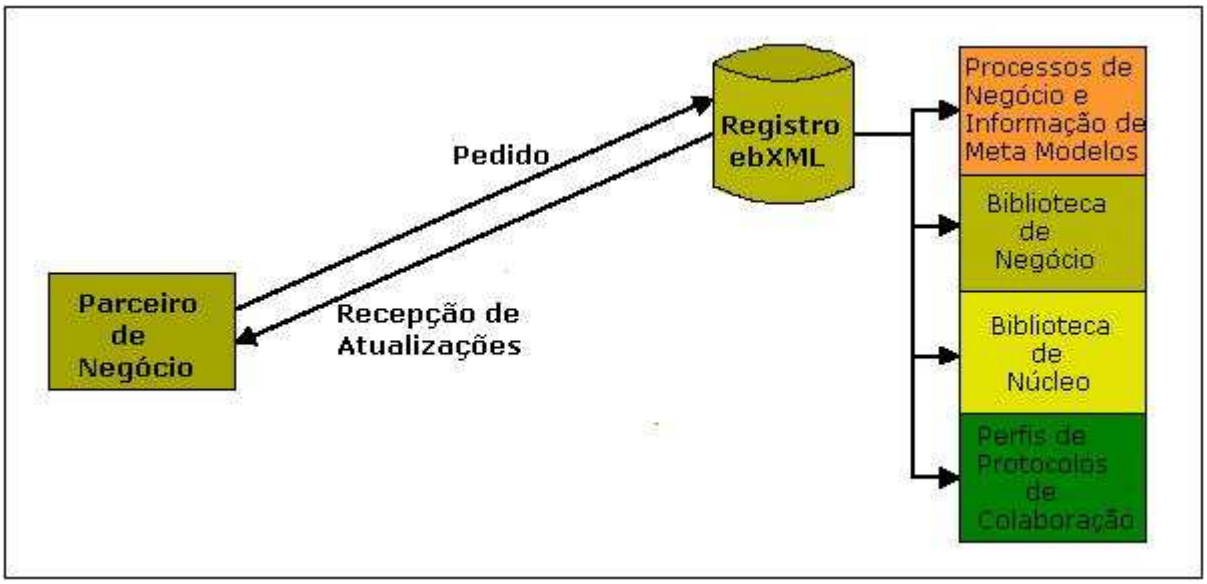
Nas 3 seções a seguir, serão apresentadas as fases para a implantação do ebXML. Fase de implementação, descoberta e recuperação e a fase de execução.

2.4.3.1 Fase de Implementação

A fase de implementação trata especificamente dos procedimentos para criar uma infra-estrutura de ebXML. Um parceiro de negócio que deseja realizar uma transação ebXML deveria adquirir cópias das especificações ebXML primeiro. O parceiro de negócio estuda

estas especificações e subseqüentemente carrega a Biblioteca Central ou Núcleo e a Biblioteca de Negócio. O parceiro comercial também pode pedir as informações do processo de negócio de outro parceiro comercial (armazenado no seu perfil empresarial) para análise e revisão. O parceiro comercial também pode submeter sua própria informação de processo de negócio a um serviço de registro ebXML concordante.

A Figura 5 abaixo, representa uma interação básica entre um serviço de registro ebXML e um parceiro de negócio.

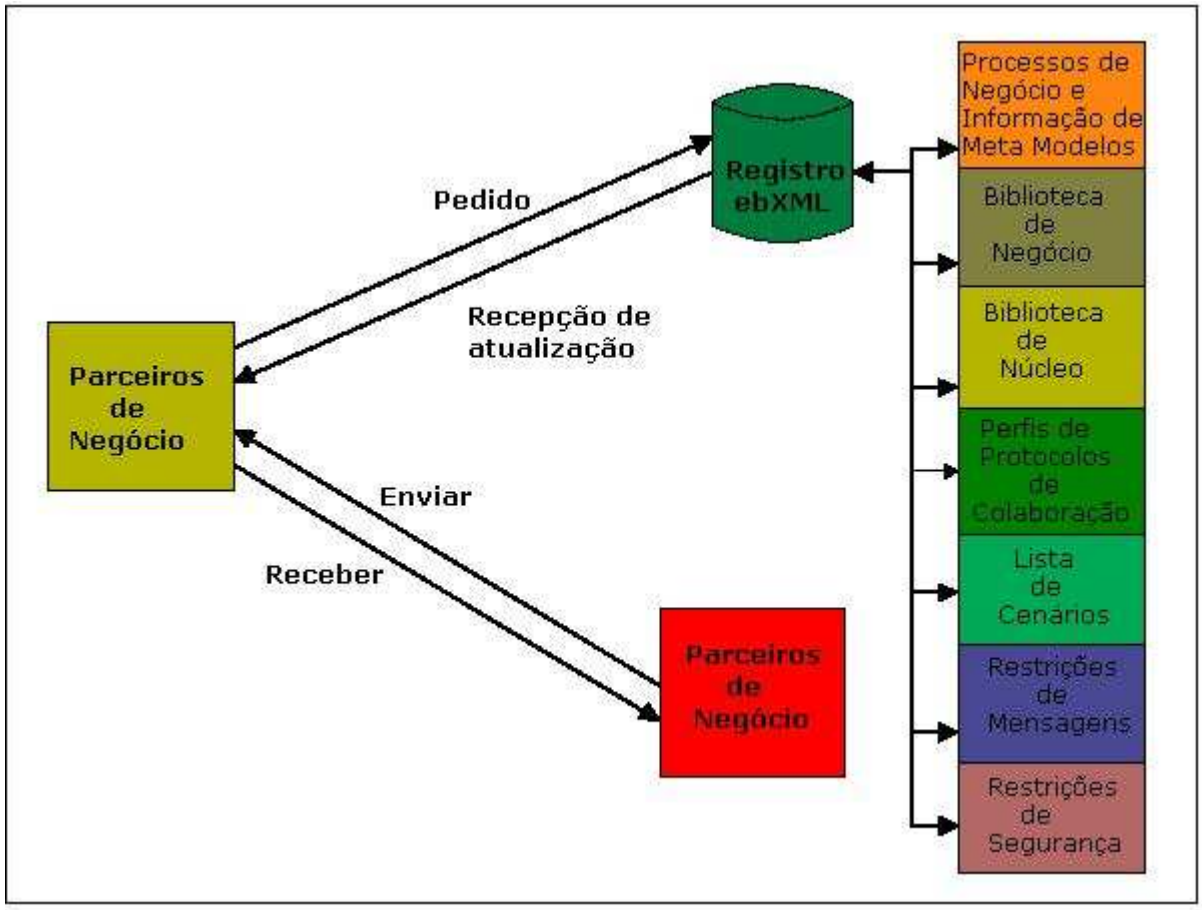


Fonte: fonte adaptado de ebTA (2001, p. 14)

Figura 5 – Fase de Implementação

2.4.3.2 Fase de Descoberta e Recuperação

Esta fase cobre todos os aspectos relacionados à descoberta de recursos relacionados ao ebXML. Um parceiro de negócio que implementou uma interface de serviço de negócios *(Business Service Interface)* agora pode iniciar o processo de descoberta e recuperação, visto na Figura 6 abaixo. Um possível método de descoberta pode ser a requisição do *Collaboration Protocol Profile* (CPP) ou Perfil de Protocolo de Colaboração, de outro parceiro comercial. Pedidos de atualizações para bibliotecas centrais, biblioteca de negócios e processo de negócios atualizado ou um novo processo de negócios e meta modelos de informação deveriam ser suportados pela interface de serviços de negócios. Esta é a fase onde parceiros descobrem o significado das informações do negócio que está sendo requisitado por outros parceiros.



Fonte: fonte adaptado de adaptado de ebTA (2001, p. 15)

Figura 6 - Fase de Descoberta e Recuperação

2.4.3.3 Fase de Execução

A fase de execução cobre a execução de um cenário de ebXML com as transações ebXML reais associadas. Nesta fase as mensagens do ebXML são trocadas entre os parceiros comerciais que utilizam o serviço de mensagens do ebXML.

Fonte: fonte adaptado de ebTA (2001, p. 15)

Figura 7 - Fase de Execução

2.4.4 INFORMAÇÃO DOS PARCEIROS DE NEGÓCIO CPP e CPA's

Para facilitar o processo de administrar *e-Business*, parceiros de negócios precisam de um mecanismo para publicar informações sobre seus processos de negócio suportados, junto com detalhas técnicos da implementação para a troca da informação empresarial. Isto é realizado através do uso do Perfil de Protocolo de Colaboração (CPP). O CPP é um documento que permite um parceiro de negócio expressar seus processos de negócio suportados e as exigências da interface de serviço de mensagens de uma maneira que possam ser entendidos universalmente por outros parceiros ebXML.

Um acordo empresarial chamado *Collaboration Protocol Agreement* (CPA) ou Acordo de Protocolo de Colaboração, é derivado da interseção de dois ou mais CPP's. O CPA serve como um "aperto de mão" entre dois ou mais parceiros de negócio.

De acordo com ebCPP (2001) as informações contidas no CPA é similar as especificações de tecnologia e informações contidas nos acordos entre parceiros de negócios EDI. Diferem-se entanto, por não serem documentos de papel, e sim, documentos eletrônicos que podem ser processados por cada organização de forma a iniciar e executar a troca eletrônica de informações.

O CPP descreve as capacidades específicas que um parceiro suporta como também as exigências das interfaces de serviços que precisam ser conhecidas para trocar documentos empresariais com ele.

O CPP contém informações essenciais sobre o parceiro, incluindo:

a) informações de contato;
b) processos de negócio suportados;
c) exigências de interface e exigências do serviço de mensagens.

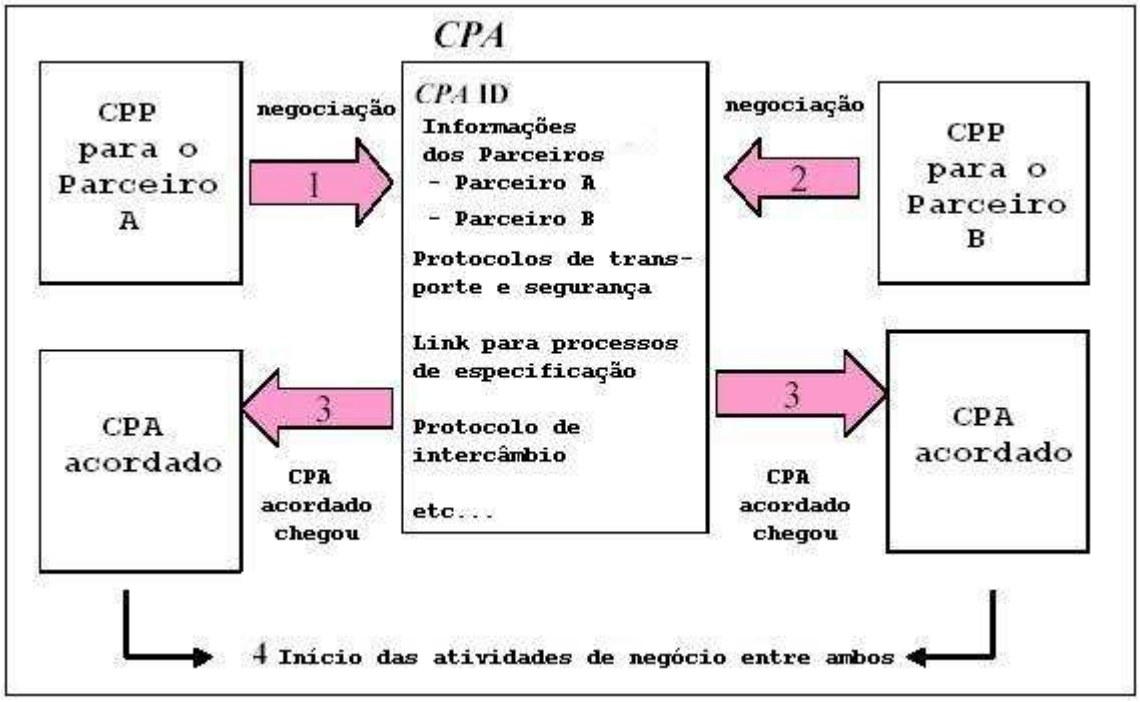
O CPP pode conter também detalhes de segurança e outros detalhes específicos da implementação.

A definição do CPP não deve permitir ambigüidade de escolha em todos os casos onde pode haver seleção múltipla (por exemplo, HTTP ou SMTP).

O CPA descreve:

a) o serviço de mensagens;
b) as exigências do processo de negócios que já está de acordo com dois ou mais parceiros.

A Figura 8 a seguir, demonstra a interseção de dois CPP's formando assim um CPA.



Fonte: fonte adaptado de ebCPP (2001, p. 14)
Figura 8 - Visão geral do CPA

Para ver exemplo de CPP e CPA consultar Anexos A e B respectivamente. A especificação para criar CPP's e CPA's pode ser encontrada em (ebCPP, 2001).

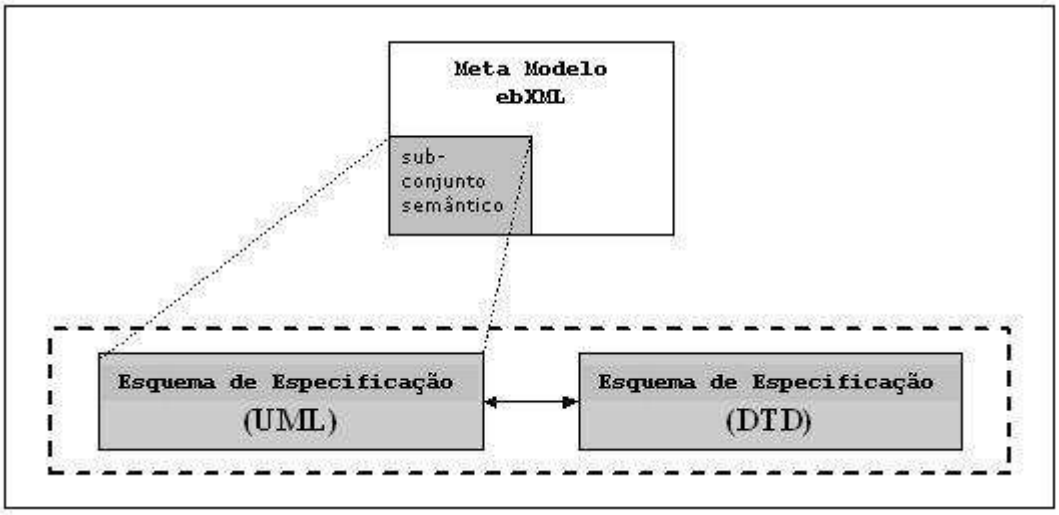
2.4.5 PROCESSO DE NEGÓCIO E MODELAGEM DA INFORMAÇÃO

O processo de negócio ebXML e meta modelo de informação *(Business Process and Information Meta Model)* é um mecanismo que permite aos parceiros comercias capturar os detalhes para a execução de um cenário empresarial específico que usa uma metodologia de modelagem consistente. Um processo de negócio descreve em detalhes como o parceiro assume papéis, relações e responsabilidades para facilitar a interação com outros parceiros em colaborações compartilhadas, a interação entre o lugar para pegar os papéis como a coreografia de transações comerciais. Cada transação comercial é expressa como uma troca de documento de negócio eletrônico.

O processo de negócio ebXML e o meta modelo de informação suportam exigências, análises e pontos de vista de projeto que provêem um conjunto de semântica (vocabulário) que formam a base da especificação dos artefatos que são exigidos para facilitar a integração da informação e interoperabilidade.

Uma visão adicional do meta modelo, a *Specification Schema* ou Esquema de Especificação, é provido também para suportar a especificação direta do conjunto de elementos para configurar um sistema para executar um conjunto de transações empresariais ebXML. As informações contidas dentro da especificação servem como contribuição para a formação de CPP's e CPA's. O esquema de especificação está disponível na forma de duas representações, UML e DTD.

A relação entre processo de negócio e meta modelo de informação e o esquema de especificação pode ser visto na Figura 9.



Fonte: fonte adaptado de ebTA (2001, p. 19)

Figura 9 – Meta modelo ebXML

Para prover facilidade na criação de processos de negócios e modelos de informações consistentes é desejado que se crie um conjunto de processos de negócio comuns em paralelo com uma biblioteca central. É possível que os usuários da infra-estrutura do ebXML possam desejar estender este conjunto ou usar seus próprios processos de negócio.

Mais informações sobre o processo de negócio e meta modelo de informação e a especificação do esquema pode ser encontrada em ebBPSS (2001).

Um exemplo de esquema de especificação de processos de negócio pode ser visto no Anexo C.

2.4.6 COMPONENTES CENTRAIS E FUNCIONALIDADES DA BIBLIOTECA DE COMPONENTES

Um componente central captura informação sobre o conceito do mundo real do negócio e os relacionamentos entre esses conceitos (semânticas). Podem ser uma parte específica de uma informação de negócio e podem adotar parte e/ou estender os componentes do ebXML.

Para ser compreendidos por uma aplicação, os processos de negócios são expressos em sintaxe XML. Os componentes são armazenados e recuperados utilizando um registro ebXML.

2.4.7 REGISTRO ebXML

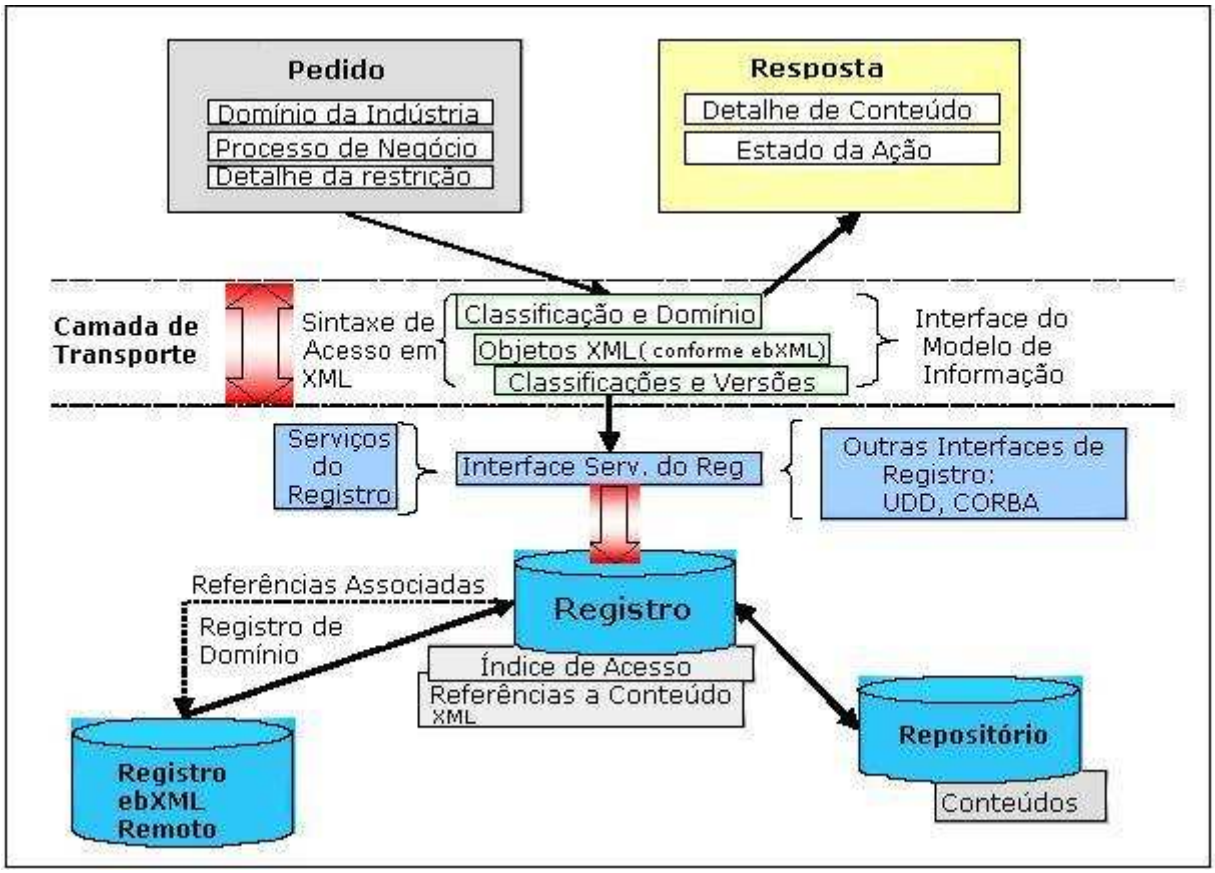
Um Registro ebXML provê um conjunto de serviços que habilitam o compartilhamento de informação entre parceiros comerciais (por exemplo, RosettaNet *Partner Interface Process* (PIP). O acesso ao Registro é provido através de Interfaces de Programação de Aplicativos (API) expostas através dos serviços do registro ou *Registry Service.*

O registro ebXML serve para algumas das funções mais importantes do ebXML, como por exemplo, a descoberta de parceiros, meio de comunicação e guardar informações compartilhadas entre eles.

Para o registro provê tais funcionalidades, o grupo de trabalho registro e repositório definiu duas especificações para auxiliar os desenvolvedores que desejam implementar um registro ebXML, são elas:

a) Modelo de Informação do Registro ou *Registry Information Model* (RIM): descreve que objetos podem ser armazenados no registro e como o registro pode ser organizado;
b) Especificação dos Serviços do Registro ou *Registry Services Specification* (RSS): descreve as interfaces disponíveis que o cliente pode utilizar.

A Figura 10, demonstra de forma geral o Registro ebXML.



Fonte: fonte adaptado de ebTA (2001, p. 25)

Figura 10 – Visão geral do Registro ebXML

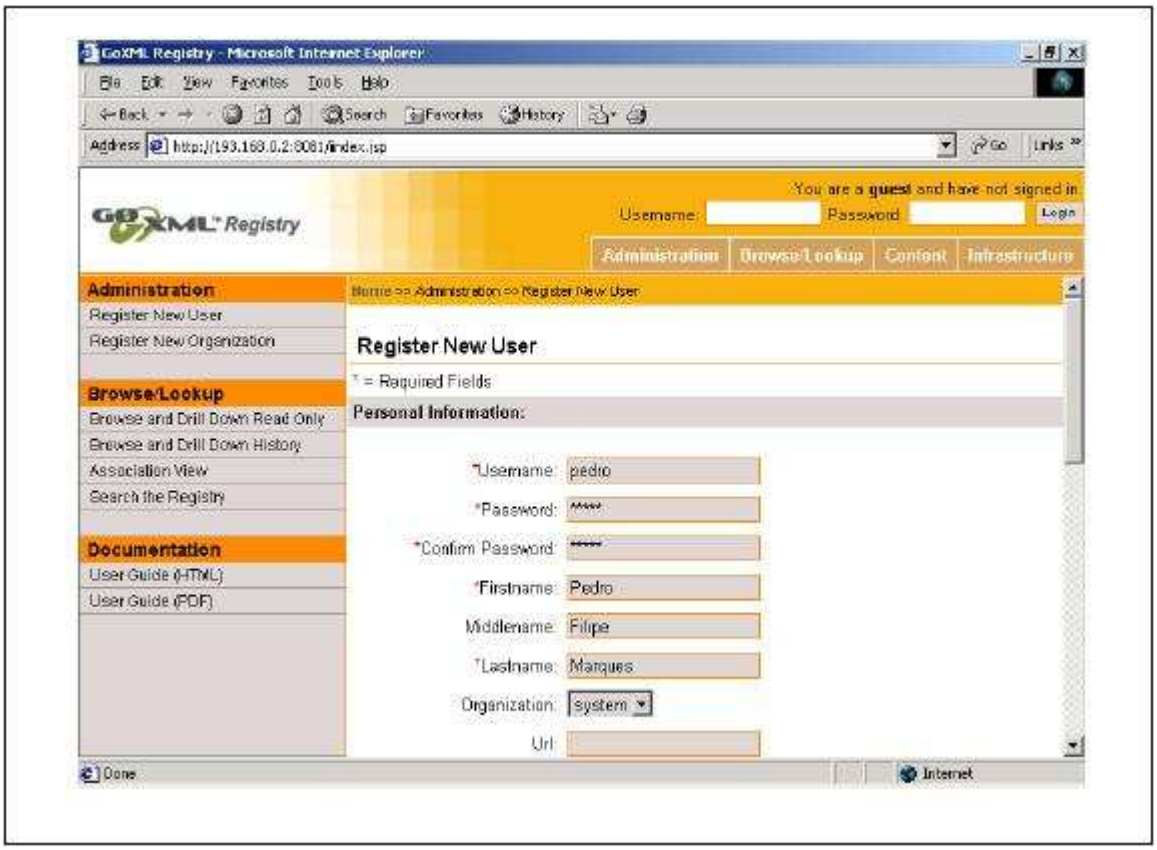
A Interface de Serviços do Registro serve como um mecanismo de acesso aplicação-para-registro. Uma aplicação para interação dos usuários com o registro pode ser construída em cima de uma interface de registro (por exemplo, um web browser).

A interface de registro deve ser projetada para ser independente da pilha de protocolo de rede (por exemplo, HTTP/SMTP em cima de TCP/IP). Instruções específicas de como interagir com a interface de registro pode estar contida na mensagem ebXML.

Uma aplicação de registro ebXML já desenvolvida é o *GoXML Registry*. O *GoXML Registry* foi desenvolvido de acordo com as especificações versão 2.0 do registro ebXML. Este registro possui interface web, permitindo aos administradores e aos proprietários das informações abrirem documentos e alterarem através do seu navegador web.

Mais informações sobre o *GoXML Registry* pode ser encontrada em GOXML (2004).

A Figura 11 a seguir, demonstra o cadastro de um novo usuário no programa *GoXML Registry.*



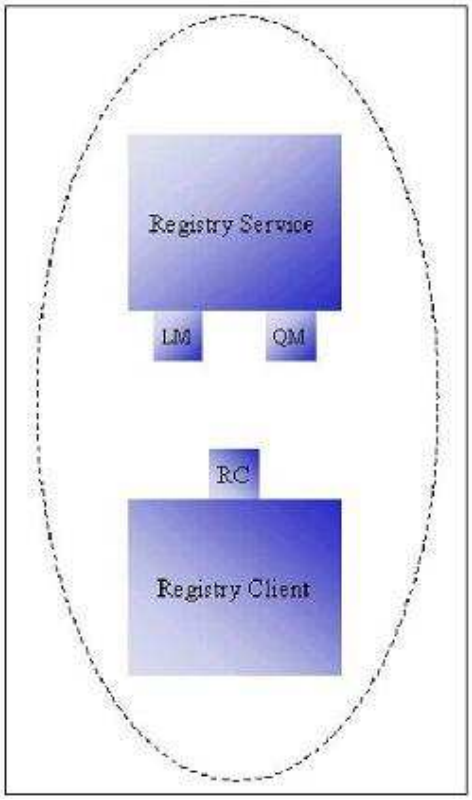
Fonte: Marques (2003, p. 111)

Figura 11 – Cadastro de um novo usuário no programa GoXML Registry

De acordo com o documento de especificação (ebRS, 2001) um Registro ebXML deve fornecer as seguintes interfaces para os clientes acessarem:

a) *Life Cycle Management* (LM) – provê uma coleção de métodos para gerenciar os objetos dentro do registro;
b) *Query Management* (QM) – controla a descoberta e recuperação de informações do registro.

O acesso as interfaces do registro ebXML deve ser feita através da *Registry Client* (RC). A Figura 12 a seguir, demonstra a arquitetura do registro.



