UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CENTRO DE INFORMÁTICA

2005.1

BIZPRO - SISTEMA DE PUBLICAÇÃO DE RELATÓRIOS FINANCEIROS EM XBRL

TRABALHO DE GRADUAÇÃO

Aluno - Ivanildo José de Sousa Aquino Júnior (ijsaj@cin.ufpe.br)

Orientadora - Valéria Cesário Times (vct@cin.ufpe.br)

Co-Orientador - Luiz Gustavo Cordeiro da Silva (lgcs@ufpe.br)

Recife, 11 de agosto de 2005.

Agradecimentos

Agradeço a todos que de uma forma ou de outra contribuíram para a realização desse trabalho em especial a:

- Minha mãe Maria Digna e meus irmãos Neto e Juliana.
- Minha namorada Tâmara pela paciência com minha ausência e apoio em todos os momentos.
- Minha orientadora Valéria Cesário Times pelo apoio, incentivo e confiança.
- Meu co-orientador Luiz Gustavo Cordeira da Silva pela ajuda no acesso às fontes de informações, contado com pesquisadores da área e incentivo.
- Milton e Mazza companheiros do projeto Ensinar pela ajuda na pesquisa.

Abstract

The fast growth of the Internet revolutionized the way that companies are delivering its financial information. Today 99% of the 500 largest companies of the world have Web Sites and 94% of these sites include financial information. The standards more used for the spreading of this information are Hypertext Markup Language (HTML) and Portable Document Format (PDF). However, the great problem inhabits in the non-standard format of this information that finishes for in such a way making it difficult its access for users as for applications. The objective of this work is to consider a system for creation, distribution and management of financial reports in XML. The development initiated with a research of the context and the existing languages for this purpose. After that we consider the architecture of BizPro system and detail the modules of management of access control polices and new object model XBRLDOM that extends model DOM of the W3C. In the module of management of access control polices, we consider the language eXtensible Business Access Control (XBAC) and describe the syntax and the semantics of its of access control polices. In model XBRLDOM we present its specification and we show how it was constructed as the extension of model DOM of the W3C.

Resumo

O rápido crescimento da Internet revolucionou a forma como as empresas estão disponibilizando suas informações financeiras. Hoje 99% das 500 maiores empresas do mundo possui *Web Sites* e 94% desses sites inclui informações financeiras. Os padrões mais utilizados para a divulgação dessas informações são *Hypertext Markup Language* (HTML) e *Portable Document Format* (PDF). No entanto, o grande problema reside no caráter não padronizado dessa informação, que acaba por dificultar seu acesso tanto por usuários como por aplicações.

O objetivo desse trabalho é propor um sistema para criação, distribuição e gerenciamento de relatórios financeiros em XML. O desenvolvimento iniciou com um estudo sobre as linguagens existentes para esse propósito. Em seguida, especificamos o sistema BizPro e detalhamos os módulos de gerenciamento de políticas de controle de acesso e então propomos um novo modelo de objeto XBRLDOM que estende o modelo DOM da W3C. No módulo de gerenciamento de políticas de controle de acesso, propomos a linguagem *eXtensible Business Access Control* (XBAC) e descrevemos a sintaxe e a semântica de suas políticas de controle de acesso. Para o modelo XBRLDOM apresentamos sua especificação e mostramos como o mesmo foi construído a partir da extensão do modelo DOM da W3C.

Sumário

1. INTRODUÇÃO 11

1.1. OBJETIVOS 11

1.2. ORGANIZAÇÃO DO TRABALHO 12

2. TRABALHOS CORRELATOS 13

2.1. CONTROLE DE ACESSO A DOCUMENTOS XML 13

2.2. MODELO DE OBJETO PARA DOCUMENTOS XML 13

2.3. LINGUAGENS BASEADAS EM XML PARA INFORMAÇÕES FINANCEIRAS 14

3. CONTEXTO 15

3.1. LINGUAGENS DE MARCAÇÃO PARA INFORMAÇÕES FINANCEIRAS 15

3.1.1. OFX 16

3.1.2. FpML 16

3.1.3. ebXML 17

3.1.4. RIXML 18

3.1.5. FIX 19

3.1.6. MDDL 19

3.1.7. MDML 20

3.1.8. XBRL 21

3.1.9. Considerações Finais 22

3.2. API'S PARA XML 22

3.2.1. Modelo DOM 23

4. XBRL 25

4.1. TAXONOMIA 26

4.1.1. Link definition 27

4.1.2. Link calculation 28

4.1.3. Link presentation 28

4.1.4. Link label 28

4.1.5. Link reference 29

4.1.6. Instância *29*

5. DESCRIÇÃO DO SISTEMA BIZPRO **31**

5.1. FLUXO DE INTERAÇÃO ENTRE OS MÓDULOS 33

5.2. DESCRIÇÃO DAS PRINCIPAIS FUNCIONALIDADES 33

6. CONTROLE DE ACESSO COM XBAC **34**

6.1. SEMÂNTICA DE XBAC 35

6.2. GRANULARIDADE 35

6.3. CORRETUDE 35

6.4. SINTAXE DA LINGUAGEM XBAC 36

6.4.1. Elemento policeLink *36*

6.4.2. Elemento police *37*

6.4.3. Elemento policeArc *38*

6.5. RECURSIVIDADE 40

6.5.1. Algoritmo de Propagação *41*

6.6. APLICADO AS COLEÇÕES E INSTÂNCIAS 42

7. DOCUMENT OBJECT MODEL PARA XBRL **43**

7.1. ESTRUTURA DO MODELO XBRLDOM 43

7.2. EXEMPLO DE APLICAÇÃO 48

8. CONCLUSÃO E TRABALHOS FUTUROS **50**

8.1. PRINCIPAIS CONTRIBUIÇÕES 50

8.2. TRABALHOS FUTUROS 50

REFERÊNCIAS BIBLIOGRÁFICAS **52**

APÊNDICE A – ESPECIFICAÇÃO DE XBRLDOM **57**

Lista de Figuras

Figura 1 – Exemplo de Solicitação no Padrão OFX 16

Figura 2 – Exemplo de FpML: Especificação de um preço inicial como um preço atual que também inclui comissões e termos FX. 17

Figura 3 – Exemplo de ebXML: Transação de Notificação 18

Figura 4 – Documento RIXML que descreve um relatório financeiro da Companhia Vale do Rio Doce 19

Figura 5 – Exemplo de ordem de compra no padrão FIXML 19

Figura 6 – Exemplo de um documento de instância MDDL com uma série temporal .. 20

Figura 7 – Exemplo de Solicitação em MDML 21

Figura 8 – Exemplo de Resposta em MDML 21

Figura 9 – Diagrama UML de DOM especificando as relações de generalização 24

Figura 10 – Construindo um documento XML 25

Figura 11 – Exemplo de definição de dois conceitos em uma taxonomia 26

Figura 12 – Exemplo de link definition do tipo generalização-especialização 27

Figura 13 – Exemplo de link calculation 28

Figura 14 – Exemplo de link presentation que estabelece relacionamento pai-filho 28

Figura 15 – Exemplo de link label 29

Figura 16 – Exemplo de link reference 29

Figura 17 – Exemplo de documento de instância XBRL 30

Figura 18 – Módulos do sistema BizPro 31

Figura 19 – Etapas de consulta a relatórios XBRL 33

Figura 20 – Definição do elemento policeLink 37

Figura 21 – Definição do Elemento police 38

Figura 22 – Definição do elemento policeArc 38

Figura 23 – Exemplo de linkbase com dois policeLink 40

Figura 24 – Tipos de ciclos de uma taxonomia 41

Figura 25 – Algoritmo de Propagação das Regras 42

Figura 26 – Diagrama UML com as relações de especialização gerais de XBRL 44

Figura 27 – Diagrama UML das relações de generalização dos elementos da Instância 45

Figura 28 – Diagrama UML com as relações de generalização dos elementos do Schema da Taxonomia 46

Figura 29 – Diagrama UML com as relações de generalização dos elementos do Linkbase 47

Figura 30 – Criação de documento de schema de uma taxonomia utilizando DOM 48

Figura 31 – Criação de documento de schema de uma taxonomia utilizando XBRLDOM 48

Lista de Quadros

Quadro 1 – Descrição dos módulos do sistema BizPro 32

Quadro 2 – Lista de funcionalidades do sistema BizPro 34

Quadro 3 – Valores padrões para atributo role 39

1. Introdução

O rápido crescimento da Internet revolucionou a forma como as empresas estão disponibilizando suas informações financeiras. Hoje 99% das 500 maiores empresas do mundo possui *Web Sites* e 94% desses sites inclui informações financeiras [11]. Os padrões mais utilizados para a divulgação dessas informações são *Hypertext Markup Language* (HTML) e *Portable Document Format* (PDF). O grande problema reside no caráter não padronizado dessas informações, que acaba por dificultar seu acesso tanto por usuários como por aplicações.

No âmbito das aplicações, destaca-se a necessidade de conversão entre diversos formatos de dados. Normalmente, os dados provenientes de relatórios financeiros são reentrados (muitas vezes manualmente) para interpretação em aplicações de computador causando prejuízos e atrasos [10]. Já no âmbito dos usuários, existe a dificuldade da utilização da informação em planilhas eletrônicas e software de análise financeira.

Para resolver esses problemas, foram propostas diversas linguagens para padronizar a forma como as informações financeiras são trocadas. Entre elas pode-se citar: *eXtensible Business Reporting Language* (XBRL) [1], *Financial Information eXchange* (FIX) [32], *Financial products Markup Language* (FpML) [24] e *Open Financial Exchange* (OFX) [23].

Dentre as várias linguagens para representação de informações financeiras, XBRL é a mais adequada à divulgação de relatórios financeiros. Isso se deve ao seu caráter não transacional, ou seja, ela não está voltada para transações financeiras e comerciais, incluindo apenas relatórios financeiros e extensivamente contemplando detalhes na representação e uso de convenções contábeis [10].

1.1. Objetivos

Esse trabalho tem como objetivo principal propor o sistema BizPro para o gerenciamento de relatórios financeiros em XBRL. Esse sistema é composto por vários módulos que estão mais bem detalhados na seção 5. Para o escopo desse trabalho, detalharemos o módulo de gerenciamento de política de controle de acesso, onde apresentaremos a sintaxe e semântica de XBAC e o modelo de objeto para XBRL que

estende o modelo DOM adicionando recursos importantes para aplicações que manipulam relatórios XBRL.

1.2. Organização do Trabalho

O trabalho está organizado em oito capítulos. Seguindo essa breve introdução, a seção 2 discorrerá sobre os trabalhos correlatos. A seção 3 descreve o contexto em que estão inseridos os relatórios financeiros na Internet, apresenta os principais padrões XML existentes para informações financeiras e introduz os modelos de objeto para XML. A seção 4 descreve a linguagem XBRL descrevendo suas características e detalhando seus principais elementos. A seção 5 apresenta a arquitetura do sistema BizPro e descreve brevemente o funcionamento do sistema. A seção 6 especifica a linguagem XBAC através de definições formais da semântica e especificação de sua sintaxe. A seção 7 apresenta a construção do modelo XBRLDOM e valida suas características através de exemplos comparativos. Finalmente, na seção 8 é apresentado a conclusão, as principais contribuições decorrentes desse estudo e os trabalhos futuros do mesmo são apresentados.

2. Trabalhos Correlatos

Essa seção revisa os principais trabalhos existentes sobre linguagens baseadas em XML para informações financeiras, as políticas de controle de acesso a documentos XML e os modelos de objeto para documentos XML.

2.1. Controle de Acesso a Documentos XML

Controle de acesso a documentos XML não é um assunto trivial, o que pode ser verificado através de sua ampla literatura [3][4][5][6][7][8][9][10][12][14][17][18][19]. O primeiro trabalho a abordar essa questão foi desenvolvido por Bertino et al. [5] que desenvolveram o sistema Author-X baseado em Java que dá suporte a políticas de controle de acesso a documentos XML em vários níveis de granularidade e credenciais. Em seguida, Akker et. al. [21] descreveram o sistema YGuard que é baseado no trabalho de Bertino. Entretanto, YGuard estende o modelo original, permitindo a aplicação de políticas de acesso a documentos heterogêneos utilizando a noção de conjunto ao invés da utilização de DTD. Então, diversos outros trabalhos se seguiram a esses. Chung-Hwan et al. [4] propuseram um modelo de controle de acesso que considera não apenas a leitura como a maior parte da literatura, mas aborda também as operações de atualização. Fundulaki et al. [3] propuseram uma método formal de representar a semântica de uma política de controle de acesso XML utilizando XPath, servindo de base para as formalizações que serão apresentadas nesse trabalho.

2.2. Modelo de Objeto para Documentos XML

Diversas extensões foram propostas para o modelo DOM da W3C. Murray-Rust et. al. [34] desenvolveram a extensão CMLDOM para representar documentos XML no padrão *Chemical Markup Language* (CML). Chen. et. al. [35] desenvolveram outra extensão de DOM para operações "set-at-a-time". Outras linguagens como *Scalable Vector Graphics* (SVG) [36], *Mathematical Markup Language* (MathML) [35] e *Synchronized Multimedia Integration Language* (SMIL) [42] também especificam modelos de objeto dos documentos baseados em DOM.

Os trabalhos desenvolvidos para a extensão das linguagens citadas anteriormente constituiu uma base importante para a especificação do modelo XBRLDOM proposto nesse trabalho.

2.3. Linguagens Baseadas em XML para Informações Financeiras

A construção desse trabalho envolveu a realização de uma pesquisa sobre diversas linguagens de marcação para informações financeiras [1][23][24][25][29][32][33][39][42][45] com o objetivo de identificar a mais adequada para a divulgação de relatórios. Silva [10] analisou em sua dissertação, a implantação de XBRL para o ambiente do Banco Central do Brasil e apresentou um estudo comparativo entre algumas linguagens. Boritz. et. al. [7] propuseram uma arquitetura segura para divulgação de relatórios XBRL baseada em assinatura digital e *Web Service*.

3. Contexto

Um relatório financeiro constitui um conjunto de informações referentes aos recursos e obrigações de uma entidade aos seus acionistas e investidores que devem ser elaborados em intervalos regulares. No caso do Brasil, as sociedades anônimas (SA) devem apresentar suas demonstrações financeiras seguindo as diretrizes da Lei 6.404/76 (Lei das S.A) e também as normas brasileiras de contabilidade, reiteradas nas instruções normativas da Comissão de Valores Mobiliários (CVM) [49].

