UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Leonardo Fossati Silveira

UM FRAMEWORK DE DESENVOLVIMENTO XML PARA A GERAÇÃO DE SOLUÇÕES EM PRODUÇÃO E GERENCIAMENTO DE CONTEÚDO

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Dr. Fernando Ostuni Gauthier

Florianópolis, Setembro de 2003

Sumário

Lista de Figuras

Lista de Tabelas

RESUMO

A grande quantidade de informação disponível na web, atualmente, e a atual falta de estruturação para estes dados, traz uma grande preocupação para a comunidade científica em desenvolver um conjunto de padrões para a descrição de conteúdos, tornando-os inteligentes e reutilizáveis. Ainda hoje, encontra-se em grandes proporções a utilização da linguagem HTML como veículo de difusão da informação através da Internet. Contando com uma sintaxe cujo objetivo é descrever a formatação dos documentos, esta linguagem não fornece estrutura semântica para os dados de forma a permitir que uma máquina os interprete. Os dados na web são em grande parte apenas interpretáveis pelos seres humanos. Assim, ainda existe uma grande falta de agentes de softwares inteligentes manipuladores de conteúdos na web. As ferramentas ainda são muito pobres e o atual desenvolvimento de soluções torna-se limitado.

O principal objetivo deste trabalho é o desenvolvimento de um framework para a autoria, gerenciamento e reutilização de conteúdos estrutrurados para a web baseados em XML. Tecnologias como XML (*eXtensible Marckup Language*), XSL (*eXtensible Stylesheet Language*), SVG (*Scalable Vector Graphics*) e MathML (*MATHmatical Marckup Language)*, serão utilizadas para compor as soluções propostas neste trabalho.

Palavras-Chave : XML, XSL, XQuery, SVG, MathML, reusabilidade, ensino à distância, web semântica, RDF, ontologias.

ABSTRACT

Since the world wide web has increased so much, the available data on the Internet are becoming so massive that, nowadays, its impossible to keep structure for this ones only with technologies such as HTML. This fact shows a big problem which must be solved by the web community. Even so, the HTML language is the predominant technologie for the data share on the web. With a sintaxe that aim at the data rendering description, this language don't provide semantics for this data. Hence, these ones are not understandable by the computers. Moreover, because of this, the majority of the available web information is understandable only by humans and there is a great lack of tools that are able to manipulate information and to provide a web data automation.

The main goal of this work is the development of a framework for the authorship, management and reusability of structured web contents using XML language. Technologies as XML (eXtensible Marckup Language), XSL (eXtensible Stylesheet Language), SVG (Scalable Vector Graphics) and MathML (MATHmatical Marckup Language), will be used to compose the solutions proposed in this work.

Key Words : XML, XSL, XQuery, SVG, MathML, reusability, learning distance, semantic web, RDF, ontologies.

Capítulo 1

Introdução

1.1 Introdução

A grande quantidade de informação disponível na web, atualmente, e a atual falta de estruturação para estes dados, traz uma grande preocupação para a comunidade científica em desenvolver um conjunto de padrões para a descrição de conteúdos, tornando-os inteligentes e reutilizáveis. Ainda hoje, encontra-se em grandes proporções a utilização da linguagem HTML como veículo de difusão da informação através da Internet. Contando com uma sintaxe cujo objetivo é descrever a formatação dos documentos, esta linguagem não fornece estrutura semântica para os dados de forma a permitir que uma máquina os interprete. Os dados na web são em grande parte apenas interpretáveis pelos seres humanos. Assim, ainda existe uma grande falta de agentes de softwares inteligentes manipuladores de conteúdos na web. As ferramentas ainda são muito pobres e o atual desenvolvimento de soluções torna-se limitado.

Berners-Lee, em seu artigo "*The Semantic Web"* [BER 01], cita : "Atualmente, a maior parte do conteúdo da web é projetada para o entendimento por humanos, e não para programas de computadores manipularem o seu significado". Foi justamente esta problemática uma das pricipais motivações para o surgimento da tecnologia XML. Em sua especificação [W3C 03d], XML é proposta como um formato universal para a estruturação e intercâmbio de dados. Com a capacidade de definição de um conjunto de *tags* (marcas) destinadas à estruturação de qualquer tipo de conteúdo, esta tecnologia viabilizou um grande passo da comunidade científica em direção a uma web com conteúdos semânticos.

Com a crescente necessidade de uma estruturação eficiente dos conteúdos, XML tornou-se uma estratégia que, sozinha, não satisfazia às necessidades de fornecer significado semântico aos dados da web. Para o entendimento do ser humano, dados em HTML[W3C 03e], bem como em XML, tornam-se facilmente interpretados e seu significado é extraído de forma direta. Infelizmente, os computadores não conseguem imitar o raciocínio humano para realizar a tarefa de interpretação intuitiva de dados. Na verdade, essa é uma das grandes metas de pesquisa da área de Inteligência Artificial.

Para uma máquina, dados em XML fornecem apenas significado sintático, mas a semântica não é preservada. Torna-se necessário algum mecanismo capaz de descrever a semântica das informações XML. Para Lucena [LUC 02], XML permite que usuários adicionem uma estrutura arbitrária a seus documentos mas não diz nada sobre o que estas estruturas significam. Visando prover uma descrição dos metadados XML, utiliza-se RDF (*Resource Description Framework*) [W3C 99b], uma linguagem que fornece a descrição dos dados XML em um formato padronizado. Enquanto que XML define a estrutura de um documento, RDF define sua ontologia, compondo a semântica da web, ou ”SemanticWeb” [W3C 03i]. Segundo Lucena, as tecnologias disponíveis para Web Semântica são a eXtensible Markup Language (XML) e o Resource Description Framework (RDF) [LUC 02]. A utilização de XML em grande escala para a estruturação das informações, juntamente com a descrição desta estruturação (definição de ontologias) através de RDF, tornaria a web atual muito mais poderosa, pois com recursos de dados altamente estruturados possuiria agentes de software capazes de manipular as informações realizando tarefas muito mais sofisticadas.

Este trabalho propõe um conjunto de soluções integradas baseadas em XML compondo uma metodologia para o desenvolvimento de soluções que visam a estruturação, reutilização e definição da semântica dos conteúdos na web. Três metodologias serão propostas : baseada em XML, para a estruturação e reutilização de conteúdos; baseada em MathML, para a publicação de conteúdos matemáticos na web e baseada em SVG, para a autoria e gerenciamento de gráficos estatísticos dinâmicos, multimedia e interativos para a Web. Para a validação destas propostas, três softwares serão apresentados : Gerenciador de Hipermídias, uma solução que integra todas as estratégias propostas e viabiliza o gerenciamento de dados e autoria de conteúdos baseados em fragmentos de dados reutilizáveis; Editor de Fórmulas, um software de autoria de conteúdos matemáticos codificados em MathML, capaz de produzir documentos neste formato a partir de dados em LaTeX; e DataGraphMaker, um editor que permite a produção de gráficos estatísticos (barras, pizza, pontos e linhas) em formato SVG.

Neste contexto, as contribuições deste trabalho são as seguintes: Metodologia RCM (*Reusable Content Methodology*); fornece a descrição de todo o processo de gerenciamento e reutilização de conteúdos bem como os módulos necessários para o encontro da solução; Linguagem RCML (*Reusable Content Markup Language*), um

aplicativo XML proposto para a descrição de conteúdos reutilizáveis para a web; RCM Script, uma linguagem de script, com sintaxe baseada em XML, com o objetivo de trabalhar de forma integrada com outras linguagens como XSL, provendo a autoria dinâmica de documentos compostos por conteúdos compartilhados por diferentes aplicações; Metodologia MathTeX, que provê estratégias para a implementação de ferramentas de autoria de conteúdos em formato MathML; Metodologia GraphML, um processo para a produção de gráficos estatísticos dinâmicos através da integração das tecnologias XML, XSL e SVG e Linguagem GraphML (*Graph Marckup Language*), um aplicativo XML destinado à estruturação e descrição de dados de gráficos estatísticos.

A integração de todos esses recursos compõe uma solução que traz muitas vantagens para a produção de documentos estruturados e reutilizáveis, fornecendo uma alternativa para a comunidade científica que atualmente busca o encontro de soluções como estas para compor a “Web Semântica”.

1.2 Justificativa

Sendo, XML, uma linguagem genérica para a estruturação de dados, torna-se difícil a difusão de ferramentas de autoria e gerenciamento que abrangem todas as possíveis utilizações desta tecnologia ao mesmo tempo. Contando com muitas técnicas e soluções que surgem à medida que seu crescimento se torna cada vez maior, a XML fornece uma ampla capacidade de crescimento e difusão de ferramentas de suporte. Possuindo um formato cujo padrão é universal, torna-se muito clara a alta expansão do desenvolvimento de soluções em software para tentar suprir todo esse espantoso crescimento. Embora soluções como o DOM (*Document Object Model*) [W3C 03c] vêm prover tal capacidade, soluções que usufruem destas facilidades geralmente são muito específicas, à medida que as aplicações XML são muito específicas. Como resultado, após vários anos desde seu surgimento, a linguagem de marcação extensível ainda conta com poucas ferramentas para seu suporte e manutenção capazes de proporcionar soluções particulares.

A atual preocupação da comunidade científica em encontrar uma web semântica com soluções de autoria e padronização de conteúdos altamente inteligentes na Internet traz grandes possibilidades de desenvolvimento de soluções que busquem o encontro de dados inteligentes não apenas interpretáveis pelos humanos, mas por agentes de

softwares, tornando a web mais inteligente e integrada [BER 01]. Infelizmente, ainda existe uma grande carência no que diz respeito à estratégias de soluções para a autoria de dados estruturados e reutilização da informação através da web semântica.

O novo formato de imagem para a Web, SVG [W3C 03h], é uma linguagem XML para a descrição de gráficos vetoriais escaláveis na Internet. Contando com codificação aberta, e em modo texto, esse formato possui inúmeras vantagens quando comparado a outros padrões de imagens já existentes. SVG herda todas as aplicabilidades e poder que XML possui, como, por exemplo, detém um conteúdo estruturado e pesquisável. Atualmente muitos trabalhos exploram esta tecnologia como uma estratégia para soluções em que representações gráficas de dados são necessárias. A grande utilização nesta área, embora ainda em meio acadêmico, é justificada pelo fato desta tecnologia poder ser integrada com padrões XML, como pode ser descrito em [BUL 00]. Com tantas vantagens em sua utilização, SVG possui uma sintaxe muito complexa. Para tanto, a necessidade de ferramentas de autoria torna-se evidente. Embora empresas como HP, Xerox e Corel já adotaram suporte total ou parcial para esta tecnologia [RANDY 00], ainda existe uma certa carência quanto à disponibilidade de softwares de autoria SVG no que diz respeito a aplicações específicas. Por ser integrado com XML, aplicações SVG podem fornecer uma representação gráfica de dados estruturados. Gráficos dinâmicos direcionados aos dados possuem muitas vantagens quando comparados aos outros tipos de gráficos atualmente utilizados. Através da tecnologia SVG integrada com as tecnologias XML e XSL torna-se possível a disponibilização de gráficos dinâmicos onde XML realiza o armazenamento e estruturação dos dados e SVG sua representação gráfica.

Juntos, gráficos estatísticos em SVG e conteúdo matemático no formato MathML [MATHML 03], fornecem uma poderosa solução para a produção de conteúdos didáticos e/ou científicos na web. Após muito tempo de pesquisa e busca de soluções para o problema de publicação de notação matemática na web [DS1 03], surgiu MathML, uma linguagem XML para a representação desses conteúdos na web. Além de conter todas as vantagens que uma linguagem XML possui, esta nova tecnologia traz consigo a capacidade de estruturar e fornecer significado aos conteúdos matemáticos, tornando-os capazes de serem processados e/ou pesquisados [CLA03].

1.3 Objetivos

1.3.1 Objetivos Gerais

O principal objetivo deste trabalho é o desenvolvimento de um framework para a autoria, gerenciamento e reutilização de conteúdos estruturados baseados em XML visando fornecer uma solução alternativa para a semântica da web. Tecnologias como XML (*eXtensible Marckup Language*), XSL (*eXtensible Stylesheet Language*), SVG (*Scalable Vector Graphics*) e MathML (*MATHmatical Marckup Language)* serão utilizadas para compor as soluções propostas neste trabalho.