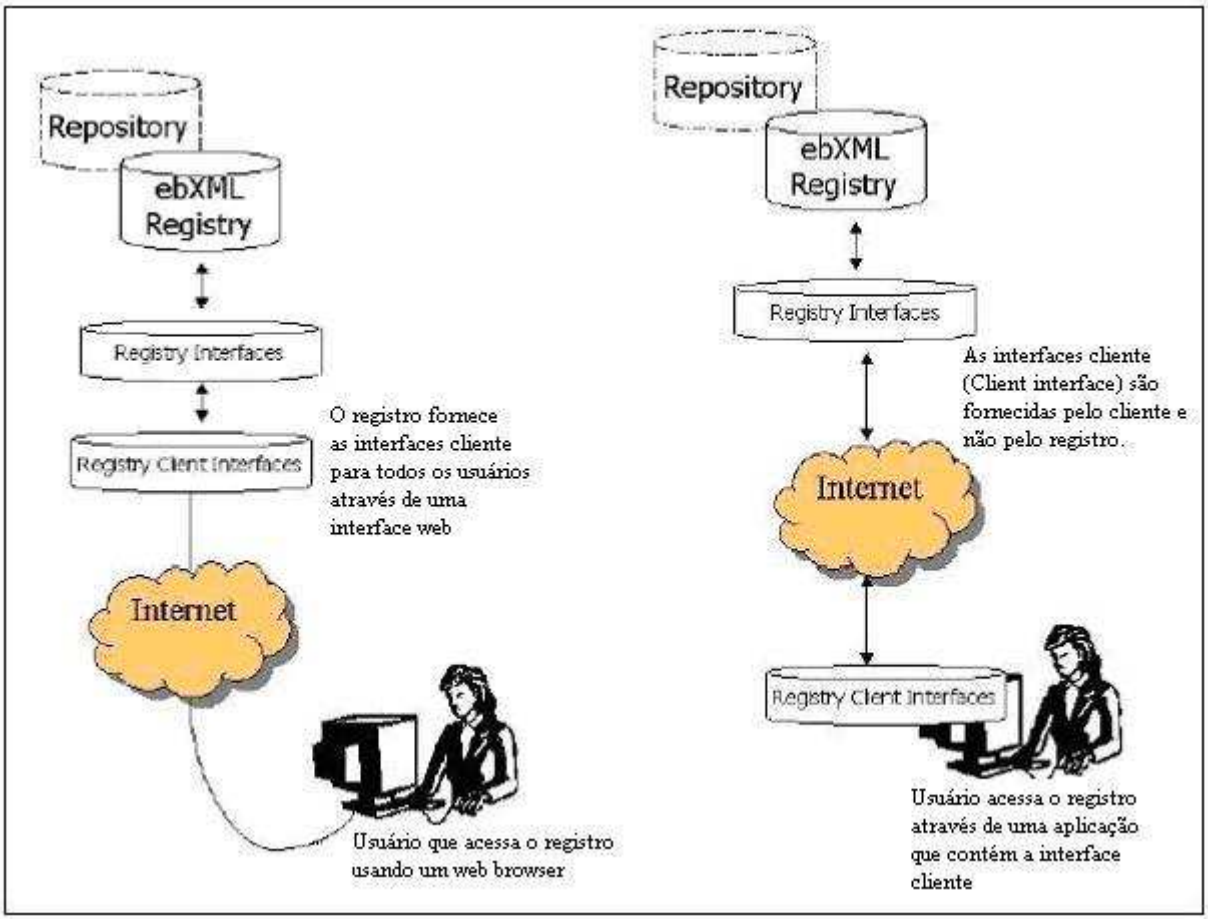
Fonte: ebRS (2001, p. 17)
Figura 12 – Arquitetura do Registro ebXML

Os clientes do registro podem acessar suas interfaces de duas formas:

a) através do SOAP usando o protocolo HTTP;
b) através do serviço de mensagens ebXML ou *ebXML Messaging Service*.'

De acordo com ebRS (2001), um registro pode implementar uma ou ambas as formas de acesso. O cliente utiliza o método de acesso que lhe seja mais apropriado.

A interface RC pode ser implementada de duas formas: local ao registro, onde a implementação do registro é transparente, por exemplo, o usuário acessa o registro através de um web site da internet ou local para o usuário, onde a interface de acesso RC tenha sido implementada na sua organização. A Figura 13 a seguir, demonstra as duas formas de implementar a interface de acesso RC.



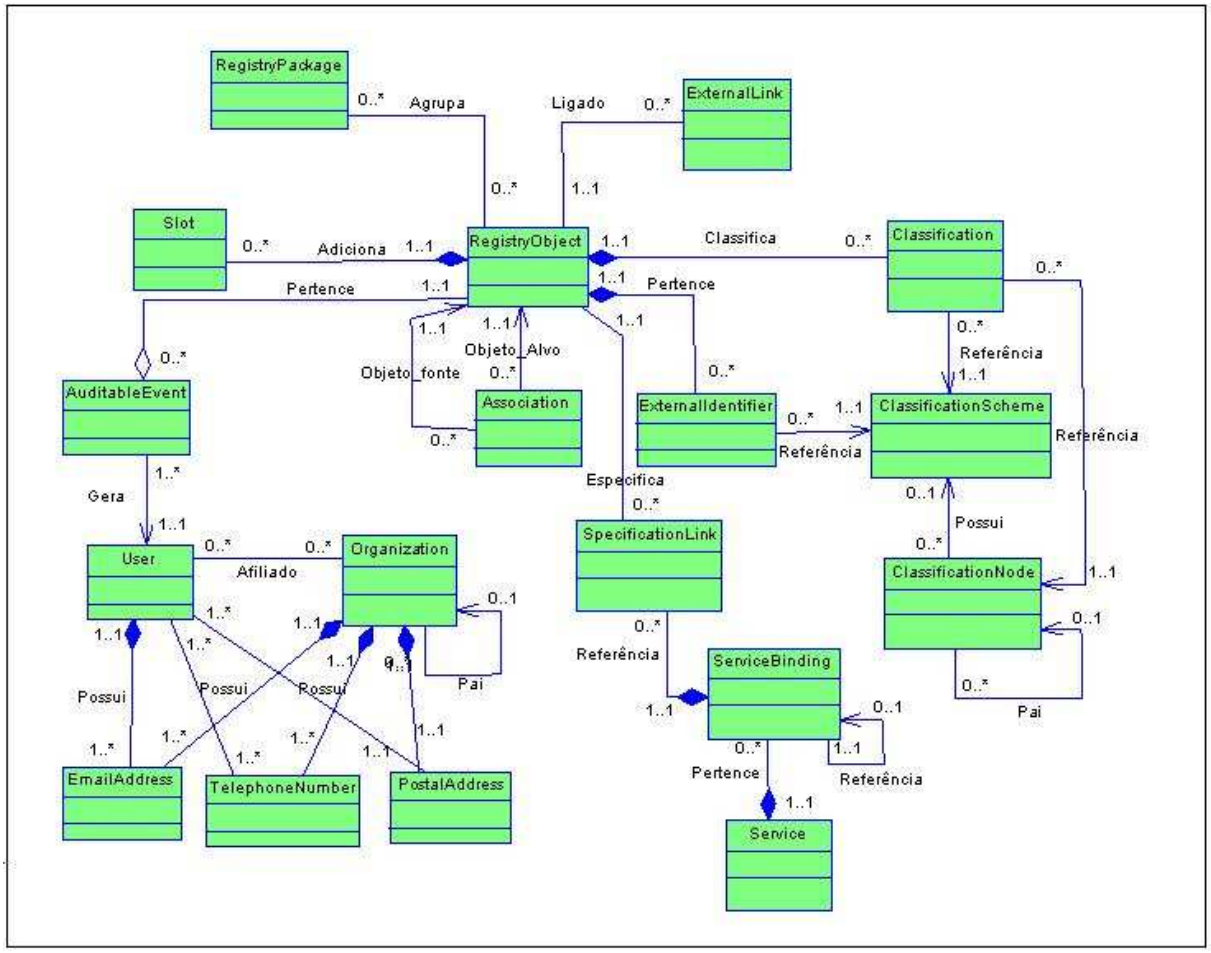
Fonte: adaptado de ebRS (2001, p. 23)

Figura 13 – Formas de acesso ao registro ebXML

O esquema XML definido pelo grupo de trabalho registro e repositório para o envio de requisições ao registro pode ser visto no Anexo D.

Para facilitar a visualização o diagrama de classes foi dividido em dois. O diagrama da Figura 14, demonstra os relacionamentos entre as classes e o diagrama da Figura 15, demonstra a hierarquia entre as mesmas.

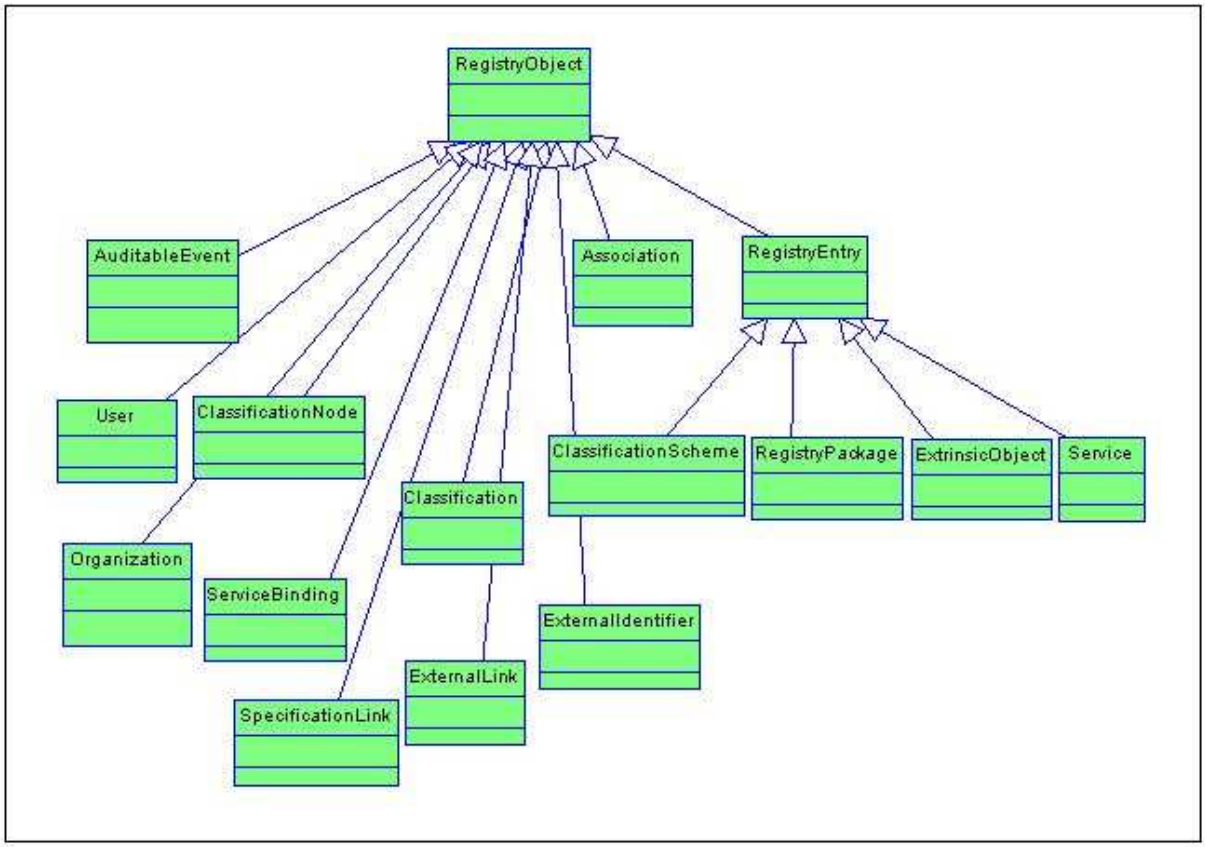
Estas classes têm como objetivo fornecer uma estrutura para suportar as solicitações dos clientes do registro ebXML.



Fonte: fonte adaptado de ebRIM(2002, p.11)

Figura 14 - Diagrama de classes do modelo de informações do registro ebXML

O esquema XML que mapeia estas classes do diagrama acima foi definido pelo grupo de trabalho registro e repositório e pode ser encontrado em (RIM, 2002).



Fonte: fonte adaptado de ebRIM(2002, p.15)

Figura 15 - Diagrama de classes demostrando a hierarquia entre as classes

Nas próximas seções, na forma de tabelas, será descrito cada classe, como também seus atributos e métodos, propostos pelo padrão (ebRIM, 2002).

2.4.7.1 Classe RegistryObject

A classe *RegistryObject* é uma classe base utilizada pela maioria das classes dentro do modelo. Provê os mínimos atributos para os objetos do registro. Também provê métodos para acessar objetos relacionados. O significado de cada atributo pode ser visto na Tabela 1.

Tabela 1 – Atributos da classe RegistryObject

Nome	**Descrição**	**Requerido**	**Especificado pelo**
accessControlPolicy	Cada objeto do registro pode ter uma política de controle de acesso que define o que é permitido fazer com o objeto.	Não	Registro
description	Cada objeto do registro pode ter uma descrição textual.	Não	Cliente
Id	Cada objeto do registro deve ter um id universalmente sem igual. Objetos do registro utilizam este atributo para referenciar outros objetos.	Sim	Cliente ou Registro
Name	Cada objeto do registro pode ter um nome.	Não	Cliente
objectType	Cada objeto do registro deve ter um *objectType*. Com exceção do objeto *ExtrincicObject* onde é o usuário que define o tipo e o dado associado, todas os outros objetos tem como valor de objectType o nome da classe.	Sim	Registro

A Tabela 2 a seguir mostra os valores pré-definidos para os tipos de objetos do registro.

Tabela 2 – Valores pré-definidos para o tipo de objeto do registro

Nome	**Descrição**
Unknown	Um *ExtrinsicObject* que identifica o conteúdo do objeto quando o tipo é desconhecido ou não especificado.
CPA	Um *ExtrinsicObject* deste tipo identifica um CPA, documento representando um acordo técnico entre parceiros comerciais e como eles planejam se comunicar.
CPP	Um *ExtrinsicObject* deste tipo identifica um CPP, documento que provê informações de um parceiro de *eBusiness.* Para mais detalhes ver ebRIM(2002).
Process	Um *ExtrinsicObject* deste tipo identifica um documento de descrição do processo.
SoftwareComponent	Um *ExtrinsicObject* deste tipo identifica um componente de software (exemplo, EJB, ou biblioteca de classes, etc.).
UMLModel	Um *ExtrinsicObject* deste tipo identifica um modelo UML.
XMLSchema	Um *ExtrinsicObject* deste tipo identifica um XML schema (DTD, XSD, etc.).
RegistryPackage	Identifica um objeto *RegistryPackage*.
ExternalLink	Identifica um objeto *ExternalLink*.
ExternalIdentifier	Identifica um objeto *ExternalIdentifier*.
Association	Identifica um objeto *Association*.
ClassificationScheme	Identifica um objeto *ClassificationScheme*.
Classification	Identifica um objeto *Classification*.
AuditableEvent	Identifica um objeto *AuditableEvent*.
User	Identifica um objeto *User*.
Organization	Identifica um objeto *Organization*.
Service	Identifica um objeto *Service*.
ServiceBinding	Identifica um objeto *ServiceBinding*.
SpecificationLink	Identifica um objeto *SpecificationLink*.

Fonte: adaptado de ebRIM(2002, p. 19).

Seus métodos podem ser visto na Tabela 3.

Tabela 3 – Métodos da classe RegistryObject

Nome	**Descrição**
getAuditTrail	Retorna a auditoria completa das alterações realizadas no objeto.
getClassifications	Retorna a classificação do objeto.
getExternalIdentifiers	Retorna uma coleção de *ExternalIdentifiers* associados com o objeto.
getExternalLinks	Retorna os *ExternalLinks* associados com o objeto.
getRegistryPackages	Retorna os *RegistryPackages* que o objeto é membro.
getSlots	Retorna os *Slots* associados com o objeto.

2.4.7.2 Classe RegistryEntry

A classe *RegistryEntry* é uma classe base para classes do modelo que provê atributos adicionais além dos herdados da classe *RegistryObject*. Suas sub-classes tipicamente

necessitam de maior gerenciamento (por exemplo, necessitar de aprovação). Essas classes geralmente têm poucas instâncias, mas servem como raiz de uma composição de hierarquia consistindo vários objetos que são sub-classes de *ReigstryObject* mas não *RegistryEntry*. O significado dos seus atributos pode ser vistos na Tabela 4. Atributos herdados da classe *RegistryObject* foram omitidos.

Tabela 4 – Atributos da classe RegistryEntry

Nome	**Descrição**	**Requerido**	**Especificado pelo**
expiration	Este atributo define um tempo limite sobre a estabilidade provida pelo atributo stability. Uma vez o tempo de expiração sendo alcançado o atributo stability se torna *STABILITY_DINAMIC* que indica que o item pode ser alterado em qualquer hora e de qualquer maneira.	Não	Cliente
majorVersion	Indica o número de revisão do objeto. Inicialmente é atribuído 1 (um), sendo atualizado pelo registro sempre que o objeto sofre alteração.	Sim	Registro
minorVersion	Indica um número de revisão secundário. Inicialmente é atribuído 0 (zero), sendo atualizado pelo registro sempre que o objeto sofre alteração.	Sim	Registro
stability	Indica a estabilidade do objeto.	Não	Cliente
status	Indica o status do objeto no repositório.	Sim	Registro
userVersion	Possui a mesma função dos atributos *majorVersion* e *minorVersion*, a diferença é que o cliente que define.	Não	Cliente

A Tabela 5 abaixo, demonstra os valores pré-definidos para o atributo stability.

Tabela 5 – Valores pré-definidos do atributo stability

Nome	**Descrição**
Dynamic	Indica que o conteúdo do objeto é dinâmico e pode ser alterado arbitrariamente pelo cliente a qualquer momento.
DynamicCompatible	Indica que o conteúdo do objeto é dinâmico e pode ser alterado pelo cliente a qualquer momento para um modo compatível anteriormente estabelecido.
Static	Indica que o conteúdo do objeto é estático e não pode ser alterado.

Fonte: adaptado de ebRIM(2002, p. 22).

A Tabela 6, demonstra os valores pré-definidos para o atributo *status*.

Tabela 6 – Valores pré-definidos do atributo status

Nome	**Descrição**
Submitted	Estado indica que o conteúdo do objeto foi submetido ao registro.
Approved	Estado indica que o conteúdo submetido do objeto ao registro foi aprovado.
Deprecated	Estado indica que o conteúdo submetido do objeto ao registro foi desaprovado.
Withdraw	Estado indica que o conteúdo do objeto foi retirado do registro.

Fonte: adaptado de ebRIM(2002, p. 23).

Esta classe não define nenhum novo método além dos herdados da classe *RegistryObject.*

2.4.7.3 Classe Slot

A classe *Slot* provê um modo dinâmico para adicionar atributos arbitrários aos objetos do registro. Esta habilidade de adicionar atributos arbitrários dinamicamente aos objetos do registro habilita extensabilidade ao modelo de informações do registro.

Cada objeto pode ter 0 (zero) ou mais *Slots*. Um *Slot* é composto de um nome, um tipo e uma coleção de valores. Seus atributos podem ser vistos na Tabela 7.

Tabela 7 – Atributos da classe Slot

Nome	**Descrição**	**Requerido**	**Especificado pelo**
name	Armazena o nome do *Slot*. O nome deve ser único no registro.	Sim	Cliente
slotType	Permite agrupar deferentes *Slots*.	Não	Cliente
Values	Coleção de valores. A coleção de valores pode ser vazia, desde que um *Slot* represente um atributo extensível cujo valor pode ser uma coleção de valores, conseqüentemente um *Slot* permite ter uma coleção de valores em lugar de um único valor.	Sim	Cliente

Esta classe não define nenhum novo método além dos herdados da classe *RegistryObject.*

2.4.7.4 Classe ExtrinsicObject

A classe *ExtrinsicObject* provê atributos que descrevem o conteúdo submetido cujo tipo não é internamente conhecido pelo registro, então deve ser descrito por meio de atributos adicionais (por exemplo, *mimeType*). Seus atributos podem ser vistos na Tabela 8.

Tabela 8 – Atributos da classe ExtrinsicObject

Nome	**Descrição**	**Requerido**	**Especificado pelo**
isOpaque	Indica que o conteúdo não é legível (por exemplo, codificado).	Não	Cliente
mimeType	Provê informação sobre o tipo de item submetido ao registro.	Não	Cliente

Exemplos de conteúdos descritos pela classe *ExtrinsicObject* inclui CPP's, descrições de processo de negócios e esquemas.

2.4.7.5 Classe RegistryPackage

A classe *RegistryPackage* agrupa logicamente objetos do registro que estão relacionados, mesmo sendo objetos de diferentes organizações. A classe *RegistryPackage* não define nenhum atributo novo além dos herdados da classe *RegistryObject e RegistryEntry*.

Esta classe define um novo método além dos herdados da classe *RegistryObject*, visto na Tabela 9.

Tabela 9 – Método da classe RegistryPackage

Nome	**Descrição**
getMemberObjects	Retorna os objetos membros do *RegistryPackage.*

2.4.7.6 Classe ExternalIdentifier

A classe *ExternalIdentifier* provê atributos adicionais para identificação dos objetos do registro (por exemplo, número DUNS, Previdência Social, ou um pseudônimo para a organização. Cada objeto do registro pode ter 0 (zero) ou mais instâncias de *ExternalIdentifier*. Seus atributos podem ser vistos na Tabela 10.

Tabela 10 – Atributos da classe ExternalIdentifier.

Nome	**Descrição**	**Requerido**	**Especificado pelo**
identificationScheme	Este atributo serve para referenciar um *ClassificationScheme*. Este *ClassificationScheme* define o espaço identificador (coleção de nomes, identificados por uma referência URI que é usada nos documentos XML como tipos de elementos e nomes de atributos) ao qual um identificador é definido usando o atributo *value* para o objeto do registro referenciado pelo atributo *registryObject.*	Sim	Cliente
registryObject	Define o objeto para o qual este é um *ExternalIdentifier.*	Sim	Cliente
Value	Armazena o valor do identificador (por exemplo, número DUNS).	Sim	Cliente

2.4.7.7 Classe ExternalLink

A classe *ExternalLink* utiliza *Uniform Resource Identifier* (URI) para associar conteúdos do registro com conteúdos armazenados fora dele. Por exemplo, uma organização que submete um DTD poderia usar um *ExternalLink* para associar o DTD com a sua *home page*. Seu atributo pode ser visto na Tabela 11. Os atributos herdados de sua classe base não são mostrados.

Tabela 11 – Atributos da classe ExternalLink

Nome	**Descrição**	**Requerido**	**Especificado pelo**
externalURI	Este atributo provê um URI para recursos externos do registro. Se o URI for um *Uniform Resource Locator* (URL), então o registro deve validar esta URL na hora que for submetido, evitando que o registro aceite um *ExternalLink* inválido.	Sim	Cliente

Esta classe define um novo método além dos herdados da classe *RegistryObject*, visto na Tabela 12.

Tabela 12 – Método da classe ExternalLink

Nome	**Descrição**
getLinkedObjects	Retorna os objetos ligados pelo *ExternalLink* para conteúdos de fora do registro.

2.4.7.8 Classe AuditableEvent

A classe *AuditableEvent* provê um registro de eventos de alterações feitas em um objeto do registro. Um objeto do registro é associado com uma coleção de instâncias de *AuditableEvent* que provê uma auditoria completa para este objeto.

Os eventos são registrados pelo registro quando o usuário, cria, atualiza, desaprova ou deleta um objeto. Pedidos de somente leitura não geram eventos. Nenhum evento é gerado para um objeto quando ele é classificado, atribuído a um *RegistryPackage* ou associado com outro objeto do registro. Seus atributos podem ser vistos na Tabela 13.

Tabela 13 - Atributos da classe AuditableEvent

Nome	**Descrição**	**Requerido**	**Especificado pelo**
eventType	Identifica o tipo de evento.	Sim	Registro
registryObject	Identifica o objeto do registro que foi afetado pelo evento.	Sim	Registro
timestamp	Registra a data e hora que o evento ocorreu.	Sim	Registro
User	Identifica o usuário que enviou o pedido que gerou o evento.	Sim	Registro

A Tabela 14 a seguir, mostra os valores pré-definidos para o atributo *eventType.* Um registro deve suportar esses eventos.

Tabela 14 – Valores pré-definidos para o atributo eventType

Nome	**Descrição**
Created	Evento gerado quando é criado um objeto no registro.
Deleted	Evento gerado quando é deletado um objeto do registro.
Deprecated	Evento gerado quando é desaprovado um objeto do registro.
Updated	Evento gerado quando é atualizado um objeto do registro.
Versioned	Evento gerado quando é alterado a versão de um objeto do registro.

Fonte: adaptado de ebRIM(2002, p. 28).

Esta classe não define nenhum novo método além dos herdados da classe *RegistryObject.*

2.4.7.9 Classe User

A classe *User* é utilizada para manter a auditoria de eventos dos objetos do registro. Seus atributos podem ser vistos na Tabela 15.

Tabela 15 – Atributos da classe User

Nome	**Descrição**	**Requerido**	**Especificado pelo**
address	Endereço postal. Instância da classe *PostalAddress.*	Sim	Cliente
emailAddressess	Coleção de instâncias da classe *EmailAddress.*	Sim	Cliente
organization	Identifica a organização que o usuário pertence.	Sim	Cliente
personName	Nome do usuário. Uma instância da classe *PersonName*.	Sim	Cliente
telephoneNumbers	Uma coleção de números telefônicos do usuário. Instâncias da classe *TelephoneNumber*.	Sim	Cliente
url	Endereço URL associado com o usuário.	Não	Cliente

Esta classe não define nenhum novo método além dos herdados da classe *RegistryObject.*

2.4.7.10 Classe Organization

A classe *Organization* provê informações sobre a organização que submete objetos ao registro. Seus atributos podem ser vistos na Tabela 16.

Tabela 16 – Atributos da classe Organization

Nome	**Descrição**	**Requerido**	**Especificado pelo**
Address	Endereço postal. Instância da classe *PostalAddress.*	Sim	Cliente
Parent	Utilizado para referenciar uma organização pai.	Não	Cliente
primaryContact	Identifica o usuário que deve ser primariamente contatado pela organização.	Sim	Cliente
telephoneNumbers	Uma coleção de números telefônicos do usuário. Instâncias da classe *TelephoneNumber*.	Sim	Cliente

Esta classe não define nenhum novo método além dos herdados da classe *RegistryObject.*

2.4.7.11 Classe PostalAddress

A classe *PostalAddress* é uma simples classe reutilizável que define atributos de um endereço postal. Seus atributos podem ser vistos na Tabela 17.

Tabela 17 – Atributos da classe PostalAddress

Nome	**Descrição**	**Requerido**	**Especificado pelo**
city	Armazena a cidade.	Não	Cliente
country	Armazena o país.	Não	Cliente
postalCode	Armazena o código postal.	Não	Cliente
state	Armazena o estado.	Não	Cliente
street	Armazena a rua.	Não	Cliente
streetNumber	Armazena o número da rua.	Não	Cliente

Esta classe define apenas um método, visto na Tabela 18.

Tabela 18 – Método da classe PostalAddress

Nome	**Descrição**
getSlots	Retorna os *Slots* associados a este objeto.

2.4.7.12 Classe TelephoneNumber

A classe *TelephoneNumber* é uma simples classe reutilizável que define atributos de numero telefônico. Seus atributos podem ser vistos na Tabela 19.