3.1. Linguagens de Marcação para Informações Financeiras

O principal objetivo da construção de relatórios financeiros é divulgar informações úteis aos usuários tomadores de decisão. Provendo informações financeiras em *Websites*, as empresas ajudam os usuários a obter mais facilmente essas informações. Entretanto, não existem padrões largamente aceitos para descrever informações financeiras. Logo, usuários que obtém relatórios postados em *Websites* precisam reentrar manualmente as informações em planilhas e aplicações, o que é claramente ineficiente e induz a um grande número de erros [42].

Um dos aspectos chave para uma solução ideal de intercâmbio de informações é a utilização de uma linguagem comum. Isso vem sendo resolvido através da adoção de XML como linguagem padrão. XML tem sido largamente aceita e vem provendo os fundamentos para o desenvolvimento de sistemas de computação com interoperabilidade.

Como o próprio nome sugere, XML (*eXtensible Markup Language*) [13] é uma linguagem de marcação extensível derivada de SGML [57]. Originalmente desenvolvida para a publicação eletrônica em larga escala, ela ultimamente vem sendo bastante utilizada como um padrão para a troca de informações eletrônicas. Entretanto, XML não é uma solução completa, pois é uma linguagem universal. Portanto, é necessário definir padrões e restrições apropriadas. Nesse cenário, diversas linguagens de marcação foram propostas para troca de informações financeiras, tais como OFX [23], SWIFT [25] e FpML [24]. A seguir abordaremos várias dessas linguagens, destacando suas principais características e domínio em que estão inseridas.

3.1.1. OFX

Open Financial Exchange (OFX) é uma especificação para a troca eletrônica de dados financeiros entre instituições financeiras, negócios e consumidores através de Internet. Criada pelo consórcio formado por CheckFree, Intuit e Microsoft em meados de 1997, OFX dá suporte a uma grande variedade de atividades financeiras incluindo atividades bancárias, pagamentos, transferência de fundos, entre outras. OFX é um padrão com arquitetura cliente-servidor que sincroniza a troca de informações através de solicitações de *request/response* (solicitação/resposta) [23]. A Figura 1 contém um exemplo de uma solicitação para *Internet Banking* executada pelo software Microsoft Money [37] no padrão OFX.

```
<RequestStatement>
<BankAccount>
    <BankID>888</BankID>
    <AccountID>9394</AccountID>
    <AccountType>CHECKING</AccountType>
</BankAccount>
</RequestStatement>
```

Figura 1 – Exemplo de Solicitação no Padrão OFX

3.1.2. FpML

Financial Product Markup Language (FpML) [24] é um padrão aberto baseado em XML para troca de informações financeiras no âmbito dos derivativos. Esse padrão estabelece um novo protocolo para troca de informações em negociação de *swaps*, derivativos e produtos estruturados. Seu objetivo é automatizar o fluxo de informações entre todos os parceiros de derivativos e a rede de clientes. Entre os principais processos automatizados por FpML destacam-se:

- Negociação e estruturação da transação;
- Análise de risco e;
- Execução e confirmação de uma transação.

A Figura 2 mostra um exemplo de um documento com informações sobre operações SWAP no padrão FpML.

```
<initialPrice>
  <commission>
    <commissionDenomination>BPS</commissionDenomination>
    <commissionAmount>5</commissionAmount>
  </commission>
  <grossPrice>
    <currency>CAD</currency>
    <amount>113228777</amount>
    <priceExpression>AbsoluteTerms</priceExpression>
  </grossPrice>
  <netPrice>
    <currency>USD</currency>
    <amount>84089141</amount>
    <priceExpression>AbsoluteTerms</priceExpression>
  </netPrice>
  <fxConversion>
    <fxRate>
      <quotedCurrencyPair>
        <currency1>CAD</currency1>
        <currency2>USD</currency2>
        <quoteBasis>Currency2PerCurrency1</quoteBasis>
      </quotedCurrencyPair>
      <rate>1.34665</rate>
    </fxRate>
  </fxConversion>
</initialPrice>
```

Figura 2 – Exemplo de FpML: Especificação de um preço inicial como um preço atual que também inclui comissões e termos FX.

3.1.3. ebXML

Eletronic Business using XML (ebXML) é um conjunto de especificações que habilitam empresas a conduzirem transações financeiras na Internet. Usando ebXML empresas podem trocar mensagens financeiras, conduzir negociações e trocar informações [33] de forma padronizada. Desenvolvida em meados de 1999 como uma iniciativa do grupo OASIS e da agência CEFACT das Nações Unidades, ebXML foi originalmente projetada para entregar cinco camadas de especificação de informações, incluindo padrões XML para:

- Processos de negócio;
- Componentes para núcleos de informação;
- Mensagens;
- Registro e repositórios e;
- Acordos de protocolo de colaboração.

Uma transação financeira é a unidade atômica de trabalho em uma negociação entre dois parceiros e consiste de uma atividade de solicitação, uma atividade de resposta e um ou dois fluxos de documentos entre eles. Uma transação pode ter suporte em um ou mais sinais financeiros que governam o uso e o significado de confirmação em uma transação [39]. O exemplo da Figura 3 ilustra uma transação de notificação que possui dois fluxos de documentos e três sinais de negócio. Nesse exemplo, o solicitante requer a confirmação tanto de recebimento quanto de aceitação. A resposta requer apenas a confirmação de recebimento.

```
<BusinessTransaction name="Create Order">
      <RequestingBusinessActivity name=""
                  isNonRepudiationRequired="true"
                  timeToAcknowledgeReceipt=”P2D"
                  timeToAcknowledgeAcceptance=”P3D">
            <DocumentEnvelopeBusinessDocument="Purchase Order"/>
      </RequestingBusinessActivity>
      <RespondingBusinessActivity name=""
                  isNonRepudiationRequired="true"
                  timeToAcknowledgeReceipt=”P5D">
            <DocumentEnvelope isPositiveResponse="true"
                  BusinessDocument="PO Acknowledgement"/>
            </DocumentEnvelope>
      </RespondingBusinessActivity>
</BusinessTransaction>
```

Figura 3 – Exemplo de ebXML: Transação de Notificação

3.1.4. RIXML

Todo fundo de investimento recebe milhares de informações que podem ser importantes para tomadores de decisão quando estão analisando quando devem comprar e quando devem vender. Estas informações vêm de diversas fontes, tais como bancos, analistas financeiros e outros. Normalmente essas informações são enviadas nos formatos Word, Excel e pdf [40]. O principal objetivo da especificação de *Research Information Exchange Markup Language* (RIXML) [41] é definir um padrão de metadados para descrever essas informações financeiras para que analistas e investidores busquem, filtrem e ordenem pesquisas publicadas com o objetivo de promover informações relevantes para os tomadores de decisão. A Figura 4 descreve metadados do relatório financeiro contido no arquivo “relatorio.pdf” utilizando o padrão RIXML.

```
<Content>
   <Title>Vale do Rio Doce e o Mercado Internacional de
Ferro</title>
```

```
   <Abstract>Apesar de dificuldades do mercado internaciona, os
resultados apresentados pela Companhia Vale do Rio Doce vem se
mostrando satisfatórias e contribuindo para o crescimento de
participação no mercado.
   </Abstract>
   <Synopsis>Esse documento apresenta o relatório anual da Vale do
Rio Doce</Synopsis>
   <Resource resourceid="relatorio.pdf"
             Language="pt_BR"
             SizeInBytes="562340"
             PrimaryIndicator="Yes">
      <Length lengthunit="Pages">30</Length>
      <MIMEType>application/pdf</MIMEType>
      <Name>relatorio.pdf</Name>
   </Resource>
</Content>
```

Figura 4 – Documento RIXML que descreve um relatório financeiro da Companhia Vale do Rio Doce

3.1.5. FIX

Financial Information eXchange (FIX) [32] é uma especificação técnica, aberta e gratuita usada para comunicação eletrônica de mensagens de negociação. Mais precisamente, o protocolo FIX é uma série de especificações de mensagens desenvolvidas para colaboração de bancos, corretoras, investidores e outros. A Figura 5 mostra uma ordem de compra para ações da IBM e especifica a quantidade e outras informações necessárias para uma transação desse tipo.

```
<FIXML xmlns="http://www.fixprotocol.org/FIXML-4-4"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.fixprotocol.org/FIXML-4-4
../../schema/fixml-main-4-4.xsd" v="4.4" r="20030618" s="20040109">
      <Order ID="123456" Side="2" TxnTm="2001-09-11T09:30:47-05:00"
Typ="2" Px="93.25" Acct="26522154">
            <Hdr Snt="2001-09-11T09:30:47-05:00" PosDup="N"
PosRsnd="N" SeqNum="521" SID="AFUNDMGR" TID="ABROKER"/>
            <Instrmt Sym="IBM" ID="459200101" Src="1"/>
            <OrdQty Qty="1000"/>
      </Order>
</FIXML>
```

Figura 5 – Exemplo de ordem de compra no padrão FIXML

3.1.6. MDDL

Market Data Definition Language (MDDL) [56] é uma especificação XML desenvolvida com o objetivo habilitar o intercâmbio de dados necessários para

contadores, auditores e investidores analisarem dados do mercado financeiro. MDDL procura, a partir da definição de termos comuns, provê um vocabulário para que dados do mercado possam ser trocados sem ambigüidades. A Figura 6 contém um documento de instância no padrão MDDL. Nesse exemplo, é apresentada uma série temporal composta por dois eventos referentes à última negociação realizada.

```
<mddl version="1.0-final"
      xmlns="http://www.mddl.org/mddl/2001/1.0-final">
  <header>
    <dataDateTime>2001-11-02T16:47:04Z</dataDateTime>
    <source>Your Data Provider</source>
  </header>
  <timeseries>
    <instrumentIdentifier>
      <name>Some Hong Kong Company</name>
      <code
        scheme="http://www.ypd.net/XML/scheme/ydpSymbols.xml">
        <mdString>HKSE-XC2</mdString>
        <nameRef>../../name</nameRef>
      </code>
    </instrumentIdentifier>
    <exchangeIdentifier>
      <code
        scheme=http://www.ypd.net/XML/scheme/ydpExchanges.xml
        >HKSE</code>
    </exchangeIdentifier>
    <currency>HKD</currency>
    <dataDateTime>2001-11-01</dataDateTime>
    <event>
      <equityDomain><commonClass><trade><last>
        <mdDecimal>188.50</mdDecimal>
        <dataDateTime>2001-11-01T16:20:12Z</dataDateTime>
      </last></trade></commonClass></equityDomain>
    </event>
    <event>
      <equityDomain><commonClass><trade><last>
        <mdDecimal>188.32</mdDecimal>
        <dataDateTime>2001-11-01T16:20:18Z</dataDateTime>
      </last></trade></commonClass></equityDomain>
    </event>
  </timeseries>
</mddl>
```

Figura 6 – Exemplo de um documento de instância MDDL com uma série temporal

3.1.7. MDML

Market Data Markup Language (MDML) [39] é uma implementação de XML utilizada para distribuir informações financeiras. O principal objetivo dessa implementação é ser flexível suficiente para transportar diversos tipos de informações financeiras sobre o mercado incluindo dados sobre câmbio, bancos e corretoras. A Figura 7 contém um exemplo de solicitação (*request*) seguindo a especificação de MDML e a Figura 8

contém a resposta (*response*) para esta solicitação. As principais características de MDML são:

- Uso genérico de elementos XML;
- Múltiplas consultas em uma única requisição e;
- Especificação define um conjunto de propriedades principais.

```
<BusinessTransaction name="Create Order">
      <RequestingBusinessActivity name=""
                  isNonRepudiationRequired="true"
                  timeToAcknowledgeReceipt=”P2D"
                  timeToAcknowledgeAcceptance=”P3D">
            <DocumentEnvelopeBusinessDocument="Purchase Order"/>
      </RequestingBusinessActivity>
      <RespondingBusinessActivity name=""
isNonRepudiationRequired="true"
            timeToAcknowledgeReceipt=”P5D">
            <DocumentEnvelope isPositiveResponse="true"
                  BusinessDocument="PO Acknowledgement"/>
            </DocumentEnvelope>
      </RespondingBusinessActivity>
</BusinessTransaction>
```

Figura 7 – Exemplo de Solicitação em MDML

```
<?xml version="1.0" ?>
<mdml:mdml xmlns:mdml="mdml">
      <mdml:vehicle id="msft" source="bridge">
            <mdml:property name="bid"? 1/8</mdml:property>
            <mdml:property name="ask"? 3/16</mdml:property>
            <mdml:property name="last"? 1/8</mdml:property>
            <mdml:property name="open"? 7/16</mdml:property>
            <mdml:property name="change">-3 9/16</mdml:property>
            <mdml:property name="volume"?</mdml:property>
      </mdml:vehicle>
</mdml:mdml>
```

Figura 8 – Exemplo de Resposta em MDML

3.1.8. XBRL

A linguagem *eXtensible Business Reporting Language* (XBRL) [1] foi desenvolvida para padronizar a divulgação de relatórios financeiros no formato XML. Dentre as linguagens apresentadas, XBRL é a que melhor se adapta à divulgação de relatórios financeiros. Isso se deve ao seu caráter não transacional, ou seja, não está voltada para transações financeiras e comerciais, incluindo apenas relatórios financeiros e

contemplando extensivamente detalhes na representação e uso de convenções contábeis [10]. A seção 4 detalhe em mais profundidade a linguagem XBRL.

3.1.9. Aceitação de Padrões pelo Mercado

Algumas outras linguagens foram encontradas na literatura incluindo FundsXML [45], *Interact Financial eXchange* (IFX) [55], MarketsML e *Tax Markup Language* (TaxML) [56], no entanto elas são menos populares.

Segundo Park [40], para que um padrão desenvolvido para a indústria tenha sucesso é necessário que ele tenha as seguintes características:

- Deve ser desenvolvido por consórcios de grandes empresas do mercado;
- Deve haver uma clara necessidade do mercado por um padrão;
- Deve ser claro como o gerenciamento de versões é feito e;
- Deve haver disponibilidade de software compatível com o padrão disponível no mercado.