1.3.2 Objetivos Específicos

1.3.2.1 Proposta de um framework para a reutilização de conteúdos

Visando obter um conjunto de estragégias que permitam o desenvolvimento de conteúdos estruturados, vistos como objetos reutilizáveis, e a manipulação destes dados de forma ágil e dinâmica, o framework proposto será dividido em três componentes :

* Metodologia RCM

Buscando encontrar soluções para o gerenciamento, estruturação e reutilização de conteúdos será desenvolvida uma metodologia, inicialmente denominada RCM (*Reusable Content Methodology*) que demonstrará como a integração das tecnologias XML e XSL, juntamente com as ferramentas que serão propostas neste trabalho, será utilizada para fornecer a produção de conteúdos estruturados e reutilizáveis. Serão apresentadas, também, estratégias de implementação mostrando como este modelo poderá ser implementado em diferentes aplicações.

* Linguagem RCML

Para encontrar a estruturação e a semântica dos objetos de dados modelados através do framework proposto, será desenvolvida uma linguagem baseada em XML, cuja denominção inicial será RCML (*Reusable Content Markup Language*), Linguagem de Marcação para Conteúdos Reutilizáveis.

* Linguagem RCM Script;

Será desenvolvida uma linguagem de *script* para permitir a manipulação dos objetos de dados através do framework proposto. A RCM Script, denominação inicial desta linguagem, terá como principal objetivo a descrição da reusabilidade dos

conteúdos de uma maneira explícita. Dentre outros, a transformação de conteúdo, a busca de fragmentos de objetos de dados e a integração com linguagens de manipulação já consagradas, como XSL, também serão possíveis através da utilização desta linguagem.

Para a validação das estratégias propostas, será desenvolvida uma ferramenta, inicialmente denominada “Gerenciador de Hipermídias”.

1.3.2.2 Desenvolvimento de duas soluções para validar a capacidade de integração do modelo proposto

Para comprovar a capacidade de integração do framework que será desenvolvido neste trabalho, duas soluções, totalmente independentes deste framework, serão desenvolvidas e integradas através de dois softwares que serão construídos para este fim. A integração destas duas soluções com o framework proposto será demonstrada em detalhes. As soluções propostas serão as que seguem :

* Soluções SVG

Será demonstrada uma estratégia de representação dinâmica de dados XML em forma de gráficos estatísticos no formato SVG através da utilização de folhas de estilos XSLT. Para viabilizar esta solução, uma ferramenta foi desenvolvida com o objetivo de gerar, gerenciar,visualizar e importar dados em XML e SVG. A integração com o framework proposto será demonstrada através da utilização desses dados (SVG) como objetos reutilizáveis (RCML) e manipulação (RCM Script e XSL) através da ferramenta Gerenciador de Hipermídias.

* Soluções MathML

Será proposta uma metodologia para a produção e conversão de conteúdos matemáticos para a web, baseando-se em tecnologias como MathML e LaTeX.

Para validar a metodologia proposta, será apresentado um software de autoria de conteúdos em formato MathML, através da importação de dados codificados em LaTeX.. A integração com o framework proposto será demonstrada através da utilização desses dados (MathML) como objetos reutilizáveis (RCML) e manipulação (RCM Script e XSL) através da ferramenta Gerenciador de Hipermídias.

Os seguintes pontos irão compor as soluções que desenvolver-se-á neste trabalho:

1. Definição de ontologias para domínios particulares;

2. Definições estruturais destas ontologias através do desenvolvimento de vocabulários XML específicos;

3. Desenvolvimento/Aplicação de folhas de estilos para a formatação e transformação dos dados;

4. Estratégias de implantação e implementação destas soluções;

5. Soluções para o gerenciamento e reusabilidade destes dados;

6. Capacidade de generalização para as soluções propostas;

7. Desenvolvimento de ferramentas para a manipulação, autoria e gerenciamento dos metadados.

1.4 O Estado da Arte e trabalhos Correlatos

Atualmente há uma grande preocupação em encontrar soluções para a padronização e estruturação dos conteúdos web. Uma nova área tem se tornado cada vez mais abrangente: a “The Semantic Web” [WS 03]. Esta área tem sido o foco das pesquisas que buscam uma estrutura padrão para definir conteúdos inteligentes na internet. A tecnologia que vem sendo utilizada como base para prover tal solução é XML. Muitos trabalhos têm sido desenvolvidos buscando encontrar soluções novas e definições de novos padrões. Enquanto que a grande preocupação da comunidade científica da web é a definição de padrões semânticos [BER 01], pouco tem sido feito no que diz respeito a definições estruturais para conteúdos particulares, ou seja, para aplicações específicas, bem como modelos de implementação para a produção de ferramentas de manipulação e autoria de metadados. O principal benefício que a padronização de conteúdos traz é a possibilidade de intercâmbio de dados. Quando define-se um padrão, todas as aplicações devem ser capazes de “entender” este novo formato, mas isto não significa que seus dados internos tenham que ser manipulados neste formato. O que geralmente ocorre é a utilização de formatos internos particulares para cada aplicação, mas com a capacidade de exportação em um formato padrão para prover a comunicação entre entidades heterogêneas.

A tecnologia RDF (*Resource Description FrameWork*), busca encontrar uma forma de descrever os metadados XML, mas não especifica como estes dados XML devem ser criados [W3C 99b]. Esta preocupação é particular para cada entidade que faça uso deste mecanismo de estruturação. Mostrar-se-á, abaixo, alguns trabalhos atuais, evidenciando o foco principal em padrões de intecâmbio, e pouca preocupação em estratégias específicas para a estruturação de conteúdos para a manipulação interna e implementação de uma determinada aplicação.

1.4.1 **"Explorando Linguagens de Markup Extensíveis na Construção de Sistemas de Educação Baseados na Web"** ***[SAN 03]***

1.4.1.1 Descrição do trabalho

Santanche, em seu trabalho, faz uma abordagem buscando encontrar uma arquitetura baseada em XML para o desenvolvimento de sistemas de educação baseados na web. Faz, também, um estudo sobre a definição de ontologias para objetos de ensino, seguindo os padrões Learning Objects da IEEE, [IEE 03]. Cita : "*Este trabalho apresenta uma nova abordagem na construção de tais sistemas, baseada na linguagem XML e no padrão RDF, a fim de desenvolver uma arquitetura aberta, extensível e cooperativa. A introdução de semântica ao conteúdo dos recursos, viabilizada pela XML e a abordagem adotada, abre novas e interessantes possibilidades na exploração das informações*".

1.4.1.2 Soluções propostas

Este trabalho busca encontrar um vocabulário XML para a descrição e semântica da informação utilizada em sistemas de educação baseados na web. Justifica-se na atual utilização de dados em formatos proprietários por esses sistemas, o que ocasiona muitas limitações. Este trabalho resume-se em revisão bibliográfica sobre XML, RDF e algumas outras tecnologias, seguindo da definição dos objetos utilizados em um sistema de ensino na web (forum de discussão, por exemplo) associada com metadados propostos correspondentes. Finalmente, uma linguagem XML e um esquema RDF completo são definidos para estruturar todas as informações necessárias para a implantação e utilização destes sistemas.

1.4.1.3 Resultados

O resultado deste trabalho é a definição estrutural do vocabulário XML proposto. Segue, na Figura 1.1, um exemplo de utilização desta linguagem.

```
<?xml version="1.0"?>

<?xml:namespace ns = "http://www.w3.org/RDF/RDF/" prefix ="RDF" ?>

<?xml:namespace ns = "http://purl.oclc.org/DC/" prefix = "DC" ?>

<?xml:namespace ns = “http://www.unifacs.br/EDUC/” prefix = “EDUC” ?>

<RDF:RDF>

<EDUC:ADiscussionGroup>

<DC:Identifier> FORUM005 </DC:Identifier> <EDUC:Type> Forum
</EDUC:Type>

<EDUC:Policy>

<EDUC:InscriptionPolicy policy=“moderated”/> <EDUC:ParticipationPolicy
policy=“free”/>

<EDUC:ParticipationDomain domain=“members”/>

<EDUC:AccessRestriction restriction=“open”/>

</EDUC:Policy>

<DC:Title> Educação a distância via Internet </DC:Title>

<DC:Subject> Educação a distância e ensino via Internet </DC:Subject>

<DC:Description> Debate sobre diversos temas relacionados a Educação a
distância e em

especial a Educação a distância que usa a Internet como mediadora
</DC:Description>

<EDUC:Members>

<EDUC:Member ID = “Gustavo”>

<EDUC:Reference reference=“cadastro.xml#gustvo”/> <EDUC:Status
status=“moderator”/>

</EDUC: Member>

<EDUC:Member ID = “Douglas”>

<EDUC:Reference reference=“cadastro.xml#douglas”/> <EDUC:Status
status=“member”/>

</EDUC: Member>

</EDUC:Members>

<EDUC:Contents>
```

```
<EDUC:Message>

<DC:Identifier> MSG1237 </DC:Identifier> <EDUC:Creator reference =
“#douglas”/>

<DC:Subject> Origem da EAD </DC:Subject> <DC:Date> 1997-03-05
</DC:Date>

<EDUC:Contents>Gostaria de saber se a educação a distância

surgiu com o computador ou antes dele.</EDUC:Contents>

</EDUC:Message>

</EDUC:Contents>

</EDUC:ADiscussionGroup>

</RDF:RDF>
```

Figura 1.1 Estado da Arte – [SAN 03].

1.4.1.4 Considerações

Como pôde ser observado, este trabalho preocupa-se apenas em mostrar uma definição estrutural baseada em metadados XML para uma determinada ontologia (Sistemas de educação baseados na web), sem referenciar ferramentas de suporte para viabilizar esta solução. Estratégias de utilização desta tecnologia em uma aplicação prática também não foram evidenciadas. Forneceu-se apenas a descrição dos metadados.

1.4.2 **“Proposta de um Modelo de Comunicação entre Sistemas de Informação em Saúde: INTERACTMed”** ***[GUS 01]***

1.4.2.1 Descrição do trabalho

Gusmão, em seu trabalho, mostra estratégias de estruturação de conteúdos para a web baseados em XML para aplicações na área da saúde. Cita como um dos objetivos de sua tese de doutorado : “*Avaliar e validar a Linguagem XML no intercâmbio de informações entre instituições de saúde*”.

1.4.2.2 Soluções propostas

Este trabalho busca encontrar um vocabulário XML para ser aplicado em Sistemas de Informação em Saúde. Justifica-se em alguns pontos, como a desigualdade de distribuição de recursos da assistência médica, a heterogeneidade e descentralização de bases de dados de saúde e a promoção de cuidado contínuo ao paciente. Assim, o principal objetivo deste trabalho é a construção e avaliação de um modelo de captura de

informações demográficas e de saúde de pacientes. O propósito da linguagem XML neste trabalho é a possibilidade de intercâmbio de informações entre instituições de saúde.

1.4.2.3 Resultados

Desenvolvimento de um modelo integrado baseado na troca de dados na web, de apoio à Telemonitoração de Saúde e à Segunda Opnião Médica. Este modelo propõe XML como a tecnologia que fornecerá a estrutura necessária para este intercâmbio de dados. O sistema desenvolvido denominou-se INTERACTMed.

1.4.2.4 Considerações

Este trabalho lida com uma aplicação XML interessante que é a telemedicina. Basicamente fez um estudo sobre as necessidades do intercâmbio de dados na área da saúde, revisão bibliográfica sobre XML e propôs uma ontologia XML para ser aplicada em sistemas de saúde. Não foram apresentados detalhes sobre a estrutura deste vocabulário, mas um aplicativo destinado à autoria deste conteúdo foi desenvolvido e demonstrado. Nenhuma evidência de como a solução proposta poderá ser implantada é apresentada.

1.4.3 **"Uma Ontologia para Apoio na Administração de Conteúdo Educativo na Web"** ***[MUN 02]***

1.4.3.1 Descrição do trabalho

Munoz, em seu trabalho, procura encontrar soluções baseadas em XML para ajudar na administração e estruturação de cursos em um ambiente web adaptativo. Cita : "*O trabalho proposto trata do estudo de metodologias para o desenvolvimento de ontologias e linguagens para a sua representação com o objetivo de implementar um protótipo para o auxílio na administração de partes de cursos armazenadas em documentos XML, no ambiente AdaptWeb*.