Tabela 19 – Atributos da classe TelephoneNumber

Nome	**Descrição**	**Requerido**	**Especificado pelo**
areaCode	Armazena a código da área.	Não	Cliente
countryCode	Armazena o código do país.	Não	Cliente
extension	Armazena o número de extensão.	Não	Cliente
number	Armazena o número.	Não	Cliente
phoneType	Armazena o tipo de telefone (por exemplo, casa, escritório).	Não	Cliente
url	Armazena o endereço URL associado com o telefone.	Não	Cliente

2.4.7.13 Classe PersonName

A classe *PersonName* simplesmente define atributos de um nome pessoal. Seus atributos podem ser visto na Tabela 20.

Tabela 20 – Atributos da classe PersonName

Nome	**Descrição**	**Requerido**	**Especificado pelo**
firstName	Armazena o primeiro nome.	Não	Cliente
lastName	Armazena o último nome.	Não	Cliente
middleName	Armazena o nome do meio.	Não	Cliente

Esta classe não define nenhum método.

2.4.7.14 Classe Service

A classe *Service* provê informações de serviços, como por exemplo, *web services*. Esta classe não define nenhum novo atributo além dos herdados das classes *RegistryEntry* e *RegistryObject.*

Esta classe define um novo método além dos herdados da classe *RegistryObject*, visto na tabela 21.

Tabela 21 – Método da classe Service

Nome	**Descrição**
getServiceBindings	Retorna os *ServiceBindings* definidos para o objeto.

2.4.7.15 Classe ServiceBinding

A classe *ServiceBinding* representa informações técnicas sobre um modo específico de acessar um serviço. Um serviço tem uma coleção de *ServiceBindings.* A descrição atribuída ao *ServiceBinding* provê detalhes sobre o relacionamento entre várias ligações de especificação (classe *SpecificationLink*) que inclui a ligações de serviço (classe *ServiceBinding*). Esta descrição pode ser utilizada para compreender como configurar um sistema. Seus atributos podem ser visto na Tabela 22.

Tabela 22 – Atributos da classe ServiceBinding

Nome	**Descrição**	**Requerido**	**Especificado pelo**
accessURI	Define o URI para acessar o *ServiceBinding.* Este atributo é ignorado se o atributo *targetBinding* for informado. Se o URI for um URL, então o registro deve validar esta URL na hora que for submetido, evitando que o registro aceite um *ServiceBinding* inválido.	Não	Cliente
targetBinding	Define referências a outros *ServiceBindings.* Um *targetBinding* pode ser especificado quando um serviço for direcionado a outro serviço. Este atributo permite o *rehosting* de um serviço para outro provedor de serviço.	Não	Cliente

Esta classe define um novo método além dos herdados da classe *RegistryObject*, visto na tabela 23.

Tabela 23 – Método da classe ServiceBinding

Nome	**Descrição**
getSpecificationLinks	Retorna os *SpecificationLinks* definidos para o objeto.

2.4.7.16 Classe SpecificationLink

A classe *SpecificationLink* provê o acoplamento entre um *ServiceBinding* e uma de suas especificações técnicas que descrevem como usar o serviço usando o *ServiceBinding.* Por exemplo, um *ServiceBinding* pode ter um *SpecificationLink* que descreve como acessar o serviço utilizando uma especificação técnica na forma de um documento WSDL. Seus atributos podem ser visto na Tabela 24.

Tabela 24 – Atributos da classe SpecificationLink

Nome	**Descrição**	**Requerido**	**Especificado pelo**
specificationObject	Provê uma referência para um objeto do registro que provê uma especificações técnica para o *ServiceBinding* pai. Geralmente, este é um *ExtrinsicObject* que representa uma especificação técnica (como por exemplo, um documento WSDL).	Sim	Cliente
usageDescription	Provê uma descrição textual de como usar o atributo *usageParameters* .	Não	Cliente
usageParameters	Especifica os parâmetros necessários para utilizar a especificação técnica (como por exemplo, um documento WSDL) referenciado pelo atributo *specificationObject.*	Não	Cliente

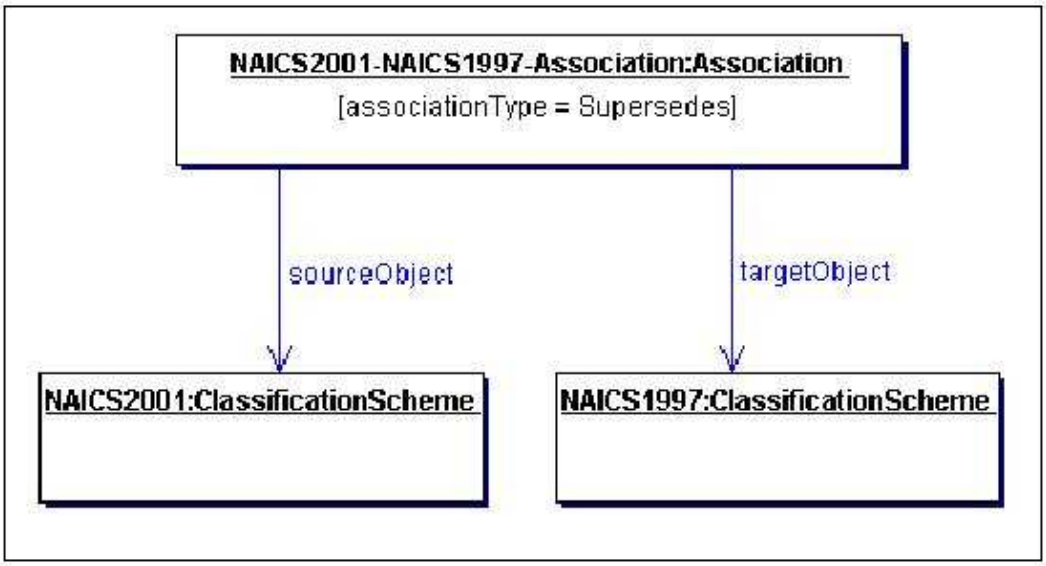
Esta classe não define nenhum novo método além dos herdados da classe *RegistryObject.*

2.4.8 ASSOCIAÇÃO DE OBJETOS DO REGISTRO

Um objeto do registro pode ser associado com 0 (zero) ou mais objetos. O modelo de informação (ebRIM, 2002) define uma classe de Associação (classe *Association*), que pode ser usada para associar qualquer dois objetos do registro.

Um exemplo de associação entre objetos do registro pode ser entre dois *ClassificationScheme,* onde um *ClassificationScheme* substitui o outro. Este pode ser o caso quando uma nova versão de um *ClassificaionScheme* é submetido.

Na Figura 16 a seguir, podemos ver a associação entre uma versão nova do NAICS *ClassificationScheme* e a versão mais velha do mesmo.



Fonte: ebRIM(2002, p. 37)

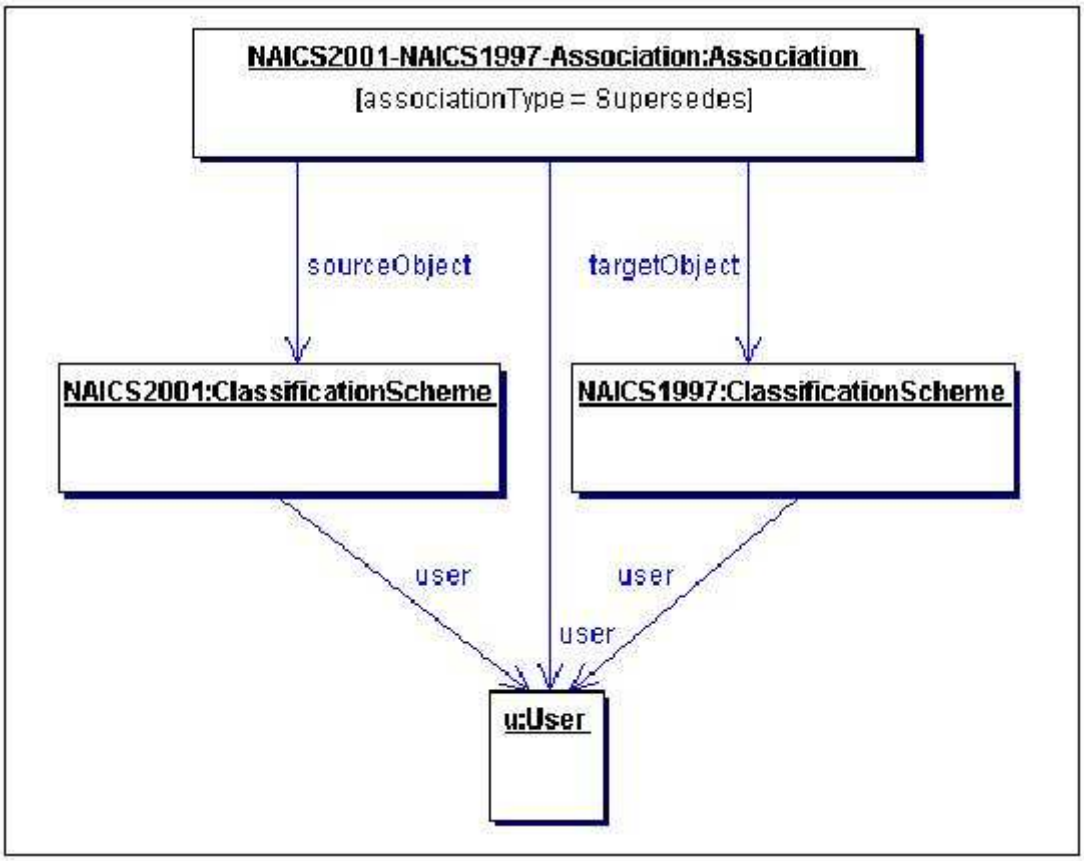
Figura 16 – Exemplo de associações entre dois objetos do registro

Uma associação é representada entre um objeto fonte e um objeto alvo. Estes atributos são chamados de *sourceObject* e *targetObject* como determina a dinâmica direcional de uma associação. No exemplo da Figura 16, vemos que a versão mais nova do NAICS *ClassificationScheme* é o *sourceObject* e a versão mais velha é o *targetObject*, isto porque o tipo da associações *(associationType=Supersedes)* indica que o *sourceObject* substitui o *targetObject* (e não ao contrário). Todas as associações devem ter um tipo de associação.

2.4.8.1 Associação Intramural

Um caso comum para a classe de associação é quando um usuário “u” cria uma associação “a” entre dois objetos do registro “o1” e “o2” onde a associação “a” e os objetos “o1” e “o2” são criados pelo mesmo usuário “u”. Este é o caso de associação mais simples, onde o associação é entre dois objetos que são do mesmo usuário que é o mesmo usuário que definiu a associação. Esta associação é chamada de Associação Intramural.

A Figura 17, extende a associação representada na Figura 16, para um caso de associação intramural.



Fonte: ebRIM(2002, p. 38)

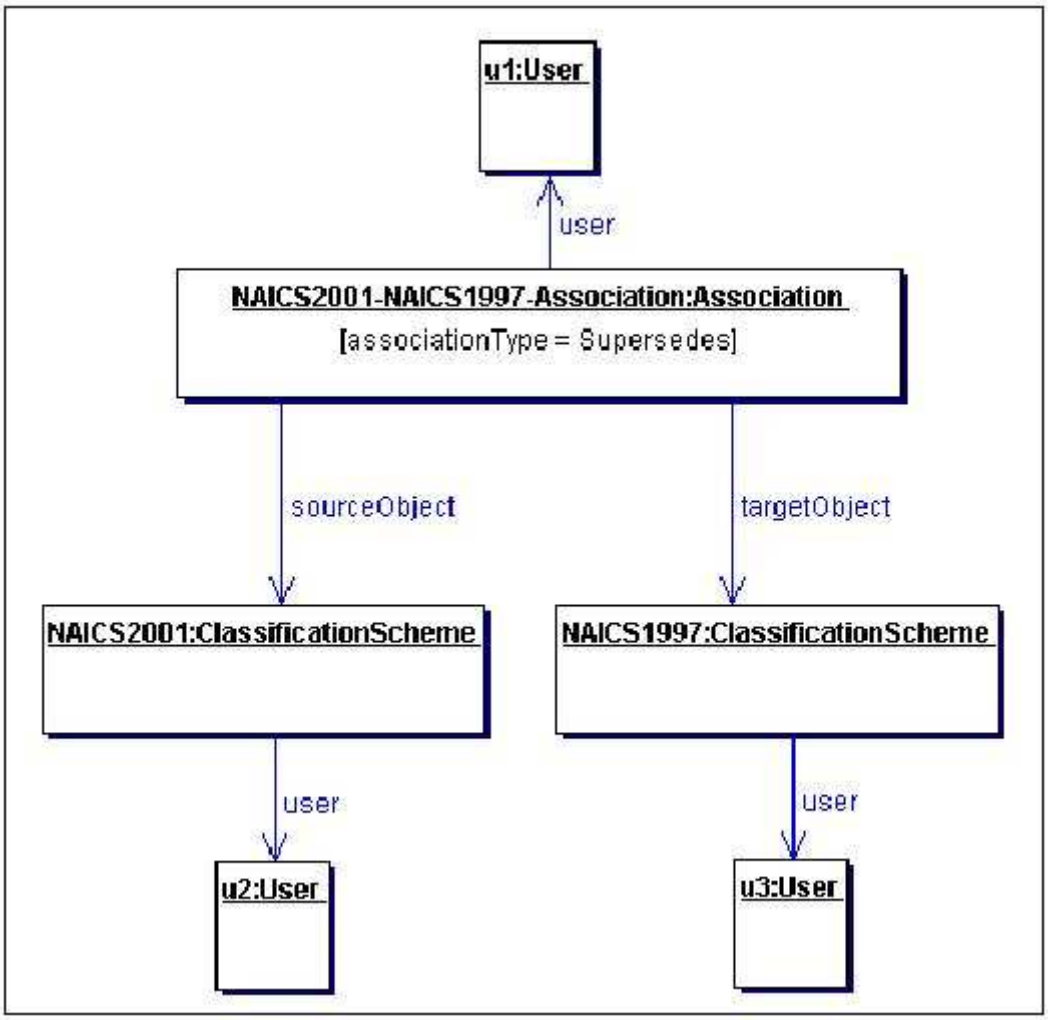
Figura 17 – Exemplo de um caso de associação intramural

2.4.8.2 Associação Extramural

O modelo de informação também permite casos de uso mais sofisticados. Por exemplo, um usuário “u1” cria uma associação “a” entre dois objetos do registro “o1” e “o2”, onde a associação “a” pertence ao usuário “u1”, mas os objetos “o1” e “o2” pertencem aos usuários “u2” e “u3” respectivamente. Neste caso de uso uma associação é definida onde ambos os objetos que estão sendo associados pertencem a um usuário diferente do usuário que definiu a associação. Esta associação é chamada de Associação Extramural.

A classe de associação (classe *Association*) define um método chamado *isExtramural* que retorna “true” se a associação é uma associação extramural.

A Figura 18 a seguir, extende a associação reprensentada na Figura 16, para um caso de associação extramural.



Fonte: ebRIM(2002, p. 39)

Figura 18 – Exemplo de um caso de associação extramural

2.4.8.3 Confirmação de uma associação

Uma associação pode precisar de confirmação pelas partes cujos objetos estão envolvidos numa associação, como o *sourceObject* e o *targetObject*. Associações do caso intramural são consideradas confirmadas implicitamente.

No caso de uma associação extramural, pode ser pensada como confirmação unilateral que não pode ser vista como verdade até que a outra parte envolvida faça a confirmação (usuários “u2” e “u3” da seção anterior). Para confirmar a associação cada parte (parte que possui o *sourceObject* ou *targetObject*) tem que submeter uma associação idêntica (associação clone) confirmando a associação utilizando um *SubmitObjectRequest* definido em

(ebRS, 2001). A associação clone tem que ter o mesmo *id* que a associação original. Uma associação extramural é deletada como qualquer outro objeto do registro, usando o *RemoveObjectRequest* como definido em (ebRS, 2001). Porém, em alguns casos deletando uma associação extramural ela pode não ser deletada de fato, ao invés disso, inverte seu estado de confirmada para não confirmada.

Uma associação sempre deve ser deletada, quando a solicitação for do proprietário da associação, independente do estado de confirmação. No caso de uma associação extramural as partes proprietárias dos objetos *source/target* devem ficar com o estado não confirmado, quando não forem proprietários da associação.

2.4.8.4 Classe Association

A classe *Association* é utilizada para definir relações *many-to-many* para os objetos do registro no modelo de informação. Seus atributos podem ser visto na Tabela 25.

Tabela 25 – Atributos da classe Association

Nome	**Descrição**	**Requerido**	**Especificado pelo**
associationType	Identifica o tipo de associação.	Sim	Cliente
sourceObject	Referência o objeto do registro que é a fonte da associação.	Sim	Cliente
targetObject	Referência o objeto do registro que é o alvo da associação.	Sim	Cliente
isConfirmedBySourceOwner	Identifica se é verdade que foi confirmada a associação pelo dono do *sourceObject.*	Não	Registro
isConfirmedByTargetOwner	Identifica se é verdade que foi confirmada a associação pelo dono do *targetObject.*	Não	Registro

A Tabela 26 a seguir, mostra os valores pré-definidos para o atributo *associationType.* Um registro deve suportar estes tipos de associações.

Tabela 26 – Valores pré-definidos para o atributo associationType

Nome	**Descrição**
RelatedTo	Define que o objeto fonte é relacionado com o objeto alvo.
HasMember	Define que o objeto fonte *(RegistryPackage)* tem o objeto alvo como um membro.
ExternallyLinks	Define que o objeto fonte *(ExternalLink)* tem ligação externa com o objeto alvo. Reservado para associações de *ExternalLinks* com *RegistryEntries.*
Contains	Define que o objeto fonte contém o objeto alvo.
EquivalentTo	Define que o objeto fonte é equivalente ao objeto alvo.
Extends	Define que o objeto fonte herda de ou é uma especialização do objeto alvo.
Implements	Define que o objeto fonte implementa as funcionalidades definidas pelo objeto alvo.
InstanceOf	Define que o objeto fonte é uma instância do objeto alvo.
Supersedes	Define que o objeto fonte (mais novo) toma o lugador do objeto alvo (mais velho), exemplo visto anteriormente.
Uses	Define que o objeto fonte usa o objeto alvo de alguma maneira.
Replaces	Define que o objeto fonte substitui o objeto alvo.
SubmitterOf	Define que o objeto fonte *(Organization)* submeteu o objeto alvo.
ResponsibleFor	Define que o objeto fonte *(Organization)* é a responsável pela manutenção contínua do objeto alvo.
OffersService	Define que o objeto fonte *(Organization)* oferece o objeto alvo *(Service)* como um serviço. Reservado para uso em indicações onde uma Organização oferece um serviço.

2.4.9 CLASSIFICAÇÃO DE OBJETOS DO REGISTRO

Um objeto pode ser classificado de diversas manerias. Por exemplo, um mesmo CPP pode ser classificado pela indústria, por seus produtos vendidos e por sua localização geográfica.

De forma geral um esquema de classificação *(classificationScheme)* pode ser visto como uma árvore de classificação.

2.4.9.1 Classe classificationScheme

A classe *classificationScheme* provê atributos que descrevem de forma estruturada como classificar ou categorizar objetos no registro. A estrutura do esquema de classificação pode ser definida como interna ou externa ao registro. Um exemplo muito comum de um esquema de classificação na área da ciência é a classificação dos seres vivos. Seus atributos podem ser vistos na Tabela 27.

Tabela 27 – Atributos da classe ClassificationScheme

Nome	**Descrição**	**Requerido**	**Especificado pelo**
isInternal	Atributo identifica se o *classificationScheme* é interno ou externo ao registro. Permite ao registro manter a consistência ao longo do ciclo de vida do *classificationScheme.*	Sim	Cliente
nodeType	Define a estrutura de nodos que o esquema irá representar. Este atributo é uma enumeração de valores.	Sim	Cliente

A Tabela 28 a seguir, mostra os valores pré-definidos para o atributo *nodeType.* Estes valores podem ser extendidos para atender novas necessidades.

Tabela 28 – Valores pré-definidos para o atributo nodeType

Nome	**Descrição**
UniqueCode	Define que cada nodo tem um código sem igual.
EmbeddedPath	Define que o código especificado *(UniqueCode)* para cada nodo especifica o seu caminho ao mesmo tempo.
NonUniqueCode	Em alguns casos os nodos não possuem código sem igual. Para tal, este valor especifica que é necessário informar o caminho completo para identificar o nodo.

2.4.9.2 Classe ClassificationNode

A classe *ClassificationNode* é utilizada para definir uma estrutura de árvores, onde cada nodo da árvore é um *classificationNode.* Estas árvores de classificação são construídas através de instâncias de *classificationNode* abaixo de uma instância de *classificationScheme.* Seus atributos podem ser visto na Tabela 29.

Tabela 29 – Atributos da classe ClassificationNode

Nome	**Descrição**	**Requerido**	**Especificado pelo**
parent	Utilizado para referenciar um nodo *(classificationNode)* pai ou um esquema de classificação *(classificationNode)* no caso de ser o primeiro nodo da árvore.	Não	Cliente
code	Este atributo contém um código dentro de um esquema de codificação padrão.	Não	Cliente
path	Este atributo deve ser presente quando um nodo *(classificatioNode)* é recuperado do registro. Este atributo deve ser ignorado quando for informado pelo cliente.	Não	Registro

2.4.9.3 Classe Classification

A classe *Classification* é utilizada para classificar objetos referenciando nodos definidos dentro de um esquema de classificação particular. Uma classificação interna irá referenciar o nodo diretamente, através do seu atributo *id*, enquanto uma classificação externa irá referenciar o nodo inderetamente através da especificação representada pelo seu valor que é único dentro do esquema de classificação externo. Seus atributos podem ser visto na Tabela 30.

Tabela 30 – Atributos da classe Classification

Nome	**Descrição**	**Requerido**	**Especificado pelo**
classificationScheme	Atributo utilizado para referenciar um esquema de classificações *(classificationScheme).*	Sim, para classificações externas.	Cliente
classificationNode	Atributo utilizado para referenciar um nodo *(classificationNode).*	Sim, para classificações internas.	Cliente
classifiedObject	Atributo utilizado para referenciar o objeto classificado por esta classificação.	Sim	Cliente
nodeRepresentation	Este atributo é a representação de um elemento de um esquema de classificação.	Sim, para classificações externas.	Cliente

2.4.9.4 Exemplo de um esquema de classificação

O exemplo do Quadro 6 a seguir, demonstra a submissão de um esquema de classificação pela localização geográfica. A submissão é feita através de um pedido *SubmitObjectsRequest* como definido em (ebRS, 2001).

```
<?xml version = "1.0" encoding = "UTF-8"?>
<SubmitObjectsRequest
    xmlns = "urn:oasis:names:tc:ebxml-regrep:registry:xsd:2.0"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:rim = "urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.0"
    xmlns:rs = "urn:oasis:names:tc:ebxml-regrep:registry:xsd:2.0">

    <rim:LeafRegistryObjectList>
        <rim:ClassificationScheme
             id="urn:uuid:C99A818D-355B-4f05-B873-C0ADDF760E9C"
             isInternal="true" nodeType="UniqueCode">
             <rim:Name>
```

```
                <rim:LocalizedString value="Geografia" />
            </rim:Name>
            <rim:Description>
                <rim:LocalizedString value="Simples exemplo de um
                                esquema de localização geográfica" />
            </rim:Description>
            <rim:ClassificationNode
                id="urn:uuid:C6A402F2-021A-46fb-8891-78265C11F2F7"
                parent="urn:uuid:C99A818D-355B-4f05-B873-C0ADDF760E9C"
                code="AmericaNorte">
                <rim:ClassificationNode
                    id="urn:uuid:5F19CCC3-8EC0-470c-956B-CCF6E3D1079A"
                    parent="urn:uuid:C6A402F2-021A-46fb-8891-
78265C11F2F7"
                    code="EstadosUnidos" />
                <rim:ClassificationNode
                    id="urn:uuid:1C4013DF-154D-498d-897E-D43CF9C3BD34"
                    parent="urn:uuid:C6A402F2-021A-46fb-8891-
78265C11F2F7"
                    code="Canada" />
            </rim:ClassificationNode>
            <rim:ClassificationNode
                id="urn:uuid:B82C6A92-3C91-454d-B32D-49C682BED76F"
                parent="urn:uuid:C99A818D-355B-4f05-B873-C0ADDF760E9C"
                code="Asia">
                <rim:ClassificationNode
                    id="urn:uuid:B6893D31-1B79-435d-A08E-B0A963AA7741"
                    parent="urn:uuid:B82C6A92-3C91-454d-B32D-
49C682BED76F"
                    code="Japao">
                    <rim:ClassificationNode
                        id="urn:uuid:1E30EEA8-F6F2-4c93-BD68-
BF82978BD2C2"
                        parent="urn:uuid:B6893D31-1B79-435d-A08E-
B0A963AA7741"
                        code="Toquio" />
                </rim:ClassificationNode>
            </rim:ClassificationNode>
        </rim:ClassificationScheme>
    </rim:LeafRegistryObjectList>
</SubmitObjectsRequest>
```

Quadro 6 – Exemplo de submissão de um esquema de classificação

O esquema de classificação mostrado no Quadro 6, pode ser representado na forma de uma árvore de classificação. Esta árvore pode ser vista na Figura 19.