Em algumas situações, diferentes padrões XML são desenvolvidos para resolver o mesmo tipo de problema. RIXML por exemplo, atua no mesmo campo que o IRML. No final das contas, o mercado acaba decidindo qual padrão será aceito [40].

3.2. API para XML

Além de definir a sintaxe das informações trocadas, é importante padronizar a forma como as aplicações manipulam essas informações. O modelo de objeto de um documento define uma interface que permite que programas acessem dinamicamente o conteúdo e a estrutura de um documento. Duas API são comumente utilizadas para processar documentos XML: Simple API for XML (SAX) [47] e Document Object Model (DOM) [30]. Um *parser* SAX provê uma interface simples, tal que informações sobre a hierarquia de um documento não são passadas para uma aplicação. Por outro lado, a interface DOM representa um documento XML como uma estrutura abstrata de árvore consistindo de objetos. Como resultado, árvores DOM podem acessar as informações em qualquer ordem de forma mais eficiente.

3.2.1. Modelo DOM

O modelo DOM nível 1 [26] especificado pelo W3C especifica um conjunto mínimo de objetos e interfaces. Ele define a estrutura lógica de documentos e o modo como um documento é manipulado. Os principais benefícios do modelo DOM são: (1) a árvore completa está sempre disponível para análise em qualquer momento, (2) pode ser processada navegando pela mesma.

3.2.1.1. Estrutura do modelo DOM

A especificação do DOM [30] proposta pelo W3C define um total de 12 interfaces representando os tipos de nós de um documento XML. DOM também especifica as interfaces *NodeList* para gerenciar listas ordenadas de nós e *NamedNodeMap* para gerenciar conjuntos não ordenados de nós referenciados pelo nome de seus atributos. *NodeLists* e *NamedNodeMap* no DOM são considerados "*live*", ou seja, as alterações na estrutura do documento afetam automaticamente esses objetos. O diagrama UML da Figura 9 mostra como são os relacionamentos de generalização do modelo DOM.

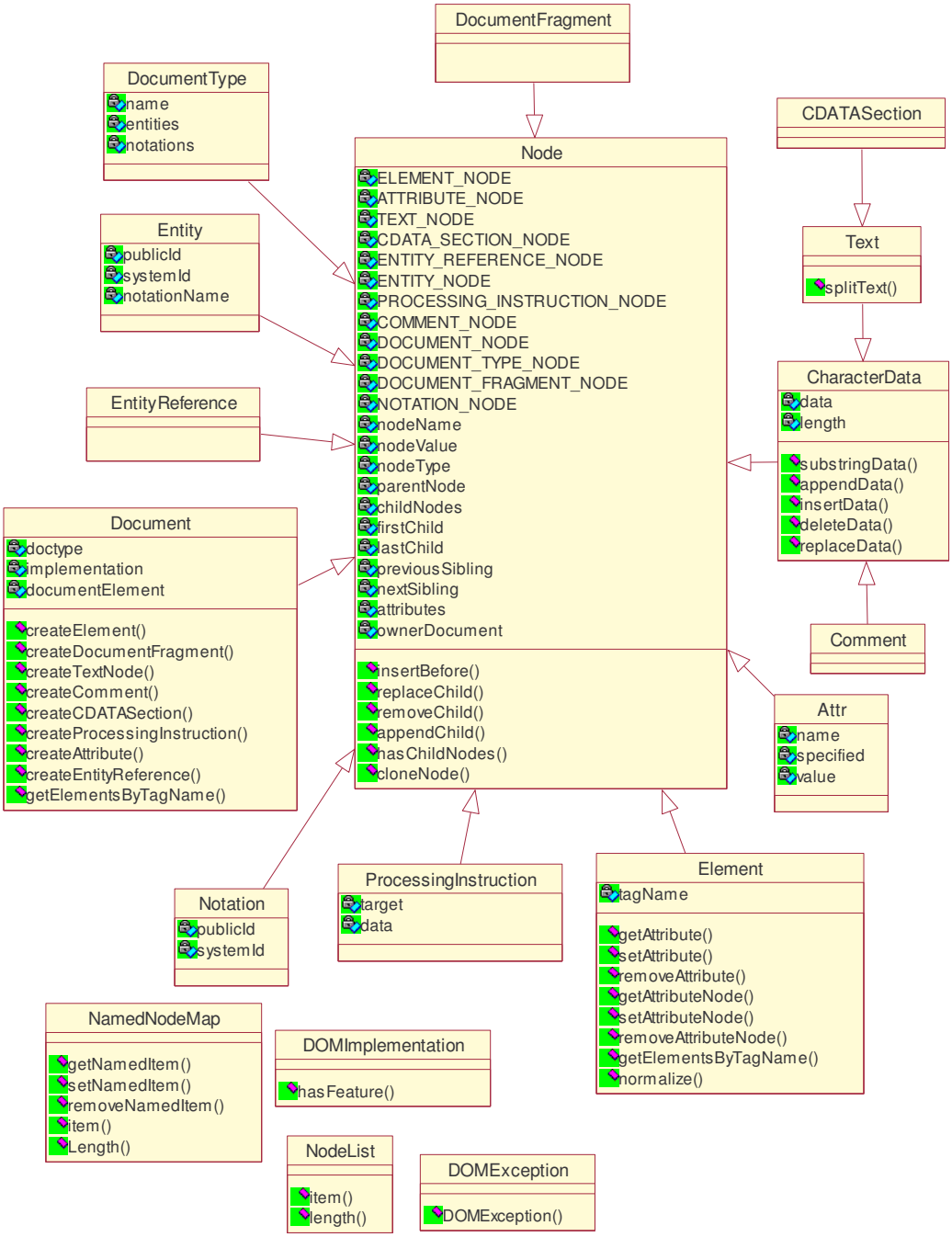

Figura 9 – Diagrama UML de DOM especificando as relações de generalização

3.2.1.2. Construção de Documentos DOM

Um documento DOM pode ser construído de duas formas. A primeira forma utiliza um documento XML pré-existente. Esse documento é lido e a partir dele construído o documento. Outra forma de construir um documento DOM é criar um documento vazio e adicionar nós utilizando o método *appendChild(Node)*. A Figura 10 exemplifica a

construção do documento da Figura 11 utilizando os objetos que implementam a API DOM.

```
//cria um novo documento
Document taxonomia = new DocumentImpl();
//criamos o elemento schema obrigatório
Element schema = new ElementImpl("schema", taxonomia);
schema.setAttribute("xmlns", "http://www.w3.org/2001/XMLSchema");
schema.setAttribute("xmlns:xbrli",
http://www.xbrl.org/2003/instance);
Element elemento1 = new ElementImpl("element", taxonomia);
elemento1.setAttribute("id", "br_ativo");
elemento1.setAttribute("name", "ativo");
elemento1.setAttribute("xbrli:periodType", "instant");
elemento1.setAttribute("type", "xbrli:monetaryItemType");
elemento1.setAttribute("substitutionGroup", "xbrli:item");
elemento1.setAttribute("nillable", "true");
schema.appendChild(elemento1);
Element elemento2 = new ElementImpl("element", taxonomia);
elemento2.setAttribute("id", "br_policitaCompensacao");
elemento2.setAttribute("name", "policitaCompensacao");
elemento2.setAttribute("xbrli:periodType", "duration");
elemento2.setAttribute("type", "xbrli:stringItemType");
elemento2.setAttribute("substitutionGroup", "xbrli:item");
elemento2.setAttribute("nillable", "true");
schema.appendChild(elemento2);
```

Figura 10 – Construindo um documento XML

4. XBRL

Em abril de 1998, Charles Hoffman iniciou suas pesquisas sobre a utilização de XML em relatórios financeiros eletrônicos. No mesmo ano, ele sugeriu à *American Institute of Certified Public Accountants* (AICPA) que os seus estudos deveriam se basear em XML. A AICPA formou um grupo para desenvolver o protótipo de um conjunto de fatos financeiros. Em janeiro de 1999, esse protótipo foi apresentado e seus resultados foram considerados satisfatórios. Em seguida, a AICPA formou uma força-tarefa para desenvolver o uso de XML em relatórios financeiros. O resultado desenvolvido por essa força-tarefa foi originalmente chamado de *eXtensible Financial Reporting Markup Language* (XFRML) que posteriormente foi renomeado para XBRL.

A linguagem XBRL é um padrão aberto e gratuito que vem sendo desenvolvida por aproximadamente 250 empresas, organizações e agências governamentais [20]. Baseada em XML, ela foi concebida para criação, intercâmbio e análise de demonstrações financeiras na Internet. Como tal, permite que investidores individuais e profissionais

do mercado financeiro analisem e extraiam informações em suas aplicações, simplificando uma das fases chaves da análise financeira [42].

XBRL define a sintaxe utilizada para reportar o valor de um fato financeiro baseado em um conjunto de conceitos bem definidos dentro de um contexto particular. XBRL divide a informação do relatório financeiro em dois componentes distintos: instância e taxonomia. A Instância contém os fatos reportados enquanto a taxonomia define os conceitos comunicados pelos fatos. A combinação de uma instância XBRL, o Schema de sua taxonomia e o conjunto de *linkbases* associados, constituem um relatório financeiro XBRL. A seguir descreveremos cada um desses componentes.

4.1. Taxonomia

Uma taxonomia é definida por um XML *Schema* e um conjunto de *links* estendidos diretamente referenciados ou qualquer link estendido que esteja dentro do XML *Schema* [1]. A taxonomia define os conceitos relacionados a um relatório financeiro. A terminologia de XBRL descreve um conceito como a definição de um fato reportado. Conceitos são criados através da definição de um elemento de XML *Schema*. No *Schema* de uma taxonomia, um conceito recebe um nome concreto e um tipo. A Figura 11 mostra um *schema* de uma taxonomia definindo dois elementos e especificando seus tipos.

```
<schema
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xbrli="http://www.xbrl.org/2003/instance">
<element
  id="br_ativo"
  name="ativo"
  xbrli:periodType="duration"
    type="xbrli:monetaryItemType"
    substitutionGroup="xbrli:item" nillable="true"/>
<element
  id="br_policitaCompensacao"
  name="policitaCompensacao"
  xbrli:periodType="duration"
    type="xbrli:stringItemType"
    substitutionGroup="xbrli:item" nillable="true"/>
</schema>
```

Figura 11 – Exemplo de definição de dois conceitos em uma taxonomia

Outro importante componente de uma taxonomia são os *links* estendidos. Os *links* estendidos documentam o significado dos conceitos, expressando o relacionamento

entre eles e relacionando conceitos com sua documentação. Relações entre fragmentos XML ocorrem de diversas formas em XBRL. Existem relações entre uma instância e sua Taxonomia. Existem relações entre fatos e as notas de rodapé que descrevem esses fatos. A semântica dos conceitos é expressa através de uma rede de relações que constituem os *linkbases*. XBRL expressa todos esses relacionamentos usando as sintaxes de ligações simples e ligações estendidas definidas na especificação de XLink [27]. Existem cinco tipos de ligações estendidas utilizadas em uma taxonomia: *definition*, *calculation*, *presentation*, *label* e *reference*. Os três primeiros tipos expressam relacionamento entre conceitos e os dois últimos, expressam relacionamento entre conceitos e suas documentações.

4.1.1. Link definition

Os *links definitions* provêem quatro tipos de relacionamentos entre conceitos de uma taxonomia. O relacionamento "*general-special*" conecta um conceito generalista a um conceito especialista. Apenas os ciclos não direcionais da Figura 24 são permitidos nesse tipo de relacionamento. O segundo tipo de relacionamento denominado "*essence-alias*" que estabelece um arco entre um conceito essencial e o seu apelido. Apenas ciclos não direcionais são permitidos nesse tipo de relacionamento. No tipo "*similar-tuples*", tuplas que possuem definições semelhantes são relacionadas, mesmo quando elas possuem modelos de conteúdo XML diferentes. Por ser um tipo de relacionamento simétrico, qualquer tipo de ciclo é coerente. Finalmente, o relacionamento "*requires-element*" indica que a ocorrência de um conceito implica na presença obrigatória de outro. Para esse relacionamento todos os tipos de ciclos são permitidos. A Figura 12 mostra um exemplo de arco em um *link definition* que estabelece uma relacionamento de generalização-especialização entre os conceitos "codigoPostal" e "CEP" (codigoPostal é uma generalização de CEP). O atributo "*order*" define a ordem de apresentação desse link para o usuário.

```
<definitionArc
    xlink:type="arc"
    xlink:from="codigoPostal"
    xlink:to="CEP"
    xlink:arcrole="http://www.xbrl.org/2003/arcrole/general-special"
    order="1"/>
```

Figura 12 – Exemplo de link definition do tipo generalização-especialização

4.1.2. Link calculation

O *link calculation* é utilizado para estabelecer uma relação somatória entre conceitos, ou seja, especifica como o valor de um conceito contribui para a definição do valor de outro. A Figura 13 mostra um arco de um *link calculation* que estabelece uma relação somatória na qual o conceito "ativoCirculante" contribui para o valor de "ativo" com peso 1.0.

```
<calculationArc
    xlink:type="arc"
    xlink:arcrole="http://www.xbrl.org/2003/arcrole/summation-item"
    xlink:from="ativo"
    xlink:to="ativoCirculante"
    weight="1.0"
    order="1"/>
```

Figura 13 – Exemplo de link calculation

4.1.3. Link presentation

O *link presentation* define a hierarquia e a ordem de apresentação dos conceitos em uma taxonomia. A Figura 14 mostra um arco de um *link presentation* que estabelece uma relação do tipo pai-filho no qual o conceito "passivoCirculante" é filho do conceito "passivo".

```
<presentationArc
    xlink:type="arc"
    xlink:from="passivo"
    xlink:to="passivoCirculante"
    xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
    order="3"/>
```

Figura 14 – Exemplo de link presentation que estabelece relacionamento pai-filho

4.1.4. Link label

O *link label* contém o relacionamento entre conceitos e documentos textuais e rótulos desses conceitos. Em outras palavras, os *link labels* são utilizados para prover documentação explicativa e rótulos legíveis para o leitor. A Figura 15 mostra um exemplo de *link label* no qual o conceito "ativoCirculante" recebe um rótulo legível "Ativo Circulante". O atributo xml:lang especifica o idioma do rótulo.

```
<label
    xlink:type="resource"
    xlink:role=http://www.xbrl.org/2003/role/label
    xlink:label="ativoCirculante"
    xml:lang="pt_BR">Ativo Circulante</label>
```

Figura 15 – Exemplo de link label

4.1.5. Link reference

O *link reference* é utilizado para estabelecer um relacionamento entre conceitos e referências normativas em publicações de negócios, literatura financeira e contábil, dando significado ao conceito. No exemplo da Figura 16 [1], duas referências são adicionadas ao conceito "s_customerSales".

```
<referenceLink xlink:type="extended"
    xlink:role="http://www.xbrl.org/2003/role/link">
      <loc xlink:type="locator"
            xlink:href="samp001.xsd#s_customerSales"
            xlink:label="s_customerSales"/>
      <referenceArc xlink:type="arc" xlink:from="s_customerName"
            xlink:to="s_customerName_REF"
xlink:arcrole="http://www.xbrl.org/2003/arcrole/concept-reference"/>
      <reference xlink:type="resource"
          xlink:label="s_salesBycustomer_REF"
          xlink:role="http://www.xbrl.org/2003/role/definitionRef">
        <ref:name>Handbook of Business Reporting</ref:name>
        <ref:pages>5</ref:pages>
      </reference>
      <reference
        xlink:type="resource" xlink:label="s_salesBycustomer_REF"
        xlink:role="http://www.xbrl.org/2003/role/measurementRef">
        <ref:name>Handbook of Business Reporting</ref:name>
        <ref:pages>45-50</ref:pages>
      </reference>
</referenceLink>
```

Figura 16 – Exemplo de link reference

4.1.6. Instância

Os conceitos definidos na taxonomia não contêm os valores atuais dos fatos financeiros. Os valores dos fatos são informados no documento de instância XBRL. A terminologia XBRL descreve esses fatos financeiros simplesmente como "fatos".