1.4.3.2 Soluções propostas

Este trabalho propõe uma ontologia particular através do projeto e representação de metadados que explicitem informação semântica sobre as partes de um curso baseado na web armazenadas em documentos XML. O objetivo é apoiar a utilização destes dados em diferentes cenários. É um trabalho colaborativo para o sistema AdaptWeb, um projeto que busca encontrar soluções para o desenvolvimento de sistemas de ensino web adaptativos

[MUN 02]. Assim, este trabalho preocupa-se com a definição estrutural das informações baseando-se na proposta de uma ontologia através da linguagem XML.

1.4.3.3 Resultados

A principal contribuição deste trabalho foi o estudo da aplicação das metodologias de construção e das linguagens de representação de ontologias voltadas para o problema da administração de partes de conteúdo educacional no projeto AdaptWeb.

1.4.3.4 Considerações

Este trabalho tratou de estratégias para a definição de ontologias destinadas a sistemas de ensino web adaptavidos (AdaptWeb), descreveu a arquitetura do sistema AdaptWeb e não mostrou resultados práticos de implementação. Como pode ser observado, o foco deste trabalho foi o estudo e busca de definições estruturais aos dados (XML) através de ontologias para um domínio particular que é o projeto AdaptWeb.

1.4.4 **"Uso de ontologias para gerenciamento e acesso a documentos na web"** ***[CHA 03]***

1.4.4.1 Descrição do trabalho

Chaves, em seu trabalho, desenvolve soluções baseada em XML e RDF para encontrar a semântica dos conteúdos na web. Cita : "*Este trabalho apresenta um estudo sobre técnicas para organização, apresentação e busca de informações na Internet e explora um exemplo de aplicação destas técnicas. A tecnologia XML, juntamente com RDF e RDF Schema, é utilizada para fornecer um suporte estrutural para a representação das informações* ".

1.4.4.2 Soluções propostas

Este trabalho propõe a definição de um vocabulário XML para a descrição de dados em um contexto "Universidade". Um forte enfoque é dado à descrição das informações referentes às disciplinas de graduação de uma universidade. Folhas de estilos foram desenvolvidas para a formatação destes dados e alguns *scripts* de código em Xquery foram implementados para a consulta dos dados no vocabulário proposto.

Um exemplo do vocabulário desenvolvido é ilustrado na Figura 1.2.

```
<BIBLIOGRAFIA>

<BIB_BAS>Bibliografia básica

<LIVRO>

<AUTOR>BERNSTEIN, P. A. </AUTOR>

<TITULO>Concurrency control and recovery in database systems</TITULO>

<LOCAL> Reading</LOCAL>

<EDITORA> Addison-Wesley</EDITORA>

<ANO_PUBLIC>1987</ANO_PUBLIC>

<NRO_PAGINAS>370 </NRO_PAGINAS>

</LIVRO>

<LIVRO>

<AUTOR>DATE, C. J. </AUTOR>

<TITULO>Introducao a sistemas de bancos de dados</TITULO>

<NRO_EDICAO> 2</NRO_EDICAO>

<LOCAL> Rio de Janeiro</LOCAL>

<EDITORA> Campus</EDITORA>

<ANO_PUBLIC>1993</ANO_PUBLIC>

<NRO_PAGINAS>513</NRO_PAGINAS>

</LIVRO>

...
```

Figura 1.2 Estado da Arte – [CHA 03].

1.4.4.3 Resultados

A principal contribuição deste trabalho está direcionada para o tratamento semântico de informações na Internet, uma vez que possibilita a extração de informação baseada em conteúdos do contexto “universidade”. A aplicação deste trabalho se restringiu às disciplinas dos cursos de graduação de uma universidade podendo ser estendido para outras aplicações cujo domínio seja “universidade” Outra contribuição foi o desenvolvimento de folhas de estilos para formatar dados do vocabulário proposto, bem como estratégias de busca de informação XML através de Xquery.

1.4.4.4 Considerações

Este trabalho focalizou um estudo sobre a utilização de tecnologias XML para prover a semântica da informação em um domínio particular. Citou estratégias e vantagens de utilização de folhas de estilos XSL, e mostrou as vantagens e aplicações de ontologias. Contribuiu com o desenvolvimento de um vocabulário XML para o domínio "Universidade", folhas de estilos XSL para a formatação destes dados e documentos Xquery para a consulta ou busca de informação baseada neste vocabulário. Poucas novidades foram evidenciadas neste trabalho, uma vez que a utilização de XML para a estruturação dos dados e XSL para sua formatação, já são estratégias consagradas. Nada foi mostrado no que diz repeito à autoria e implementação de ferramentas de suporte e gerenciamento destes dados. Modelos de implantação, buscando prover uma aplicação prática da solução proposta, também não foram evidenciados.

1.4.5 **"XML Support for an Operational Enterprise Ontology"** ***[GEE 03]***

1.4.5.1 Descrição do trabalho

Geerts, em seu trabalho, mostra como XML pode ser aplicado para a definição de ontologias. Cita "*This paper discusses the use of XML in support of an operational enterprise ontology*".

1.4.5.2 Soluções propostas

Este trabalho propõe a definição de uma ontologia para um domínio particular cuja definição foi "*REA Resource-Event-Agent framework*". Uma arquitetura baseada em XML foi proposta para prover a definição estrutural deste domínio baseado na ontologia REA. Assim, um vocabulário XML foi desenvolvido para fornecer esta estrutura. A Figura 1.3 mostra um exemplo deste vocabulário.

```
<dual inflow = "cashreceipt" iminc = "0" imaxc = "1" outflow = "sale"
ominc = "0" omaxc = "1"/>

<sale>

<invoiceno> 1 </invoiceno>

<amount> 200 </amount>

<agent type = "customer"> 1 </agent></sale>
```

Figura 1.3 Estado da Arte – [GEE 03].

1.4.5.3 Resultados

As principais contribuição deste trabalho foram : a proposta da ontologia "*REA Resource-Event-Agent framework*" e as definições estruturais desta ontologia através de um vocabulário XML específico.

1.4.5.4 Considerações

Este trabalho resume-se em mostrar como XML pode ser aplicado para a definição estrutural de domínios de dados particulares através do uso de ontologias específicas. Faz uma revisão bibliográfica sobre XML e ontologias, propõe uma ontologia particular, "*REA Resource-Event-Agent framework*", e desenvolve um vocabulário XML para as definições da estrutura desta ontologia. Nada mais é apresentado além de uma simples definição de uma ontologia e de um vocabulário. Folhas de estilos XSL não foram apresentadas e nada foi mostrado como validação do modelo proposto.

1.4.6 Considerações finais

Como pôde ser observado acima, grande parte dos trabalhos relacionados à estruturação de conteúdos web preocupa-se, principalmente, com a definição de padrões web e de ontologias. Estas definições são de essencial importância, porém precisa-se observar que para a utilização dessas estratégias em um domínio particular é necessário o estudo e pesquisa de metodologias que viabilizem a aplicação prática destas soluções. Enquanto que trabalhos como [MUN 02], [TAR 03] e [GUZ 01] citam estratégias para a definição de conteúdos, bem como a definição de sua ontologia e estruturação baseando-se em tecnologias XML, nada é dito sobre modelos de implantação e implementação destas soluções. Baseado no tipo de conteúdo que foi definido utilizando ontologias descritas em muitos trabalhos acima, é preciso o desenvolvimento de soluções integradas e de ferramentas para a autoria e manipulação dos metadados.

Este trabalho, além da definição de ontologias baseadas em XML, busca propor modelos de implementação e ferramentas de autoria. Outros aplicativos XML, que são padrões ontológicos universais, como MathML (*Mathematical Marckup Language*) e SVG (*Scalable Vector Graphics*), também serão demonstrados e um modelo de implementação e produção de ferramentas de autoria nestes formatos também serão evidenciados.

Os items numerados de 1 a 7, definidos na Seção 1.3.2, são mostrados na Tabela 1.1 para a realização de uma comparação entre os trabalhos correlatos apresentados nesta seção.

Itens / Trabalhos	1	2	3	4	5	6	7
[SAN 03]	X	X					
[GUS 01]	X	X					X
[MUN 02]	X	X					
[CHA 03]	X	X	X		X		
[GEE 03]	X	X					

Tabela 1.1 Estado da Arte – Comparação dos trabalhos correlatos.

1.5 Conclusão

Este capítulo fez uma breve introdução ao trabalho, mostrando os objetivos, justificativas, trabalhos correlatos e outros aspectos importantes para esta dissertação. O Capítulo 2 faz uma revisão bibliográfica sobre algumas tecnologias responsáveis pela descrição e estruturação dos dados na web : XML (*eXtensible Marckup Language*), XSL (*eXtensible Marckup Language*), SVG (*Scalable Vector Graphics*), MathML (*MATHematical Marckup Language*), *Semantic Web* e Ontologias.

Capítulo 2

Definições estruturais e semânticas para os conteúdos Web

O objetivo deste capítulo é fazer uma revisão bibliográfica sobre as tecnologias XML (*eXtensible Marckup Language)*, XSL (*eXtensible StyleSheet Language)*, MathML (*Mathematical Marckup Language*) , SVG (*Scalable Vector Graphics)*, Semantic Web e Ontologias.

2.1 XML - eXtensible Markup Language

2.1.1 Introdução

Com o enorme crescimento da Web, tornou-se necessária a criação de padrões para a comunicação de dados. Com isso, foram criadas linguagens de marcação para serem utilizadas com a internet [MOU 00].

Segundo [W3C 03e], a linguagem HTML foi inicialmente o padrão pelo qual páginas Web podiam ser facilmente criadas em uma plataforma, colocadas em um servidor e vistas em diversas plataformas diferentes. A sua simplicidade, tanto de sua sintaxe como da sua forma de utilizar, impulsionou o crescimento da Web e a divulgação da mesma. Mesmo assim, a HTML possui muitas desvantagens. Embora necessária para exibir informações, é relativamente insuficiente para a representação de dados reais na forma como eles devem ser estruturados. A HTML oferece ao usuário apenas uma camada visual do documento com uma navegação unidirecional [MOU 00]. Abaixo estão descritas algumas características da linguagem HTML:

* Possui número limitado e não extensível de tags;
* Descreve o formato de apresentação;
* Inadequado para o gerenciamento de grande volume de dados;
* Não oferece a funcionalidade requerida pelo comércio eletrônico.

Com todas essas carências, surgiu a necessidade de uma linguagem mais ampla que fosse capaz de supri-las. Conforme [W3C 03b], a SGML (*Standard Generalized Markup Language*) é uma linguagem muito poderosa, e que a HTML é apenas uma aplicação dela. A SGML solucionaria todos os problemas da Internet gerados pelas

limitações da HTML. Mesmo assim, a sua alta complexidade inviabilizou a sua utilização. Seria necessária a aquisição de conhecimentos experientes para a produção de documentos nesse formato. Os principais problemas encontrados na SGML são os seguintes [W3C 03b]:

* Falta de suporte pelos browsers mais populares;
* Uso de SGML na Web envolve a tradução para HTML;
* Perda de informações;
* Intercâmbio e automação de dados ficam muito mais difíceis;
* Impossível reconstruir o arquivo original a partir do arquivo HTML;
* Falta de suporte para folhas de estilos.

Assim, tornou-se importante que se criasse um projeto de uma linguagem nova, abrangente e que combinasse a flexibilidade e capacidade da SGML com a ampla aceitação e simplicidade da HTML. A XML (*eXtensible Marckup Language*) [W3C 03d] surgiu para resolver todos esses problemas. O padrão XML foi criado para tornar possível a entrega de informações SGML na Web. Ele elimina as limitações do padrão SGML e, ao mesmo tempo, oferece todos os seus benefícios. Essa linguagem teve como base as especificações da SGML e, realmente, foi especificada para ser um sub-conjunto desta linguagem. Sendo, portanto, um sub-conjunto, a XML reúne muitos dos recursos SGML, mas de uma forma muito mais simples. O XML é o sub-conjunto da SGML que reúne todos os recursos referentes à web de uma forma simplificada. XML é o SGML adaptado à Web. [W3C 03d] define as principais características da linguagem XML:

* Sua estrutura define o conteúdo (dados);
* Suas tags descrevem os dados fornecendo um significado aos documentos;
* Fornece a capacidade de criar um conjunto próprio de marcas;
* A apresentação é definida externamente através de folhas de estilo como, por exemplo, o XSL e o CSS;
* Os dados são separados da apresentação e do processamento dos dados

A linguagem XML possui muitas vantagens e pode ser aplicada em diferentes situações. Em muitas delas onde se precisa trabalhar com dados estruturados pode-se utilizar a XML para esse fim.