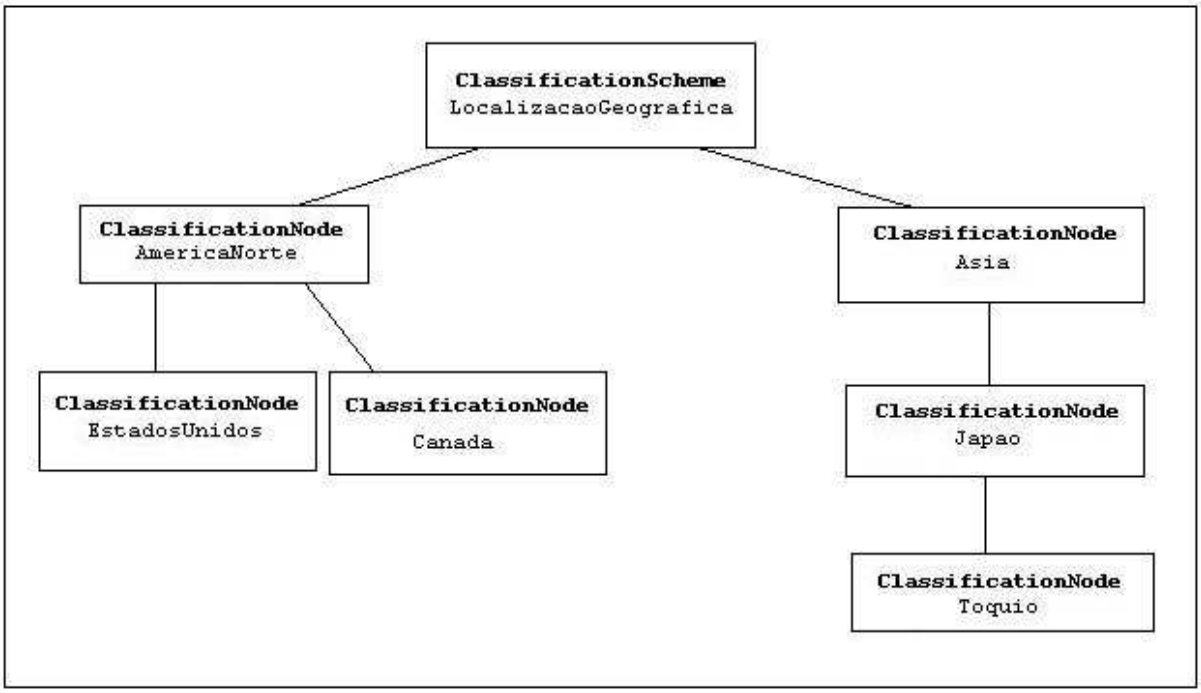


Figura 19 – Estrutra em árvore de um esquema de classificação geográfica

2.4.10 SERVIÇO DE MENSAGENS DO ebXML

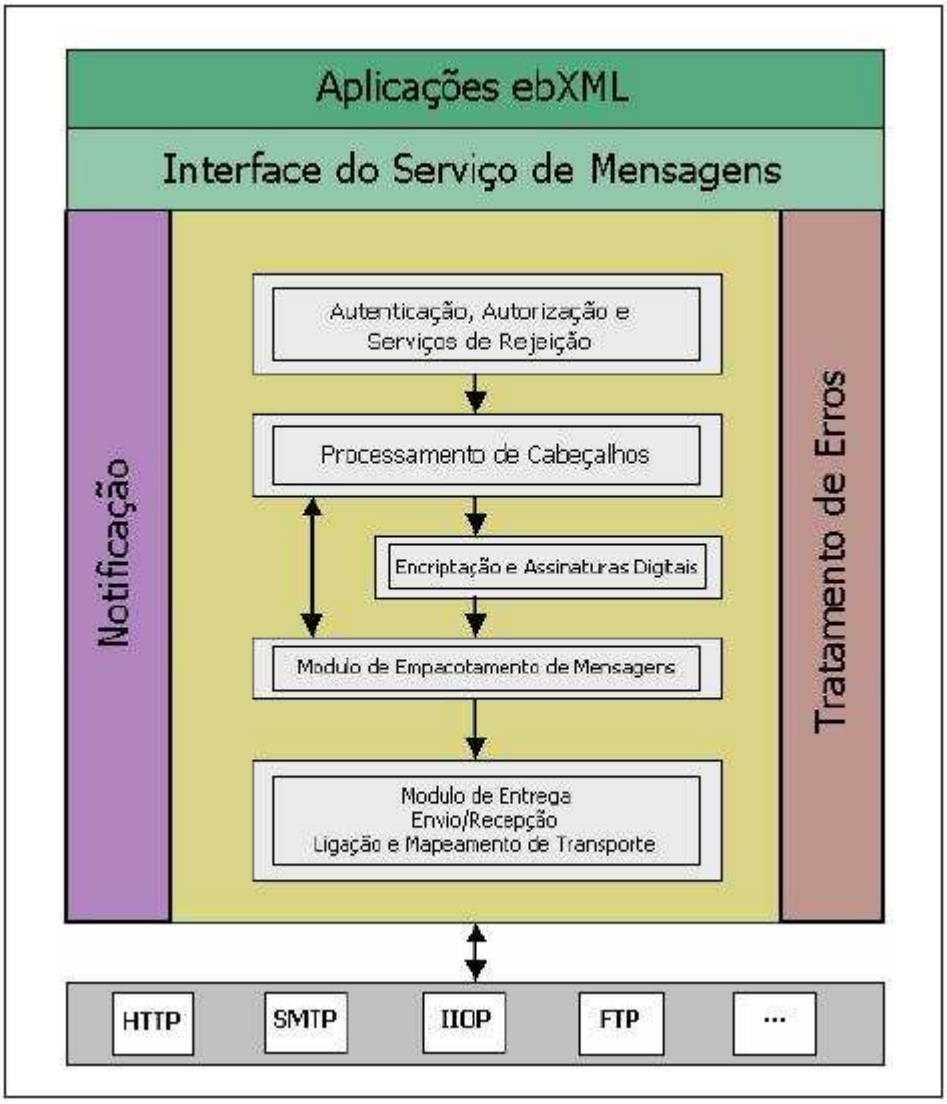
O serviço de mensagens ebXML provê um modo padrão seguro para a troca de mensagens de negócios entre parceiros de negócio sem confiar em tecnologias e soluções proprietárias. Uma mensagem ebXML contém uma estrutura para o cabeçalho da mensagem (necessário para o roteamento e entrega) e uma seção de *Payload.*

De acordo com ebTA (2001) o serviço de mensagens ebXML está dividido em 3 partes: (1) uma interface abstrata do serviço, (2) funções providas por camadas do serviço de mensagens e (3) mapeamento para serviços de transportes. O relacionamento entre interface abstrata, camadas do serviço de mensagens e serviço de transporte pode ser visto na Figura 20.

Fonte: adaptado de ebTA (2001, p. 28)

Figura 20 – Serviço de mensagens ebXML

O diagrama da Figura 21, descreve mais detalhadamente os módulos existentes dentro da arquitetura do serviço de mensagens ebXML. Estes módulos são organizados para indicar as suas inter-relações e dependências. Também descreve formatos para todas as mensagens trocadas por componentes ebXML incluindo os registros e aplicações de usuários concordantes com o ebXML não colocando nenhuma restrição ao conteúdo da mensagem.

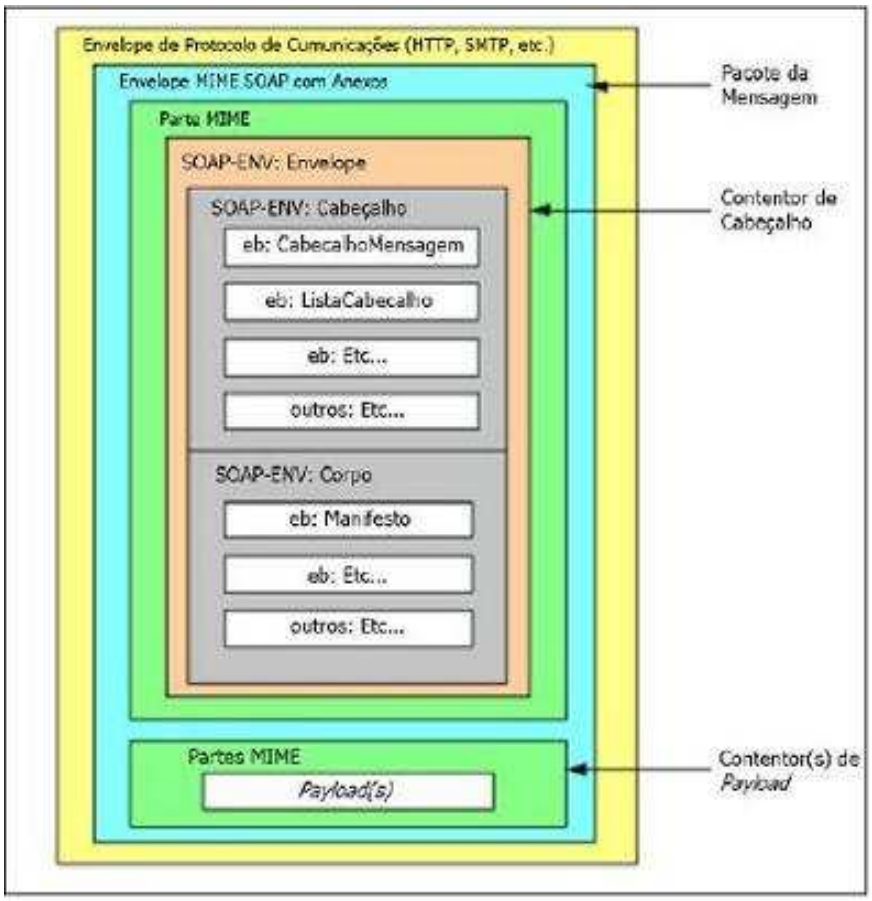


Fonte: Marques (2003, p. 76)

Figura 21 – Arquitetura do serviço de mensagens

Segundo ebTA (2001) o serviço de mensagens ebXML provê um mecanismo seguro, consistente e confiável para a troca de mensagens de usuários da infra-estrutura ebXML em cima de vários protocolos de transporte (por exemplo, SMTP, HTTP/S, FTP).

A estrutura da mensagem pode ser vista na Figura 22.



Fonte: Marques (2003, p. 78)

Figura 22 – Estrutura de uma mensagem ebXML

Uma mensagem ebXML é considerado um protocolo de comunicação estruturado e esta em conformidade com a especificação de mensagens SOAP com anexos (SOAPAttach).

O envelope da mensagem ebXML é empacotado usando o *Multiporpose Internet Mail Extension* (MIME) *Multipart/related*. O MIME é utilizada por causa da natureza diversa das informações trocadas entre os parceiros comerciais em *e-Business.* Por exemplo, uma transação comercial complexa entre dois ou mais parceiros comerciais onde se deseja trocar conteúdos que contenham uma ordem de documentos empresariais (XML ou outros formatos de documentos), imagens binárias, ou outra informação de negócio relacionada. Podem existir zero ou mais partes adicionais MIME *(Contentor's* de *Payload).*

2.4.11 TRABALHOS CORRELATOS

Existem várias propostas de padrão para a troca eletrônica de dados utilizando XML. Podem-se citar algumas delas:

a) Open Travel Alliance (OTA) - define estratégias para o intercâmbio de informações para a indústria de viagens;
b) HR-XML – dedica-se ao desenvolvimento de padrões para habilitar o *e-commerce* e a automação de troca de dados da área de Recursos Humanos;
c) RosettaNet – seu objetivo é desenvolver um conjunto de padrões para a troca de negócios eletrônicos em toda indústria, ajudando as indústrias a integrarem suas conversas de negócios eletrônicos;
d) *Commerce XML* (cXML) – define documentos padrões para definições de pedido, definições *PunchOut* (permite a organização manter o controle de seu processo interno de compra), definições de catálogo, definições de gerenciamento de inscrição e definições de recuperação de mensagens;
e) Grupo XML/EDI – seu objetivo não é substituir o EDI, mas definir padrões e práticas novas que utilizam à sintaxe XML. Sua meta é combinar XML e EDI para criar um novo paradigma poderoso que utilize os ambos pontos fortes;
f) BizTalk – iniciado pela Microsoft. Possui a intenção de guiar a criação e manutenção de esquemas de dados XML para permitr o comércio eletrônico e integração de aplicações.

Diversas *softwares houses* já desenvolveram soluções utilizando o padrão ebXML, por exemplo a empresa Sybase já disponibilizou no seu software de modelagem Power Designer a opção de criar Processos de Negócio, Trasações de Negócio no padrão ebXML.

Na *home page* do Grupo de Trabalho ebXML (ebXML, 2004) pode ser encontrado diversos estudos de caso de empresas que utilizaram o ebXML para desenvolver suas soluções de *e-Business.*

3 DESENVOLVIMENTO DO TRABALHO

Nas seções anteriores foram apresentadas as tecnologias utilizadas no desenvolvimento deste trabalho. Nesta seção será apresentado o uso dessas tecnologias no desenvolvimento do protótipo de registro ebXML.

O presente protótipo se baseou na especificação versão 2.1 do modelo de informação do registro (ebRIM, 2002) e na versão 2.0 da especificação dos serviços do registro (ebRS, 2001). Como as especificações são constantemente revistas e atualizadas para atender novas necessidades das organizações e o surgimento de novas tecnologias é muito provável que existam versões mais recentes que estas utilizadas no desenvolvimento do protótipo. Para verificar especificações mais recentes consultar (ebXML, 2004).

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O objetivo do protótipo é possibilitar que empresas submetam informações e que elas possam ser descobertas e recuperadas para facilitar transações B2B que desejam utilizar o padrão ebXML.

O presente protótipo implementou as interfaces LM e QM através de *web services* utilizando o protocolo SOAP, possibilitando que aplicações ebXML clientes, utilizem os serviços web para fornecerem seus perfis e assim realizarem a troca de informações entre os parceiros comerciais.

De acordo com a especificação ebRS (2001) um repositório deve permitir a capacidade dos clientes recuperarem as informações do registro de duas formas:

a) *Filter Query*: utiliza tags XML;
b) SQL *Query*: utiliza a linguagem SQL.

Pelo fato da especificação do esquema XML ser bastante extensa e o tempo para desenvolvimento do protótipo ser bastante limitado, o protótipo se limitou no desenvolvimento da requisição *OrganizationQueryType,* que possibilita descobrir as organizações cadastradas no protótipo de registro ebXML. Para mais detalhes sobre esta e demais requisições consultar (ebRS, 2001).

3.2 ESPECIFICAÇÃO

Nesta seção são abordadas as técnicas utilizadas para especificar o desenvolvimento do protótipo. A especificação do protótipo desenvolvido se baseou na técnica de orientação a objetos e utilizou-se da UML, através da ferramenta PowerDesigner 9 da Sybase.

3.2.1 CASOS DE USO

Para visualizar a interação com o protótipo desenvolvido, utilizou-se de Diagramas de Casos de Uso.

Este diagrama contém oito casos de uso possíveis e pode ser visto na Figura 23.

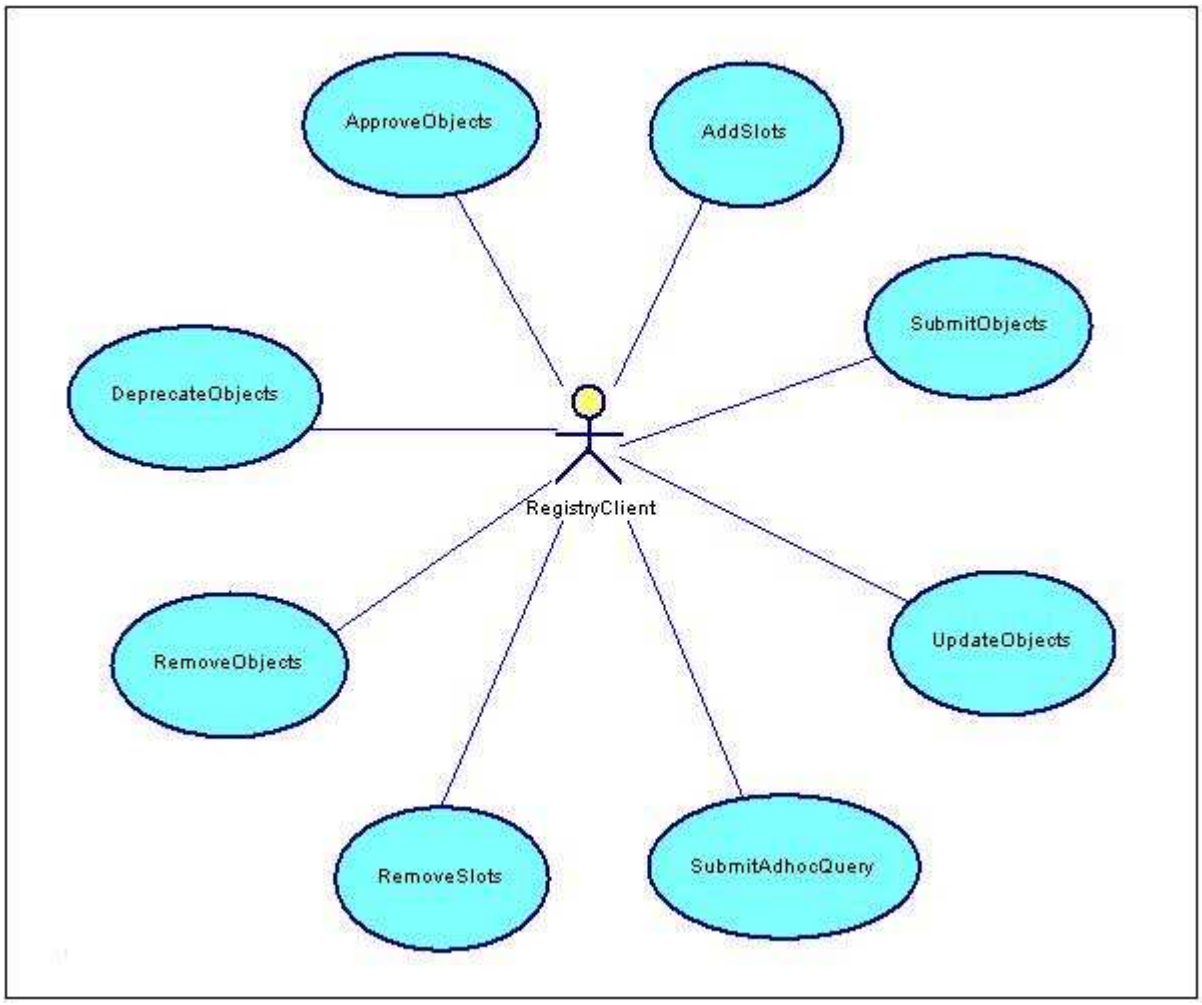


Figura 23 – Diagrama de casos de uso

Pode-se observar que todas as ações partem de uma aplicação cliente *(RegistryClient)* que deseja realizar *e-Business* através do padrão ebXML, podendo submeter, atualizar, remover ou consultar informações no registro.

Os casos de uso representam as ações possíveis, que são:

a) *SubmitObjects* – permite a aplicação cliente submeter informações ao registro ebXML;
b) *UpdateObjects* – permite a aplicação cliente atualiza informações existentes no registro ebXML;
c) *ApproveObjects* – permite aos clientes aprovar um ou mais itens existentes no registro ebXML;
d) *DeprecateObjects* – permite aos clientes desaprovar um ou mais itens existentes no registro ebXML;
e) *RemoveObjects* – permite aos clientes remover um ou mais itens existentes no registro ebXML;
f) *AddSlots* – permite aos clientes adicionar *slots* para um Registry Entry existente no registro ebXML;
g) *RemoveSlots* – permite aos clientes remover um ou mais s*lots* existentes no registro ebXML;
h) *SubmitAdhocQuery* – permite aos clientes submeter consultas ao registro ebXML.

3.2.2 DIAGRAMA DE CLASSES

O diagrama da Figura 24 a seguir, demonstra as classes, seus respectivos atributos e métodos para satisfazer as requisições solicitadas pelos clientes do registro ou *Registry Client.* Estas requisições podem ser para criar, atualizar ou modificar objetos do registro ebXML, conforme especificado em ebRS (2001). Estas classes são utilizadas pelo *web service LifeCycleManager.*

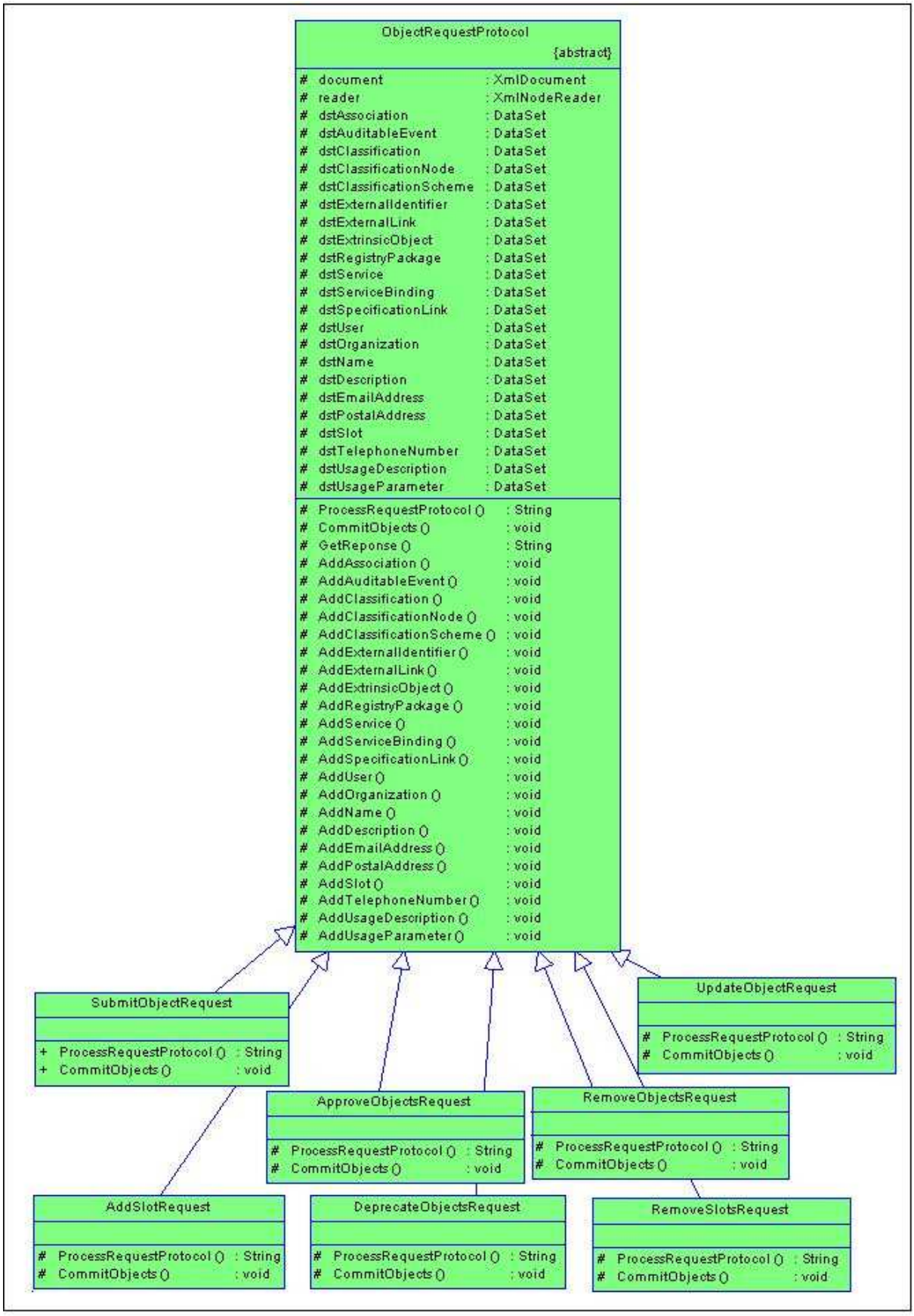


Figura 24 – Diagrama de classes

Como pode ser visto no diagrama da Figura 24, todas as classes do modelo são sub-classes da classe abstrata *ObjectRequestProtocol*. Esta classe além dos atributos *DataSets* (mini banco de dados localizado na memória, namespace *System.Data*) utilizados para persistir os objetos no banco de dados, possui os atributos *document* e *reader*. O atributo *document* é uma referência a um objeto *XmlDocument* que representa um documento XML. Este documento é armazenado como estruturas de árvore na memória. Essa estrutura de árvore é chamada de *Document Object Model* (DOM) ou Modelo de Objeto de Documento e o analisador sintático que cria esta estrutura é conhecido como Analisador Sintático DOM. O atributo *reader* é uma referência a um objeto *XmlNodeReader* que itera por cada nó no documento XML. No Visual Basic.NET as classes para criar, ler e manipular documentos XML estão localizadas no namespace *System.Xml*.

Esta classe provê dois métodos abstratos que devem ser sobrepostos pelas suas sub-classes, são eles:

a) ProcessRequestProtocol: método que processa a requisição do cliente do registro, que está em formato XML, armazenando os objetos nos seus respectivos *DataSets*, utilizando os métodos herdados da super classe, por exemplo, *addSlot* que adiciona ao *DataSet* um objeto *Slot;*
b) CommitObjects: este método manipula os objetos (armazenados em seus *DataSets* pelo método descrito no item a) no banco de dados do registro ebXML. Cada requisição submetida ao registro deve ser atômica, ou seja, caso haja algum erro, nenhum objeto submetido será persistido. Cada classe deve implementar este método com sua respectiva funcionalidade, podendo incluir, excluir ou alterar os objetos do registro. Por exemplo, a classe *SubmitObjectsRequest* insere os objetos no banco de dados, então, nesta classe a rotina é de inserção no banco de dados.

O método *GetResponse* retorna a resposta para o cliente. Esta classe também possui, diversos métodos para incluir os objetos nos seus respectivos *DataSets*, por exemplo, *addSlot, addAssociation, addOrganization,* etc. Estes métodos são utilizados para popular os *DataSets*, possibilitando a atualização dos objetos em lote no banco de dados.

3.2.3 WEB SERVICE QueryManager (QM)

O *web service* QM, além da classe *QueryManager* que expõe o método *SubmitAdhocQuery* para que possa ser acessado e utilizado por um cliente através de uma rede de computadores, apenas possui a classe *FilterQuery* que processa a requisição do cliente, através do método *getAdhocQueryResponse* que retorna uma *string AdhocQueryResponse* conforme especificado em (ebRS, 2001).

O esquema XML definido para a requsição de consultas ao registro pode ser encontrado em (QUERY, 2002).

3.2.4 DIAGRAMA DE SEQÜÊNCIA

Para representar a interação entre as classes do protótipo utilizou-se de diagramas de seqüências. Como pode ser visto nos diagramas a seguir, toda iteração parte de uma aplicação cliente *(RegistryClient)* que acessa a classe *LifeCycleManger* através da internet. A classe *LifeCycleManager* é uma classe que habilita a computação distribuída permitindo que seus métodos sejam chamados por outras máquinas através de uma rede, ou seja, um *web service.* Cada diagrama de seqüência representa um caso de uso.

O caso de uso *SubmitObjects* permite ao cliente do registro submeter um ou mais objetos em nome de uma Organização. O processo pode ser visto no diagrama de seqüência, representado na Figura 25.

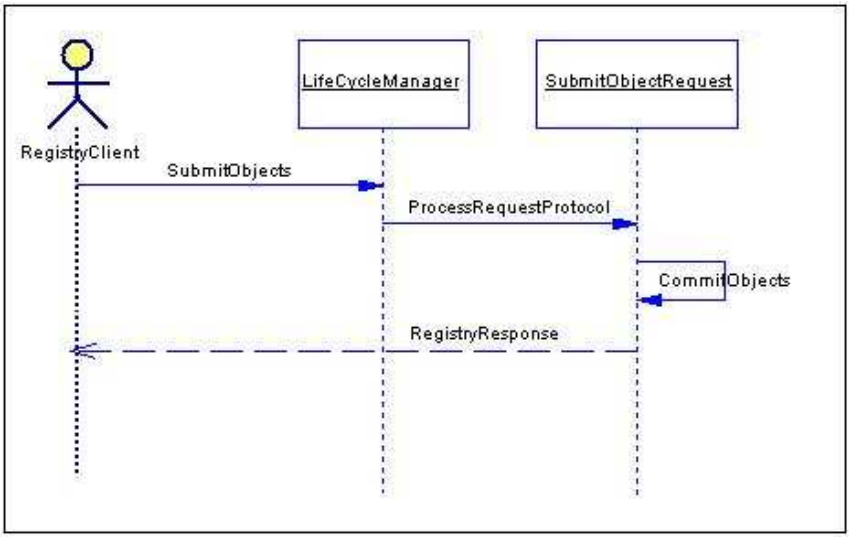


Figura 25 – Diagrama de seqüência SubmitObjects

Caso não haja erros no processamento da requisição é retornado uma *string* no formato XML *(RegistryResponse)* como definido em ebRS(2001) com status “Success” ao cliente, caso contrário retorna status “Failure”. Esta mesma regra vale para os demais casos de uso.

O caso de uso *UpdateObjects* permite ao cliente do registro atualizar um ou mais objetos em nome de uma Organização. O processo pode ser visto no diagrama de seqüência, representado na Figura 26.