A estrutura de um documento de instância XBRL é definida no documento XBRL *Instance Schema* que além de definir seus atributos e tipos de dados, especifica os elementos: (1) *item* – representa um único fato ou medida financeira, (2) *tuple* - agrupa fatos que não podem ser interpretados individualmente, (3) *context* - usado para definir

o contexto em que o relatório financeiro se encontra, (4) xbrl – corresponde ao elemento raiz da instância XBRL do documento financeiro.

Um documento de instância XBRL pode ter suporte de mais de uma taxonomia. Assim como as taxonomias também podem ser interconectadas, estendendo e modificando umas às outras de várias formas. Geralmente, é necessário considerar múltiplas taxonomias relacionadas quando se interpreta uma instância XBRL. O conjunto de taxonomias relacionadas a uma instância é chamado de *Discoverable Taxonomy Set* (DTS). A Figura 17 mostra um documento de instância que contém os valores para quatro fatos financeiros.

```
<xbrl xmlns="http://www.xbrl.org/2003/instance"
      xmlns:xlink="http://www.xbrl.org/2001/XLink"
      xmlns:link="http://www.xbrl.org/2003/linkbase"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:br="http://www.exemplo.com.br">
      <br:ativo precision="3" unitRef="u1"
            contextRef="c1">6784</br:ativo>
      <br:ativoCirculante precision="3" unitRef="u1"
            contextRef="c1">5684</br:ativoCirculante>
      <br:passivo precision="3" unitRef="u1"
            contextRef="c1">635</br:passivo>
      <br:passivoCirculante precision="3" unitRef="u1"
            contextRef="c1">235</br:passivoCirculante >
      <context id="c1"><!-- ... --></context>
      <unit id="u1"><!-- ... --></unit>
</xbrl>
```

Figura 17 – Exemplo de documento de instância XBRL

5. Descrição do Sistema BizPro

O sistema BizPro foi concebido numa arquitetura cliente/servidor onde a comunicação é realizada através de *Web Service*. A Figura 18 destaca a arquitetura do sistema BizPro e os principais módulos que a constituem.

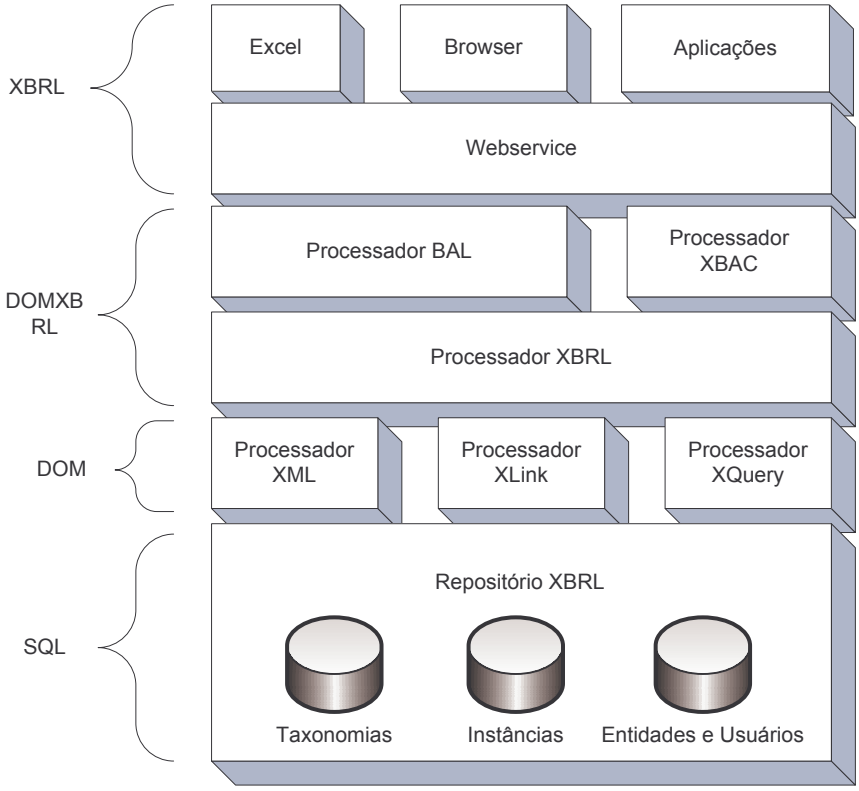

Figura 18 – Módulos do sistema BizPro

Dos módulos descritos na arquitetura do sistema, vários possuem versões comerciais e acadêmicas disponíveis o que propicia um bom nível de reuso. O Quadro 1 descreve os principais módulos do sistema e apresenta soluções existentes no mercado ou no meio acadêmico.

Quadro 1 – Descrição dos módulos do sistema BizPro

Módulos	Descrição	Soluções Existentes
Web Service	Esse módulo é a interface de comunicação entre as aplicações clientes e o sistema BizPro e utiliza o padrão SOAP [50] para implementar o Web Services.	
Processador BAL	Módulo responsável por processar e interpretar os comandos de análise escritos na linguagem BAL.	
Processador XBAC	Módulo responsável por gerenciar as políticas de controle de acesso aos relatórios financeiros no sistema.	
Processador XBRL	Módulo responsável por processar os relatórios financeiros habilitando as demais aplicações a manipular os relatórios através de uma API.	Interstage XWand [51]
Processador XML	Módulo responsável por processar os documentos XML fornecendo uma API para a manipulação dos mesmos e execuções de tarefas freqüentes.	Existe uma infinidade de processadores XML tanto comerciais como acadêmicos como: Microsoft MSXML
Processador XQuery	Módulo responsável pela execução dos comandos Xquery ao nível de *middleware*.	NUX [52]
Processador XLink	Módulo responsável por processar os *links* simples e estendidos presentes nos relatórios.	XLiP [53]
Repositório XBRL	Responsável pelo armazenamento e recuperação dos relatórios XBRL e registro dos usuários e empresas em banco de dados relacional.	Altova Xquery [54]

Para o escopo deste trabalho, detalharemos o processador XBAC, o repositório XBRL e especificaremos o modelo XBRL DOM utilizado pelo Processador XBRL.

5.1. Fluxo de Interação Entre os Módulos

A Figura 19 descreve as etapas realizadas na execução de um comando BAL no sistema BizPro. Na etapa 0 o usuário acessa uma determinada aplicação (Excel, *Browser* ou Ferramenta de Auditoria) e descreve a análise que deseja realizar. Em seguida, na etapa 1, a aplicação assina o documento BAL solicitada pelo usuário e envia para o sistema BizPro através do Web Service. No Processador BAL o comando é interpretado e os relatórios contidos no comando são solicitados ao Processador XBAC na etapa 3. O processador XBAC, por sua vez, solicita os relatório ao repositório na etapa 4 e gera os sub-relatórios a partir da análise das credenciais do usuário e verificando seu nível de acesso aos relatórios. O Processador BAL então efetua a análise dos sub-relatórios e envia os resultados na etapa 8. Os resultados são então assinados e enviados para a aplicação do usuário.

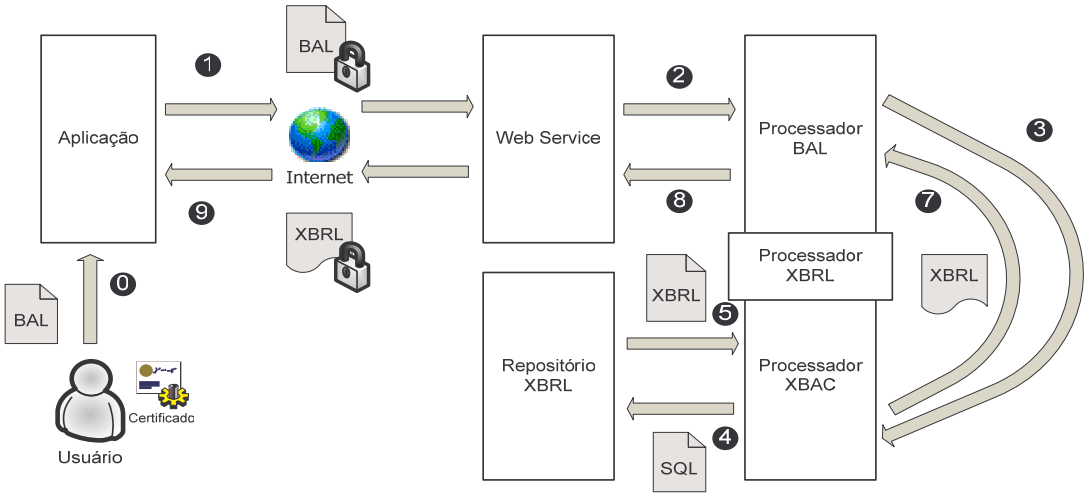

Figura 19 – Etapas de consulta a relatórios XBRL

5.2. Descrição das Principais Funcionalidades

A Quadro 2 descreve as principais funcionalidades idealizadas para o sistema BizPro. Como dito anteriormente, para o escopo desse trabalho apenas algumas dessas funcionalidades foram contempladas.

Quadro 2 – Lista de funcionalidades do sistema BizPro

Funcionalidade	Descrição
Histórico de atualizações.	Manter o histórico de alterações nos documentos para possibilitar futuras auditorias.
Executar comandos de análise financeira.	Criar uma linguagem para análise financeira onde os comandos possam ser aplicados a conjuntos de
Gerenciar controle de acesso.	Permitir que o estabelecimento de visões diferente de um mesmo relatório financeiro, permitindo que as necessidades de diferentes usuários sejam satisfeitas. Exemplos de usuários incluem: auditor interno, investidor e diretor.
Permitir cadastro, consulta e alteração de relatórios XBRL.	O sistema deve permitir cadastro, consulta e alteração dos relatórios financeiros.
Validar documentos XBRL.	Todos os relatórios XBRL armazenados devem ser validados de acordo com seu Schema e com as regras definidas pela especificação da linguagem XBRL versão 2.1.
Multi-empresas	Permitir que várias empresas utilizem o sistema de forma independente.
Garantir que as respostas sejam confiáveis	Além de utilizar conexões seguras, é necessário implementar mecanismos de assinatura digital e sistema de detecção de mudanças baseado em DOMHASH.

6. Controle de Acesso com XBAC

A maioria das políticas de controle de acesso a documentos XML utiliza o nó XML como a menor unidade de proteção e consideram que a propagação de uma política é realizada sobre a hierarquia do documento. Entretanto, no âmbito dos relatórios financeiros XBRL, a hierarquia das informações é definida em um documento diferente daquele que contém os dados financeiros. Logo, a utilização dos modelos tradicionais mostra-se dispendioso e complexo. Para resolver esses problemas, propomos um novo modelo de política de controle de acesso denominado XBAC (*eXtensible Business Access Control*). As principais características e propriedades semânticas e sintáticas de XBAC serão detalhadas a seguir.

6.1. Semântica de XBAC

Uma política de controle de acesso P a um documento XBRL é especificada por P = (U, A, C, T, R) na qual: (1) U é um usuário ou perfil de usuário, (2) A é a ação que pode ser ler, atualizar, excluir e criar, (3) C é o conceito ao qual a política é aplicada, (4) T é o tipo da política que se classifica em permissão ou negação e (5) R define se a política é recursiva, ou seja, se também pode ser aplicada aos conceitos da rede de relacionamentos de C.

6.2. Granularidade

Na maioria dos trabalhos propostos [1][2][6], a menor unidade de proteção é o nó XML e XPath 1.0 é utilizada para identificar os nós impactados por uma regra de acesso. Em nossa abordagem, a menor unidade é o subdocumento e XBAC é utilizada para identificar os conceitos XBRL. A grande vantagem da utilização de sub-relatórios é a construção mais natural e concisa das políticas.

6.3. Corretude

A fim de garantir que toda requisição XBAC possua uma resposta válida, é necessário definir um valor padrão quando um conceito não possuir nenhuma política e definir um critério para a resolução de conflitos.

A maioria das abordagens [3][5][6] considera existentes na literatura que se não há nenhuma regra de controle de acesso a um determinado nó, então o nó é considerado não acessível. Em nossa política de controle de acesso, utilizaremos esta mesma idéia, pois a mesma contribui para a qualidade de menor privilégio, descrita na seção anterior.

XBAC permite que em alguns casos, um determinado conceito possua uma política de acesso de negação e permissão para um mesmo usuário. Nesse caso, a abordagem escolhida por XBAC é de que a *negação prevalece*, ou seja, uma política de negação tem prioridade sobre uma política de permissão. A seguir, apresentaremos uma definição formal para essa escolha.