2.1.2 Elementos

2.1.2.1 Definição

Um *elemento* é um *contêiner* de conteúdo – pode conter dados de caractere, outros elementos e/ou outras marcações (comentários, referências de entidade etc.). Uma vez que representam objetos discretos, os elementos podem ser considerados como "substantivos " da XML. Para simplificar, podemos dizer que, na XML, um elemento é algo que descreve um dado [MOU 00].

Um elemento é diferente de uma marca usada no HTML, [W3C 03e], pois a marca HTML realmente descreve a marca (*tag)* e não o conteúdo, já o elemento descreve o conteúdo.

A XML permite a criação de seu próprio conjunto de elementos para uma aplicação específica. Os nomes de elementos a serem criados devem atender aos padrões XML e ser descritivos. Quando for necessário aumentar o significado do conteúdo de um determinado elemento, pode-se utilizar os *atributos*. A estrutura de um elemento pode ser observada na Figura 2.1

```
<NOME-ELEMENTO>     Marca de Abertura

</NOME-ELEMENTO>    Marca de Fechamento
```

Figura 2.1 XML – Exemplo de elementos XML.

2.1.2.2 Tipos de Conteúdo

Dentro de um elemento XML, podem ser inseridas as seguintes informações:

* Dados de caracteres;

Os elementos que podem conter apenas dados de caracteres são os mais simples. Os dados de caracteres são apenas texto, ou seja, um conjunto de strings que não contenha os caracteres reservados (tags, símbolos especiais). A Figura 2.2 apresenta um exemplo deste tipo de conteúdo.

```
<NOME-ELEMENTO>

   Texto qualquer

</NOME-ELEMENTO>
```

Figura 2.2 XML – Exemplos de elementos contendo dados de caracteres.

* Elementos;

Elementos XML podem ser ordenados hierarquicamente compondo um conjunto de elementos aninhados. Um exemplo da utilização de elementos como conteúdo de outros elementos pode ser observado na Figura 2.3.

```
<NOME-ELEMENTO>

   <ElementoFilho> ... </ElementoFilho>

</NOME-ELEMENTO>
```

Figura 2.3 XML – Exemplo de elementos contendo outros elementos.

* Referências a Entidades;

As referências a Entidades, para um processador XML qualquer, são consideradas como dados de caracteres, não possuindo uma declaração específica para as mesmas. A Figura 2.4 apresenta um exemplo de utilização.

```
<!ENTITIE Titulo “Um Framework XML”>

<NOME-ELEMENTO>

   &Titulo;

</NOME-ELEMENTO>
```

Figura 2.4 XML – Exemplo de elementos contendo referências a entidades.

* Dados mistos

Os elementos podem conter uma combinação dos tipos de conteúdos explicados acima, formando um conjunto de dados misto conforme pode ser observado na Figura 2.5.

```
<!ENTITIE Titulo “Um Framework XML”>

<NOME-ELEMENTO>

 O Título do trabalho é   &Titulo;

 <ElementoFIlho> ... </ElementoFilho>

</NOME-ELEMENTO>
```

Figura 2.5 XML – Exemplo de elementos contendo dados mistos.

2.1.3 Atributos

Muito embora se saiba que os elementos da linguagem XML fornecem informações referentes ao seu conteúdo, fornecendo estrutura ao mesmo, às vezes são necessárias informações adicionais. Quando em uma determinada estrutura XML se fazem necessárias informações adicionais, além das fornecidas pelos elementos, ou seja, quando somente os elementos não são suficientes para estruturar um determinado conteúdo, então utilizam-se os atributos, [MOU 00].

Conforme pode ser observado na Figura 2.6, um atributo é definido pelo par “nome-do-atributo” e “valor-do-atributo” que é inserido na marca de abertura de um determinado elemento. Pode-se ter uma lista de atributos em cada elemento, desde que esta seja declarada na DTD do documento.

```
<ELEMENTO atributo1 = “valoratributo1” atributo2 = “valor atributo2”>

</ELEMENTO>
```

Figura 2.6 XML – Exemplo de atributos.

A XML fornece três tipos de atributos, conforme os seus conteúdos:

* Atributos string;
* Atributos contendo tokens;
* Atributos enumerados;

2.1.4 Entidades

2.1.4.1 Definição

Entidades são referências a um determinado recurso. Este recurso fica armazenado em uma entidade, ou seja, uma entidade é uma unidade de armazenamento.

Um uso muito comum de entidade é quando se utiliza, repetidamente, um determinado fragmento de texto. Assim se torna importante a criação de uma entidade para referenciar este fragmento de texto. Uma entidade pode referenciar praticamente qualquer tipo de conteúdo [MOU 00]. São listados, a seguir, alguns dos tipos de conteúdos que uma entidade pode armazenar :

* Dados de caracteres;
* Segmento de código XML;
* Segmento de código de uma DTD;
* Caracteres reservados;
* Recursos binários.

Uma entidade deve ser declarada com a palavra-chave “Entitie”, seguida do nome da entidade e de seu conteúdo, como pode ser observado na Figura 2.7.

```
<!ENTITIE EntidadeTeste “Conteúdo da Entidade”>
```

Figura 2.7 XML – Exemplo de declaração de entidades.

A utilização de entidades é feita através de referências conforme sua declaração. Para tanto é necessária a utilização do símbolo “&” seguido do nome da entidade e do símbolo “;”, ou seja, &nomeentidade; conforme pode ser observado na Figura 2.8.

```
Este texto faz referência à entidade “EntidadeTeste” cujo conteúdo é
&EntidadeTeste;
```

Figura 2.8 XML – Exemplo de referência a entidades.

2.1.4.2 Tipos de Entidades

Em [W3C 03d], define-se que as entidades podem ser classificadas quanto à localização de seus recursos. Se uma determinada entidade referenciar um certo conteúdo que não esteja no mesmo local que a sua declaração, ou seja, não sendo uma entidade auto-suficiente, então trata-se de uma *Entidade Externa.* Se o recurso desta entidade estiver localizado junto com a sua declaração, dentro da DTD, então tem-se uma *Entidade Interna.*

Quando se trata de *Entidades Externas,* pode-se, ainda, classificá-las em *entidades de sistema e entidades públicas.* Quando o recurso está localizado separado da DTD, mas dentro da mesma máquina, então se tem uma *Entidade Externa de*

Sistema. Já quando o recurso está localizado remotamente em outra máquina, então temos uma *Entidade Externa Pública.*

Pode-se classificar as entidades, também, em *Entidades de Texto,* que armazenam dados de caracteres; *Entidades Binárias,* cujo recurso armazenado é binário e *Entidades Paramétricas,* entidades especiais que armazenam conteúdos que são utilizados dentro da DTD, ou seja, existem para serem utilizadas apenas dentro da DTD e não em documentos XML.

Para finalizar, serão citados os usos mais comuns de entidades [CAG 01] :

* Nomes alternativos para Marcações e Textos usados freqüentemente;
* Incluir documentos XML dentro de outros documentos XML;
* Organizar a DTD através de Entidades Paramétricas;
* Incluir recursos não-XML;
* Representar caracteres Não-ASCII.

2.1.5 DTD (Document Type Definition)

2.1.5.1 Introdução

A linguagem XML é uma linguagem muito rígida em relação às regras sintáticas. Diferentemente da HTML, um documento XML precisa estar sintaticamente correto para ser utilizado.

Moutis [MOU 00] classifica os documentos XML, quanto às regras sintáticas, em dois tipos : *documentos bem formados* e *documentos válidos.*

2.1.5.2 Definição

Uma DTD (*Document Type Definition*) é a base de dados que define toda a estrutura sobre a qual uma determinada fonte será criada [W3C 03d]. Numa DTD são definidos, entre outros:

* O nome de todos os elementos que poderão ser utilizados em um documento XML;
* A lista de atributos para cada elemento declarado anteriormente;
* O tipo de dado que os valores de cada atributo definidos irão conter;
* A lista de elementos filhos que cada elemento poderá conter;
* As regras para esses elementos filhos, como ordem e aninhamento, etc.

Para exemplificar, supõe-se um aplicativo XML que destina-se à descrição de dados para representação em gráficos estatísticos. A DTD para este aplicativo define, entre outros, quais os elementos pertencentes a esta linguagem e qual a ordem hierárquica que estes devem ser utilizados. Assim, se este aplicativo conter um elemento chamado “Grafico”, com um atributo obrigatório chamado “Tipo” e com uma ou mais instâncias de um elemento chamado “Item”, a DTD correspondente, para a definição desta estrutura, é a que segue na Figura 2.9.

```
<!ELEMENT Grafico (Item)+>

<!ELEMENT Item (#PCDATA)>

<!ATTLIST Grafico Tipo CDATA #Required>
```

Figura 2.9 XML – Exemplo de utilização de uma DTD.

2.1.5.3 Documentos Bem Formados

Para que um documento seja bem formado, é necessário que este obedeça um conjunto de regras específicas. Essas regras descrevem a sintaxe XML. Por exemplo, para que um documento seja bem formado, para cada marca (*tag*) de abertura deve haver uma marca de fechamento correspondente. O documento exposto na Figura 2.10 não é bem formado.

```
<Grafico Tipo = “Barras”>

  <Item>

      15

   </Item>

  <Item>

      33

   </Item>
```

Figura 2.10 XML – Exemplo de um documento não bem formado.

Pôde-se observar na Figura 2.10 que não existe uma marca de fechamento para o elemento”Grafico”. Logo, este documento não é bem formado.

2.1.5.4 Documentos Válidos

Para que um documento XML seja considerado um *documento válido*, será necessário que este, além de ser bem formado, obedeça ao conjunto de regras declarado na DTD deste documento.

Por exemplo, para a DTD da Figura 2.9, o documento da Figura 2.11 é considerado bem formado mas não é considerado válido.

```
<Grafico>
   <Item>
      15
   </Item>
   <Item>
      33
   </Item>
   <Valor>
     444
   </Valor>
</Grafico>
```

Figura 2.11 XML – Exemplo de um documento não válido.

Nota-se na Figura 2.11 que, conforme a DTD representada na Figura 2.9, o elemento “Grafico” deve conter um atributo chamado “Tipo”, e nada é dito sobre a existência de um elemento filho chamado “Valor”. Logo, este documento não é considerado válido.

2.1.6 Estrutura de Documentos XML

Esta seção está baseada em [CAG 01].

O objetivo das marcas XML é estruturar o conteúdo dando significado ao mesmo. Já foi observado que o conjunto de elementos XML assume uma forma hierárquica “pai-filho”. Isto indica que a estrutura das bases de dados XML que serão interpretadas por um processador qualquer assume forma de uma “árvore”. A hierarquia aqui citada é classificada como hierarquia de contenção, ou seja, o elemento “pai”

contém o elemento “filho”. A estrutura básica para representar uma fonte de dados XML é apresentada na Figura 2.12.