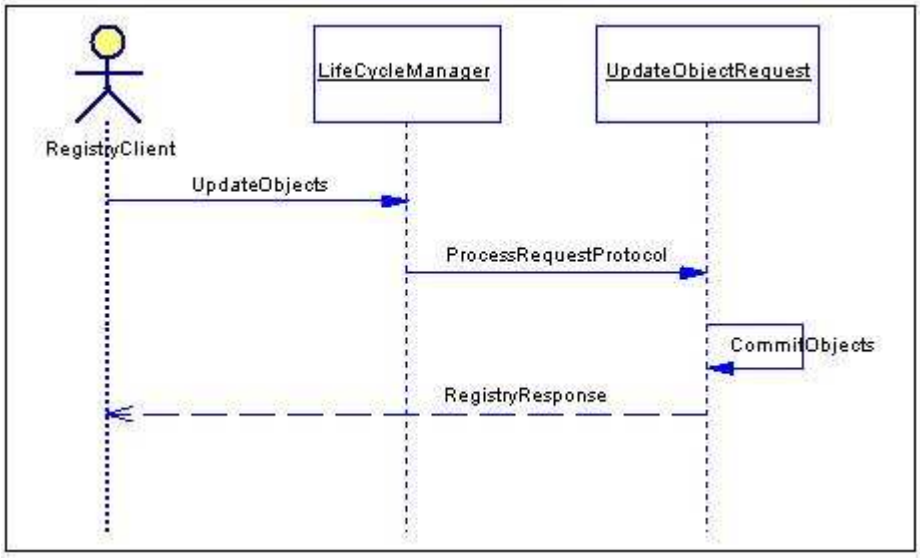


Figura 26 – Diagrama de seqüência UpdateObjects

O caso de uso *AddSlots* permite ao cliente do registro adicionar um ou mais *Slots* aos objetos do registro. O processo pode ser visto no diagrama de seqüência, representado na Figura 27.

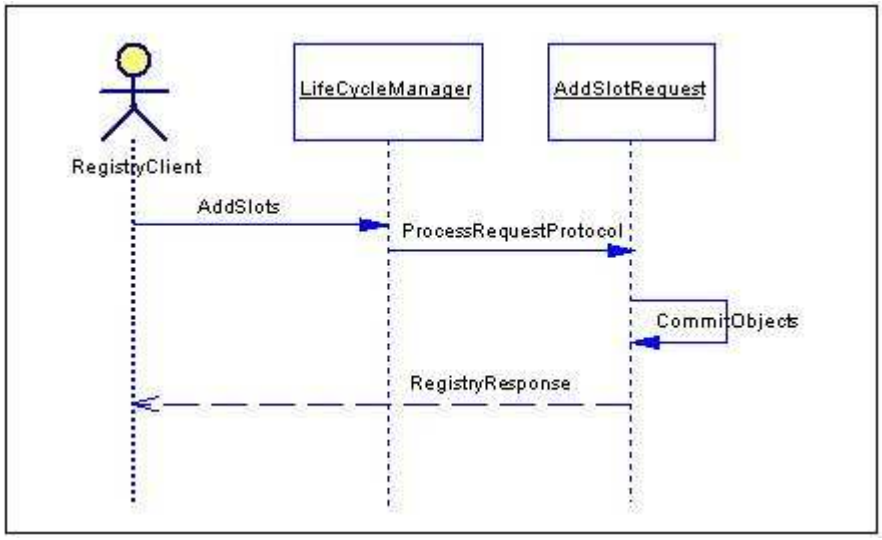


Figura 27 – Diagrama de seqüência AddSlots

O caso de uso *RemoveSlots* permite ao cliente do registro remover um ou mais *Slots* dos objetos do registro. O processo pode ser visto no diagrama de seqüência, representado na Figura 28.

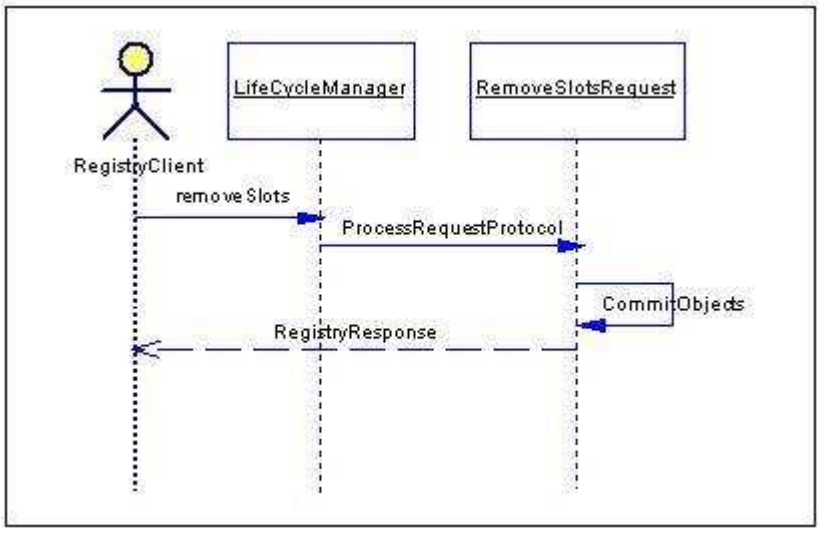


Figura 28 – Diagrama de seqüência RemoveSlots

O caso de uso *ApproveObjects* permite ao cliente do registro aprovar um ou mais objetos do registro. O processo pode ser visto no diagrama de seqüência, representado na Figura 29.

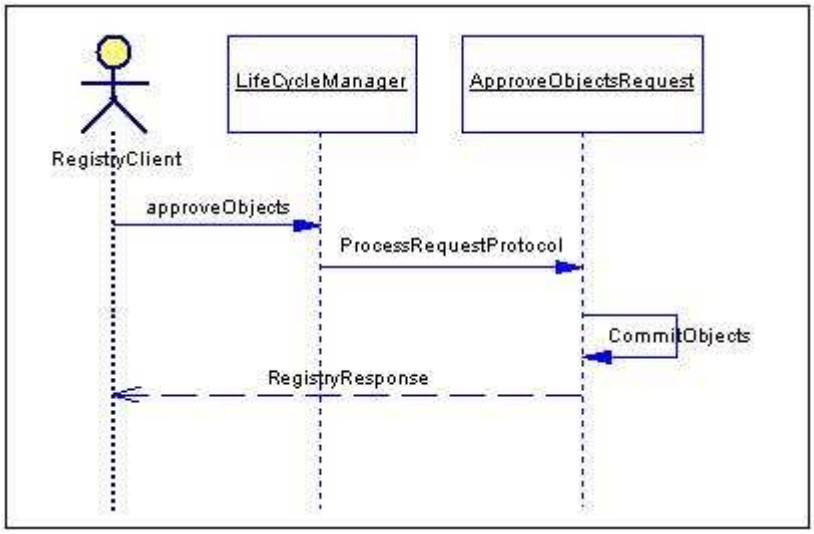


Figura 29 – Diagrama de seqüência ApproveObjects

O caso de uso *DeprecateObjects* permite ao cliente do registro desaprovar um ou mais objetos do registro. O processo pode ser visto no diagrama de seqüência, representado na Figura 30.

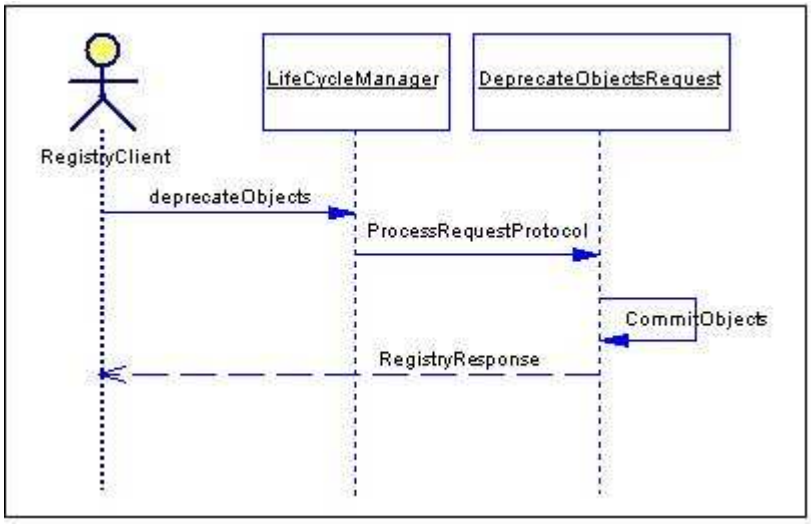


Figura 30 – Diagrama de seqüência DeprecateObjects

O caso de uso *RemoveObjects* permite ao cliente do registro remover um ou mais objetos do registro. O processo pode ser visto no diagrama de seqüência, representado na Figura 31.

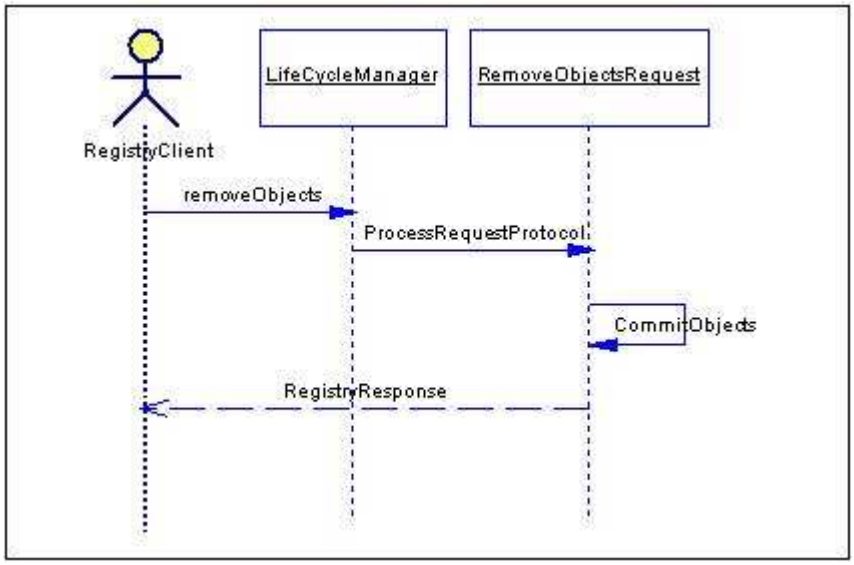


Figura 31 – Diagrama de seqüência RemoveObjects

O caso de uso *SubmitAdhocQuery* permite ao cliente recuperar informações dos objetos do registro. O processo pode ser visto no diagrama de seqüência, representado na Figura 32.

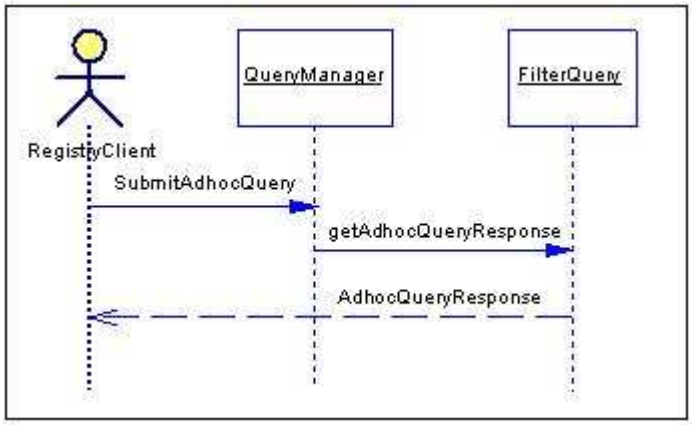


Figura 32 – Diagrama de seqüência SubmitAdhocQuery

3.2.5 ARQUITETURA DO PROTÓTIPO

A arquitetura do protótipo está organizada da seguinte forma: uma aplicação cliente e dois *web services* (LM e QM) que acessam o mesmo banco de dados, que armazena as informações submetidas pelas aplicações clientes.

O Diagrama da Figura 33, demonstra como está organizada a arquitetura do protótipo.

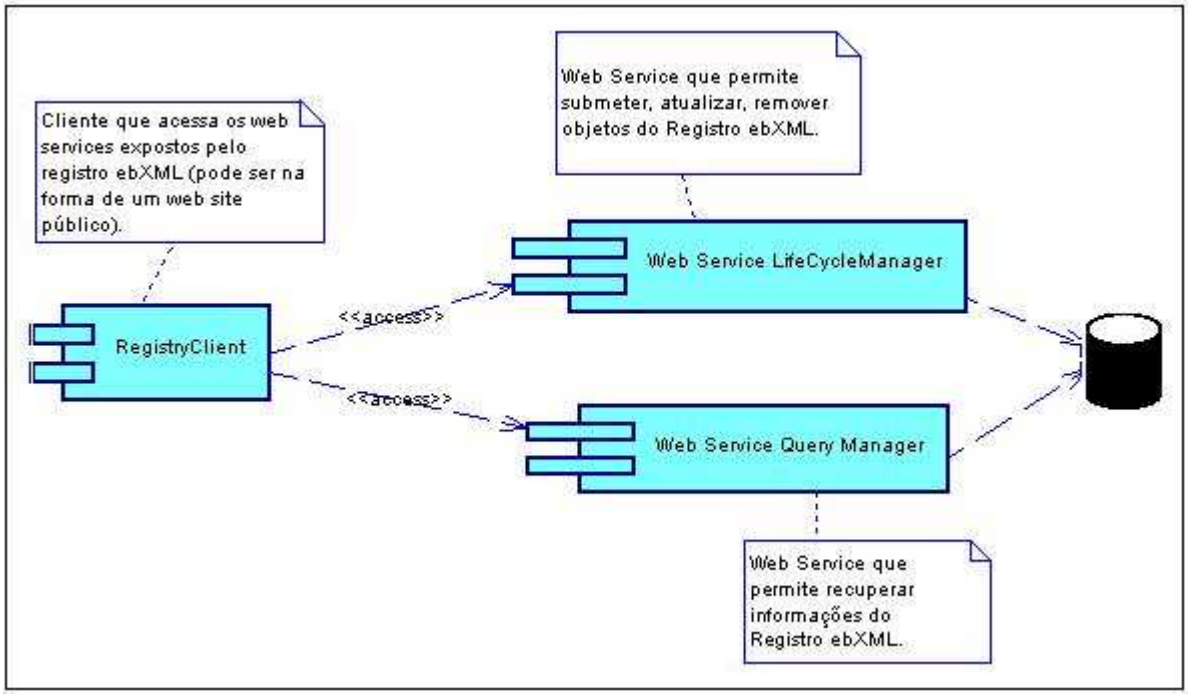


Figura 33 – Arquitetura do protótipo

Uma aplicação cliente *RegistryClient* que deseja realizar *e-Business* através do padrão ebXML acessa os *web services* através da internet. Os *web services* processam a requisição da aplicação cliente que retorna uma resposta de acordo com a requisição do cliente.

3.3 IMPLEMENTAÇÃO

Nesta seção são discutidos os aspectos rereferentes à implementação do protótipo do registro ebXML.

3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

O protótipo desenvolvido, como dito anteriormente, utilizou-se das especificações do padrão ebXML para o desenvolvimento de registros ebXML.

Para desenvolver os *web services* LM e QM utilizou-se o ambiente de desenvolvimento *Visual Studio .Net* 2003 e a linguagem escolhida foi *Visual Basic*. Para armazenar as informações submetidas pelos clientes do registro foi utilizado o Sistema Gerenciador de Banco de Dados PostgreSQL 7.2 versão Windows, fornecido pela empresa DBEXPERTS. O servidor HTTP utilizado para armazenar os *web services* é o *Internet Information Services* versão 5.1 da Microsoft.

Optou-se em desenvolver as interfaces LM e QM através de dois *web services*, mas poderia ter sido desenvolvido apenas um, com as duas funcionalidades (cadastro e pesquisa).

O protótipo está organizado em seis módulos, são eles:

a) *Business*: armazena as classes de negócio, por exemplo, *Organization*, *User,* etc.;
b) *Data*: contém as classes dos *DataSets*;
c) *DataAccess*: contém as classes para acesso ao banco de dados, possuindo métodos para inserir, atualizar, excluir ou recuperar informações;
d) *Principal*: contém as classes que processam as requisições dos clientes do registro ebXML, por exemplo, *SubmitObjectRequest, UpdateObjectRequest,* etc.;
e) *LifeCycleManager*: projeto *ASP.NET Web Service* que disponibiliza seis métodos e pode ser visto no Quadro 7;
f) *QueryManager*: projeto *ASP.NET Web Service* que disponibiliza apenas o método *SubmitAdhocQuery* e pode ser visto no Quadro 8.

Os métodos do módulo principal além de possuir o parâmetro que recebe a requisição do cliente, possui o parâmetro *UserID*, que representa o identificador único do usuário no registro. Este parâmetro serve para fazer a auditoria das ações realizadas sobre um objeto, por exemplo, criar, alterar, excluir.

Todos estes módudos foram empacotados dentro do *namespace* ebRS. No Quadro 6 pode ser visto o *import ebRS.Principal*, para que possa ser utilizada as classes que processam as requisições dos clientes do registro ebXML.

Os detalhes de implementação da classe abstrata *ObjectRequestProtocol* e a sua sub-classe *SubmitObjectRequest* podem ser vistas detalhadamente nos Apêndices A e B, respecitivamente.

3.3.2 PROTÓTIPO DE REGISTRO ebXML

O protótipo como dito anteriormente, disponibiliza dois *web services* para as organizações que desejam realizar *e-Business* utilizando o padrão ebXML.

Os métodos expostos por estes *web services* são marcados com o atributo *WebMethod* e são executados por meio de uma chamada a procedimento remoto (RPC).

O arquivo *asmx* onde são definidos os métodos do *web service LifeCycleManger* pode ser visto no Quadro 7.

```
Option Strict On
Option Explicit On

Imports System.Web.Services
Imports System.Data.OleDb
Imports ebRS.Principal

<System.Web.Services.WebService(Namespace:="http://localhost/
                                LifeCycleManager/LifeCycleManager", _
  Description:="Web Service que provê métodos para manipular objetos
  dentro do Registro ebXML", _
  Name:="Web Service LifeCycleManager")> _

Public Class LifeCycleManager
    Inherits System.Web.Services.WebService

Web Services Designer Generated Code

<WebMethod(Description:="Permite aos clientes Adicionar Slots para
           um Registry Entry existente no registro ebXML")> _
Public Function addSlots(ByVal AddSlotsRequest As String, _
                         ByVal UserID As String) As String
```

```
     Dim ProcessAddSlotsProtocol As New addSlotRequest(UserID, _
                                                       AddSlotsRequest)
     ProcessAddSlotsProtocol.ProcessRequestProtocol()
     ProcessAddSlotsProtocol.CommitObjects()
     Return ProcessAddSlotsProtocol.getResponse()
End Function

<WebMethod(Description:="Permite aos clientes Aprovar um ou mais
          itens existentes no registro ebXML")> _
Public Function approveObjects(ByVal ApproveObjectsRequest As String,
                               ByVal UserID As String) As String
    Dim ProcessApproveObjectsProtocol As New ApproveObjectsRequest(UserID,_
                                                      ApproveObjectsRequest)
   ProcessApproveObjectsProtocol.ProcessRequestProtocol()
   ProcessApproveObjectsProtocol.CommitObjects()
   Return ProcessApproveObjectsProtocol.getResponse()
End Function

<WebMethod(Description:="Permite aos clientes Deprecate(desaprovar)
          um ou mais itens existentes no registro ebXML")> _
Public Function deprecateObjects(ByVal DeprecateObjectsRequest As String, _
                                 ByVal UserID As String) As String

      Dim ProcessDeprecateObjects As New DeprecateObjectsRequest(UserID, _
                                                    DeprecateObjectsRequest)
      ProcessDeprecateObjects.ProcessRequestProtocol()
      ProcessDeprecateObjects.CommitObjects()
      Return ProcessDeprecateObjects.getResponse()
End Function


  <WebMethod(Description:="Permite aos clientes Remover um ou mais
            itens existentes no registro ebXML")> _

  Public Function removeObjects(ByVal RemoveObjectsRequest As String, _
                                ByVal UserID As String) As String
      Dim ProcessRemoveObjectsProtocol As New RemoveObjectsRequest(UserID, _
                                                       RemoveObjectsRequest)
      ProcessRemoveObjectsProtocol.ProcessRequestProtocol()
      ProcessRemoveObjectsProtocol.CommitObjects()
      Return ProcessRemoveObjectsProtocol.getResponse()
  End Function

  <WebMethod(Description:="Permite aos clientes Remover um ou mais
            Slots existentes no registro ebXML")> _
  Public Function removeSlots(ByVal RemoveSlotsRequest As String, _
                              ByVal UserID As String)As String
       Dim ProcessRemoveSlotsProtocol As New RemoveSlotsRequest(UserID, _
                                                         RemoveSlotsRequest)
       ProcessRemoveSlotsProtocol.ProcessRequestProtocol()
       ProcessRemoveSlotsProtocol.CommitObjects()
       Return ProcessRemoveSlotsProtocol.getResponse()
  End Function

  <WebMethod(Description:="Permite aos clientes Submeter um ou mais
            itens ao registro ebXML")> _
  Public Function submitObjects(ByVal SubmitObjectsRequest As String, _
                                ByVal UserID As String) As String
       Dim ProcessSubmitProtocol As New SubmitObjectRequest(UserID, _
                                                       SubmitObjectsRequest)
       ProcessSubmitProtocol.ProcessRequestProtocol()
       ProcessSubmitProtocol.CommitObjects()
       Return ProcessSubmitProtocol.getResponse()
  End Function

  <WebMethod(Description:="Permite aos clientes Atualizar um ou mais
            itens submetidos")> _
```

```
  Public Function updateObjects(ByVal UpdateObjectsRequest As String, _
                                ByVal UserID As String) As String
      Dim ProcessUpdateObjectsProtocol As New UpdateObjectRequest(UserID, _
                                                  UpdateObjectsRequest)
      ProcessUpdateObjectsProtocol.ProcessRequestProtocol()
      ProcessUpdateObjectsProtocol.CommitObjects()
      Return ProcessUpdateObjectsProtocol.getResponse()
  End Function
End Class
```

Quadro 7 – Arquivo asmx web service LifeCycleManager

A Figura 34, demonstra o Internet Explorer processando o arquivo *asmx LifeCycleManager.*

Figura 34 – Visualização do arquivo *asmx* LifeCycleManager

O arquivo *asmx* do *web service QueryManager* pode ser visto no Quadro 8.

```
Imports System.Web.Services
Imports ebRS.Business
Imports ebRS.Principal

<System.Web.Services.WebService(Namespace:="http://localhost/QueryManager/
    QueryManager", _
    Description:="Web Service que provê aos clientes do registro descobrir
    e recuperar objetos.", _
    Name:="Web Service QueryManager")> _
```

```
Public Class QueryManager
    Inherits System.Web.Services.WebService

Web Services Designer Generated Code

  <WebMethod(Description:="Permite aos clientes submeter consultas
             ao registro ebXML")> _
  Public Function SubmitAdhocQuery(ByVal AdhocQueryRequest As String) As
  String
     Dim ProcessSubmitAdhocQuery As New FilterQuery(AdhocQueryRequest)
     Return ProcessSubmitAdhocQuery.getAdhocQueryResponse()
  End Function
End Class
```

Quadro 8 – Arquivo asmx web service QueryManger

A Figura 35 abaixo, demonstra o Internet Explorer processando o arquivo *amsx QueryManger*.

Figura 35 – Visualização do arquivo *asmx* QueryManager

3.3.3 OPERACIONALIDADE DA IMPLEMENTAÇÃO

A operacionalidade do sistema será apresenta através dos Casos de Uso *SubmitObjects* e *AdhocQueryResponse*.

Para demonstrar a operacionalidade da implementação criou-se uma pequena aplicação para acessar os *web services* do protótipo ebXML (apesar de não ser necessário, pois é possível testar os métodos através do Internet Explorer). A aplicação cliente *(RegistryClient)* pode ser vista na Figura 36.

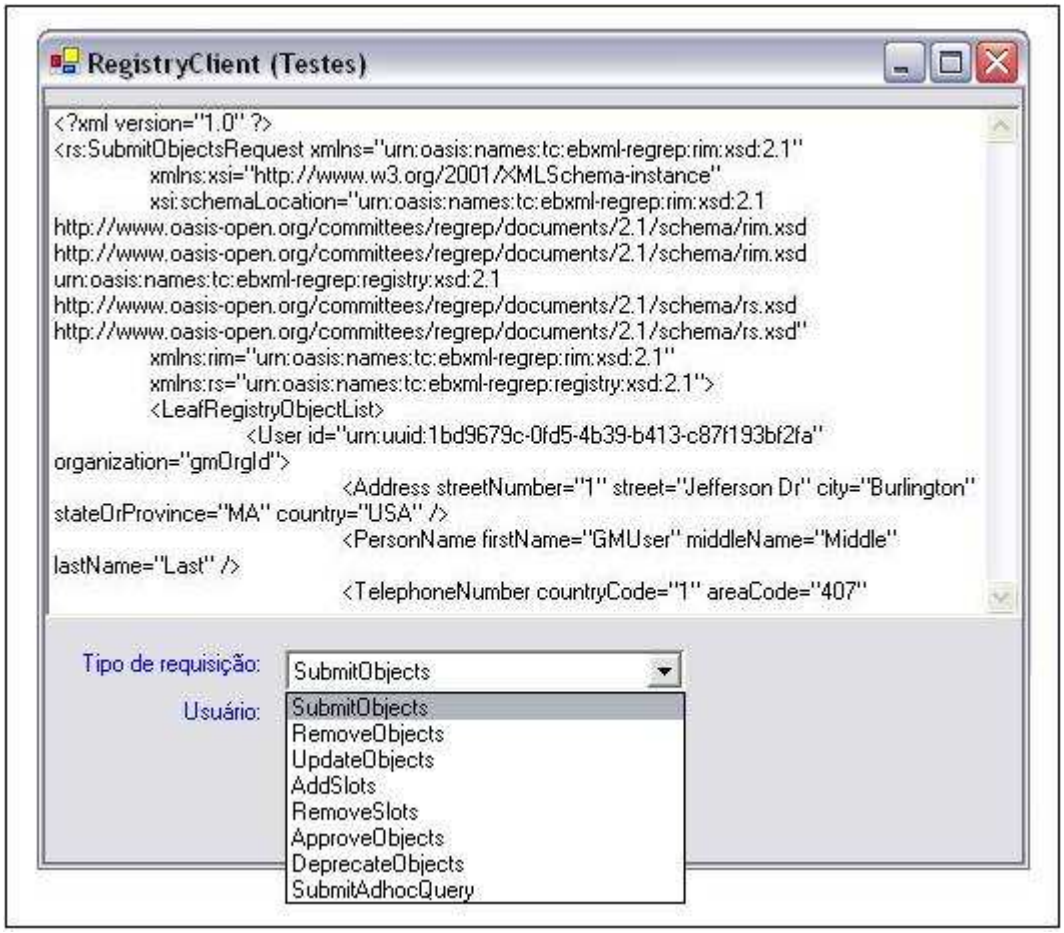


Figura 36 – Aplicação para testes do protótipo ebXML

A Figura 36 acima, demonstra as requisições que o usuário pode executar. Nesta figura é mostrada a submissão de novos objetos no registro.

A Figura 37 a seguir, demonstra a submissão da requisição de teste através do aplicativo *RegistryClient* utilizado para testes. O detalhamento da mensagem de teste pode ser vista no Apêndice C.