Definição 3:

Dado P, uma política de acesso identificada por P = (U, A, C, T, R), a união dessa política com uma segunda P' = (U', A, C', T', R) resultará numa terceira política T'' = (U'', A, C'', T'', R) onde:

$U'' = U \cap U'$ e $C'' = C \cup C'$ se $T = T'$ e

$U'' = U \cap U'$ e $C'' = C \cap C'$ se $T \neq T'$

A definição formal das operações de união e interseção no âmbito de subdocumento está definida em [2].

6.4. Sintaxe da Linguagem XBAC

Os *links estendidos* em uma taxonomia, provêem informação adicional sobre os conceitos, expressando relacionamentos entre eles ou associando-os com suas documentações. Cada um dos elementos *policeLink* de uma política de acesso deve estar inserido em um contêiner XLink. O container XLink deve ser um elemento linkbase que pode estar localizado no caminho "schema/annotation/appinfo/*" no *schema* da taxonomia ou deve ser o elemento raiz de um documento separado. Para a definição de uma política XBAC são utilizados os elementos *policeLink*, *police* e *policeArc*.

6.4.1. Elemento policeLink

O elemento *policeLink* é um link estendido utilizado para definir as políticas de controle de acesso a relatórios XBRL. Sua definição é apresentada na Figura 20 e foi criada a partir de restrições aos *links* estendidos.

```
<schema targetNamespace="http://www.xbrl.org/2003/linkbase"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:link="http://www.xbrl.org/2003/linkbase"
  xmlns:xl="http://www.xbrl.org/2003/XLink"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  elementFormDefault="qualified">

  <element name="policeLink" substitutionGroup="xl:extended">
    <annotation>
      <documentation>
      Definição do link estendido police
      </documentation>
    </annotation>
    <complexType>
      <complexContent>
        <restriction base="xl:extendedType">
          <choice minOccurs="0" maxOccurs="unbounded">
            <element ref="xl:title"/>
            <element ref="link:documentation"/>
            <element ref="link:loc"/>
            <element ref="link:policeArc"/>
            <element ref="link:police"/>
          </choice>
          <anyAttribute
namespace="http://www.w3.org/XML/1998/namespace"
processContents="lax" />
        </restriction>
      </complexContent>
    </complexType>
  </element>

</schema>
```

Figura 20 – Definição do elemento policeLink

6.4.2. Elemento police

O elemento *police* define o perfil ao qual a política é aplicada. Sua definição é apresentada na Figura 21 e foi criada a partir de restrições ao elemento *resource*.

```
<schema targetNamespace="http://www.ensinar.org/xbrl/2005/xbac"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:link="http://www.xbrl.org/2003/linkbase"
  xmlns:xl="http://www.xbrl.org/2003/XLink"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  elementFormDefault="qualified">
  <element name="police" substitutionGroup="xl:resource">
    <annotation>
      <documentation>
      Definição de um elemento police
      </documentation>
    </annotation>
    <complexType mixed="true">
      <complexContent mixed="true">
        <extension base="xl:resourceType">
          <sequence>
            <any namespace="http://www.w3.org/1999/xhtml"
              processContents="skip" minOccurs="0"
maxOccurs="unbounded"/>
          </sequence>
          <anyAttribute
namespace="http://www.w3.org/XML/1998/namespace"
            processContents="lax" />
        </extension>
      </complexContent>
    </complexType>
  </element>
</schema>
```

Figura 21 – Definição do Elemento police

6.4.3. Elemento policeArc

O arco *policeArc* estabelece relacionamento entre o conceito especificado em um elemento *locator* e um elemento *police*.

```
<schema targetNamespace="http://www.ensinar.org/xbrl/2005/xbac "
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:link="http://www.xbrl.org/2003/linkbase"
  xmlns:xl="http://www.xbrl.org/2003/XLink"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  elementFormDefault="qualified">

  <element name="policeArc" type="xl:arcType"
substitutionGroup="xl:arc">
    <annotation>
      <documentation>
      Arco usado em links estendidos police
      </documentation>
    </annotation>
  </element>

</schema>
```

Figura 22 – Definição do elemento policeArc

O atributo *arcrole* descreve o tipo de política de acesso. A Quadro 3 contém os quatro possíveis valores para o atributo role e suas respectivas descrições. O atributo *label* identifica o grupo de usuários.

Quadro 3 – Valores padrões para atributo role

Valor do Atributo	Descrição
http://www.ensinar.org/xbrl/2005/role/positive_local	Política que garante acesso apenas ao conceito informado.
http://www.ensinar.org/xbrl/2005/role/positive_recursive	Política que garante acesso ao conceito selecionado e a todos os seus filhos.
http://www.ensinar.org/xbrl/2005/role/negative_local	Política que nega acesso apenas ao conceito informado.
http://www.ensinar.org/xbrl/2005/role/negative_recursive	Política que nega acesso ao conceito selecionado e a todos os seus filhos.

A Figura 23 ilustra a utilização do elemento *police*. Nesse exemplo é estabelecida uma política de acesso para o perfil "Auditor Interno" onde é garantido o acesso recursivo aos conceitos "br_ativo" e "br_passivo".

```
<linkbase
  xmlns="http://www.ensinar.org/xbrl/2005/xbac"
  xmlns:samp="http://www.ensinar.org/xbrl/exemplo"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ensinar.org/xbrl/exemplo
exemplo.xsd"
  xml:base="http://www.ensinar.org/xbrl/exemplo">
      <policeLink xlink:type="extended"
                  xlink:role="http://www.xbrl.org/2003/role/link">
            <loc xlink:type="locator"
                  xlink:href="exemplo.xsd#br_ativo"
                  xlink:label="br_ativo"/>
            <loc xlink:type="locator"
                  xlink:href="exemplo.xsd#br_passivo"
                  xlink:label="br_passivo"/>
            <policeArc
                  xlink:type="arc"xlink:from="br_ativo"
                  xlink:to="auditor_interno"
                  xlink:arcrole="http://www.ensinar.org/xbrl/2005
                        /role/positive_local"/>
            <policeArc
                  xlink:type="arc" xlink:from="br_passivo"
                  xlink:to="auditor_interno"
                  xlink:arcrole="http://www.ensinar.org/xbrl/2005
                        /role/positive_local"/>
            <police xlink:type="resource"
                  xlink:role=" http://www.ensinar.org/xbrl
                        /2005/arcrole/concept-police"
                  xlink:label="auditor_interno">Auditor Interno
            </police>
</linkbase>
```

Figura 23 – Exemplo de linkbase com dois policeLink

6.5. Recursividade

A possibilidade de criar políticas de controle de acesso recursivas é um recurso bastante desejável, pois permite a construção de regras de acesso mais sucintas. Ao contrário do relacionamento hierárquico típico de um documento XML, uma taxonomia XBRL possui um conjunto de *links* estendidos que definem uma complexa rede de relacionamentos entre os conceitos. A Figura 24 mostra algumas das possíveis formações de redes de relacionamentos de uma taxonomia XBRL.

Em uma taxonomia, os arcos dos *links* estendidos *definition*, *calculation* e *presentation* organizam os conceitos XBRL em redes de relacionamentos. Os arcos dos *links* estendidos *label* e *reference* organizam redes de relacionamentos entre conceitos e sua documentação. Para o escopo de XBAC consideraremos apenas os três primeiros *links* estendidos.

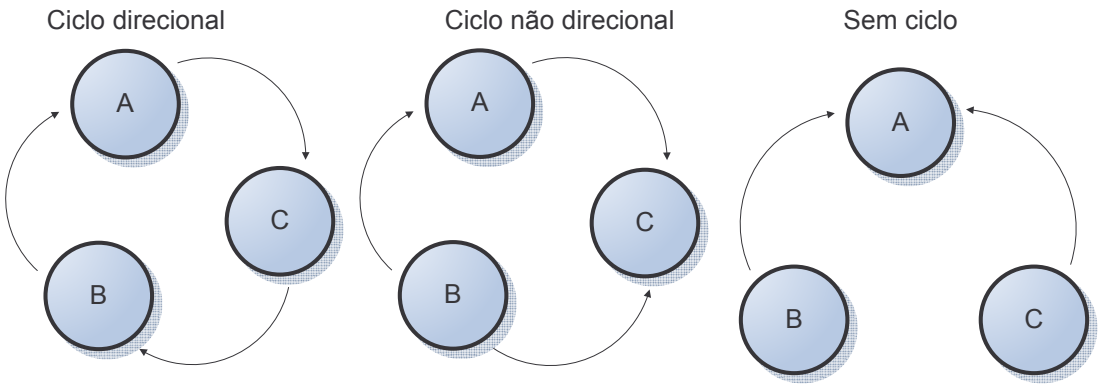

Figura 24 – Tipos de ciclos de uma taxonomia

6.5.1. Algoritmo de Propagação

O algoritmo de propagação de uma política em XBAC mostrado na Figura 25 é constituído de duas fases principais. Na primeira fase, todos os conceitos que possuem uma política de permissão são adicionados a uma lista. Em seguida, para cada conceito da lista que possui permissão recursiva, navega-se pelos arcos que são originados deles, percorrendo assim, toda a rede de relacionamentos desses conceitos. Todos os conceitos encontrados nas redes são adicionados à lista de permissão. Na segunda fase, o processo é semelhante ao ocorrido na primeira fase. Todos os conceitos que possuem política de negação que estiverem na lista de permissão são removidos. Em seguida, para cada conceito que foi removido, navega-se pelos arcos que são originados dele e retira-se da lista todos os conceitos navegados que estiverem nela. Essa ordem se deve à adoção da estratégia de prevalecimento da negação, descrita na seção 5.2.

```
ProcessarPermissoes(IList regras) {
      foreach (Regra regra in regras) {
            if (regra.Tipo == permissao) {
                  if (regra.Recursiva) {
                        IList conceitos =
GetConceitosRede(regra.Conceito);
                        AdicionarConceitos(conceitos);
                  } else {
                        AdicionarConceitos(regra.Conceito);
                  }
            }
      }
      foreach (Regra regra in regras) {
            if (regra.Tipo == negacao) {
                  if (regra.Recursiva) {
                        IList conceitos =
GetConceitosRede(regra.Conceito);
                        RemoverConceitos(conceitos);
                  } else {
                        RemoverConceitos(regra.Conceito);
                  }
            }
      }
}
```

Figura 25 – Algoritmo de Propagação das Regras

6.6. Aplicado as Coleções e Instâncias

Um dos requisitos de uma política de controle de acesso é a possibilidade de aplicar regras de controle de acesso a documentos específicos. Entretanto, esse requisito torna sua implantação muito trabalhosa e complicada. Por isso, XBAC utiliza uma abordagem que realiza o controle de acesso tanto no nível de coleções quanto ao nível de instâncias.

No caso da especificação de uma regra de acesso a uma instância, a solução é simples e consiste apenas em inserir o *linkbase* contendo as políticas de segurança diretamente no documento de instância XBRL. No caso da especificação de uma regra de acesso baseada em um conjunto de relatórios, sua implementação é mais complexa e se baseia na relação contida entre um documento de instância XBRL e sua Taxonomia. Nesse caso, consideramos que a regra aplica-se ao conjunto de instâncias que implementam uma determinada taxonomia.

7. Document Object Model para XBRL

Nessa seção, apresentamos uma especificação para o modelo de objetos de XBRL que estende a Core API do DOM nível 1 para descrever objetos e métodos específicos para elemento XBRL. Com o objetivo de provê uma API que pode ser utilizada em uma grande variedade de ambientes e aplicações e também com o objetivo de manter a padrão utilizado pelo DOM, escolhemos definir a especificação utilizando OMG IDL [28]. A especificação completa do XBRLDOM encontra-se no Apêndice A.

A especificação do DOM [30] proposta pelo W3C define um total de 12 interfaces representando os tipos de nós de um documento XML. No entanto essas interfaces não possuem as funcionalidades específicas para XBRL o que torna sua aplicação em documentos XBRL dispendiosa e ineficiente. Por isso, desenvolvemos uma extensão do W3C DOM com o objetivo de construir classes especializadas para XBRL. Para a definição do XBRLDOM, especializamos os tipos de nó *Document*, *Element*, *Text* e *Attribute* além da interface *NodeList* e *DocumentImplementation*. Os principais objetivos da construção do XBRLDOM foram:

- Especializar e adicionar funcionalidades para elementos específicos de XBRL e;
- Adicionar formas convenientes de executar tarefas freqüentes em elementos XBRL.

Para uma aplicação identificar se está utilizando um modelo DOM ou a especialização XBRLDOM ela deve utilizar o método *hasFeature* da interface *DOMImplementation* e informar o parâmetro “org.ensinar.dom.xbrl”.

7.1. Estrutura do Modelo XBRLDOM

As principais especializações propostas por XBRLDOM foram:

- As interfaces *XBRLInstanceDocument*, *XBRLTaxonomySchemaDocument* e *XBRLLinkbaseDocument* especializaram a interface *Document* e possuem todos os métodos necessário para criar os elementos do modelo XBRLDOM, seguindo o padrão de projeto de projeto *Factory* [43];

- Interfaces *XBRLItemFactElement* e *XBRLTupleFactElement* que especializaram a interface *XBRLFactElement* para representar os fatos financeiros de um documento de instância. Como as informações contidas nesses elementos são as mais acessadas, foram adicionados métodos específicos para localizar e processar objetos que implementem essa interface e;
- Interfaces *XBRLCalculationElement*, *XBRLDefinitionElement*, *XBRLPresentationElement*, *XBRLReferenceElement* e *XBRLLabelElemet* que especializaram a interface *XBRLExtendedLinkElement* para representar a os links que foram a rede de relacionamento dos conceitos de uma interface. A criação dessas interfaces possibilita a navegação pela rede de forma mais eficiente, além de possibilitar o tratamento das muitas regras de XBRL aplicadas aos links.

A Figura 26 mostra um diagrama UML com as relações de generalização dos elementos distintos do modelo XBRLDOM.