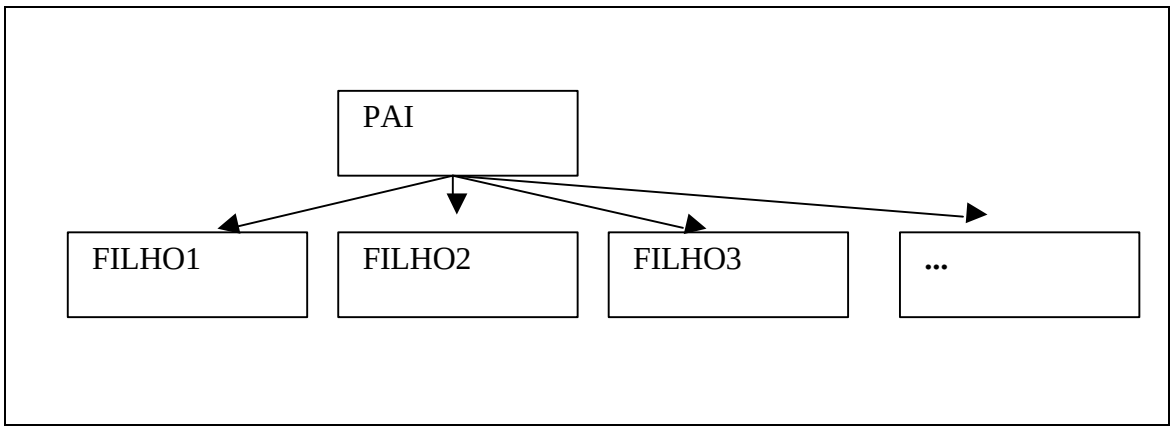


Figura 2.12 XML – Estrutura Hierárquica.

2.1.7 Interpretando documentos XML

A interpretação da estrutura de uma determinada base de dados XML é feita por módulos de softwares que são denominados *processadores XML*. Esses processadores geralmente possuem analisadores sintáticos para verificar a validade dos documentos. Os processos através dos quais um certo documento XML deverá passar, desde a criação da DTD, são os seguintes: análise da aplicação, criação da DTD, criação da fonte de dados XML, processar dados e exibir dados através de folhas de estilo. A Figura 2.13 mostra a seqüência desses processos:

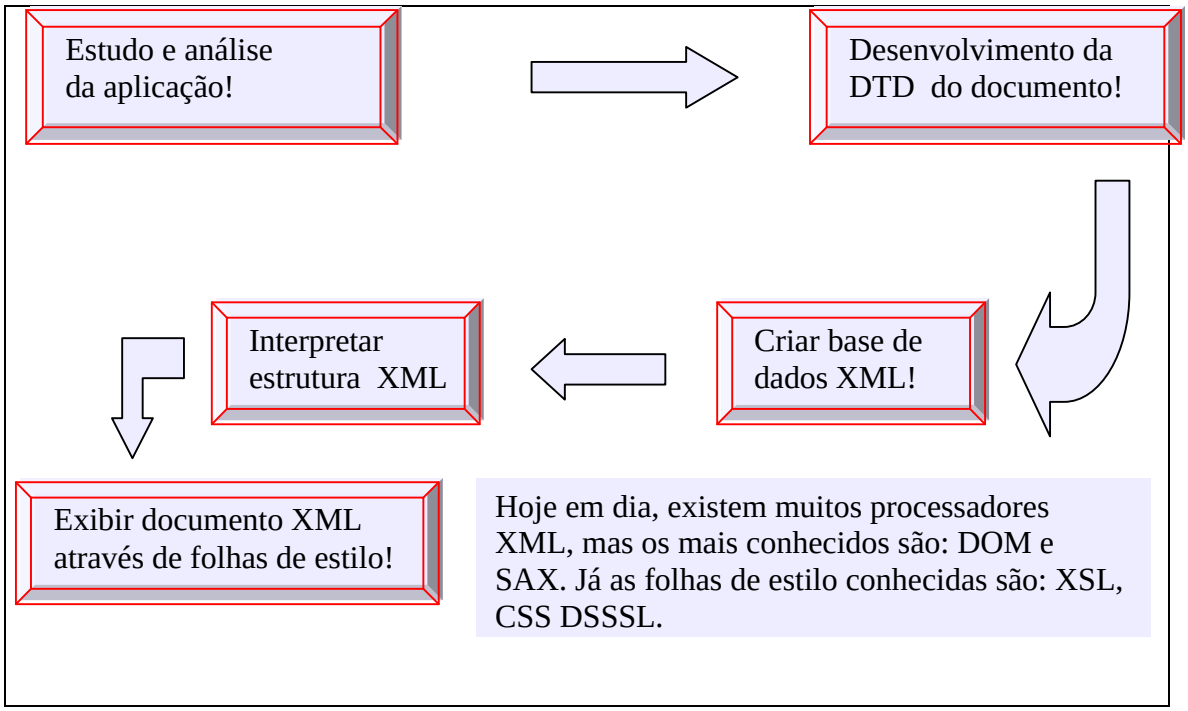


Figura 2.13 XML – Processos.

Como citado, os principais processadores XML são o DOM [W3C 03c] e o Sax [SAX 03]. O DOM (*Document Object Model*) é um modulo de software utilizado e acessado principalmente através de *scripts*, que possui como função interpretar e analisar uma determinada estrutura XML.

O termo *Document Object Model* foi aplicado a navegadores da web por algum tempo. Os objetos como janela, documento e histórico foram considerados parte do modelo orientado a objetos do navegador. Qualquer pessoa que tivesse envolvida em desenvolvimento da web, no entanto, observaria que o meio no qual este modelo foi implementado variou conforme diferentes navegadores. A fim de criar um meio mais padronizado de acessar e manipular estrutura de documentos na web, o W3C [W3C 03m] produziu especificações resultando no DOM atual [W3C 03c].

Segundo [W3C 03m], o DOM é uma definição neutra de plataforma e de linguagem, ou seja, as interfaces são definidas para os diferentes objetos englobando o DOM, mas nenhuma característica específica da implementação é fornecida. Assim, ele poderia ser feito em qualquer linguagem de programação. O DOM define uma funcionalidade padrão para navegação e manipulação de documentos do conteúdo e estrutura dos documentos XML e HTML.

Pode-se acessar o DOM, além de scripts, através da importação de bibliotecas em ferramentas de programação, como o Delphi. Se instalada a API (*Application Programming Interface*) – interface de programação de aplicativos - do DOM no computador, pode-se fazer a importação. Assim, converte-se a API em um arquivo Delphi.

Já o SAX, [SAX 03], é um estilo de interface bem diferente do DOM. Com o DOM, sua aplicação “pergunta” o que está no documento ao seguir uma referência de objeto na memória; com SAX, o analisador diz à aplicação o que está no documento ao notificar a aplicação de um fluxo de eventos analisadores. SAX significa “Simple API for XML”. SAX é uma interface que permite que se escrevam aplicações para ler dados contidos em um documento XML.

[MOU 00] compara o DOM com o SAX, tirando a seguinte conclusão: Enquanto que o DOM é muito mais eficiente quanto à funcionalidade, o SAX necessita de bem menos memória para ser acessado. Detalhes do *Document Object Model* podem ser encontrados em [W3C 03c].

2.1.8 Considerações

XML é um padrão emergente para a web e a tendência é que, cada vez mais, sua utilização tornar-se-á evidente. Apesar do grande esforço em pesquisa na busca de soluções XML para diversas aplicações, as ferramentas de suporte e autoria, geralmente não satisfazem soluções específicas. Com toda a extensibilidade que XML provê através do desenvolvimento de tags específicas para cada aplicação, torna-se muito difícil o desenvolvimento de alguma solução genérica para a manipulação e autoria de dados em formato XML.

Além disso, a comunidade da web ainda não utiliza XML para a estruturação de conteúdos de forma integral. Muitas aplicações ainda fazem uso de HTML e outras tecnologias para armazenamento e visualização de dados, tornando a interpretação e o compartilhamento destas informações muito pobres. Uma mudança de cultura na web será necessária de forma a encontrar esta integridade gerando uma web com conteúdos mais inteligentes e com agentes de softwares com capacidades mais eficientes e robustas de manipulação de dados.

Embora esta linguagem possua um alto grau de estruturação de conteúdos, ainda torna-se ineficiente para a preservação da semântica da informação. Muitas pesquisas estão surgindo atualmente em torno da *"SemanticWeb"*, que busca a integração de XML com outras tecnologias que descrevam estes metadados, preservando uma informação mais detalhada dos dados, tornando-os interpretáveis pela máquina.

2.2 XSL – eXtensible Style Sheet

2.2.1 Introdução

Enquanto XML é muito eficiente para a estruturação de dados, torna-se difícil tal estruturação e representação destas informações ao mesmo tempo. De fato, esta é uma das principais razões para o uso desta linguagem: a separação de dados da formatação permite uma produção de documentos muito mais efetiva e eficiente através de dados reutilizáveis [CAG 01]. Mesmo assim, existe uma grande necessidade que tais dados sejam formatados para visualização em diversos meios, como, por exemplo, *browsers* (HTML). Para suprir esta necessidade, será necessária a transformação dos dados em algum formato legível pelos seres-humanos. Enquanto que XML [W3C 03d], baseado em formato texto, é "legível", com certeza seu formato não é o mais adequado para aplicações em que a renderização dos documentos é prioritária [W3C 03j].

Composta por uma sintaxe baseada em XML, XSL, sigla de *eXtensible StyleSheet Language,* é definida como a linguagem de estilos para a XML [W3C 03j]. Enquanto que CSS (*Cascading Style Sheets*) [W3C 03a] é a tecnologia de estilos para formatar documentos HTML [W3C 03e], XSL foi projetada para trabalhar com XML. Tornando-se uma recomendação da W3C [W3C 03m] em novembro de 1999 [W3C 03j], esta linguagem vem prover mecanismos para formatar, transformar e manipular dados em XML. À medida que um documento XML contém informações apenas sobre o conteúdo, torna-se necessária a utilização de mecanismos externos para fornecer a formatação deste documento. XSL surgiu com o objetivo de fornecer tal mecanismo, mas, contando com recursos tão sofisticados, acabou destinando-se a propostas bem mais ambiciosas. Através de uma folha de estilos nesse formato, é possível não apenas uma formatação dos dados, mas, também, qualquer manipulação com essas informações, como, por exemplo, ordenação, transformação em outros formatos, filtragem de dados, etc [CAG 01].

Segundo [CAG 01], a linguagem de estilos da XML é dividida em três partes : XSLT (*eXtensible StyleSheet Language Transformation*) [W3C 99d], XPath (*XML Path Language*) [W3C 99c] e XSL-FO (*XSL Formating Objects*) [W3C 03j]. XSLT é a tecnologia mais utilizada e divulgada que, juntamente com Xpath, fornece recursos avançados para transformar e manipular documentos [CAG 01]. Enquanto que XSLT compõe um conjunto de modelos (*templates*) para manipular e transformar cada nó

XML, Xpath, viabiliza a localização destes nós, ou seja, fornece uma linguagem de expressões para indicar qual elemento será aplicado num determinado contexto, bem como a forma de encontrá-lo. Já a linguagem XSL-FO, é um vocabulário de formatação baseado em XML cujo objetivo é renderizar conteúdos em diferentes dispositivos e formatos. Um documento em XSL-FO pode ser renderizado em diversos meios, como, por exemplo, em PDF. XSLT e XSL-FO devem trabalhar juntas para a obtenção deste resultado. Partindo de uma base de dados em XML, utiliza-se uma folha de estilos XSLT para transformá-lo em um documento XSL-FO, para então, através de um processador apropriado, ser renderizado no formato desejado. Ocasionalmente, XSLT não é a ferramenta mais efetiva para o uso de transformações. Em tais situações, utiliza-se tecnologias alternativas que são interfaces programáticas, como, por exemplo o DOM e o SAX [W3C 03m].

Como mencionado acima, uma das grandes vantagens da XML é a capacidade de separação dos dados da formatação, que, juntamente com XSL, provê excelentes mecanismos de manutenção e produção de documentos dinâmicos. Esta poderosa linguagem de estilos permite a renderização dos mesmos dados em diversos meios. Contando com diversas visualizações das mesmas informações, torna-se clara a fácil manutenção desses documentos.

2.2.2 O Modelo da Linguagem XSL

Conforme [CAG 01], uma folha de estilos XSLT consiste em uma série de um ou mais *templates*, juntos com instruções baseadas em expressões Xpath que informam a um processador XSLT como encontrar os elementos em um documento de entrada XML. Para cada *template*, o processador, através de uma expressão Xpath, busca o elemento XML correspondente no documento de entrada e, então, aplica as transformações correspondentes. O resultado de todas as transformações aplicadas em um documento é conhecido como *result document.* Um exemplo da sintaxe XSLT pode ser encontrado na Figura 2.15. A representação gráfica do modelo da linguagem XSL pode ser observado na Figura 2.14.