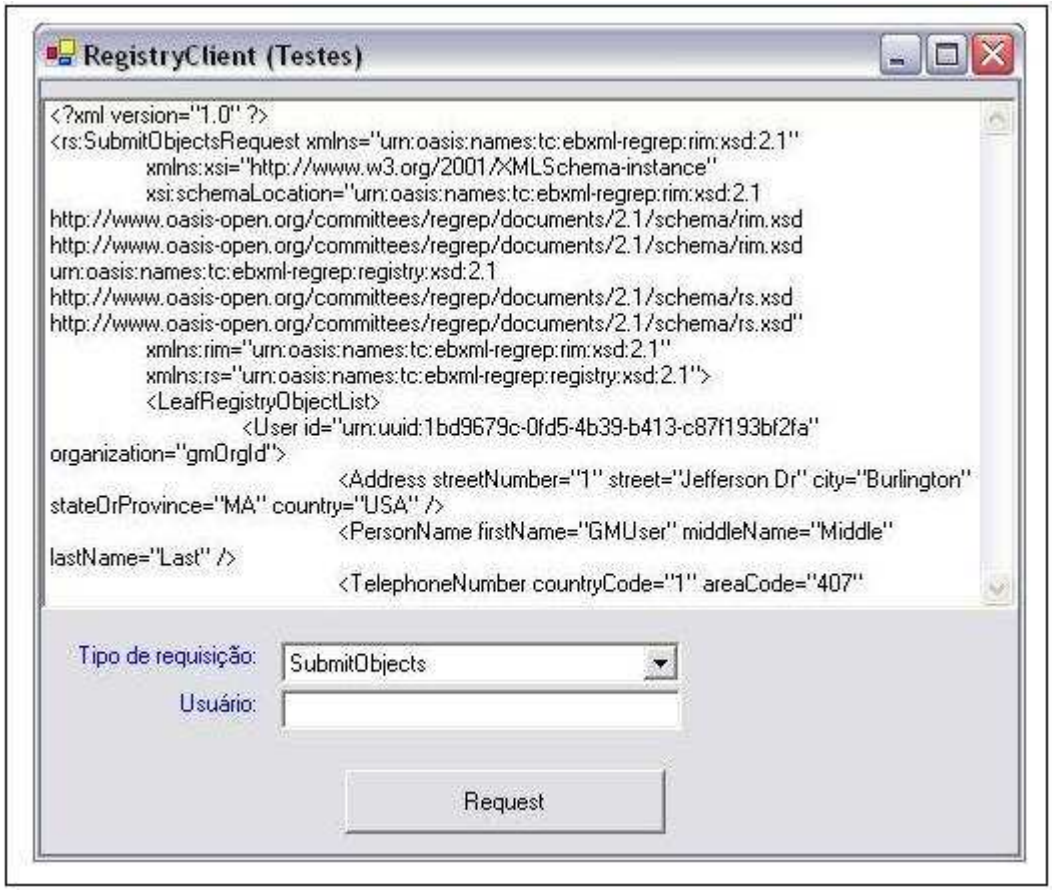


Figura 37 – Tela submissão da requisição de teste

Caso aconteça algum erro no processamento da requisição, o registro retornará uma mensagem com *status "Failure"*. Erros ocorrem quando um identificador de objeto (atributo *id*) não é único, quando um usuário não possui permissão sobre o objeto, quando referências para objetos do registro não forem encontradas, etc.

O identificador do objeto deve ser um *Universally Unique Identifier* (UUID) ou Identificador Único Universal. O Visual Basic .Net provê a classe GUID e o método *NewGuid* para a geração de identificadores único.

A Figura 38 a seguir, demonstra a resposta do protótipo de registro ebXML para a requsição de teste. Como não ocorreu nenhum erro no processamento da requisição, a mensagem retorna com *status "Success"*.

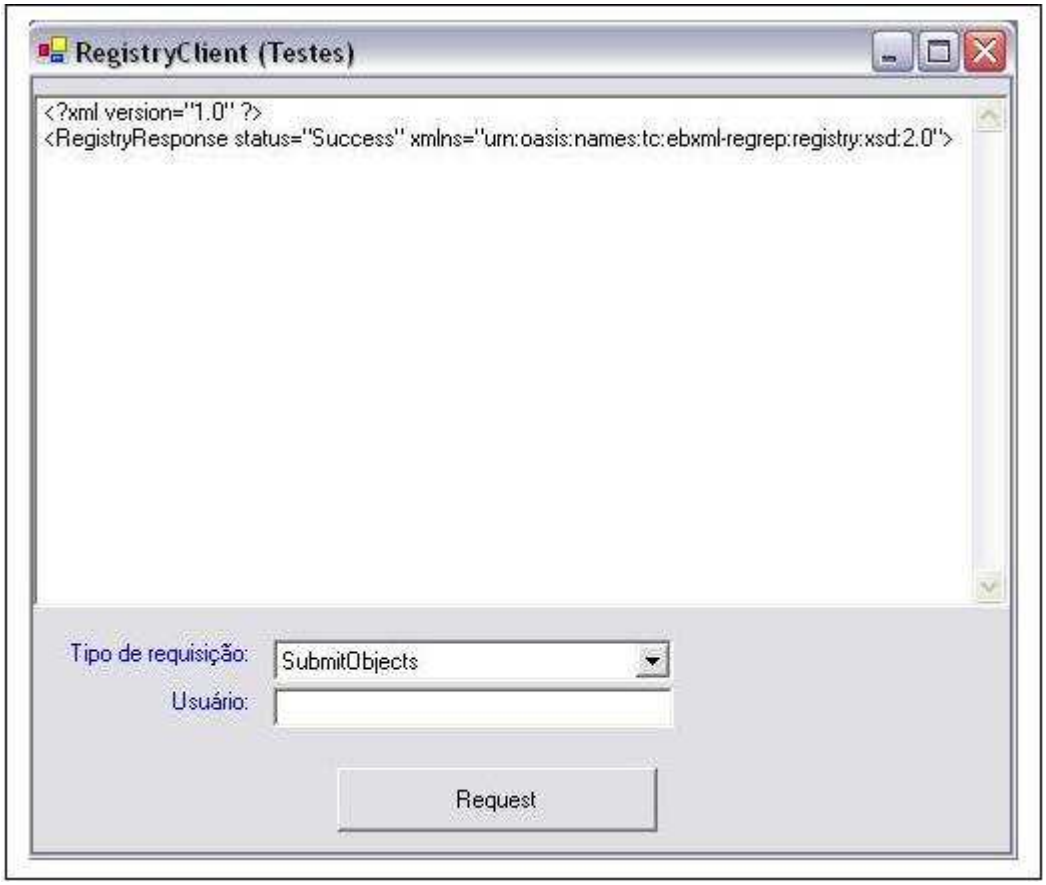


Figura 38 – Resposta da requisição SubmitObjectsRequest

O Quadro 9 abaixo, representa o caso de uso *SubmitAdhocQuery*, onde é requisitado todas as organizações cujo o nome contém “General Motors”.

```
<?xml version="1.0" encoding="UTF-8" ?>
<AdhocQueryRequest>
    <ResponseOption returnType="LeafClass"/>
    <FilterQuery>
        <OrganizationQuery>
            <NameBranch>
                <LocalizedStringFilter>
                    <Clause>
                       <SimpleClause leftArgument="name">
                           <StringClause stringPredicate="Contains">
                                    General Motors
                           </StringClause>
                    </Clause>
                </LocalizedStringFilter>
            </NameBranch>
        </OrganizationQuery>
    </FilterQuery>
</AdhocQueryRequest>
```

Quadro 9 – Requisição de teste FilterQuery

O resultado da requisição de teste *OrganizationQuery*, pode ser visto na Figura 39.

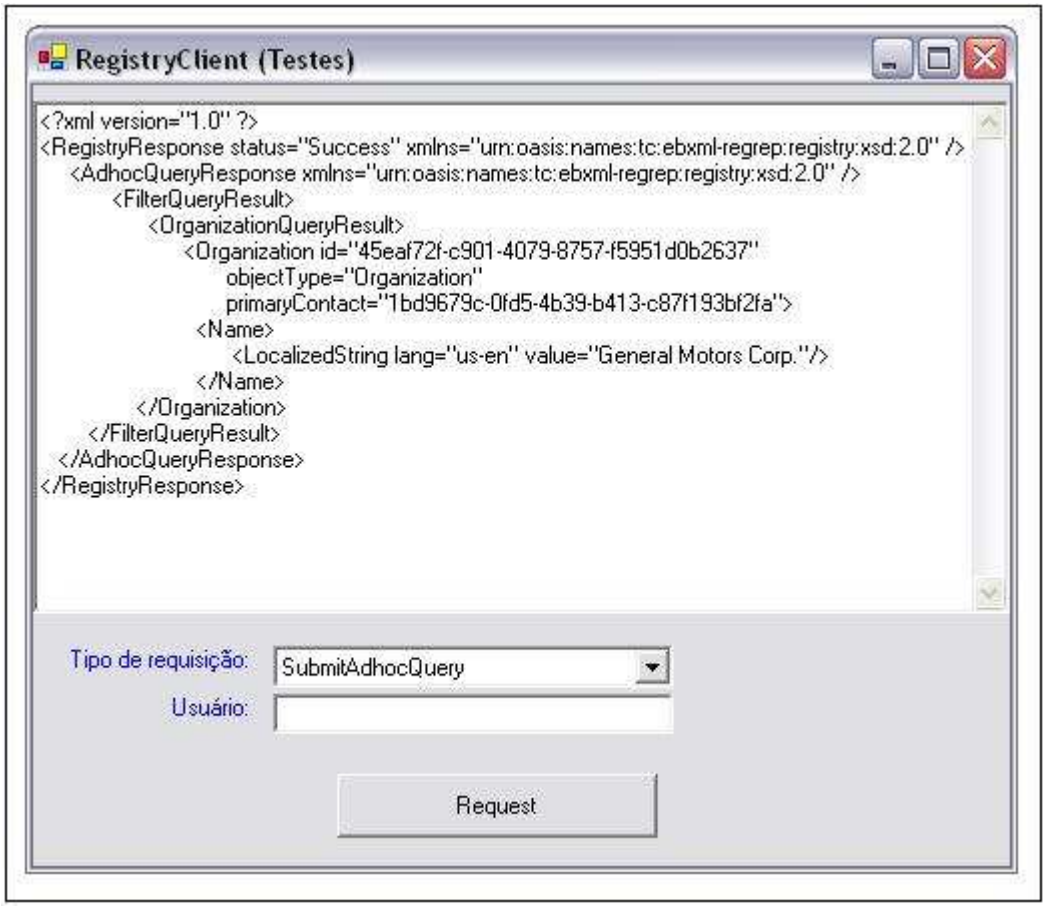


Figura 39 – Resposta da requisição AdhocQueryRequest

3.4 RESULTADOS E DISCUSSÃO

As telas apresentadas na operacionalidade da implementação, representam um cliente do protótipo de registro ebXML utilizado para facilitar os testes e a visualização do envio e recebimento das mensagens entre registro e aplicação cliente, pois a visualização pelo Internet Explorer não seria muito adequada.

A aplicação cliente utilizada para testes também demonstrou a dificuldade para um usuário interagir com o protótipo se ele tiver que conhecer o padrão ebXML para o envio e recebimento de informações ao registro.

A criação de uma aplicação cliente com uma interface amigável, possibilitando que o usuário gerencie objetos do registro é essencial para as organizações que desejam realizar *e-Business* utilizando o padrão ebXML.

Apesar de a sintaxe XML ser mais fácil para os programadores compreender, não é este o motivo que levará as empresas a abandonar o EDI tradicional para utilizar XML como formato das mensagens trocadas entre parceiros comerciais. Pois, com as mensagens EDI já se alcançou à interoperabilidade entre sistemas para o envio e recebimento de informações eletronicamente.

Um dos motivos para as mensagens EDI serem enxutas, e assim, tornando-se difícil para se compreender é a cobrança por byte trafegado nas mensagens trocadas entre os parceiros de negócio pela *Value Added Network* (VAN) ou Rede de Valor Agregado. A contração de uma VAN, para o envio e recebimento de informações, não é um requisito obrigatório para a realização de EDI.

Com a utilização da *internet* e padrões como ebXML não há mais necessidade de contratar serviços de uma VAN, assim a XML não enfrenta este problema, permitindo que mais byte sejam trafegados na rede e possibilitando que as mensagens sejam de mais fácil entendendimento pelos desenvolvedores.

O grande benefício da XML para o *e-Business* é sua grande adoção pelo mercado, a XML é utilizada em diversas aplicações para delimitar e representar dados, fazendo com que surgissem diversos desenvolvedores que programam com XML e linguagens de programação que fornecem funcionalidades para a criação e processamento de mensagens XML.

O EDI contribuiu significativamente para o *e-Business* e com o uso da XML os custos para o desenvolvimento de novas aplicações para intercâmbio eletrônico de dados poderão ser menores, em relação ao EDI tradicional, possibilitando que pequenas e médias empresas possam dispor das vantagens de trocar informações comerciais eletronicamente.

4 CONCLUSÕES

O padrão ebXML se mostrou um *framework* poderoso para a realização de *e-Business*. Este padrão possui mecanismos para a descoberta de novos parceiros comerciais uma vantagem em relação ao EDI. A descoberta de novos parceiros é feita através de registros/repositórios e os acordos entre os parceiros através de CPP's e CPA's. Além disso, as mensagens trocadas estão no formato XML, onde as mensagens praticamente se descrevem a si própria, diferente das mensagens do padrão EDI.

Certamente cada vez mais o XML será utilizado por novas aplicações para o intercâmbio eletrônico de dados, porém, apenas a utilização do XML como formato das mensagens ou simplismente convertendo as mensagens EDI em XML não trará grandes vantagens em relação ao EDI tradicional, por isso, a utilização de padrões como ebXML se torna necessário.

O protótipo de registro ebXML desenvolvido apesar de possibilitar o envio e armazenamento de informações, apresenta limitações para informar e também para a descoberta das mesmas pelos parceiros comerciais, possibilitando apenas o descobrimento das organizações cadastradas no registro. Também não é implementado nenhum mecanismo de segurança, possibilitando que as informações possam manipuladas por qualquer usuário, inclusivo a de outros parceiros comerciais.

Por ser um padrão aberto e global para a realização de *e-Business*, os documentos de especificação são bastante extenso e demandam um grande tempo para o estudo, compreensão e desenvolvimento dos mesmos. Estes documentos definem diversas regras de negócio para diversas situações, que não foram possíveis de ser implementadas durante o tempo estipulado para o desenvolvimento do protótipo.

O armazenamento de documentos que descrevem os processos de negócio, CPP's, CPA's e outros documentos que facilitem o desenvolvimento de aplicações de *e-Business* utilizando o padrão ebXML, não podem ser enviadas pelas aplicações clientes, pois o *web service* LM não trata requisições SOAP com anexo (SOAPAttach), a maneira pelo qual seria enviado estes documentos.

Apesar do protótipo não ter atingido todos seus objetivos, o trabalho conseguiu demonstrar como o padrão ebXML está organizado para possibilitar a realização de *e-Business*, possibilitando que organizações interessadas no padrão ebXML possam ter uma idéia de como desenvolver sua própria aplicação de interação com um registro ebXML ou até mesmo desenvolver seu próprio registro.

A escolha da plataforma .Net para o desenvolvimento dos *web services* e o ambiente de desenvolvimento Visual Studio .Net, facilitou bastante o desenvolvimento do protótipo, pois a criação de *web services* nesta plataforma é de fácil implementação.

4.1 EXTENSÕES

Aprimorar o protótipo desenvolvido para fornecer as funcionalidades descritas no documento de especificação (ebRS, 2001) ou uma versão mais recente.

Como dito anteriormente, pode-se criar uma aplicação cliente que forneça uma interface amigável ao usuário para interagir com o protótipo.

Pelo fato de várias empresas utilizarem soluções EDI, o desenvolvimento de ferramentas que fizessem a conversão das mensagens EDI em mensagens XML, possibilitando a intregação destes padrões seria de grande valia.

Desenvolver ferramentas que facilitem a geração de CPP’s, CPA’s e os processos de negócio.

Acompanhar o desenvolvimento das especificações do padrão ebXML é extremamente necessário para as empresas que desejam se manter atualizadas em relação ao desenvolvimentos de aplicações para comércio eletrônico.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDERSON, Richard et al. **Professional XML.** Tradução Mônica Santos Sarmento, Rejane Freitas. Rio de Janeiro: Ciência Moderna Ltda., 2001. 1266 p.

CONALLEN, Jim. **Desenvolvimento de aplicações Web com UML.** Tradução Altair Dias Caldas de Moraes, Claudio Belleza Dias. Rio de Janeiro: Campus, 2003. 476 p.

DEITEL, H. M., DEITEL, P. J., NIETO, T. R. **Visual Basic .Net: como programar.** Tradução Célia Yumi Okano. São Paulo: Pearson Education do Brasil, 2004. 1088 p.

EAN BRASIL. **Manual EANCOM Brasil (UN/EDIFACT)**. EAN BRASIL, 1994.

ebBPSS. **ebXML Business process specification schema v1.01,** business process project team, 2001. Disponível em: <http://www.ebxml.org/specs/ ebbpss.pdf>. Acesso em: 6 de abr. 2004.

ebCPP. **ebXML collaboration-protocol profile and agreement specification v2.0,** OASIS/ebXML collaboration protocol profile and agreement technical commitee, 2002. Disponível em: <http://www.ebxml.org/specs/ebcpp-2.0.pdf>. Acesso em: 6 de abr. 2004.

ebRIM. **ebXML registry information model v2.1,** OASIS/ebXML registry technical commitee, 2002. Disponível em: <http://www.ebxml.org/specs/ebrim_v2.1.pdf>. Acesso em 6 de abr. 2004.

ebREQ. **ebXML requirements specifiaction v1.06,** ebXML requirements team, 2001. Disponível em: <http://www.ebxml.org/specs/ebreq.pdf>. Accesso em 6 de abr. 2004.

ebRS. **ebXML registry services specification v2.0,** OASIS/ebXML registry technical commitee, 2001. Disponível em <http://www.ebxml.org/specs/ebrs2.pdf>. Acesso em 6 de abr. de 2004.

ebTA. **ebXML technical architecture specification v1.0.4**, ebXML technical architecture project team, 2001. Disponível em: <http://www.ebxml.org/specs/ebta.pdf>. Acesso em: 6 de abr. 2004.

ebXML. **ebXML enabling a global electronic market,** [s.l.], [2004]. Disponível em: <http://www.ebxml.org>. Acesso em: 6 de abr. 2004.

GOXML. **Transform re-purpose and distribute documents statements ans reports – Xenos,** [s.l.], 2004. Disponível em: <http://www.xmlglobal.com>. Acesso em: 9 de jul. 2004.

MARQUES, Pedro Filipe de Jesus Vieira. **Troca de informação de negócio para negócio – do EDI ao XML/EDI e ebXML.** 2003. 136 f. Trabalho de Conclusão de Curso (Licenciado em Engenharia da Comunicação) – Universidade Fernando Pessoa, Porto.

MÜLLER JÚNIOR, Jalmor. **Protótipo de um formatador de dados utilizando as normas de sintaxe ISO 9735 – EDIFACT.** 1997. 64 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

QUERY. **Query XML schema,** [s.l], [2002?]. Disponível em: <http://www.oasis-open.org/committees/regrep/documents/2.0/schema/query.xsd>. Acesso em: 10 de ago. de 2004.

RIM. **RIM XML schema,** [s.l.], [2002?]. Disponível em: <http://www.oasis-open.org/committees/regrep/documents/2.0/schema/rim.xsd>. Acesso em: 10 de ago. de 2004.

SAMPLES. **ebXML samples,** [s.l], [2004?]. Disponível em: <http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/ebxmlrr/ebxmlrr-spec/misc/samples>. Acesso em: 15 de ago. de 2004.

W3C. **World Wide Web Consortium,** United States, 2004. Disponível em: <http://www.w3c.org>. Acesso em: 9 de abr. 2004.

XML SOLUTIONS. **XML and e-business. A new revolution on the horizon,** [s.l], [2004]. Disponível em: <http://www. sterlingcommerce.com/solutions/products/ebi/wp/xmlebiz_wp.html>. Acesso em: 22 de dez. de 2004.

APÊNDICE A – Classe ObjectRequestProtocol

O quadro a seguir demonstra a implementação da classe abstrata *ObjectRequestProtocol*. Para não se tornar muito extenso alguns trechos de código foram omitidos, colocando no seu lugar (...).

```
Option Explicit On
Option Strict On

Imports System
Imports System.Text
Imports System.Xml
Imports System.IO
Imports System.Data.Odbc
Imports ebRS.DataAccess
Imports ebRS.Business
Imports ebRS.Data

Public MustInherit Class ObjectRequestProtocol

    Protected wrkUserID As String
    Protected wrkOrganization As String

    Private wrkFileName As String
    Protected document As XmlDocument
    Protected reader As XmlNodeReader

    Private wrkErro As Boolean
    Protected Property Erro() As Boolean
        Get
            Return wrkErro
        End Get
        Set(ByVal Value As Boolean)
            wrkErro = Value
        End Set
    End Property

    Protected ObjResponse As New RegistryResponse

    Private wrkTable As Hashtable
    Protected Sub AddTableKey(ByVal pKey As String, ByVal pValue As String)
        Try
            wrkTable.Add(pKey, pValue)
        Catch ex As ArgumentException
            Me.Erro = True
            Me.ObjResponse.Status = StatusResponse.Failure.ToString
            Me.ObjResponse.Documentation = "Error during the process of the
request!"
        End Try
    End Sub

    Protected Function getTableValue(ByVal pKey As String) As String
        Dim wrkResult As String = Convert.ToString(wrkTable(pKey))
        If Not wrkResult Is Nothing Then
            Return wrkResult
        Else
            Me.Erro = True
            Me.ObjResponse.Status = StatusResponse.Failure.ToString
            Me.ObjResponse.Documentation = "Error during the process of the
request!"
        End If
    End Function
```

```
    Protected Function existeTableKey(ByVal pKey As String) As Boolean
        Return wrkTable.Contains(pKey)
    End Function

    Protected Sub clearTable()
        wrkTable.Clear()
    End Sub

    'DataSet para armazenar informações submetidas
    Protected dstAssociation As AssociationData
    Protected dstAuditableEvent As AuditableEventData
    Protected dstClassification As ClassificationData
    Protected dstClassificationNode As ClassificationNodeData
    Protected dstClassificationScheme As ClassificationSchemeData
    Protected dstExternalIdentifier As ExternalIdentifierData
    Protected dstExternalLink As ExternalLinkData
    Protected dstExtrinsicObject As ExtrinsicObjectData
    Protected dstRegistryPackage As RegistryPackageData
    Protected dstService As ServiceData
    Protected dstServiceBinding As ServiceBindingData
    Protected dstSpecificationLink As SpecificationLinkData
    Protected dstUser As UserData
    Protected dstOrganization As OrganizationData
    Protected dstName As NameData
    Protected dstDescription As DescriptionData
    Protected dstEmailAddress As EmailAddressData
    Protected dstPostalAddress As PostalAddressData
    Protected dstSlot As SlotData
    Protected dstTelephoneNumber As TelephoneNumberData
    Protected dstUsageDescription As UsageDescriptionData
    Protected dstUsageParameter As UsageParameterData

    Sub New(ByVal UserID As String, ByVal Request As String)
        Me.wrkUserID = UserID
        Me.wrkOrganization = ""
        Try
            document = New XmlDocument
            document.LoadXml(Request)

            Me.InitVars()

        Catch ex As Exception
            Me.Erro = True
            Me.ObjResponse.Status = StatusResponse.Failure.ToString
            Me.ObjResponse.Documentation = "Exception: " & ex.Message
        End Try
    End Sub

    Private Sub InitVars()
        reader = New XmlNodeReader(document)
        wrkTable = New Hashtable

        dstAssociation = New AssociationData
        dstAuditableEvent = New AuditableEventData
        dstClassification = New ClassificationData
        dstClassificationNode = New ClassificationNodeData
        dstClassificationScheme = New ClassificationSchemeData
        dstExternalIdentifier = New ExternalIdentifierData
        dstExternalLink = New ExternalLinkData
        dstExtrinsicObject = New ExtrinsicObjectData
        dstRegistryPackage = New RegistryPackageData
        dstService = New ServiceData
        dstServiceBinding = New ServiceBindingData
        dstSpecificationLink = New SpecificationLinkData
        dstUser = New UserData
        dstOrganization = New OrganizationData
        dstName = New NameData
```

```
        dstDescription = New DescriptionData
        dstEmailAddress = New EmailAddressData
        dstPostalAddress = New PostalAddressData
        dstSlot = New SlotData
        dstTelephoneNumber = New TelephoneNumberData
        dstUsageDescription = New UsageDescriptionData
        dstUsageParameter = New UsageParameterData

        'Assume como sucesso
        Me.Erro = False
        Me.ObjResponse.Status = StatusResponse.Sucess.ToString

        'Se não foi passado como parâmetro o ID do usuário, então procura na
requisição do cliente
        If Me.wrkUserID = "" Then
            Dim wrkTempUserID As String = XMLProcessUtils.getUserID(Me.document)
            'verifica se foi informado na requisição
            If wrkTempUserID = "" Then
                'Erro not authorized
                Me.Erro = True
                Me.ObjResponse.Status = StatusResponse.Failure.ToString
                Me.ObjResponse.Documentation = "User not authorized!"
                Exit Sub
            Else
                'verifica se ID é único
                Dim daRegistryObject As New daRegistryObject
                If Not daRegistryObject.existeID(wrkTempUserID) Then
                    Me.wrkUserID = wrkTempUserID
                Else
                    'Erro InvalidIDError
                    Me.Erro = True
                    Me.ObjResponse.Status = StatusResponse.Failure.ToString
                    Me.ObjResponse.Documentation = "Invalid Id: " & wrkTempUserID
                    Exit Sub
                End If
                daRegistryObject.Dispose()
                Me.AddTableKey(wrkTempUserID, wrkUserID)
                Me.getOrganizaton()
                Exit Sub
            End If
        End If
        'Se foi informado o usuário, verifica se usuário existe
        If Me.wrkUserID <> "" Then
            Dim dsUser As DataSet
            Dim dtaUser As New daUser
            dsUser = dtaUser.LoadUser(Me.wrkUserID)
            If dsUser.Tables(0).Rows.Count = 0 Then
                'Erro not authorized
                Me.Erro = True
                Me.ObjResponse.Status = StatusResponse.Failure.ToString
                Me.ObjResponse.Documentation = "User not authorized! User: " &
wrkUserID
                Exit Sub
            End If
            Me.getOrganizaton()
        End If

    End Sub

    'Métodos que devem ser sobreposto pelas classes filhas
    Public MustOverride Function ProcessRequestProtocol() As String
    Public MustOverride Sub CommitObjects()

    'Métodos para adicionar os objetos em seus respectivos dataSets
    ...
    Protected Sub AddUser(ByVal User As User)
        Dim rowUser As UserData.user_Row
```

```
        rowUser = dstUser.user_.Newuser_Row
        rowUser.id = User.Id
        rowUser.accesscontrolpolicy = User.AccessControlPolicy
        rowUser.objecttype = User.ObjectType
        rowUser.organization = User.Organization
        rowUser.personname_firstname = User.PersonName.FirstName
        rowUser.personname_lastname = User.PersonName.LastName
        rowUser.personname_middlename = User.PersonName.MiddleName
        rowUser.url = User.URL
        dstUser.user_.Adduser_Row(rowUser)
    End Sub

    Protected Sub AddOrganization(ByVal Organization As Organization)
        Dim rowOrganization As OrganizationData.organizationRow
        rowOrganization = dstOrganization.organization.NeworganizationRow
        rowOrganization.id = Organization.Id
        rowOrganization.accesscontrolpolicy = Organization.AccessControlPolicy
        rowOrganization.objecttype = Organization.ObjectType
        rowOrganization.parent = Organization.Parent
        rowOrganization.primarycontact = Organization.PrimaryContact
        dstOrganization.organization.AddorganizationRow(rowOrganization)
    End Sub

    'Método retorna a organização que usuário é filiado
    Private Sub getOrganizaton()
        'Procura organização que o usuário pertence primeiro na requisição do
cliente, caso esteja sendo cadastrado novo usuário
       Dim wrkTempOrganization As String = XMLProcessUtils.getAttribute(document, _
                                "User", "urn:uuid:" & Me.wrkUserID, "organization")
        Me.wrkOrganization = XMLProcessUtils.getReferenceObject(document, _
                                             "Organization", wrkTempOrganization)
        Dim daRegistryObject As New daRegistryObject
        If daRegistryObject.existeID(Me.wrkOrganization) Then
            'Erro InvalidIDError
            Me.Erro = True
            Me.ObjResponse.Status = StatusResponse.Failure.ToString
            Me.ObjResponse.Documentation = "Invalid Id: " & wrkOrganization
            Exit Sub
        End If
        daRegistryObject.Dispose()

        'Se não encontrou a organização na requisição do cliente, então procura no
banco de dados
        If Me.wrkOrganization = "" Then
            Dim daUser As New daUser
            Me.wrkOrganization = daUser.getOrganization(Me.wrkUserID)
            daUser.Dispose()
        End If
        Me.AddTableKey(wrkTempOrganization, Me.wrkOrganization)
    End Sub
End Class
```