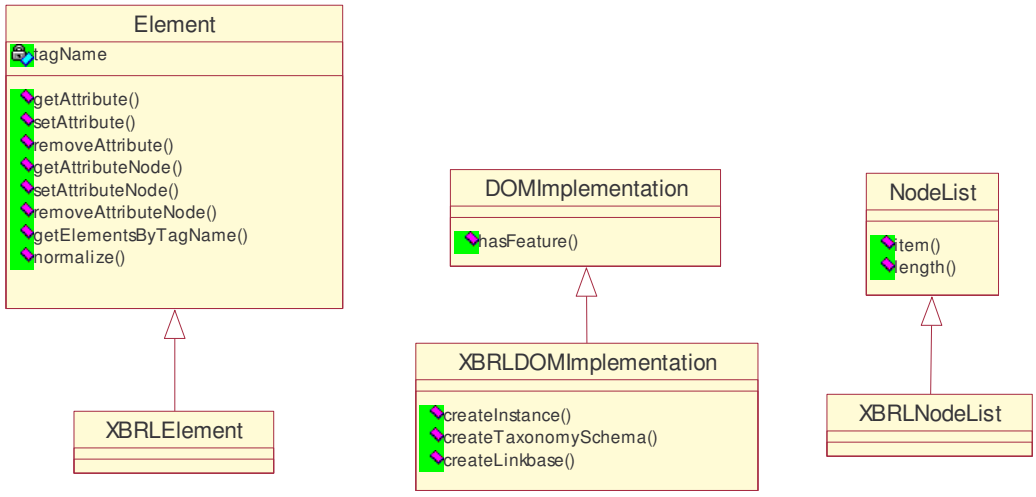

Figura 26 – Diagrama UML com as relações de especialização gerais de XBRL

A Figura 27 mostra o diagrama UML com as relações de generalização das interfaces de XBRLDOM utilizadas em documentos de instância.

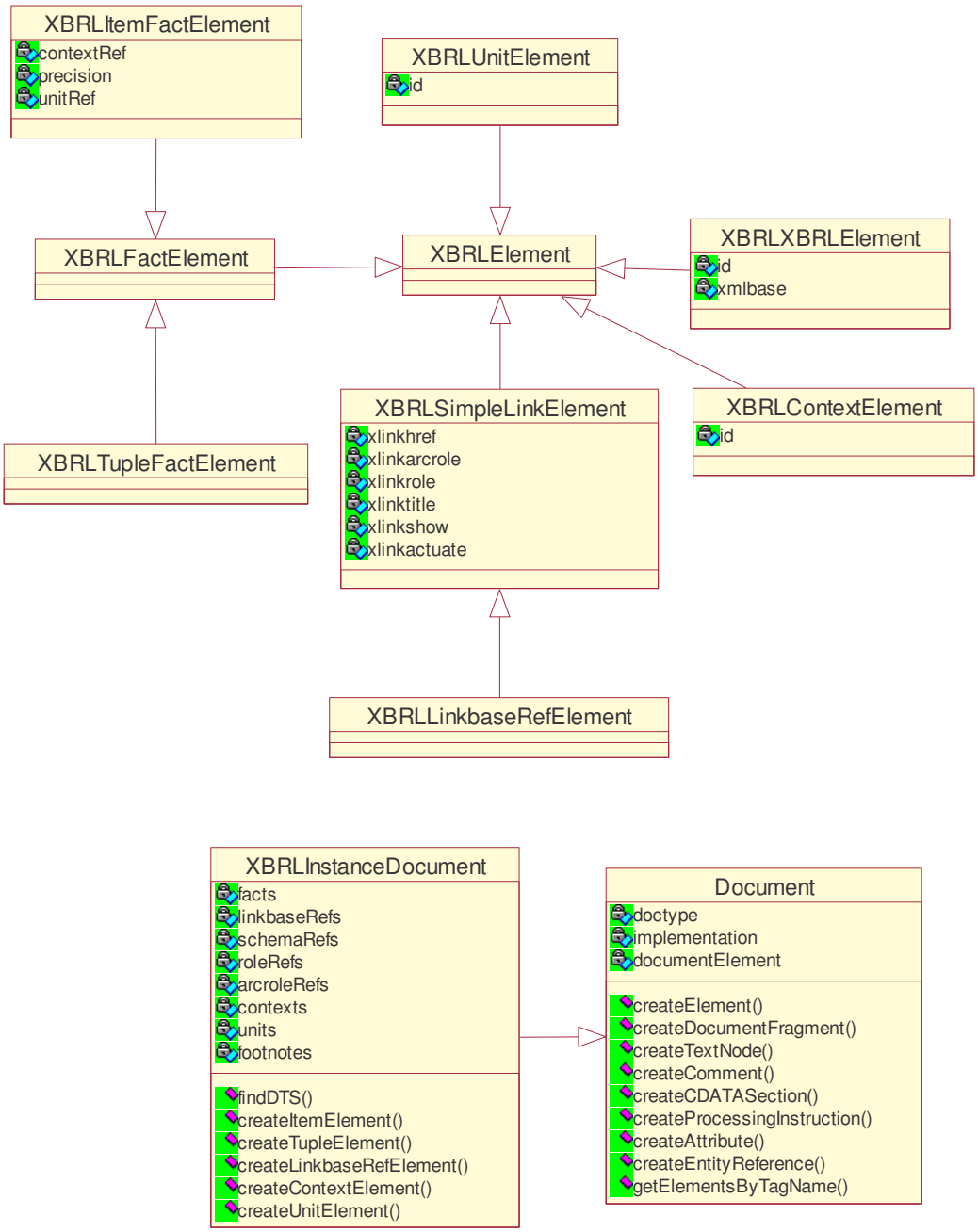

Figura 27 – Diagrama UML das relações de generalização dos elementos da Instância

A Figura 28 mostra o diagrama UML com as relações de generalização das interfaces de XBRLDOM utilizadas em documentos de taxonomia.

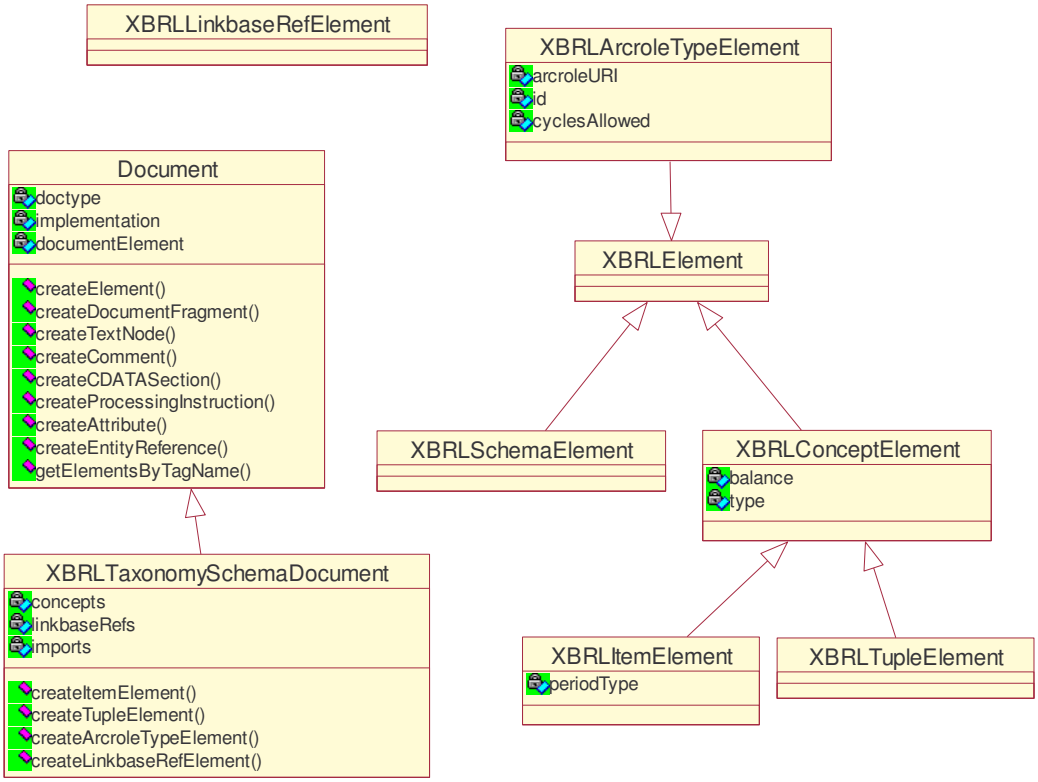

Figura 28 – Diagrama UML com as relações de generalização dos elementos do Schema da Taxonomia

A Figura 29 mostra o diagrama UML com as relações de generalização das interfaces de XBRLDOM utilizadas em documentos *linkbase*.

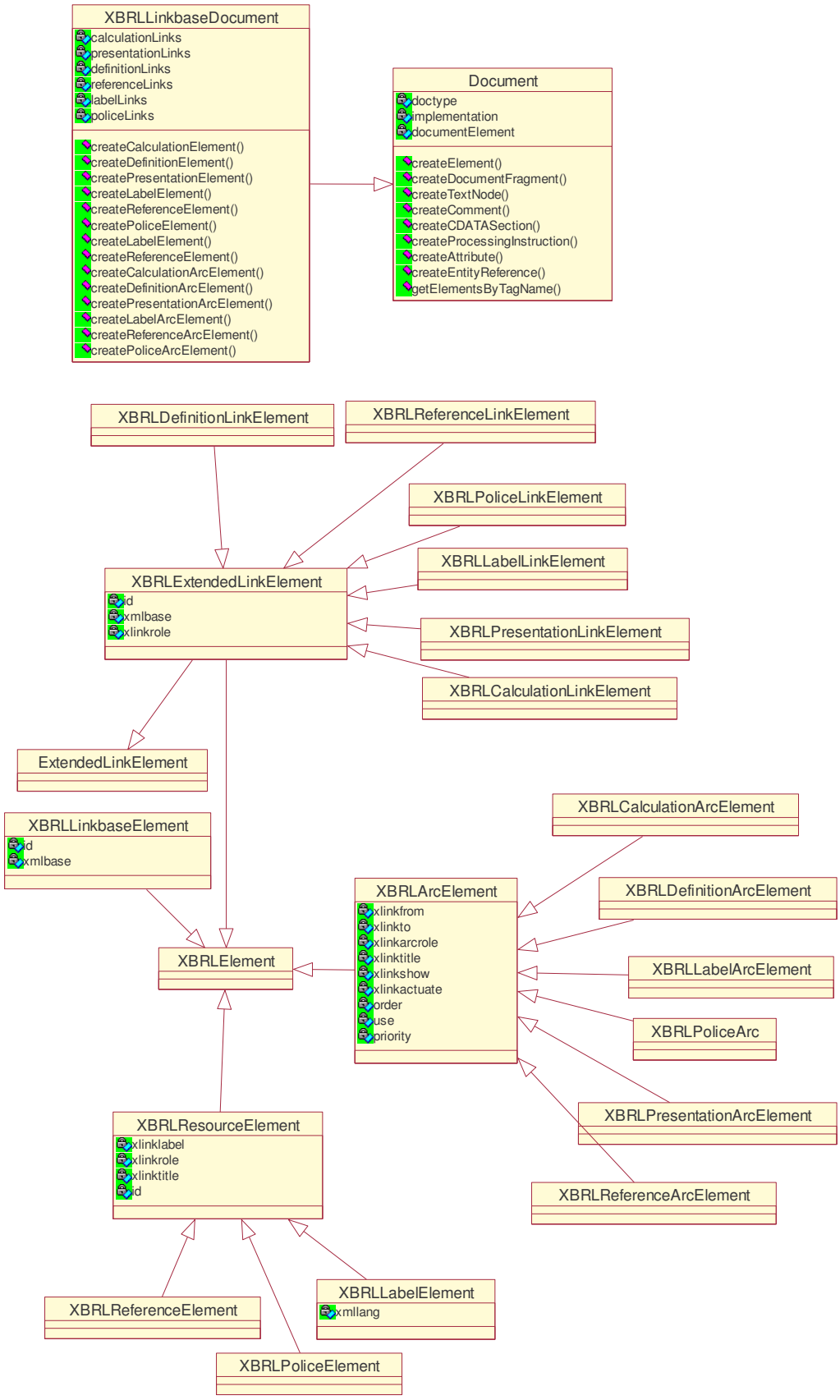

Figura 29 – Diagrama UML com as relações de generalização dos elementos do Linkbase

7.2. Exemplo de Aplicação

A fim de demonstrar a vantagem da utilização de XBRLDOM sobre o W3CDOM, considere o documento de *schema* da taxonomia exibida na Figura 11. A seguir, mostraremos a construção desse documento utilizando o DOM e em seguida, faremos o mesmo procedimento para o mesmo documento só que usando XBRLDOM. A Figura 30 mostra como o *schema* foi construído utilizando o DOM e a Figura 31 mostra como o mesmo *schema* foi construído utilizando XBRLDOM. Esse exemplo deixa claro que a utilização de XBRLDOM além de diminuir o número de linhas de código, também mantém o código mais legível e consistente.

```
Document taxonomia = new DocumentImpl();
//criamos o elemento schema obrigatório
Element schema = new ElementImpl("schema", taxonomia);
schema.setAttribute("xmlns", "http://www.w3.org/2001/XMLSchema");
schema.setAttribute("xmlns:xbrli",
http://www.xbrl.org/2003/instance);
Element elemento1 = new ElementImpl("element", taxonomia);
elemento1.setAttribute("id", "br_ativo");
elemento1.setAttribute("name", "ativo");
elemento1.setAttribute("xbrli:periodType", "instant");
elemento1.setAttribute("type", "xbrli:monetaryItemType");
elemento1.setAttribute("substitutionGroup", "xbrli:item");
elemento1.setAttribute("nillable", "true");
schema.appendChild(elemento1);
Element elemento2 = new ElementImpl("element", taxonomia);
elemento2.setAttribute("id", "br_policitaCompensacao");
elemento2.setAttribute("name", "policitaCompensacao");
elemento2.setAttribute("xbrli:periodType", "duration");
elemento2.setAttribute("type", "xbrli:stringItemType");
elemento2.setAttribute("substitutionGroup", "xbrli:item");
elemento2.setAttribute("nillable", "true");
schema.appendChild(elemento2);
```

Figura 30 – Criação de documento de schema de uma taxonomia utilizando DOM

```
XBRLTaxonomySchema tax = new XBRLTaxonomySchemaImpl();
XBRLItemFactElement item1 = new XBRLItemFactElmentImpl("br_ativo",
"ativo");
item1.periodType = item.INSTANT;
item1.type = "xbrli:monetaryItemType";
item1.nillable = true;
tax.appendChild(item1);
XBRLItemFactElement item2 = new
XBRLItemFactElmentImpl("br_policitaCompensacao","policitaCompensacao"
);
item1.periodType = item.DURATION;
item1.type = "xbrli:stringItemType";
item1.nillable = true;
tax.appendChild(item2);
```

Figura 31 – Criação de documento de schema de uma taxonomia utilizando XBRLDOM

Outro exemplo no qual a utilização de XBRLDOM se mostra ainda mais vantajosa diz respeito a operações de busca e análise de um documento, pois a especificação de XBRL define métodos úteis que simplificam essas atividades. Exemplos dessas operações incluem: busca de fatos de um documento de instância, busca de conceitos de um *schema* de taxonomia, busca do conceito que está relacionado a um fato, entre muitos outros.