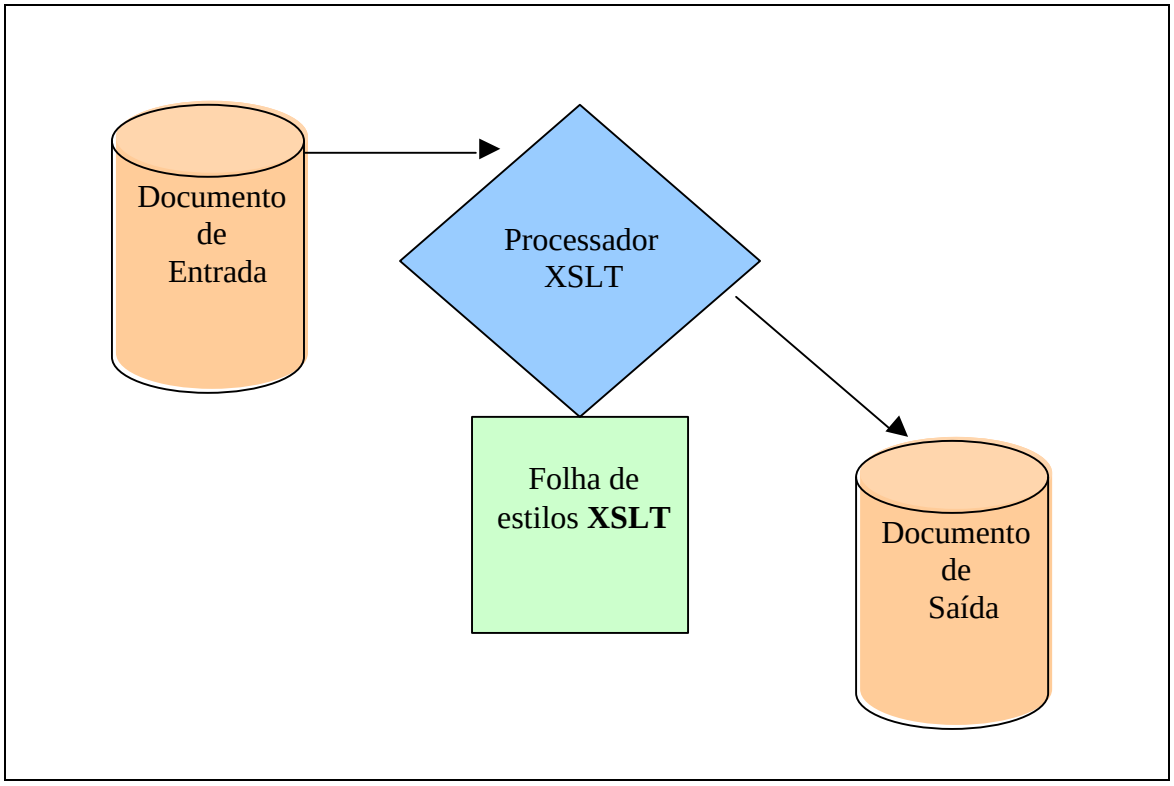


Figura 2.14 XSL – Modelo da Linguagem XSL.

```
<?xml version='1.0'?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

    xmlns:msxml="urn:schemas-microsoft-com:xslt"

    xmlns="http://www.w3.org/2000/svg-20000303-stylable"

    xmlns:math="urn:schemas-cagle-com:math"

    exclude-result-prefixes="math msxml"

    version="1.0">

    <xsl:output method="xml" media-type="image/svg" omit-xml-
declaration="yes" indent="yes"/>

    <xsl:template match="ITEM">

        <xsl:variable name="cx" select="floor(1000 div
count(../ITEM))"/>

        <xsl:variable name="cy" select="floor( $valor* 1000 div
$maxValue )"/>

        <xsl:variable name="valoritem" select="PORCENTAGEM"/>

      <g transform="translate({position()*$cx - $cx} {1000-$cy})">

      <xsl:choose>
```

```
        <xsl:when test="$valoritem = $maxValue">

            <rect width="{$cx}" height="{$cy}"
style="fill:url(#hotGradient);stroke:black;stroke-width:2;"/>

        </xsl:when>

          <xsl:when test="$valoritem = $minValue">

            <rect width="{$cx}" height="{$cy}"
style="fill:url(#coldGradient);stroke:black;stroke-width:2;"/>

        </xsl:when>

        <xsl:otherwise>

            <rect width="{$cx}" height="{$cy}"
style="fill:url(#boxGradient);stroke:black;stroke-width:2;"/>

          </xsl:otherwise>

        </xsl:choose>

        <text style="color:black;font-size:40;" y="-10"><xsl:value-of
select="PORCENTAGEM"/></text>

        <g transform="translate({$cx div 3} {$cy + 24})">

        <g transform="rotate(30)">

        <text style="color:black;font-size:30;">

            <xsl:value-of select="@TITULO"/>

        </text>

        </g>

        </g>

        </g>

    </xsl:template>

        </xsl:stylesheet>
```

Figura 2.15 XSL – Exemplo da sintaxe XSLT.

Outros exemplos de folhas de estilos XSLT podem ser encontradas no Anexo X.

Como já foi mencionado, para cada *template* é gerado um *result document* formando um conjunto de *results* que representa o documento final gerado pela transformação. Este documento pode estar em vários formatos, como XML, HTML, ou até mesmo XSL. O fluxo de processamento de uma folha de estilos aplicada em uma fonte XML (*input tree*) pode ser expressado através da Figura 2.16.

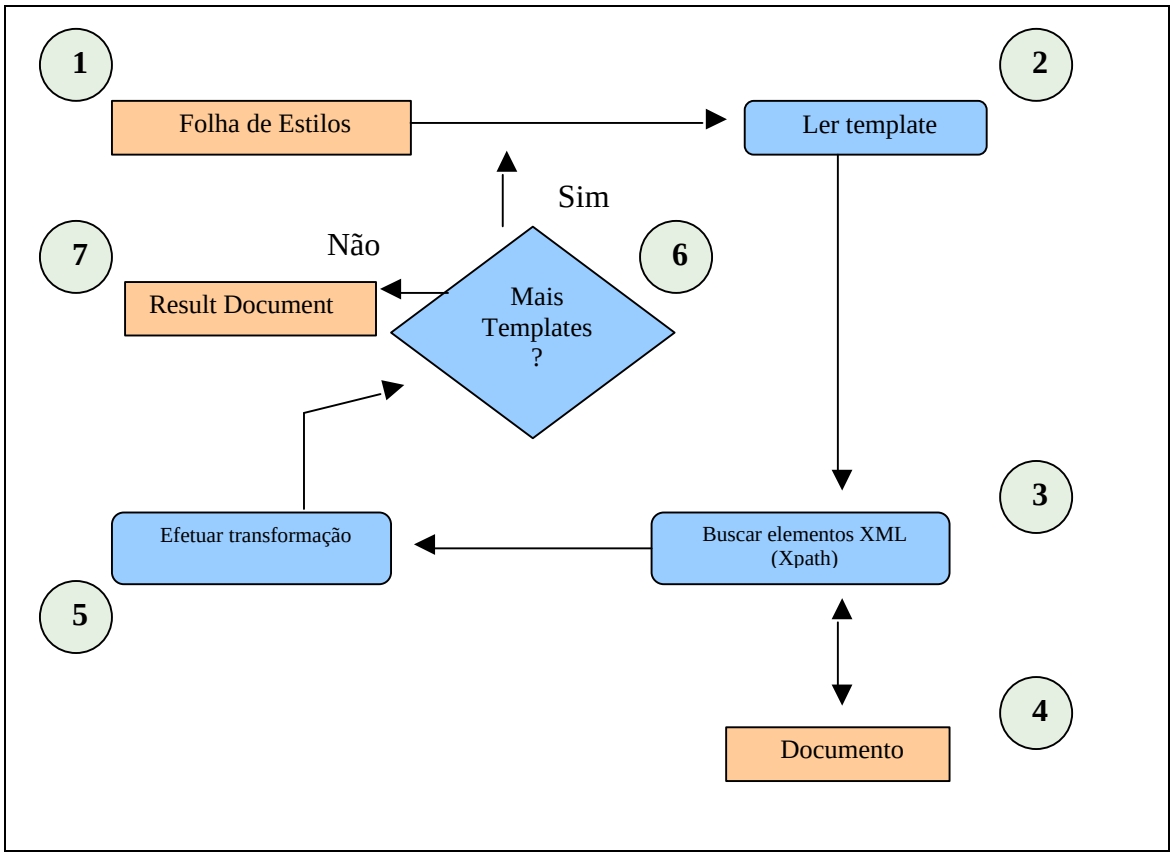


Figura 2.16 XSL - Fluxo de Processamento de uma folha de estilos.

O processo inicia-se com a aplicação de uma folha de estilos (1) a um determinado documento XML (4). O processador XSL, então, efetua a leitura dos *templates* (2) e, em seguida, faz a busca do elemento solicitado através de expressões Xpath (3). Esta busca é realizada no documento XML (4). Com o elemento encontrado, a transformação é realizada (5). O processo continua repetidamente enquanto existirem mais *templates* (6). O processo finaliza quando toda a folha de estilos (1) for processada. Finalmente o *Result Document* é entregue para o usuário (7).

2.2.3 Considerações

XSL é uma linguagem muito poderosa com recursos avançados de manipulação e transformação de dados em XML. Sua utilização torna-se cada vez mais importante, à medida que a tecnologia XML torna-se mais difundida. A grande problemática da linguagem XSL é a complexidade de sua sintaxe, tornando-se utilizada de forma avançada, apenas por usuários experientes. Com isso, sua divulgação em torno dos usuários e desenvolvedores web torna-se precária. A justificativa para tanta complexidade diz respeito ao fato de que a sua sintaxe é direcionada à manipulação dos dados em XML de forma genérica. Assim, qualquer vocabulário XML, independente de sua estrutura, poderá ser manipulado por uma folha de estilos XSL projetada para tal tarefa. Apesar de uma suposta linguagem de estilos, sem tanta generalidade, ser muito mais simples, embora menos poderosa, muitas aplicações poderiam fazer um uso facilitado, sem a necessidade de conhecimento de tanta complexidade encontrada em XSL.

2.3 MathML – Mathematical Marckup Language

2.3.1 Introdução

Uma característica diferencial da matemática é o uso de um sistema complexo e altamente evoluído de notações simbólicas bi-dimensionais. Matemática e sua notação não devem ser vistas como a mesma coisa. Idéias matemáticas existem independentemente das notações que as representam. Entretanto, a relação entre significado e notação é sutil, e parte da potência da matemática para descrever e analisar deriva da sua habilidade em representar e manipular idéias em uma forma simbólica. O desafio em colocar a matemática na Web resume-se na captura de notação e conteúdo (isto é, significado) em uma forma que documentos possam utilizar os formatos notacionais altamente evoluídos de escrita e impressão, e o potencial para a interconectividade em meio eletrônico [CLA03].

Notações matemáticas estão constantemente evoluindo à medida que pessoas continuam a fazer inovações no que diz respeito à aproximação e expressão de idéias. A notação matemática moderna é o produto de séculos de refinamento, e as convenções nacionais para a composição de conteúdo ainda são tarefas complicadas. Por exemplo, variáveis e letras que sutilmente diferem do texto usual itálico. Espaçamento entre símbolos para operações como “+,-,*,/” é levemente diferente daquele texto, para refletir convenções sobre precedência de operador. Livros completos têm sido dedicados para às convenções de composição matemática, desde o alinhamento de sobre-escritos e sub-escritos até regras para escolher tamanhos de parênteses e práticas notacionais especializadas para subcampos da matemática.

Entretanto, existe mais inserção de textos matemáticos na web do que meramente a busca de formas de visualização da notação matemática tradicional em um *Browser web* [DES 03b]. A Web representa uma alteração fundamental na metáfora subjacente para armazenamento de conhecimento, uma alteração em que a interconectividade se torna um papel importante. Isto está tornando-se crescentemente importante para encontrar formas de comunicação matemática que facilite o processamento automático, pesquisa e indexação, e reusabilidade em outras aplicações matemáticas e contextos. Com este avanço da tecnologia de comunicação, existe uma oportunidade da expansão da nossa habilidade de representar, codificar, e, ultimamente,

comunicar nossa percepção matemática e entendê-la. Acredita-se que a MathML é um passo muito importante no desenvolvimento matemático na Web.