APÊNDICE B – Classe SubmitObjectRequest

O quadro a seguir, demonstra detalhadamente a implementação da classe *SubmitObjectReques.* Para não se tornar muito extenso alguns trechos de código foram omitidos, colocando no seu lugar (...).

```
Option Explicit On
Option Strict On

Imports System
Imports System.Xml
Imports System.IO
Imports System.Data.Odbc
Imports ebRS.DataAccess
Imports ebRS.Business

Public Class SubmitObjectRequest
    Inherits ObjectRequestProtocol

    Sub New(ByVal UserID As String, ByVal Request As String)
        MyBase.New(UserID, Request)
    End Sub

    'Método sobreposto da classe pai
    Public Overrides Function ProcessRequestProtocol() As String
        Try
            Me.reader.MoveToContent()
            If Me.reader.LocalName <> "SubmitObjectsRequest" Then
                Me.Erro = True
                Me.ObjResponse.Status = StatusResponse.Failure.ToString
                Me.ObjResponse.Documentation     =     "Request     is     not     a
SubmitObjectsRequest valid!"
            End If

            While Me.reader.Read
                'Caso houve erro, para de processar
                If Me.Erro Then
                    Exit Try
                End If
                If Me.reader.NodeType = XmlNodeType.Element Then
                    ProcessaNodeReader(reader, Nothing)
                End If
            End While

        Catch ex As Exception
            Me.Erro = True
            Me.ObjResponse.Status = StatusResponse.Failure.ToString
            Me.ObjResponse.Documentation = "Exception: " & ex.Message & " - " &
ex.StackTrace
        Finally
            If Not (Me.reader Is Nothing) Then
                Me.reader.Close()
            End If
        End Try
    End Function

Private Sub ProcessaNodeReader(ByVal pNode As XmlNodeReader, ByVal pIdParent As
String)

        If pNode.EOF Then
            Exit Sub
        End If
```

```
        Select Case pNode.LocalName.ToUpper

            ...
            Case "ORGANIZATION"

                pNode.MoveToContent()
                If pNode.HasAttributes Then

                    Dim ObjOrganization As New Organization
                    'Atributos herdados da classe RegistryObject
                    ObjOrganization.Id = Me.wrkOrganization
                    ObjOrganization.AccessControlPolicy = "ContentOwner"
                    ObjOrganization.ObjectType = "Organization"
                    ObjOrganization.Parent = pNode.GetAttribute("parent")
                    ObjOrganization.PrimaryContact = Me.wrkUserID

                    XMLProcessUtils.getNameAndDescription(pNode,
CType(ObjOrganization, RegistryObject))
                    Me.AddName(ObjOrganization)
                    Me.AddDescription(ObjOrganization)

                    If pNode.NodeType = XmlNodeType.EndElement AndAlso _
                       pNode.LocalName.ToUpper = "ORGANIZATION" Then
                        Exit Sub
                    End If

                    'Passar pelo RegistryObjectType
                    Do While pNode.Read
                        If pNode.NodeType = XmlNodeType.Element Then
                            Select Case pNode.LocalName.ToUpper
                                Case "SLOT", "CLASSIFICATION", "EXTERNALIDENTIFIER"
                                    Me.ProcessaNodeReader(pNode, _
                                                          ObjOrganization.Id)
                            End Select
                        End If
                        If pNode.NodeType = XmlNodeType.EndElement AndAlso _
                           pNode.LocalName.ToUpper = "ORGANIZATION" Then
                            Exit Do
                        End If
                    Loop

                  ObjOrganization.Address=XMLProcessUtils.getAddress(Me.document, _
                                          "Organization",       ObjOrganization.Id)
ObjOrganization.TelephoneNumbers=XMLProcessUtils.getTelephoneNumbers(Me.document,_
                                         "Organization", ObjOrganization.Id)
                    Me.AddPostalAddress(ObjOrganization.Address)
                    Dim wrkTelephoneNumber As TelephoneNumber
                    For Each wrkTelephoneNumber In ObjOrganization.TelephoneNumbers
                        Me.AddTelephoneNumber(wrkTelephoneNumber)
                    Next

                    'Adiciona o objeto ao dataSet de Organization
                    Me.AddOrganization(ObjOrganization)

                    ...
                    'Cria evento de auditoria para o objeto
                    Dim ObjAuditableEvent As New AuditableEvent
                    ObjAuditableEvent.Id = Guid.NewGuid.ToString
                    ObjAuditableEvent.AccessControlPolicy = "ContentOwner"
                    ObjAuditableEvent.ObjectType = "AuditableEvent"
                    ObjAuditableEvent.RegistryObject = ObjOrganization.Id
                    ObjAuditableEvent.EventType = "Created"
                    ObjAuditableEvent.TimeStamp = Date.Now
                    ObjAuditableEvent.User = Me.wrkUserID
                    Me.AddAuditableEvent(ObjAuditableEvent)
                End If
          ...
```

```
          Case "USER"

               pNode.MoveToContent()
               If pNode.HasAttributes Then
                   Dim ObjUser As New User

                   'Atributos herdados da classe RegistryObject
                   ObjUser.Id = Me.wrkUserID
                   ObjUser.AccessControlPolicy = "ContentOwner"
                   ObjUser.ObjectType = "User"
                   'Atributos específicos desta classe
                   ObjUser.Organization = Me.wrkOrganization
                   ObjUser.Address = XMLProcessUtils.getAddress(Me.document, _
                                                          "User", ObjUser.Id)
                  ObjUser.PersonName = XMLProcessUtils.getPersonName(Me.document,_
                                                          "User", ObjUser.Id)
        ObjUser.TelephoneNumbers=XMLProcessUtils.getTelephoneNumbers(Me.document,_
                                                          "User", ObjUser.Id)
               ObjUser.EmailAddress= XMLProcessUtils.getEmailAddress(Me.document,_
                                                    "User", ObjUser.Id)
                   Me.AddPostalAddress(ObjUser.Address)
                   'Adiciona os telefones e emails nos seus respectivos datasets
                   Dim wrkEmailAddress As EmailAddress
                   For Each wrkEmailAddress In ObjUser.EmailAddress
                       Me.AddEmailAddress(wrkEmailAddress)
                   Next
                   Dim wrkTelephoneNumber As TelephoneNumber
                   For Each wrkTelephoneNumber In ObjUser.TelephoneNumbers
                       Me.AddTelephoneNumber(wrkTelephoneNumber)
                   Next

                   'Passar pelo RegistryObjectType
                   Do While pNode.Read
                       If pNode.NodeType = XmlNodeType.Element Then
                           Select Case pNode.LocalName.ToUpper
                               Case "SLOT", "CLASSIFICATION", "EXTERNALIDENTIFIER"
                                   Me.ProcessaNodeReader(pNode, ObjUser.Id)
                           End Select
                       End If
                       If pNode.NodeType = XmlNodeType.EndElement AndAlso _
                          pNode.LocalName.ToUpper = "USER" Then
                           Exit Do
                       End If
                   Loop

                   'Adiciona o objeto ao dataSet de Users
                   Me.AddUser(ObjUser)

                   'Cria uma associação do objeto submetido com a organização
submetente
                   Dim ObjAssociationOrg As New Association
                   ObjAssociationOrg.Id = Guid.NewGuid.ToString
                   ObjAssociationOrg.AccessControlPolicy = "ContentOwner"
                   ObjAssociationOrg.ObjectType = "Association"
                   ObjAssociationOrg.AssociationType = "SubmitterOf"
                   ObjAssociationOrg.SourceObject = Me.wrkOrganization
                   ObjAssociationOrg.TargetObject = ObjUser.Id
                   ObjAssociationOrg.IsConfirmedBySourceOwner = True
                   ObjAssociationOrg.IsConfirmedByTargetOwner = True
                   Me.AddAssociation(ObjAssociationOrg)

                   'Cria evento de auditoria para o objeto
                   Dim ObjAuditableEvent As New AuditableEvent
                   ObjAuditableEvent.Id = Guid.NewGuid.ToString
                   ObjAuditableEvent.AccessControlPolicy = "ContentOwner"
                   ObjAuditableEvent.ObjectType = "AuditableEvent"
```

```
                    ObjAuditableEvent.RegistryObject = ObjUser.Id
                    ObjAuditableEvent.EventType = "Created"
                    ObjAuditableEvent.TimeStamp = Date.Now
                    ObjAuditableEvent.User = Me.wrkUserID
                    Me.AddAuditableEvent(ObjAuditableEvent)
                End If
        End Select
    End Sub

    Public Overrides Sub CommitObjects()

        'Método para salvar objetos no banco de dados.
        'As transações devem ser realizadas como um todo, caso haja erro de ser
abortada
        'Se houve erro, não faz nada
        If Me.Erro Then
            Exit Sub
        End If

        Dim StringConexao As String = ebRS_Config.getDSN_BancoDados
        Dim OdbcConn As New OdbcConnection(StringConexao)

        OdbcConn.Open()
        Dim OdbcCmd As OdbcCommand = OdbcConn.CreateCommand()
        Dim OdbcTrans As OdbcTransaction

        'Inicia transação local
        OdbcTrans = OdbcConn.BeginTransaction(IsolationLevel.ReadCommitted)
        OdbcCmd.Transaction = OdbcTrans

        Try
            'Verifica se existe Objetos nos DataSet para serem salvos
            ...
            If dstUser.user_.Count > 0 Then
                Dim dtaUser As New daUser
                'Insere os usuarios no banco de dados
                dtaUser.InsertUser(OdbcCmd, dstUser)
                dtaUser.Dispose()
            End If

            If dstOrganization.organization.Count > 0 Then
                Dim dtaOrganization As New daOrganization
                'Insere as organizações no banco de dados
                dtaOrganization.InsertOrganization(OdbcCmd, dstOrganization)
                dtaOrganization.Dispose()
            End If
            ...
            OdbcTrans.Commit()
        Catch ex As Exception
            Me.Erro = True
            Me.ObjResponse.Status = StatusResponse.Failure.ToString
            Me.ObjResponse.Documentation = "Exception: " & ex.Message & " - " &
ex.StackTrace
            OdbcTrans.Rollback()
        Finally
            OdbcConn.Close()
        End Try
    End Sub
End Class
```

APÊNDICE C – Mensagem de teste do caso de uso SubmitObjects

A requisição utilizada como teste é apresentada no Quadro a seguir. Esta requisição, trata de uma requisição fictícia, onde a Organização requisitante é a General Motors Corp. (GM). Esta requisição foi utilizada como teste, pois demonstra a submissão de diversos objetos. Exemplos de testes podem ser encontrados em (SAMPLES, 2004).

Inicialmente (linhas 11-20) é realizado o cadastro de um novo usuário (objeto *User*) com o nome “GMUser”. Este usuário será afiliado à organização “General Motors Corp.” (objeto *Organization*), sendo cadastrada nas linhas (21-35).

Nas linhas (36-43) está sendo cadastrado um link (objeto *ExternalLink*) para o web site da GM. Uma associação (objeto *Association*) entre este link e a organização GM é criado nas linhas 44 e 45.

Um esquema de classificação (objeto *ClassificationScheme*) e dois nodos (objeto *ClassificationNode*) ligados a ele são criados nas linhas (47-68). Nas linhas (69-203) é criado um pacote (objeto *RegistryPackage*) utilizado para agrupar diversos DTD’s (objetos *ExtrinsicObjects*) que descrevem como realizar *e-Business* com GM para a compra de automóveis.

Finalmente, nas linhas (209-211) é criada uma associação deste pacote com a organização GM.

```
1 <?xml version="1.0" ?>
2 <rs:SubmitObjectsRequest
3   xmlns="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.1"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.1
6  http://www.oasis-open.org/committees/regrep/documents/2.1/schema/rim.xsd
7  http://www.oasis-open.org/committees/regrep/documents/2.1/schema/rs.xsd
8  xmlns:rim="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.1"
9  xmlns:rs="urn:oasis:names:tc:ebxml-regrep:registry:xsd:2.1">
10 <LeafRegistryObjectList>
11     <User id="urn:uuid:1bd9679c-0fd5-4b39-b413-c87f193bf2fa"
12        organization="gmOrgId">
13        <Address streetNumber="1" street="Jefferson Dr" city="Burlington"
14            stateOrProvince="MA" country="USA" />
15        <PersonName firstName="GMUser" middleName="Middle"
16            lastName="Last" />
17        <TelephoneNumber countryCode="1" areaCode="407"
18            number="123-1234" phoneType="office" />
19       <EmailAddress address="GMUser@GM.COM" type="work" />
20   </User>
21   <Organization id="gmOrgId" primaryContact=
```

```
                        "urn:uuid:1bd9679c-0fd5- 4b39-b413-c87f193bf2fa">
      <Name>
          <LocalizedString lang="us-en" value="General Motors Corp." />
      </Name>
      <Description>
          <LocalizedString lang="us-en" value="World's largest
              automobile manufacturer. The world's automotive sales
              leader since 1931!" />
      </Description>
      <Address streetNumber="1" street="Jefferson Dr" city="Detroit"
          stateOrProvince="MI" postalCode="01234" country="USA" />
      <TelephoneNumber countryCode="1" areaCode="351" number="123-1234"
          phoneType="office" />
</Organization>
<ExternalLink id="gmWebSiteId" externalURI="http://www.gm.com">
    <Name>
        <LocalizedString value="General Motors - Welcome to GM.com" />
    </Name>
    <Description>
        <LocalizedString value="Main web site for GM" />
    </Description>
</ExternalLink>
<Association id="gmOrgId-gmWebSiteId" associationType="ExternallyLinks"
     sourceObject="gmWebSiteId" targetObject="gmOrgId" />
<!-- Custom Classification schemes -->
<ClassificationScheme id="bpsClassificationSchemeId"
     isInternal="true" nodeType="UniqueCode"
     xmlns="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.1">
     <Name>
         <LocalizedString charset="UTF-8" value="Business Process
             Classification Scheme" />
     </Name>
     <Description>
         <LocalizedString charset="UTF-8" value="This is the
           classification scheme for classifying business processes" />
     </Description>
     <ClassificationNode id="purchasingId" code="Purchasing">
         <Name>
             <LocalizedString charset="UTF-8" value="Purchasing" />
         </Name>
     </ClassificationNode>
     <ClassificationNode id="envoicingId" code="Invoicing">
         <Name>
             <LocalizedString charset="UTF-8" value="Invoicing" />
         </Name>
     </ClassificationNode>
</ClassificationScheme>
<RegistryPackage id="rentalCarPurchasePackageId">
     <Name>
         <LocalizedString value="Rental Car Purchasing Package" />
     </Name>
     <Description>
        <LocalizedString value="Contains technical documents related to
Rental Car Purchasing process" />
     </Description>
     <RegistryObjectList>
        <ExtrinsicObject id="CPAId" objectType="CPA">
             <Name>
                 <LocalizedString value="Template CPA for Rental Car
                         Buyers" />
```

```
82                </Name>
83                <Description>
84                    <LocalizedString value="This template Colloboration
85 Protocol Agreemnt (CPA) is specified by GM for use by GM's Rental Car
86 Buyers" />
87                </Description>
88           </ExtrinsicObject>
89           <Association id="gmOrgId-CPAId"
90                 associationType="HasTemplateCPA"
91                 sourceObject="gmOrgId" targetObject="CPAId" />
92           <ExtrinsicObject id="BPSId" objectType="BusinessProcess">
93               <Name>
94                  <LocalizedString value="Rental Car Purchasing Process"/>
95               </Name>
96               <Description>
97                 <LocalizedString value="Business Process Specification
98 (BPS) for Rental Car Purchasing Process" />
99               </Description>
100              <Classification classificationNode="purchasingId"
101                   classifiedObject="BPSId">
102                   <Name>
103                      <LocalizedString lang="us-en" value="Purchasing"/>
104                   </Name>
105              </Classification>
106          </ExtrinsicObject>
107          <Association id="gmOrgId-BPSId"
108              associationType="HasBusinessProcess" sourceObject="gmOrgId"
109              targetObject="BPSId" />
110          <RegistryPackage id="rentalCarPurchaseDTDsPackageId">
111             <Name>
112               <LocalizedString value="Rental Car Purchasing DTDs
113 Package"/>
114              </Name>
115              <Description>
116                 <LocalizedString value="Contains DTDs defining messages
117 used by Rental Car Purchasing process" />
118              </Description>
119              <RegistryObjectList>
120                   <!-- Next are 8 DTDs used to define messages exchanged
121                   during the rental car purchasing process -->
122                   <ExtrinsicObject id="dtd1Id" objectType="DTD">
123                       <Name>
124                          <LocalizedString value="DTD for Available
125 Vehicles Query" />
126                       </Name>
127                       <Description>
128                         <LocalizedString value="Defines message
129 structure for Available Vehicles Query" />
130                       </Description>
131                   </ExtrinsicObject>
132                   <ExtrinsicObject id="dtd2Id" objectType="DTD">
133                       <Name>
134                          <LocalizedString value="DTD for Available
135 Vehicles Response" />
136                       </Name>
137                      <Description>
138                          <LocalizedString value="Defines message
139 structure for Available Vehicles Response" />
140                      </Description>
141                   </ExtrinsicObject>
```

```
142                 <ExtrinsicObject id="dtd3Id" objectType="DTD">
143                     <Name>
144                         <LocalizedString value="DTD for Dealer
145 Location Query" />
146                     </Name>
147                     <Description>
148                         <LocalizedString value="Defines message
149 structure for Dealer Location Query" />
150                      </Description>
151                 </ExtrinsicObject>
152                 <ExtrinsicObject id="dtd4Id" objectType="DTD">
153                      <Name>
154                         <LocalizedString value="DTD for Dealer
155 Location Response" />
156                      </Name>
157                      <Description>
158                           <LocalizedString value="Defines message
159 structure for Dealer Location Response" />
160                      </Description>
161                   </ExtrinsicObject>
162                   <ExtrinsicObject id="dtd5Id" objectType="DTD">
163                       <Name>
164                           <LocalizedString value="DTD for Vehicles
165 History Query" />
166                       </Name>
167                       <Description>
168                           <LocalizedString value="Defines message
169 structure for Vehicles History Query" />
170                       </Description>
171                 </ExtrinsicObject>
172                 <ExtrinsicObject id="dtd6Id" objectType="DTD">
173                     <Name>
174                         <LocalizedString value="DTD for Vehicles
175 History Response" />
176                       </Name>
177                       <Description>
178                          <LocalizedString value="Defines message
179 structure for Vehicles History Response" />
180                       </Description>
181                 </ExtrinsicObject>
182                 <ExtrinsicObject id="dtd7Id" objectType="DTD">
183                     <Name>
184                         <LocalizedString value="DTD for Vehicles
185 Purchase Request" />
186                     </Name>
187                     <Description>
188                         <LocalizedString value="Defines message
189 structure for Vehicles Purchase Request" />
190                       </Description>
191                 </ExtrinsicObject>
192                 <ExtrinsicObject id="dtd8Id" objectType="DTD">
193                      <Name>
194                          <LocalizedString value="DTD for Vehicles
195 Purchase Approval" />
196                      </Name>
197                      <Description>
198                          <LocalizedString value="Defines message
199 structure for Vehicles Purchase Approval" />
200                      </Description>
201                   </ExtrinsicObject>
```

```
202                </RegistryObjectList>
203           </RegistryPackage>
204           <Association id="BPSId-rentalCarPurchaseDTDsPackageId"
205               associationType="UsesDTD" sourceObject="BPSId"
206               targetObject="rentalCarPurchaseDTDsPackageId" />
207        </RegistryObjectList>
208    </RegistryPackage>
209    <Association id="gmOrgId-rentalCarPurchasePackageId"
210       associationType="HasPackage" sourceObject="gmOrgId"
211       targetObject="rentalCarPurchasePackageId" />
212    </LeafRegistryObjectList>
213 </rs:SubmitObjectsRequest>
```

ANEXO A – Exemplo de um CPP

O seguinte quadro demonstra um exemplo de um documento contendo um perfil de protocolo de colaboração. Este exemplo pode ser obtido em (ebXML, 2004).

```
<?xml version="1.0" encoding="UTF-8"?>
<tp:CollaborationProtocolProfile
      xmlns:tp="http://www.ebxml.org/namespaces/tradePartner"
      xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
      xmlns:xlink="http://www.w3.org/1999/xlink"
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
      xsi:schemaLocation="http://www.ebxml.org/namespaces/tradePartner
            http://ebxml.org/project_teams/trade_partner/cpp-cpa-v1_0.xsd"
      tp:version="1.1">
      <tp:PartyInfo>
            <tp:PartyId tp:type="DUNS">123456789</tp:PartyId>
            <tp:PartyRef tp:href="http://example.com/about.html"/>
            <tp:CollaborationRole tp:id="N00">
                <tp:ProcessSpecification tp:version="1.0"
                  tp:name="buySell"
                  xlink:type="simple"
                  xlink:href="http://www.ebxml.org/processes/buySell.xml"/>
               <tp:Role tp:name="buyer" xlink:type="simple"
                xlink:href="http://ebxml.org/processes/buySell.xml#buyer"/>
               <tp:CertificateRef tp:certId="N03"/>
               <tp:ServiceBinding tp:channelId="N04" tp:packageId="N0402">
                <tp:Service tp:type="uriReference">
                    uri:example.com/services/buyerService
                </tp:Service>
                <tp:Override tp:action="orderConfirm"
                    tp:channelId="N07" tp:packageId="N0402"
            xlink:href="http://ebxml.org/processes/buySell.xml#orderConfirm"
            xlink:href=type="simple"/>
                </tp:ServiceBinding>
            </tp:CollaborationRole>
            <tp:Certificate tp:certId="N03">
                <ds:KeyInfo/>
            </tp:Certificate>
            <tp:DeliveryChannel tp:channelId="N04" tp:transportId="N05"
                tp:docExchangeId="N06">
                <tp:Characteristics tp:syncReplyMode="none"
                    tp:nonrepudiationOfOrigin="true"
                    tp:confidentiality="true" tp:authenticated="true"
                    tp:authorized="false"/>
            </tp:DeliveryChannel>
            <tp:DeliveryChannel tp:channelId="N07" tp:transportId="N08"
                tp:docExchangeId="N06">
                <tp:Characteristics tp:syncReplyMode="none"
                    tp:nonrepudiationOfOrigin="true"
                    tp:nonrepudiationOfReceipt="false"
                    tp:secureTransport="false"
                    tp:confidentiality="true" tp:authenticated="true"
                    tp:authorized="false"/>
            </tp:DeliveryChannel>
            <tp:Transport tp:transportId="N05">
                <tp:SendingProtocol tp:version="1.1">
                    HTTP
                </tp:SendingProtocol>
```

```
                <tp:ReceivingProtocol tp:version="1.1">
                    HTTP
                </tp:ReceivingProtocol>
                <tp:Endpoint
                  tp:uri="https://www.example.com/servlets/ebxmlhandler"
                  tp:type="allPurpose"/>
                <tp:TransportSecurity>
                    <tp:Protocol tp:version="3.0">SSL</tp:Protocol>
                    <tp:CertificateRef tp:certId="N03"/>
                </tp:TransportSecurity>
            </tp:Transport>
            <tp:Transport tp:transportId="N08">
                <tp:SendingProtocol tp:version="1.1">
                    HTTP
                </tp:SendingProtocol>
                <tp:ReceivingProtocol tp:version="1.1">
                    SMTP
                </tp:ReceivingProtocol>
                <tp:Endpoint
                    tp:uri="mailto:ebxmlhandler@example.com"
                    tp:type="allPurpose"/>
            </tp:Transport>
            <tp:DocExchange tp:docExchangeId="N06">
                <tp:ebXMLBinding tp:version="0.98b">
                    <tp:ReliableMessaging
                        tp:deliverySemantics="OnceAndOnlyOnce"
                        tp:idempotency="true"
                        tp:messageOrderSemantics="Guaranteed">
                           <tp:Retries>5</tp:Retries>
                           <tp:RetryInterval>30</tp:RetryInterval>
                           <tp:PersistDuration>P1D</tp:PersistDuration>
                    </tp:ReliableMessaging>
                    <tp:NonRepudiation>
                        <tp:Protocol>http://www.w3.org/2000/09/xmldsig#
                        </tp:Protocol>
                        <tp:HashFunction>
                            http://www.w3.org/2000/09/xmldsig#sha1
                        </tp:HashFunction>
                        <tp:SignatureAlgorithm>
                            http://www.w3.org/2000/09/xmldsig#dsa-sha1
                        </tp:SignatureAlgorithm>
                        <tp:CertificateRef tp:certId="N03"/>
                    </tp:NonRepudiation>
                    <tp:DigitalEnvelope>
                        <tp:Protocol tp:version="2.0">
                            S/MIME
                        </tp:Protocol>
                        <tp:EncryptionAlgorithm>
                            DES-CBC
                        </tp:EncryptionAlgorithm>
                        <tp:CertificateRef tp:certId="N03"/>
                    </tp:DigitalEnvelope>
                </tp:ebXMLBinding>
             </tp:DocExchange>
        </tp:PartyInfo>
        <tp:Packaging tp:id="N0402">
            <tp:ProcessingCapabilities tp:parse="true" tp:generate="true"/>
            <tp:SimplePart tp:id="N40" tp:mimetype="text/xml">
                <tp:NamespaceSupported
                    tp:location=
```

```
              "http://ebxml.org/project_teams/transport/messageService.xsd"
                   tp:version="0.98b">
                      http://www.ebxml.org/namespaces/messageService
               </tp:NamespaceSupported>
               <tp:NamespaceSupported
                   tp:location=
         "http://ebxml.org/project_teams/transport/xmldsig-core-schema.xsd"
                   tp:version="1.0">http://www.w3.org/2000/09/xmldsig
               </tp:NamespaceSupported>
           </tp:SimplePart>
           <tp:SimplePart tp:id="N41" tp:mimetype="text/xml">
               <tp:NamespaceSupported tp:location=
                   "http://ebxml.org/processes/buysell.xsd"
                   tp:version="1.0">
                      http://ebxml.org/processes/buysell.xsd
               </tp:NamespaceSupported>
            </tp:SimplePart>
            <tp:CompositeList>
                <tp:Composite tp:id="N42" tp:mimetype="multipart/related"
                    tp:mimeparameters="type=text/xml;">
                    <tp:Constituent tp:idref="N40"/>
                    <tp:Constituent tp:idref="N41"/>
                </tp:Composite>
             </tp:CompositeList>
       </tp:Packaging>
       <tp:Comment tp:xml_lang="en-us">
           buy/sell agreement between example.com and contrived-example.com
       </tp:Comment>
</tp:CollaborationProtocolProfile>
```