8. Conclusão e Trabalhos Futuros

Nesse trabalho, apresentamos o sistema BizPro que possibilita a criação, gerenciamento e publicação de relatórios financeiros na Internet. Nesse contexto foram detalhados: a arquitetura do sistema, o modelo da política de controle de acesso XBAC e a especificação do modelo XBRL DOM.

Para o módulo de controle de acesso, propomos um novo modelo de política de controle de acesso o qual é mais adequado às necessidades dos sistemas de publicação de relatórios financeiros em XBRL por ser mais conciso e considerar as relações semânticas de um relatório. Nós formalizamos a definição de subdocumentos que é a base para política de controle de acesso XBAC.

O resultado obtido nesse trabalho se mostrou satisfatório, superando as expectativas inicias. As especificações de XBAC e XBRLDOM proposta nesse trabalho serão submetidas à XBRL Internacional para que sejam incluídas em futuras versões da especificação de XBRL.

8.1. Principais Contribuições

As principais contribuições desse trabalho são: (1) formalização do conceito de sub-relatório XBRL baseado na definição de um subdocumento, (2) um novo modelo de política de controle de acesso a relatórios financeiros em XBRL baseado em sub-relatórios e com propagação em redes e (3) especificação do modelo XBRLDOM que estende o modelo DOM adicionando interfaces e métodos úteis para o tratamento de relatórios XBRL.

8.2. Trabalhos Futuros

Para dar continuidade a este trabalho podem ser realizados os seguintes estudos:

- Assinatura de consultas para garantir que mesmo em servidores não confiáveis as consultas a relatórios financeiros retornem informações com características de não-repúdio e autenticidade;

- XBRL estabelece regras que não são tratadas por processadores XML genéricos. A especificação de XBRLDOM apresentada nesse trabalho, apesar de possuir um bom nível de detalhamento, não está completa, pois é necessário incluir as exceções para tratamento de violação dessas regras. Outras alterações necessárias só podem ser identificadas como a experiência de utilização do modelo;
- Definição e implementação da linguagem BAL (*Business Analysis Language*) para realizar análise financeira em relatórios financeiros XBRL e;
- Definição de uma interface gráfica para construção de comando BAL.

Referências Bibliográficas

[1] Engel, P. and Hamscher, W. and Shuetrim, G. and Kannon, D. and Wallis, H. "Extensible Business Reporting Language (XBRL) 2.1.", 2003.

[2] A. Sahuguet, B. Alexe, "Sub-Document Queries Over XML with XSQuirrel", *Proceedings of the 14th international conference on World*, ACM Press, Chiba, Japão, 2005, pp 268 – 277.

[3] I. Fundulaki, M. Marx, "Specifying Access Control Policies for XML Documents with XPath", *9th ACM Symposium on Access Control Models and Technologies*, ACM Press, Yorktown Heights, New York, USA, 2004, pp. 61 – 69.

[4] C. Lim, S. Park, S. H. Son, "Access Control of XML Documents Considering Update Operations", *Proceedings of the 2003 ACM workshop on XML security*, Fairfax, Virginia, 2003, pp. 49 – 59.

[5] E. Bertino, S. Castano, E. Ferrari, "On Specifying Security Policies for Web Documents with an XML-based Language", Proceedings of the sixth ACM symposium on Access control models and technologies, ACM Press, Chantilly, Virginia, USA, 2001, pp. 57 – 65.

[6] M. Murata, A. Tozawa, M. Kudo, "XML Access Control Using Static Analysis", Proceedings of the 10th ACM conference on Computer and communications security, ACM Press, Washington D.C., USA, 2003, pp. 73 – 84.

[7] J. E. Boritz, W. G. No, "Security in XML-based financial reporting services on the Internet", *Journal of Accounting and Public Policy*, 2005.

[9] T. Akker, Q. O. Snell, M. J. Clement, "The YGuard Access Control Model: Set-Based Access Control", Proceedings of the sixth ACM symposium on Access control, ACM Press, Chantilly, Virginia, USA 2001, pp. 75 – 84.

[10] P. C. Silva, "Explorando Linguagens de Marcação para Representação de Relatórios de Informações Financeiras", 2002.

[11] Financial Accounting Standard Board, "Business Reporting Research Project: Electronic Distribution of Business Information Norwalk", 2001

[12] G. Gottlob, C. Koch, R. Pichler. “The Complexity of XPath Query Evaluation”, *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, ACM Press, San Diego, California, 2003, pp. 179 – 190.

[13] Business Processor (BizPro) – http://www.ensinar.org/bizpro, acessado em 2005.

[14] V. Parmar, H. Shi, S. Chen - “XML Access Control for Semantically Related XML Documents”, *36th Annual Hawaii International Conference on System Sciences*, 2003

[15] E. Damiani, S.Vimercati, S. Paraboschi, P. Samarati – “Design and Implementation of an Access Control Processor for XML Documents”, *WWW9 / Computer Networks*, North-Holland Publishing Co, Amsterdam, The Netherlands, 2000, pp. 59 – 75.

[16] P. Bonatti, S. Capitani di Vimercati, P. Samarati, “A Modular Approach to Composing Access Control Policies”, *7th ACM Conference on Computer Communications Security*, ACM Press, Athens, Greece, 2000, pp. 164 – 173.

[17] E. Damiani, S. Vimercati , S. Paraboschi, P. Samarati, “A Fine-Grained Access Control System for XML Documents”, *ACM Transactions on Information and System Security*, ACM Press, 2002.

[18] P. Bonatti, S. Vimercati, S. Samarati, “An Algebra for Composing Access Control Policies”, *ACM Transactions on Information and System Security*, ACM Press, 2002.

[19] V. Gowadia, C. Farkas, “RDF Metadata for XML Access Control”, *Proceedings of the 2003 ACM workshop on XML security*, ACM Press, 2004, pp. 39 – 48.

[20] Web Site oficial XBRL, www.xbrl.org (2005) acessado em julho de 2005.

[21] T. van den Akker, Q. O. Snell, M. J. Clement, “The YGuard access control model: set-based access control”, *ACM symposium on Access control models and technologies*, ACM Press, Chantilly, Virginia, USA, 2001, pp. 75 – 84.

[22] R. Elmasri, S. B. Navathe, “Fundamentals of Database Systems”, Person Education, Inc., 2004.

[23] Open Financial Exchange Specification 1.5.1. Disponível em <www.ofx.org>. Acessado em 04 de julho. de 2005.

[24] FpML Especification. Disponível em <www.fpml.org>. Acessado em 04 de junho de 2005.

[25] SWIFT Standard. Disponível em <http://www.swift.com>. Acessado em 03 de julho de 2005.

[26] M. Champion, S. Byrne, G. Nicol, L. Wood, "Document Object Model (Core) Level 1". Disponível em <http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/level-one-core.html>. Acessado em 06 de julho de 2005.

[27] S. DeRose, E. Maler, D. Orchard, "XML Linking Language (XLink) Version 1.0". Diponível em <http://www.w3.org/TR/xlink/>. Acessado em 07/08/2005

[28] XML/Valuetype Language Mapping Specification. Disponível em <http://www.omg.org/docs/formal/03-04-01.pdf>. Acessado em 09 de julho de 2005.

[29] Straight-Through Processing Markup Language. Disponível em <http://www.stpml.org/>. Acessado em 01 de julho de 2005.

[30] Document Object Model level 1 Specification. Disponível em <http://www.w3.org/DOM/DOMTR>. Acessado em 30 de junho de 2005.

[31] P. Casarini, L. Padovani, "The Gnome DOM Engine", MIT Press, Cambridge, USA, 2001, pp. 173 – 190.

[32] Financial Information Exchange Markup Language. Disponível em <http://www.fixprotocol.org>. Acessado em 01 de agosto de 2005.

[34] P. Murray-Rust, H. S. Rzepa, "Chemical Markup, XML and the World-Wide Web. 2. Information Objects and the CMLDOM", *Inf. Comput. Sci*, 2000

[35] Mathematical Markup Language (MathML) Version 2.0 (Second Edition). Disponível em <http://www.w3.org/TR/MathML2/>. Acessado em 12 de julho de 2005.

[36] Scalable Vector Graphics (SVG) 1.1 Specification. Disponível em <http://www.w3.org/TR/SVG/>. Acessado em 11 de julho de 2005.

[37] Microsoft Money. Disponível em <http://www.microsoft.com/money/default.mspx>. Acessado em 11 de julho de 2005.

[38] ebXML Business Process Specification Schema 6 Version 1.01. Disponível em <http://www.ebxml.org/specs/ebBPSS.pdf>. Acessado em 02 de agosto de 2005.

[39] Market Data Markup Language (MDML) Specification. Disponível em <http://www.fisd.net/mdpolicy/1100_mdmlspec.pdf>. Acessado em 01 de junho de 2005.

[40] Y. Park, Xml in industries. Disponível em <http://www.diderottrack.nl/en.articles.branchestandaard.html>. Acessado em 27 de maio de 2005.

[41] Research Information Markup Language Specification Version 2.2. Disponível em <http://www.rixml.org/newsite/specification.html>. Acessado em 1 de agosto de 2005.

[42] Investiment Research Markup Language. Disponível em <http://www.irml.org/>. Acessado em 02 de agosto de 2005.

[43] E. Gama, R. Helm, R. Johnson, J. Vlissides, " Design Patterns", Addison-Wesley, 1995.

[44] SMIL Document Object Model, 2000. Disponível em <http://www.w3.org/TR/smil-boston-dom/smil.html>. Acessado em 20 de julho de 2005.

[45] FundsXML. Disponível em <http://www.funds-xml.org/>. Acessado em 10 de agosto de 2005.

[46] H. Maruyama, K. Tamura, N. Uramoto, "Digest Values for DOM (DOMHASH)", IBM Tokyo Research Laboratory, 1998.

[47] Simple API for XML. Disponível em <http://sax.sourceforge.net/>. Acessado em 26 de julho de 2005.

[48] J. Efrim Boritz, "Business Reporting with XML: XBRL (Extensible Business Reporting Language)", *Encyclopedia of the Internet*, 2004, pp. 863-885.

[49] O. Moreira, E. L. Riccio, M. G. Sakata, "A difusão do XBRL: Caso do Brasil", 2005.

[50] SOAP Specification. Disponível em <http://www.w3.org/TR/soap/>. Acessado em 10 de maio de 2005.

[51] T. Suzuki , "XBRL Processor Interstage XWand and Its Application Programs", 2003.

[52] NUX. Disponível em <http://dsd.lbl.gov/nux/>. Acessado em 4 de maio de 2005.

[53] Fujitsu XLink Processor. Disponível em <http://software.fujitsu.com/en/interstage-xwand/activity/xbrltools/xlip/>. Acessado em 28 de julho de 2005.

[54] Altova XQuery Engine. Disponível em <http://www.altova.com/features_xquery.html>. Acessado em 04 de junho de 2005.

[55] Interactive Financial eXchange Standard. Disponível em <http://www.ifxforum.org/ifxforum.org/index.cfm>. Acessado em 05 de agosto de 2005.

[56] Market Data Definition Specification. Disponível em <www.mddl.org/sub/specification.asp>. Acessado em 06 de agosto de 2005.

[57] Standard Generalized Markup Language. Disponível em <http://www.w3.org/MarkUp/SGML>. Acessado em 28 de agosto de 2005.

Apêndice A – Especificação de XBRLDOM

Nessa seção apresentaremos a definição para o modelo de objeto da linguagem XBRL. Seguindo o mesmo padrão utilizado na definição do DOM pela W3C, utilizaremos a linguagem *Interface Definition Language* (IDL) [28] publicado pela *Object Management Group* (OMG).

Interface XBRLDOMImplementation

A interface XBRLDOMImplementation estende a interface DOMImplementation adicionando o método createXBRLDocument.

```
interface XBRLDOMImplementation: DOMImplementation {
      XBRLInstanceDocument createInstanceDocument();
      XBRLTaxonomySchemaDocument createTaxonomySchemaDocument();
      XBRLLinkbaseDocument createLinkbaseDocument();
}
```

Método createInstanceDocument

Cria um documento de instância XBRL contendo uma árvore mínima que contém apenas o elemento raiz XBRLXBRLElement.

Método createTaxonomyDocument

Cria um documento XML Schema de taxonomia XBRL contendo apenas o elemento raiz *schema*.

Método createLinkbaseDocument

Cria um documento XML de um linkbase contendo o elemento raiz linkbase.

Interface XBRLElement

Todas as interfaces que correspondem diretamente a elementos na linguagem XBRL são derivadas dessa interface.

```
interface XBRLElement: Element {
}
```

Interface XBRLNodeList

A interface XBRLNodeList é uma especialização da interface *NodeList* do *DOM XML*.

```
interface XBRLNodeList: NodeList {
}
```

Interface XBRLInstanceDocument

Essa interface estende a interface *Document* e representa um documento de instância em um relatório XBRL.

```
interface XBRLInstanceDocument: Document {
      attribute XBRLNodeList facts;
      attribute XBRLNodeList linkbaseRefs;
      attribute XBRLNodeList schemaRefs;
      attribute XBRLNodeList roleRefs;
      attribute XBRLNodeList arcroleRefs;
      attribute XBRLNodeList contexts;
      attribute XBRLNodeList units;
      attribute XBRLNodeList footnotes;
      XBRLNodeList findDTS();
      XBRLItemFactElement createItemElement();
      XBRLTupleFactElement createTupleElement();
      XBRLLinkbaseElement createLinkbaseRefElement();
      XBRLContextElement createContextElement();
      XBRLUnitElement createUnitElement();
}
```

Método findDTS

Método procura recursivamente a lista de todas as taxonomias que compõem a Discovered Taxonomy Schema (DTS). A forma como essa busca deve ser realizada está descrito no documento de especificação de XBRL.