A linguagem MathML (*Mathematical Markup Language*) [W3C 03f] é um aplicativo XML para descrever notação matemática e capturar sua estrutura e conteúdo. A meta do MathML é disponibilizar matemática para ser processada através da *World wide Web* (www), da mesma forma que a HTML tem disponibilizado esta funcionalidade para textos em geral.

Segundo [W3C 03f], a MathML pode ser usada para criar fórmulas matemáticas, bem como para representar equações matemáticas avançadas, como polinômios, cálculos e equações geométricas e trigonométricas na Web. A MathML possui dois conjuntos de elementos de marcação, um para conteúdo e outro para apresentação : *elementos de conteúdo* e *elementos de apresentação*..

Dada a seguinte equação : $x^2 - 3x + 2$, o código MathML correspondente pode ser observado nas Figuras 2.17 e 2.18.

* Para a marcação de conteúdo

```
<apply>
  <apply>
    <power/>
    <ci>x</ci>
    <cn>2</cn>
  </apply>
  <apply>
    <plus/>
    <apply>
      <times/>
      <cn>3</cn>
      <ci>x</ci>
    </apply>
```

```
    <cn>2</cn>

</apply>
```

Figura 2.17 MathML – Exemplo de marcação de conteúdo.

* Para a marcação de apresentação :

```
<math displaystyle='true'>

   <semantics>

      <mrow>

         <msup>

            <mi>x</mi>

            <mn>2</mn>

         </msup>

         <mo>-</mo>

         <mn>3</mn>

         <mi>x</mi>

         <mo>+</mo>

         <mn>2</mn>

      </mrow>

   </semantics>

</math>
```

Figura 2.18 MathML – Exemplo de marcação de apresentação.

2.3.2 Considerações

Conteúdos matemáticos estruturados na web trazem soluções altamente promissoras, como pesquisa, produção de dados dinâmicos e processamento em tempo real. MathML é o padrão mundial de estruturação do universo matemático para a divulgação na web. Torna-se cada vez maior a utilização desta tecnologia por grandes empresas e, também, por muitos usuários finais. Mesmo assim, sua utilização em nível mundial é muito pequena, quando comparada com outras tecnologias emergentes. Um fator de grande importância para este fato é a atual carência de ferramentas de autoria de qualidade e não proprietárias. Embora a tendência seja uma web com conteúdos matemáticos inteligentes baseados em MathML, com uma alta disponibilização de ferramentas de autoria e de qualidade, a evolução desta linguagem ainda é muito lenta.

2.4 SVG – Scalable Vector Graphics

2.4.1 Introdução

SVG – *Scalable Vector Graphics* [W3C 03h], é uma liguagem para a descrição de gráficos de duas dimensões em XML (*eXtensible Markup Language*). SVG permite três tipos de objetos gráficos: gráficos vetoriais, imagens e texto. Os objetos gráficos podem ser agrupados, definidos por estilos e compostos em objetos renderizados anteriormente. Texto pode estar em qualquer *namespace* XML que se adapte à aplicação, melhorando as possibilidades de pesquisa e acessibilidade dos gráficos SVG.

Objetos criados em SVG podem ser dinâmicos e interativos. O DOM – *Document Object Model* [W3C 03c], para o SVG, permite animação direta e eficiente dos gráficos via *script.* Um rico conjunto de eventos manipuladores, como *onmouseover* e *onclick,* pode ser atribuído a qualquer objeto SVG [GEO 00].

2.4.2 Sintaxe do SVG

2.4.2.1 Formas Básicas

SVG contém o seguinte conjunto de formas básicas:

* rectangles;
* circles;
* ellipses;
* polylines;
* polygons

Algumas destas formas serão mostradas a seguir:

* Elemento *rect*

a. Definição :

O elemento “rect” define um retângulo alinhado com um sistema de coordenadas apropriado. Os retângulos são construídos definindo-se valores apropriados para os atributos *rx* e *ry*.

b. Exemplo :

A Figura 2.19 mostra um exemplo e a sua representação gráfica é apresentada na Figura 2.20.

```
<svg width="12cm" height="4cm" viewBox="0 0 1200 400">
    <rect x="1" y="1" width="1198"   height="398"
        fill="none" stroke="blue" stroke-  width="2"/>
  <rect x="400" y="100" width="400" height="200"
        fill="red" stroke="navy" stroke-width="10"  />
</svg>
```

Figura 2.19 SVG – Exemplo de código SVG para o elemento *rect*.

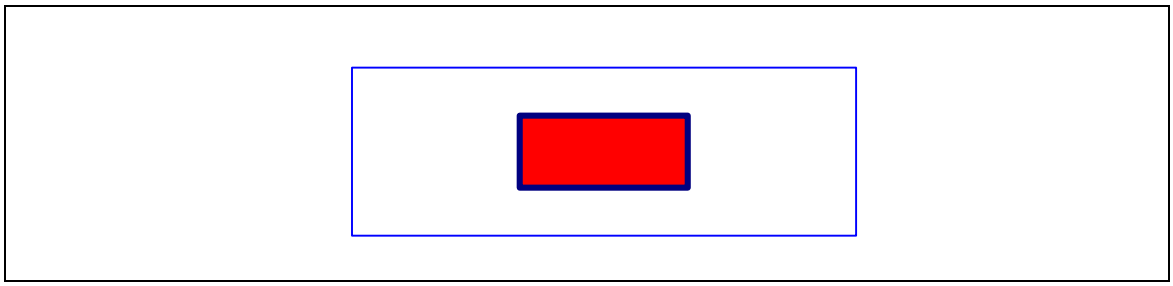


Figura 2.20 SVG – Exemplo de visualização SVG para o elemento *rect*.

* Elemento *circle*

a. Definição :

Para criar um círculo é necessária a definição do elemento *circle* com seus atributos *cx*, *cy* e *r* (eixo x, y e raio, respectivamente).

b. Exemplo :

Um exemplo de código SVG para o elemento *circle* é ilustrado na Figura 2.21. A visualização em SVG correspondente é apresentada na Figura 2.22.

```
<svg width="12cm" height="4cm" viewBox="0 0 1200 400"
     xmlns="http://www.w3.org/2000/svg" version="1.1">
  <rect x="1" y="1" width="1198" height="398"
        fill="none" stroke="blue" stroke-width="2"/>
  <circle cx="600" cy="200" r="100"
        fill="red" stroke="blue" stroke-width="10"  />
</svg>
```

Figura 2.21 SVG – Exemplo de código SVG para o elemento *circle*.

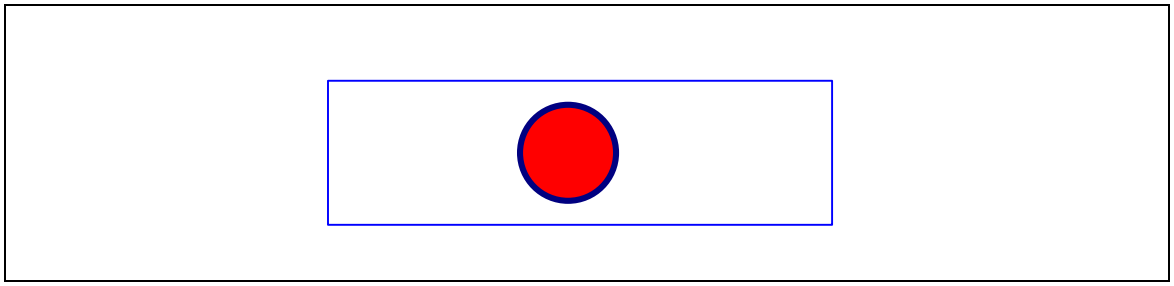


Figura 2.22 SVG – Exemplo de visualização SVG para o elemento *circle.*

SVG possui muitos outros elementos que não serão citados aqui. O objetivo principal desta seção é fornecer apenas uma idéia básica da sintaxe SVG. A questão fundamental é que, através de elementos básicos como os citados acima, torna-se possível compô-los para formar elementos complexos como, por exemplo, gráficos estatísticos (gráficos de pizza, de barra ou de pontos). SVG também permite interatividade, animação e a possibilidade de "embutir" elementos de outras linguagens para a formação documentos híbridos [W3C 03h].

2.4.3 Compondo Elementos SVG

Conforme citado, através dos elementos básicos SVG, pode-se criar elementos complexos com a composição das suas formas básicas. Abaixo pode-se observar um gráfico complexo, formado pela composição de elementos como retângulos, linhas e texto. A visualização deste gráfico está na Figura 2.23.

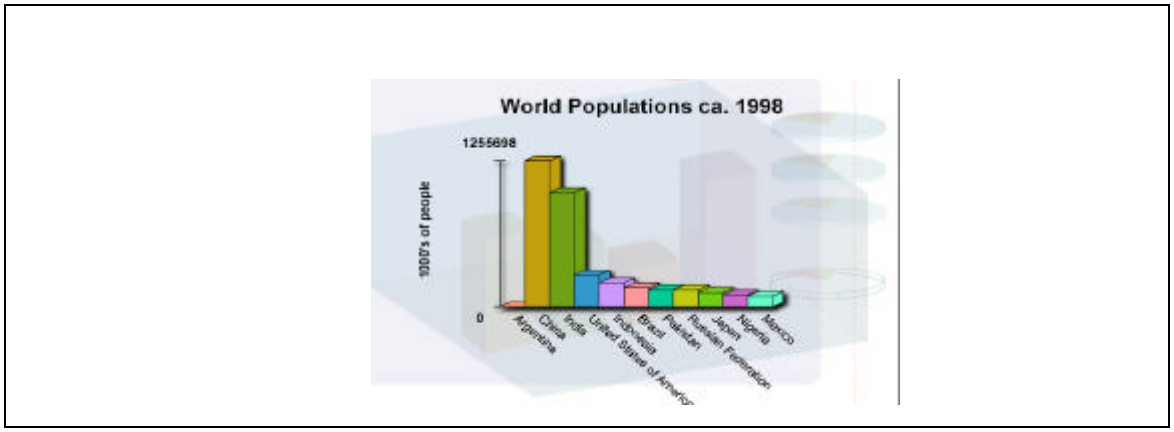


Figura 2.23 SVG – Compondo elementos SVG.

O código em formato SVG necessário para a composição do gráfico exposto na Figura 2.23 pode ser obtido em [ADO 03b].

2.4.4 Considerações

O futuro da linguagem SVG talvez seje o mais promissor dentre as tecnologias utilizadas neste trabalho. Muitos autores o referenciam como o futuro da web no que diz respeito a uma linguagem gráfica, substituindo todos os padrões e formatos de imagens e animações hoje existentes. SVG vem crescendo muito atualmente e muitas empresas o estão adotando-o. Surgem cada vez mais ferramenas de autoria e manipulação de dados neste formato. Mesmo com tanta divulgação e crescimento, por ser baseado em XML, suas aplicações tornam-se muito extensíveis, gerando uma grande dificuldade ao desenvolvimento de ferramentas que consigam suprir todas as suas possíveis utilizações. Muitos mecanismos de integração SVG com outras tecnologias baseadas em XML podem ser desenvolvidos, gerando soluções diferenciadas onde as ferramentas já existentes provavelmente não conseguirão adaptar-se. SVG traz muitas possibilidades para o desenvolvimento de soluções integradas e, também, para o desenvolvimento de ferramentas de autoria e manipulação de dados neste formato.

2.5 Uma breve introdução à semântica na web

2.5.1 Web Semântica

A Web Semântica é um espaço de informação conceitual em que os recursos podem ser processados por máquinas. Opera sobre os princípios de “entendimento parcial” e “inferência” (tornando-se capaz de inferir novos conhecimentos de termos a partir de dados já entendidos) [SEA 01].

A World Wide Web foi originalmente construída para concepção humana, entretanto ela deve ser legível para as máquinas também. É muito difícil automatizar o processamento de documentos na Web e, pelo volume de informações que ela contém, não é possível gerenciá-los manualmente [W3C 99b].

Em grande parte, estes documentos são identificados por URIs (*Uniform Resource Identifier*). Assim, os sistemas podem crescer em uma base globalmente descentralizada, similar a sistemas de documentação de hipertextos na web passada.