ANEXO B – Exemplo de um CPA

O seguinte quadro demonstra um documento contendo um acordo de protocolo de colaboração. Este exemplo pode ser obtido em (ebXML, 2004).

```
<?xml version="1.0"?>
<tp:CollaborationProtocolAgreement
    xmlns:tp="http://www.ebxml.org/namespaces/tradePartner"
    xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
    xsi:schemaLocation=
        "http://www.ebxml.org/namespaces/tradePartner
        http://ebxml.org/project_teams/trade_partner/cpp-cpa-v1_0.xsd"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    tp:cpaid="uri:yoursandmycpa"
    tp:version="1.2">
    <tp:Status tp:value="proposed"/>
    <tp:Start>2001-05-20T07:21:00Z</tp:Start>
    <tp:End>2002-05-20T07:21:00Z</tp:End>
    <tp:ConversationConstraints tp:invocationLimit="100"
        tp:concurrentConversations="100"/>
    <tp:PartyInfo>
        <tp:PartyId tp:type="DUNS">123456789</tp:PartyId>
        <tp:PartyRef xlink:href="http://example.com/about.html"/>
        <tp:CollaborationRole tp:id="N00">
            <tp:ProcessSpecification tp:version="1.0" tp:name="buySell"
                xlink:type="simple"
                xlink:href="http://www.ebxml.org/processes/buySell.xml"/>
            <tp:Role tp:name="buyer" xlink:type="simple"
                xlink:href="http://ebxml.org/processes/buySell.xml#buyer"/>
            <tp:CertificateRef tp:certId="N03"/>
            <tp:ServiceBinding tp:channelId="N04" tp:packageId="N0402">
                <tp:Service tp:type="uriReference">
                    uri:example.com/services/buyerService</tp:Service>
                <tp:Override tp:action="orderConfirm" tp:channelId="N08"
                    tp:packageId="N0402"
                    xlink:href=
                      "http://ebxml.org/processes/buySell.xml#orderConfirm"
                    xlink:type="simple"/>
           </tp:ServiceBinding>
        </tp:CollaborationRole>
        <tp:Certificate tp:certId="N03">
            <ds:KeyInfo/>
        </tp:Certificate>
        <tp:DeliveryChannel tp:channelId="N04"
            tp:transportId="N05" tp:docExchangeId="N06">
            <tp:Characteristics tp:syncReplyMode="none"
                tp:nonrepudiationOfOrigin="true"
                tp:nonrepudiationOfReceipt="false"
                tp:secureTransport="true"
                tp:confidentiality="true"
                tp:authenticated="true"
                tp:authorized="false"/>
        </tp:DeliveryChannel>
        <tp:DeliveryChannel tp:channelId="N07" tp:transportId="N08"
            tp:docExchangeId="N06">
            <tp:Characteristics tp:syncReplyMode="none"
```

```
                tp:nonrepudiationOfOrigin="true"
                tp:nonrepudiationOfReceipt="false"
                tp:secureTransport="false"
                tp:confidentiality="true"
                tp:authenticated="true"
                tp:authorized="false"/>
        </tp:DeliveryChannel>
        <tp:Transport tp:transportId="N05">
            <tp:SendingProtocol
                tp:version="1.1">HTTP</tp:SendingProtocol>
            <tp:ReceivingProtocol
                tp:version="1.1">HTTP</tp:ReceivingProtocol>
            <tp:Endpoint
                tp:uri="https://www.example.com/servlets/ebxmlhandler"
            tp:type="allPurpose"/>
            <tp:TransportSecurity>
                <tp:Protocol tp:version="3.0">SSL</tp:Protocol>
                <tp:CertificateRef tp:certId="N03"/>
            </tp:TransportSecurity>
        </tp:Transport>
        <tp:Transport tp:transportId="N18">
            <tp:SendingProtocol
                tp:version="1.1">HTTP</tp:SendingProtocol>
            <tp:ReceivingProtocol
                tp:version="1.1">SMTP</tp:ReceivingProtocol>
            <tp:Endpoint
                tp:uri="mailto:ebxmlhandler@example.com"
                   tp:type="allPurpose"/>
        </tp:Transport>
        <tp:DocExchange tp:docExchangeId="N06">
            <tp:ebXMLBinding tp:version="0.98b">
                <tp:ReliableMessaging
                    tp:deliverySemantics="OnceAndOnlyOnce"
                    tp:idempotency="true"
                    tp:messageOrderSemantics="Guaranteed">
                    <tp:Retries>5</tp:Retries>
                    <tp:RetryInterval>30</tp:RetryInterval>
                    <tp:PersistDuration>P1D</tp:PersistDuration>
                </tp:ReliableMessaging>
                <tp:NonRepudiation>
                    <tp:Protocol>
                        http://www.w3.org/2000/09/xmldsig#
                    </tp:Protocol>
                    <tp:HashFunction>
                        http://www.w3.org/2000/09/xmldsig#sha1
                    </tp:HashFunction>
                    <tp:SignatureAlgorithm>
                        http://www.w3.org/2000/09/xmldsig#dsa-sha1
                    </tp:SignatureAlgorithm>
                    <tp:CertificateRef tp:certId="N03"/>
                </tp:NonRepudiation>
                <tp:DigitalEnvelope>
                    <tp:Protocol tp:version="2.0">S/MIME</tp:Protocol>
                    <tp:EncryptionAlgorithm>
                        DES-CBC
                    </tp:EncryptionAlgorithm>
                    <tp:CertificateRef tp:certId="N03"/>
                </tp:DigitalEnvelope>
            </tp:ebXMLBinding>
        </tp:DocExchange>
```

```
    </tp:PartyInfo>
    <tp:PartyInfo>
        <tp:PartyId tp:type="DUNS">987654321</tp:PartyId>
            <tp:PartyRef xlink:type="simple"
                xlink:href="http://contrived-example.com/about.html"/>
            <tp:CollaborationRole tp:id="N30">
                <tp:ProcessSpecification tp:version="1.0"
                    tp:name="buySell" xlink:type="simple"
                    xlink:href=
                          "http://www.ebxml.org/processes/buySell.xml"/>
                <tp:Role tp:name="seller" xlink:type="simple"
                    xlink:href=
                          "http://ebxml.org/processes/buySell.xml#seller"/>
                <tp:CertificateRef tp:certId="N33"/>
                <tp:ServiceBinding tp:channelId="N34" tp:packageId="N0402">
                    <tp:Service
                        tp:type="uriReference">
                           uri:example.com/services/sellerService
                    </tp:Service>
                </tp:ServiceBinding>
            </tp:CollaborationRole>
            <tp:Certificate tp:certId="N33">
                <ds:KeyInfo/>
            </tp:Certificate>
            <tp:DeliveryChannel tp:channelId="N34"
                tp:transportId="N35" tp:docExchangeId="N36">
                <tp:Characteristics tp:nonrepudiationOfOrigin="true"
                    tp:nonrepudiationOfReceipt="false"
                    tp:secureTransport="true"
                    tp:confidentiality="true"
                    tp:authenticated="true"
                    tp:authorized="false"/>
            </tp:DeliveryChannel>
            <tp:Transport tp:transportId="N35">
                <tp:SendingProtocol
                    tp:version="1.1">HTTP</tp:SendingProtocol>
                <tp:ReceivingProtocol
                    tp:version="1.1">HTTP</tp:ReceivingProtocol>
                <tp:Endpoint tp:uri=
                  "https://www.contrived-example.com/servlets/ebxmlhandler"
                    tp:type="allPurpose"/>
                <tp:TransportSecurity>
                    <tp:Protocol
                        tp:version="3.0">SSL</tp:Protocol>
                    <tp:CertificateRef tp:certId="N33"/>
                </tp:TransportSecurity>
            </tp:Transport>
            <tp:DocExchange tp:docExchangeId="N36">
                <tp:ebXMLBinding tp:version="0.98b">
                    <tp:ReliableMessaging
                        tp:deliverySemantics="OnceAndOnlyOnce"
                        tp:idempotency="true"
                        tp:messageOrderSemantics="Guaranteed">
                        <tp:Retries>5</tp:Retries>
                        <tp:RetryInterval>30</tp:RetryInterval>
                        <tp:PersistDuration>P1D</tp:PersistDuration>
                    </tp:ReliableMessaging>
                    <tp:NonRepudiation>
                        <tp:Protocol>
                            http://www.w3.org/2000/09/xmldsig#
```

```
                        </tp:Protocol>
                        <tp:HashFunction>
                             http://www.w3.org/2000/09/xmldsig#sha1
                        </tp:HashFunction>
                        <tp:SignatureAlgorithm>
                            http://www.w3.org/2000/09/xmldsig#dsa-sha1
                        </tp:SignatureAlgorithm>
                        <tp:CertificateRef tp:certId="N33"/>
                    </tp:NonRepudiation>
                    <tp:DigitalEnvelope>
                        <tp:Protocol
                            tp:version="2.0">S/MIME</tp:Protocol>
                        <tp:EncryptionAlgorithm>
                            DES-CBC</tp:EncryptionAlgorithm>
                        <tp:CertificateRef tp:certId="N33"/>
                    </tp:DigitalEnvelope>
                </tp:ebXMLBinding>
            </tp:DocExchange>
    </tp:PartyInfo>
    <tp:Packaging tp:id="N0402">
        <tp:ProcessingCapabilities tp:parse="true" tp:generate="true"/>
        <tp:SimplePart tp:id="N40" tp:mimetype="text/xml">
           <tp:NamespaceSupported tp:location=
              "http://ebxml.org/project_teams/transport/messageService.xsd"
               tp:version="0.98b">
                  http://www.ebxml.org/namespaces/messageService
           </tp:NamespaceSupported>
           <tp:NamespaceSupported tp:location=
         "http://ebxml.org/project_teams/transport/xmldsig-core-schema.xsd"
                tp:version="1.0">
                  http://www.w3.org/2000/09/xmldsig
           </tp:NamespaceSupported>
        </tp:SimplePart>
        <tp:SimplePart tp:id="N41" tp:mimetype="text/xml">
            <tp:NamespaceSupported tp:location=
                            "http://ebxml.org/processes/buysell.xsd"
                tp:version="1.0">
                   http://ebxml.org/processes/buysell.xsd
            </tp:NamespaceSupported>
        </tp:SimplePart>
        <tp:CompositeList>
            <tp:Composite tp:id="N42"
                tp:mimetype="multipart/related"
                tp:mimeparameters="type=text/xml;">
                <tp:Constituent tp:idref="N40"/>
                <tp:Constituent tp:idref="N41"/>
            </tp:Composite>
        </tp:CompositeList>
    </tp:Packaging>
    <tp:Comment
        xml:lang="en-us">
           buy/sell agreement between example.com and contrived-example.com
    </tp:Comment>
</tp:CollaborationProtocolAgreement>
```

ANEXO C – Exemplo de um Esquema de Especificação de Processos de Negócio

O seguinte quadro demonstra um documento que define os processos de negócios suportados por um parceiro comercial. Este exemplo pode ser obtido em (ebXML, 2004).

```
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Kurt Kanaskie
(Lucent Technologies) -->
<!-- edited by Kurt Kanaskie (Lucent Technologies) -->
<!DOCTYPE ProcessSpecification SYSTEM
    "ebXMLProcessSpecification- v1.01.dtd">
<ProcessSpecification name="Simple" version="1.1"
    uuid="[1234-5678-901234]">
    <!-- Business Documents -->
    <BusinessDocument name="Catalog Request"/>
    <BusinessDocument name="Catalog"/>
    <BusinessDocument name="Purchase Order"/>
    <BusinessDocument name="PO Acknowledgement"/>
    <BusinessDocument name="Credit Request"/>
    <BusinessDocument name="Credit Confirm"/>
    <BusinessDocument name="ASN"/>
    <BusinessDocument name="CreditAdvice"/>
    <BusinessDocument name="DebitAdvice"/>
    <BusinessDocument name="Invoice"/>
    <BusinessDocument name="Payment"/>
    <BusinessDocument name="Inventory Report Request"/>
    <BusinessDocument name="Inventory Report"/>
    <BusinessDocument name="Inventory Report"/>
    <Package name="Ordering">
        <!-- First the overall MultiParty Collaboration -->
        <MultiPartyCollaboration name="DropShip">
            <BusinessPartnerRole name="Customer">
                <Performs initiatingRole="requestor"/>
                <Performs initiatingRole="buyer"/>
                <Transition fromBusinessState="Catalog Request"
                    toBusinessState="Create Order"/>
            </BusinessPartnerRole>
            <BusinessPartnerRole name="Retailer">
                <Performs respondingRole="provider"/>
                <Performs respondingRole="seller"/>
                <Performs initiatingRole="Creditor"/>
                <Performs initiatingRole="buyer"/>
                <Performs initiatingRole="Payee"/>
                <Performs respondingRole="Payor"/>
                <Performs initiatingRole="requestor"/>
                <Transition fromBusinessState="Create Order"
                    toBusinessState="Check Credit"/>
                <Transition fromBusinessState="Check Credit"
                    toBusinessState="Create Order"/>
            </BusinessPartnerRole>
            <BusinessPartnerRole name="DropShip Vendor">
                <Performs respondingRole="seller"/>
                <Performs initiatingRole="payee"/>
                <Performs respondingRole="provider"/>
            </BusinessPartnerRole>
            <BusinessPartnerRole name="Credit Authority">
                <Performs respondingRole="credit service"/>
```

```
            <Performs respondingRole="payor"/>
        </BusinessPartnerRole>
    </MultiPartyCollaboration>
    <!-- Now the Binary Collaborations -->
    <BinaryCollaboration name="Request Catalog">
        <InitiatingRole name="requestor"/>
        <RespondingRole name="provider"/>
        <BusinessTransactionActivity name="Catalog Request"
             businessTransaction="Catalog Request"
             fromAuthorizedRole="requestor"
             toAuthorizedRole="provider"/>
    </BinaryCollaboration>
    <BinaryCollaboration name="Firm Order" timeToPerform="P2D">
        <Documentation>
            timeToPerform = Period: 2 days from start of transaction
        </Documentation>
        <InitiatingRole name="buyer"/>
        <RespondingRole name="seller"/>
        <BusinessTransactionActivity name="Create Order"
            businessTransaction="Create Order"
            fromAuthorizedRole="buyer"
            toAuthorizedRole="seller"/>
    </BinaryCollaboration>
    <BinaryCollaboration
        name="Product Fulfillment" timeToPerform="P5D">
        <Documentation>
            timeToPerform = Period: 5 days from start of transaction
        </Documentation>
        <InitiatingRole name="buyer"/>
        <RespondingRole name="seller"/>
        <BusinessTransactionActivity name="Create Order"
            businessTransaction="Create Order"
            fromAuthorizedRole="buyer" toAuthorizedRole="seller"/>
        <BusinessTransactionActivity name="Notify shipment"
            businessTransaction="Notify of advance shipment"
            fromAuthorizedRole="buyer" toAuthorizedRole="seller"/>
        <Start toBusinessState="Create Order"/>
        <Transition fromBusinessState="Create Order"
            toBusinessState="Notify shipment"/>
        <Success fromBusinessState="Notify shipment"
            conditionGuard="Success"/>
        <Failure fromBusinessState="Notify shipment"
            conditionGuard="BusinessFailure"/>
    </BinaryCollaboration>
    <BinaryCollaboration name="Inventory Status">
        <InitiatingRole name="requestor"/>
        <RespondingRole name="provider"/>
        <BusinessTransactionActivity name="Inventory Report Request"
            businessTransaction="Inventory Report Request"
            fromAuthorizedRole="requestor"
            toAuthorizedRole="provider"/>
        <BusinessTransactionActivity name="Inventory Report"
            businessTransaction="Inventory Report"
            fromAuthorizedRole="provider"
            toAuthorizedRole="requestor"/>
    </BinaryCollaboration>
    <BinaryCollaboration name="Credit Inquiry">
        <InitiatingRole name="creditor"/>
        <RespondingRole name="credit service"/>
         <BusinessTransactionActivity name="Check Credit"
```

```
                businessTransaction="Check Credit"
                fromAuthorizedRole="creditor"
                toAuthorizedRole="credit service"/>
        </BinaryCollaboration>
        <BinaryCollaboration name="Credit Payment">
            <InitiatingRole name="payee"/>
            <RespondingRole name="payor"/>
            <BusinessTransactionActivity name="Process Credit Payment"
                businessTransaction="Process Credit Payment"
                fromAuthorizedRole="payee"
                toAuthorizedRole="payor"/>
        </BinaryCollaboration>
        <!-- A compound BinaryCollaboration for illustration purposes-->
        <BinaryCollaboration name="Credit Charge">
            <InitiatingRole name="charger"/>
            <RespondingRole name="credit service"/>
            <CollaborationActivity name="Credit Inquiry"
                binaryCollaboration="Credit Inquiry"
                fromAuthorizedRole="charger"
                toAuthorizedRole="credit service"/>
            <CollaborationActivity name="Credit Payment"
                binaryCollaboration="Credit Payment"
                fromAuthorizedRole="charger"
                toAuthorizedRole="payor"/>
            <Transition fromBusinessState="Credit Inquiry"
                toBusinessState="Credit Payment"/>
        </BinaryCollaboration>
        <BinaryCollaboration name="Fulfillment Payment">
            <InitiatingRole name="payee"/>
            <RespondingRole name="payor"/>
            <BusinessTransactionActivity name="Process Payment"
                businessTransaction="Process Payment"
                fromAuthorizedRole="payee"
                toAuthorizedRole="payor"/>
        </BinaryCollaboration>
        <!-- Here are all the Business Transactions needed -->
        <BusinessTransaction name="Catalog Request">
            <RequestingBusinessActivity name="">
                <DocumentEnvelope isPositiveResponse="true"
                    businessDocument="Catalog Request"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="">
                <DocumentEnvelope isPositiveResponse="true"
                    businessDocument="Catalog"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>
        <BusinessTransaction name="Create Order">
            <RequestingBusinessActivity name=""
                isNonRepudiationRequired="true"
                timeToAcknowledgeReceipt="P2D"
                timeToAcknowledgeAcceptance="P3D">
                <DocumentEnvelope isPositiveResponse="true"
                    businessDocument="Purchase Order"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name=""
                isNonRepudiationRequired="true"
                timeToAcknowledgeReceipt="P5D">
                <DocumentEnvelope isPositiveResponse="true"
                    businessDocument="PO Acknowledgement"/>
            </RespondingBusinessActivity>
```

```
        </BusinessTransaction>
        <BusinessTransaction name="Check Credit ">
            <RequestingBusinessActivity name="">
                <DocumentEnvelope isPositiveResponse="true"
                    businessDocument="Credit Request"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="">
                <DocumentEnvelope isPositiveResponse="true"
                    businessDocument="Credit Confirm"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>
        <BusinessTransaction name="Notify of advance shipment">
            <RequestingBusinessActivity name="">
                <DocumentEnvelope isPositiveResponse="true"
                    businessDocument="ASN"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name=""
                timeToAcknowledgeReceipt="P2D"/>
        </BusinessTransaction>
        <BusinessTransaction name="Process Credit Payment">
            <RequestingBusinessActivity name="">
                <DocumentEnvelope isPositiveResponse="true"
                    businessDocument="CreditAdvice"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="">
                <DocumentEnvelope isPositiveResponse="true"
                    businessDocument="DebitAdvice"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>
        <BusinessTransaction name="Process Payment">
            <RequestingBusinessActivity name="">
                <DocumentEnvelope isPositiveResponse="true"
                    businessDocument="Invoice"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="">
                <DocumentEnvelope isPositiveResponse="true"
                    businessDocument="Payment"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>
        <BusinessTransaction name="Request Inventory Report">
            <RequestingBusinessActivity name="">
                <DocumentEnvelope isPositiveResponse="true"
                    businessDocument="Inventory Report Request"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name="">
                <DocumentEnvelope isPositiveResponse="true"
                    businessDocument="Inventory Report"/>
            </RespondingBusinessActivity>
        </BusinessTransaction>
        <BusinessTransaction name="Inventory Report">
            <RequestingBusinessActivity name="">
                <DocumentEnvelope isPositiveResponse="true"
                    businessDocument="Inventory Report"/>
            </RequestingBusinessActivity>
            <RespondingBusinessActivity name=""/>
        </BusinessTransaction>
    </Package>
</ProcessSpecification>
```

ANEXO D – Esquema XML para a troca de mensagens com o registro ebXML

No quadro a seguir é demonstrado o esquema de especificação para a troca de mensagens com o registro ebXML. O esquema está de acordo com a versão 2.0 da especificação dos serviços do registro e pode ser obtido em (ebXML, 2004).

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--$Revision: 1.24 $-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
     targetNamespace="urn:oasis:names:tc:ebxml-regrep:registry:xsd:2.1"
     xmlns:tns="urn:oasis:names:tc:ebxml-regrep:registry:xsd:2.1"
     xmlns:rim="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.1"
     xmlns:query="urn:oasis:names:tc:ebxml-regrep:query:xsd:2.1">
     <annotation>
        <documentation xml:lang="en">The schema for OASIS ebXML Registry
                 Services
        </documentation>
     </annotation>
     <!-- Import the rim.xsd file with XML schema mappaing from RIM -->
     <import namespace="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.1"
      schemaLocation="http://localhost/LifeCycleManager/Schemas/rim.xsd"/>
     <!-- Import the query.xsd file with XML schema for query related
          schema -->
     <import namespace="urn:oasis:names:tc:ebxml-regrep:query:xsd:2.1"
     schemaLocation="http://localhost/LifeCycleManager/Schemas/query.xsd"/>
     <element name="RequestAcceptedResponse">
         <annotation>
              <documentation xml:lang="en">
                    Mapping of the same named interface in ebRS.
              </documentation>
         </annotation>
         <complexType />
      </element>
      <element name="SubmitObjectsRequest">
         <annotation>
             <documentation xml:lang="en">
                 The SubmitObjectsRequest allows one to submit a list
                 of RegistryObject elements. Each RegistryEntry
                 element provides metadata for a single submitted object.
                 Note that the repository item being submitted is in a
                 separate document that is not in this DTD.
                 The ebXML Messaging Services Specfication defines
                 packaging, for submission, of the metadata of a repository
                 item with the repository item itself. The value of the id
                 attribute of the ExtrinsicObject element must be the same
                 as the xlink:href attribute within the Reference
                 element within the Manifest element of the MessageHeader.
             </documentation>
         </annotation>
         <complexType>
             <sequence>
                 <element ref="rim:LeafRegistryObjectList" />
             </sequence>
         </complexType>
```

```
      </element>
      <element name="UpdateObjectsRequest">
          <annotation>
              <documentation xml:lang="en">
                  The UpdateObjectsRequest allows one to update a list of
                  RegistryObject elements. Each RegistryEntry
                  element provides metadata for a single submitted object.
                  Note that the repository item being submitted is in a
                  separate document that is not in this DTD.
                  The ebXML Messaging Services Specfication defines
                  packaging, for submission, of the metadata of a
                  repository item with the repository item itself.
                  The value of the id attribute of the ExtrinsicObject
                  element must be the same as the xlink:href attribute
                  within the Reference element within
                  the Manifest element of the MessageHeader.
              </documentation>
          </annotation>
          <complexType>
              <sequence>
                  <element ref="rim:LeafRegistryObjectList" />
              </sequence>
          </complexType>
      </element>
      <element name="AddSlotsRequest">
          <complexType>
             <sequence>
                <element ref="rim:ObjectRef" minOccurs="1" maxOccurs="1"/>
                <element ref="rim:Slot" minOccurs="1"
                          maxOccurs="unbounded" />
             </sequence>
          </complexType>
      </element>
      <element name="ApproveObjectsRequest">
          <annotation>
              <documentation xml:lang="en">
                   The ObjectRefList is the list of
                   refs to the registry entrys being approved.
              </documentation>
          </annotation>
          <complexType>
              <sequence>
                  <element ref="rim:ObjectRefList" />
              </sequence>
          </complexType>
       </element>
       <element name="DeprecateObjectsRequest">
          <annotation>
              <documentation xml:lang="en">
                  The ObjectRefList is the list of
                  refs to the registry entrys being deprecated.
              </documentation>
          </annotation>
          <complexType>
              <sequence>
                  <element ref="rim:ObjectRefList" />
              </sequence>
          </complexType>
       </element>
       <element name="RemoveObjectsRequest">
```

```
        <annotation>
             <documentation xml:lang="en">
                  The ObjectRefList is the list of
                  refs to the registry entrys being removed
             </documentation>
        </annotation>
        <complexType>
            <sequence>
               <element ref="rim:ObjectRefList" />
            </sequence>
            <attribute name="deletionScope" use="optional">
                 <simpleType>
                     <restriction base="NMTOKEN">
                          <enumeration value="DeleteAll" />
                          <enumeration value="DeleteRepositoryItemOnly"/>
                     </restriction>
                 </simpleType>
            </attribute>
        </complexType>
    </element>
    <element name="RegistryProfile">
        <annotation>
            <documentation xml:lang="en">
                 Describes the capability profile for the registry
                  and what optional features are supported
            </documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="tns:OptionalFeaturesSupported" />
            </sequence>
            <attribute name="version" use="required" type="string" />
        </complexType>
    </element>
    <element name="OptionalFeaturesSupported">
        <complexType>
             <attribute name="sqlQuery" default="false" type="boolean" />
             <attribute name="xQuery" default="false" type="boolean" />
        </complexType>
    </element>
    <simpleType name="ErrorType">
        <restriction base="NMTOKEN">
            <enumeration value="Warning" />
            <enumeration value="Error" />
        </restriction>
    </simpleType>
    <element name="RegistryErrorList">
        <annotation>
             <documentation xml:lang="en">
                 The RegistryErrorList is derived from the ErrorList
                 element from the ebXML Message Service Specification
             </documentation>
        </annotation>
        <complexType>
            <sequence>
               <element ref="tns:RegistryError" maxOccurs="unbounded" />
            </sequence>
            <attribute name="highestSeverity" use="optional"
                type="tns:ErrorType" />
        </complexType>
```

```
      </element>
       <element name="RegistryError">
           <complexType>
               <simpleContent>
                    <extension base="string">
                         <attribute name="codeContext" use="required"
                              type="string" />
                         <attribute name="errorCode" use="required"
                              type="string" />
                         <attribute name="severity" default="Error"
                              type="tns:ErrorType" />
                         <attribute name="location" use="optional"
                              type="string" />
                    </extension>
               </simpleContent>
           </complexType>
       </element>
       <element name="RegistryResponse">
           <complexType>
               <sequence>
                   <choice minOccurs="0">
                       <element ref="query:AdhocQueryResponse" />
                       <element ref="query:GetContentResponse" />
                   </choice>
                   <element ref="tns:RegistryErrorList" minOccurs="0" />
               </sequence>
               <attribute name="status" use="required">
                   <simpleType>
                       <restriction base="NMTOKEN">
                            <enumeration value="Success" />
                            <enumeration value="Failure" />
                            <enumeration value="Unavailable" />
                       </restriction>
                   </simpleType>
               </attribute>
           </complexType>
      </element>
      <element name="RootElement">
           <annotation>
               <documentation xml:lang="en">
                     The contrived root node </documentation>
           </annotation>
           <complexType>
               <choice>
                   <element ref="tns:SubmitObjectsRequest" />
                   <element ref="tns:UpdateObjectsRequest" />
                   <element ref="tns:ApproveObjectsRequest" />
                   <element ref="tns:DeprecateObjectsRequest" />
                   <element ref="tns:RemoveObjectsRequest" />
                   <element ref="query:AdhocQueryRequest" />
                   <element ref="tns:AddSlotsRequest" />
                   <element ref="tns:RemoveSlotsRequest" />
                   <element ref="tns:RegistryResponse" />
                   <element ref="tns:RegistryProfile" />
               </choice>
           </complexType>
     </element>
</schema>
```