Método createItemElement

Método que cria um novo elemento do tipo XBRLItemFactElement

Método createTupleElement

Método que cria um novo elemento do tipo XBRLTupleFactElement

Método createLinkbaseRefElement

Método que cria um novo objeto XBRLLinkbaseRef

Método createContextElement

Método que cria um novo objeto XBRLContextElement

Método createUnitElement

Método que cria um novo objeto XBRLUnitElement

Atributo facts

Retorna a lista de todos os elementos *item* e *tuple* filhos do elemento *xbrl* em um documento de instância.

Atributo linkbaseRefs

Retorna a lista de todos os elementos linkbaseRef de um documento de instância.

Atributo schemaRefs

Retorna uma lista de todos os elementos schemaRef de um documento de instância.

Atributo roleRefs

Retorna uma lista de todos os elementos roleRef de um documento de instância.

Atributo arcroleRefs

Retorna uma lista de todos os elementos arcroleRef de um documento de instância.

Atributo contexts

Retorna uma lista de todos os elementos context de um documento de instância.

Atributo units

Retorna uma lista de todos os elementos unit de um documento de instância.

Atributo footnotes

Retorna uma lista de todos os elementos footnote de um documento de instância.

Interface XBRLTaxonomyDocument

Essa interface estende a interface *Document* e representa o documento XML Schema da taxonomia. O atributo *documentElement* deve ser o *XBRLSchemaElement* que representa a raiz do documento XML Schema da taxonomia.

```
interface XBRLTaxonomyDocument: Document {
      XBRLNodeList concepts;
      XBRLNodeList linkbaseRefs;
      XBRLNodeList imports;
      XBRLItemElement createItemElement();
      XBRLTupleElement createTupleElement();
      XBRLArcroleTypeElement createArcroleTypeElement();
      XBRLLinkbaseRefElement createLinkbaseRefElement();
}
```

Método createItemElement

Cria um novo objeto do tipo XBRLItemElement.

Método createTupleElement

Cria um novo objeto do tipo XBRLTupleElement.

Método createArcroleTypeElement

Cria um novo objeto do tipo XBRLArcroleTypeElement

Método createLinkbaseRefElement

Cria um novo objeto do tipo XBRLLinkbaseRefElement

Atributo concepts

Retorna lista de objetos XBRLConceptElement.

Atributo linkbaseRefs

Retorna lista de todas as referências a linkbase contidos no schema.

Atributo imports

Retorna lista de elementos *import* utilizados para importar uma taxonomia.

Interface XBRLLinkbaseDocument

Essa interface estende a interface Document e representa um documento linkbase.

```
interface XBRLLinkbaseDocument: Document {
      attribute readonly XBRLNodeList calculationLinks;
      attribute readonly XBRLNodeList presentationLinks;
      attribute readonly XBRLNodeList definitionLinks;
      attribute readonly XBRLNodeList referenceLinks;
      attribute readonly XBRLNodeList labelLinks;
      attribute readonly XBRLNodeList policeLinks;
      XBRLLabelElement createLabelElement();
      XBRLReferenceElement createReferenceElement();
      XBRLPoliceElement createPoliceElement();
      XBRLCalculationArcElement createCalculationArcElement();
      XBRLPresentationArcElement createPresentationArcElement();
      XBRLDefinitionArcElement createDefinitionArcElement();
      XBRLLabelArcElement createLabelArcElement();
      XBRLReferenceArcElement createReferenceArcElement();
      XBRLPoliceArcElement createPoliceArcElement();
}
```

Método createLabelElement

Cria um novo objeto do tipo XBRLLabelElement

Método createReferenceElement

Cria um novo objeto do tipo XBRLReferenceElement

Método createPoliceElement

Cria um novo objeto do tipo XBRLPoliceElement.

Método createCalculationArcElement

Cria um novo objeto do tipo XBRLCalculationArcElement.

Método createPresentationArcElement

Cria um novo objeto do tipo XBRLPresentationArcElement.

Método createLabelArcElement

Cria um novo objeto do tipo XBRLLabelArcElement.

Método createReferenceArcElement

Cria um novo objeto do tipo XBRLReferenceArcElement.

Método createPoliceArcElement

Cria um novo objeto do tipo XBRLPoliceArcElement

Atributo calculationLinks

Retorna uma lista de todos os extendedLink's do linkbase do tipo calculation.

Atributo presentationLinks

Retorna uma lista de todos os extendedLink's do linkbase do tipo presentation.

Atributo definitionLinks

Retorna uma lista de todos os extendedLink's do linkbase do tipo definition.

Atributo referenceLinks

Retorna uma lista de todos os extendedLink's do linkbase do tipo reference.

Atributo labelLinks

Retorna uma lista de todos os extendedLink's do linkbase do tipo label.

Atributo policeLinks

Retorna uma lista de todos os extendedLink's do linkbase do tipo police.

Método createCalculationElement

Cria um novo elemento extendedLink do tipo calculationLink.

Método createDefinitionElement

Cria um novo elemento extendedLink do tipo definitionLink.

Método createPresentationElement

Cria um novo elemento extendedLink do tipo presentationLink.

Método createPresentationElement

Cria um novo elemento extendedLink do tipo presentationLink.

Método createLabelElement

Cria um novo elemento extendedLink do tipo labelLink.

Método createreferenceElement

Cria um novo elemento extendedLink do tipo referenceLink.

Interface XBRLXBRLElement

Essa interface representa o elemento raiz *xbrl* de um documento de instância.

```
interface XBRLXBRLElement: XBRlElement {
      attribute DOMString id;
      attribute DOMString xmlbase;
}
```

Atributo id

String que representa o atributo id do elemento xbrl.

Atributo xmlbase

String que representa o atributo xml:base do elemento xbrl.

Interface XBRLLinkbaseRefElement

Interface que representa o elemento *linkbaseRef* presente no XML Schema da taxonomia ou no documento de instância.

```
interface XBRLLinkbaseRefElement: XBRLSimpleLinkElement {
}
```

Interface XBRLSchemaElement

Interface que representa o elemento raiz de todos os documentos de schema de uma taxonomia.

```
interface XBRLSchemaElement: XBRLElement {
}
```

Interface XBRLFactElement

Interface abstrata que representa um fato de um documento de instância. Nenhum elemento implementa diretamente essa interface.

```
interface XBRLFactElement: XBRLElement {
}
```

Interface XBRLItemFactElement

Interface abstrata que representa os elementos do documento de instância que implementam o elementos do substitutionGroup item na taxonomia.

```
interface XBRLItemFactElement: XBRLFactElement {
      attribute DOMString contextRef;
      attribute DOMString unitRef;
      attribute DOMString precision;
}
```

Atributo contextRef

Uma string que representa o atributo contextRef do elemento item.

Atributo unitRef

Uma string que representa o atributo unitRef do elemento item.

Atributo precision

Uma string no fromato “unsigned integer”, representa o atributo precision do elemento item.

Interface XBRLTupleFactElement

Interface abstrata que representa os elementos do documento de instância que implementam o elementos do substitutionGroup tuple na taxonomia.

```
interface XBRLTupleFactElement: XBRLFactElement {
}
```

Interface XBRLContextElement

Interface que representa o elemento context de um documento de instância.

```
interface XBRLContextElement: XBRLElement {
      readonly attribute DOMString id;
}
```

Atributo id

Uma string que representa o atributo id do elemento context.

Interface XBRLUnitElement

Representa o elemento unit de um documento de instância.

```
interface XBRLUnitElement: XBRLElement {
      readonly attribute DOMString id;
}
```

Atributo id

Uma string que representa o atributo id do elemento unit.

Interface XBRLConceptElement

Interface abstrata que representa um conceito em uma taxonomia. Nenhum elemento é diretamente representado por essa interface.

```
interface XBRLConceptElement: XBRLElement {
      attribute DOMString balance; // debit, credit
      attribute DOMString type;
}
```

Atributo balance

Atributo balance dos elementos definidos no schema da taxonomia. Os valores possíveis para esse atributo são "*credit*" e "*debit*". Se o elemento não foi do tipo *monetaryItemType* ou derivado deste, seu valor deve ser *null*.

Atributo type

O atributo type dos elementos definidos no schema da taxonomia.

Interface XBRLItemElement

Interface dos elementos do documento de schema da taxonomia que fazem parte do substitutionGroup item.

```
interface XBRLItemElement: XBRLConceptElement {
      attribute DOMString periodType; //instant, duration
}
```

Atributo periodType

O atributo periodType retorna a string "instant" ou "duration" e especifica se o tipo de período de um fato é instante ou duração.

Interface XBRLTupleElement

Interface dos elementos do documento de schema da taxonomia que fazem parte do substitutionGroup tuple.

```
interface XBRLTupleElement: XBRLConceptElement {
}
```

Interface XBRLSimpleLinkElement

Representa o elemento SimpleLink definido pela especificação de XLink.

```
interface XBRLSimpleLinkElement: XBRLElement, SimpleLinkElement {
}
```

Interface XBRLLinkbaseElement

Interface que representa o elemento linkbase.

```
interface XBRLLinkbaseElement: XBRLElement {
      attribute DOMString id;
      attribute DOMString xmlbase;
}
```

Atributo id

String que representa o atributo id do elemento linkbase.

Atributo xmlbase

String que representa o atributo xml:base do elemento linkbase.

Interface XBRLExtendedLinkElement

Interface que representa os *links* estendidos utilizados na taxonomia.

```
interface XBRLExtendedLinkElement: XBRLElement, ExtendedLinkElement
{
}
```

Interface XBRLCalculationLinkElement

Interface que representa o elemento calculationLink de uma taxonomia.

```
interface XBRLCalculationElement: XBRLExtendedLinkElement {
}
```

Atributo order

Representa o atributo order que especifica a ordem do link.

Atributo weight

Atributo que determina o peso para a soma do elemento.

Interface XBRLReferenceLinkElement

Interface que representa o elemento referenceLink de uma taxonomia.

```
interface XBRLReferenceElement: XBRLExtendedLinkElement {
}
```

Interface XBRLPresentationLinkElement

Interface que representa o elemento presentationLink de uma taxonomia.

```
interface XBRLPresentationElement: XBRLExtendedLinkElement {
}
```

Interface XBRLLabelLinkElement

Interface que representa o elemento labelLink de uma taxonomia.

```
interface XBRLLabelElement: XBRLExtendedLinkElement {
}
```

Atributo xmllang

Atributo que identifica o idioma do rótulo.

Interface XBRLDefinitionLinkElement

Interface que representa o elemento definitionLink de uma taxonomia.

```
interface XBRLDefinitionElement: XBRLExtendedLinkElement {
}
```

Interface XBRLPoliceLinkElement

Interface que representa o elemento *police* definido pela especificação de XBAC.

```
interface XBRLPoliceElement: XBRLExtendedLinkElement {
}
```

Atributo xbaccredencial

Atributo que determina a credencial do usuário

Atributo xbacpolice

Atributo que retorna o nome do conceito ao qual a política está sendo aplicada.

Interface XBRLArcroleTypeElement

Interface que representa o elemento arcoroleType.

```
interface XBRLArcroleTypeElement: XBRLElement {
      attribute DOMString arcroleURI;
      attribute DOMString id;
      attribute DOMString cyclesAllowed;
}
```

Interface XBRLLinkbaseRefElement

Representa o elemento linkbaseRef presente em documentos de instância e schema da taxonomia.

```
interface XBRLLinkbaseRefElement: XBRLSimpleLink {
}
```

Interface XBRLArcElement

Interface abstrata que representa os arcos nos documentos linkbase.

```
interface XBRLArcElement: XBRLElement{
      attribute DOMString xlinkfrom;
      attribute DOMString xlinkto;
      attribute DOMString xlinktitle;
      attribute DOMString xlinkshow;
      attribute DOMString xlinkactuate;
      attribute DOMString order;
      attribute DOMString use;
      attribute DOMString priority;
}
```

Atributo xlinkfrom

Representa o atributo xlink:from.

Atributo xlinkto

Representa o atributo xlink:to.

Atributo xlinktitle

Representa o atributo xlink:title.

Atributo xlinkshow

Representa o atributo xlink:show.

Atributo xlinkactuate

Representa o atributo xlink:actuate.

Atributo order

Representa o atributo order.

Atributo use

Representa o atributo use.

Atributo priority

Representa o atributo priority.

Interface XBRLCalculationArcElement

Representa o elemento calculationArc.

```
interface XBRLCalculationArcElement: XBRLArcElement {
}
```

Interface XBRLPresentationArcElement

Representa o elemento presentationArc.

```
interface XBRLPresentationArcElement: XBRLArcElement {
}
```

Interface XBRLDefinitionArcElement

Representa o elemento definitionArc.

```
interface XBRLDefinitionArcElement: XBRLArcElement {
}
```

Interface XBRLLabelArcElement

Representa o elemento labelArc.

```
interface XBRLLabelArcElement: XBRLArcElement {
}
```

Interface XBRLReferenceArcElement

Representa o elemento referenceArc.

```
interface XBRLReferenceArcElement: XBRLArcElement {
}
```

Interface XBRLPoliceArcElement

Representa o elemento policeArc.

```
interface XBRLPoliceArcElement: XBRLArcElement {
}
```

Interface XBRLResourceElement

Interface abstrata que representa os elementos do tipo xlink:resource.

```
interface XBRLResourceElement: XBRLElement {
}
```

Interface XBRLPoliceElement

Representa o elemento police.

```
interface XBRLPoliceElement: XBRLResourceElement {
}
```

Interface XBRLLabelElement

Representa o elemento label.

```
interface XBRLLabelElement: XBRLResourceElement {
}
```

Interface XBRLReferenceElement

Representa o elemento reference.

```
interface XBRLReferenceElement: XBRLResourceElement {
}
```

Assinaturas

_____

Valéria Cesário Times

_____

Luiz Gustavo Cordeiro da Silva

_____

Ivanildo José de Sousa Aquino Júnior