Sendo identificado em uma URI, um documento, também identificado como um recurso, pode ser referenciado globalmente, ou seja, por qualquer sistema. Assim, relações complexas podem ser construídas, consultadas e processadas. A meta da web semântica é que todos os dados sejam armazenados seguindo estruturas padrões e relações hierárquicas entre recursos que possam ser compartilhados de uma maneira padronizada universalmente. Assim, todos os dados (recursos) em qualquer sistema poderiam ser facilmente legíveis e utilizados. A semântica nos dados da web traria uma web com conteúdos inteligentes, reutilizáveis e integrados.

Tim Berners-Lee [BER 01] afirma que a “*Semantic Web*” é composta por três componentes : XML, Ontologias e RDF [W3C 99b].

Acredita-se que a integração destas três tecnologias viabilizará o encontro da “Semântica na Web”, seguindo o seguinte modelo : enquanto que XML fornece estrutura sintática aos documentos, descrevendo os dados em um determinado domínio, RDF oferece a descrição destes metadados permitindo o desenvolvimento de relacionamentos entre os diversos recursos definidos pelos metadados XML. A definição de uma ontologia para um domínio particular vem oferecer a especificação de um vocabulário padrão para ser utilizado e compartilhado de forma universal.

Por não fazer parte do conjunto de teconlogias utilizadas, a linguagem RDF não será mencionada neste trabalho. Maiores informações sobre esta linguagem podem ser obtidas em [W3C 99b].

2.5.2 Ontologias

Para Gruber [GRU 93], uma ontologia é uma especificação formal e explícita de uma conceitualização compartilhada. Fensel [FEN 00] descreve esse conceito em partes, afirmando que uma "conceitualização" refere-se para um modelo abstrato de algum fenômeno no mundo que identifica conceitos relevantes daquele fenômeno. Guarino [GUA 97] ainda comenta que uma "conceitualização" explica o significado pretendido dos termos usados para indicar relações relevantes. "Explícito" significa que os tipos de conceitos usados e as restrições para esses conceitos são definidos explicitamente. "Formal" refere-se para o fato de que uma ontologia deve ser legível para as máquinas. "Compartilhada" reflete a noção de que uma ontologia captura o conhecimento consensual, isto é, o conhecimento não é restrito a algum indivíduo, mas aceito por um grupo. Para a ciência Inteligência Artificial e os pesquisadores Web, uma ontologia é um documento ou arquivo que formalmente define as relações entre os termos. O tipo mais comum de ontologia para a Internet tem uma taxonomia e um conjunto de regras de inferência [BER 01]. A taxonomia define as classes de objetos e as relações entre eles.

Na prática, uma outra maneira de abordar o conceito de ontologia é através de um conjunto de sentenças que descrevem a estrutura de um domínio de interesse, dando o significado aos termos identificados no domínio. Com um vocabulário comum assim definido é possível o intercâmbio inteligente de informações entre agentes (humanos ou automáticos) que concordam com a ontologia.

Como as ontologias são teorias formais que descrevem um domínio, elas requerem uma linguagem lógica para representá-las. Na área da Inteligência Artificial tem-se desenvolvido muitas linguagens, algumas baseadas em lógica de predicados de primeira ordem, outras baseadas em *frames* com expressivas primitivas de modelagem, como Ontolingua, e outras tendo como principal objetivo a robustez na inferência. Recentemente, os padrões da Web, como XML e RDF [W3C 99b], estão atraindo o interesse da comunidade científica na criação e uso de linguagens de representação de ontologias. O uso

destes padrões permite que a ontologia possa residir na Web, além da vantagem de aproveitar as ferramentas já desenvolvidas e testadas para esses padrões.

Assim, surge o RDF Schema [W3C 99b], uma linguagem de poucas primitivas de representação e muita portabilidade, que permite representar ontologias simples. É possível compartilhar especificações RDF Schema na Web através de sua sintaxe em XML. Sobre o RDF Schema tem-se construído linguagens com primitivas mais expressivas e com capacidade de inferência, como, por exemplo, DAML [DAML 01].

2.6 Conclusão

Este capítulo fez uma abordagem sobre as linguagens destinadas a fornecer estrutura aos documentos na Web. Tecnologias como XML, XSL, MathML e SVG foram descritas em detalhes. Também uma breve introdução à semântica na web foi evidenciada. O próximo capítulo, Capítulo 3, refere-se ao framework proposto neste trabalho, “Um FrameWork para a estruturação e reutilização de Hipermídias”.

Capítulo 3

Um framework para a estruturação e reutilização de Hipermídias

O framework proposto neste trabalho será apresentado através de uma "Metodologia para a Reutilização de Conteúdos", Seção 3.1. A capacidade de integração deste framework será demonstrada através de uma "Metodologia para a Autoria de Conteúdos em formato MathML", Seção 3.2, e uma "Metodologia para a Autoria de Gráficos dinâmicos e interativos em formato SVG", Seção 3.3.

3.1 Metodologia para a Reutilização de Conteúdos

3.1.1 Introdução

Esta seção irá propor uma estratégia de produção de conteúdos estruturados, reutilizáveis e inteligentes o suficiente para serem utilizados em diversas aplicações. A reusabilidade e estruturação dos conteúdos é um tópico de grande importância pois traz grandes vantagens para o gerenciamento, produção e intercâmbio de documentos na web.

Atualmente, o grande problema de difusão da informação através da web é a falta de um modelo padrão para o intercâmbio de dados. Muitas pesquisas tem sido feitas na área de "Web Semântica" [W3C 03i], onde procura-se adicionar informações semânticas aos dados na web. Assim, tais informações tornam-se mais inteligentes (no que diz respeito à interpretação feita pelas máquinas) e reutilizáveis. XML pode resolver muito os problemas de padronização de intercâmbio de dados na internet. Muitas aplicações baseadas nesta tecnologia são capazes de reutilizar documentos, trocar informações e, dentre outros, obter diversas interpretações diferentes de uma mesma base de dados.

No que diz respeito à semântica e estruturação da informação na web, novas formas de modelagem de dados tornam-se necessárias. Os dados precisam ser organizados e modelados em uma forma que possibilite a reutilização e intercâmbio desses conteúdos. Para tanto, uma introdução ao conceito de objetos de dados

reutilizáveis, ou dados inteligentes, será realizada com o objetivo de substituir as idéias existentes por novas formas de pensar e modelar informações para a divulgação através da web. Assim, muitas estratégias de produção de documentos serão evidenciadas, mostrando como esse novo paradigma de produção poderá ser útil para a confecção de material para a web.

3.1.2 Uma arquitetura para a Web Semântica

3.1.2.1 Introdução

Em um framework de dados estruturados, um documento não deve ser visto como um conjunto de dados específicos para uma determinada aplicação, mas sim como um conjunto de dados que compartilha informações contidas em diferentes **repositórios** de dados. O conteúdo destes repositórios deverá ser um conjunto de dados com informações semânticas permitindo que estas informações sejam interpretadas e manipuladas pelas máquinas através da web. Os dados finais, que serão entregues para o cliente, poderão fazer uso de vários repositórios armazenados em diferentes locais. Sua formatação (renderização) será específica para cada cliente, ou seja, a formatação será independente desses repositórios. A web poderia ser vista como um conjunto de repositórios de dados estruturados e compartilhados. A Figura 3.1 mostra esta idéia.

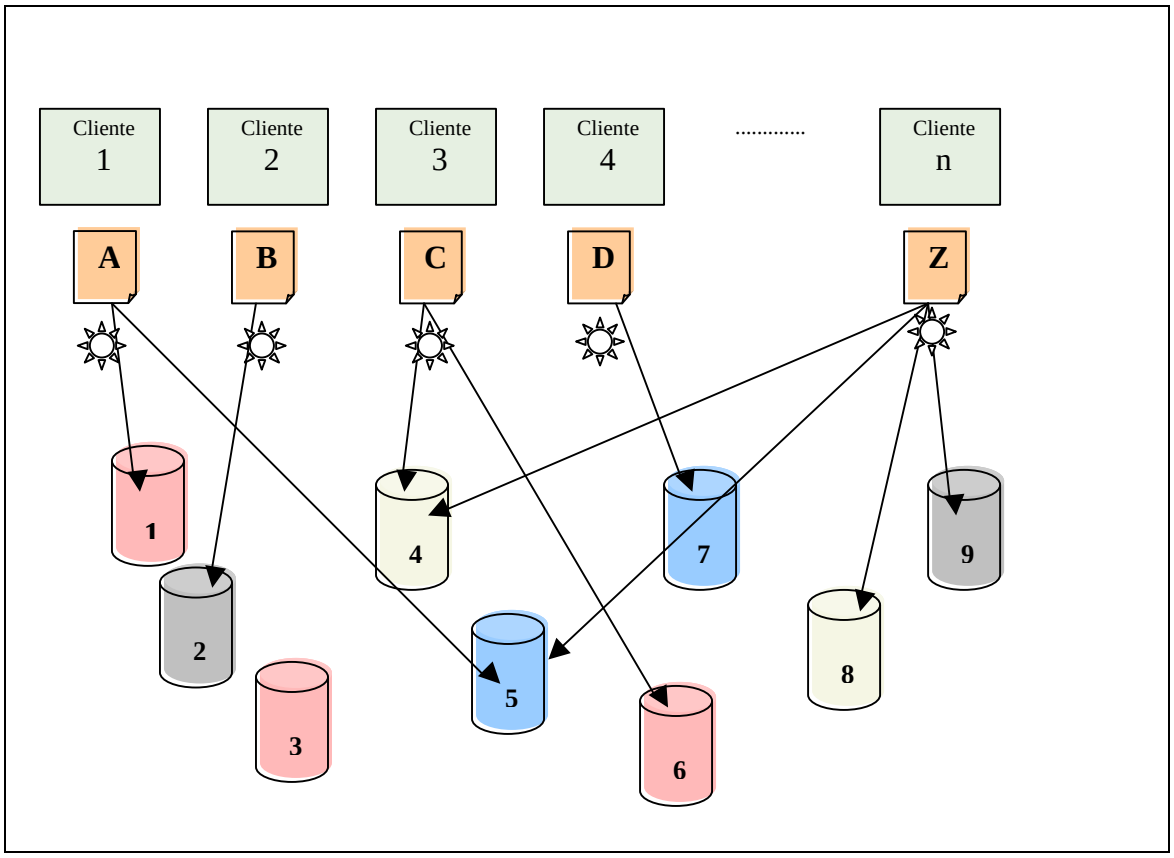


Figura 3.1 Arquitetura Web Semântica – Web como um conjunto de repositório de dados.

A Figura 3.1 ilustra a proposta de uma arquitetura para a web semântica. Esta arquitetura define que os seguintes objetos devem compor esta solução : um conjunto de **repositórios de dados** (representado pelos cilindros); um conjunto de **agentes de**

softwares (representados pelas estrelas) capazes de manipularem dados contidos em diferentes repositórios e gerarem **saídas formatadas** (representadas pelos retângulos A,B,..Z) que serão entregues para os **clientes** (retângulos verdes). Cada repositório deverá conter **dados fragmentados** com informações semânticas que serão detalhados na Seção 3.1.2.2.

Os dados contidos em cada repositório deverão ser específicos para um domínio particular e nenhuma informação relacionada à formatação visual deverá existir. A estrutura semântica de cada repositório deverá ser conhecida e interpretável pelos agentes que utilizam estas informações. Um documento web, que será entregue para um cliente, poderá compartilhar informações de diferentes repositórios. Na Figura 3.1, os cilindros numerados de 1 a 9 representam esses repositórios espalhados pela web. As estrelas representam agentes de softwares capazes de manipularem estes dados e gerarem saídas formatadas e compartilhadas para diferentes clientes.

As seguintes questões surgem no que diz respeito à implantação desta arquitetura :

* Como serão adicionadas informações semânticas para estes dados;
* Como estes dados deverão ser estruturados;
* Quais tecnologias viabilizarão a implantação desta proposta;
* Como os agentes de software serão capazes de manipularem os dados armazenados nos repositórios.

As seções que seguem mostram como as questões acima podem ser solucionadas. A Seção 3.1.2.2 refere-se à **semântica** e **estruturação** dos conteúdos. As estratégias de soluções através das **tecnologias** escolhidas serão apresentadas na Seção 3.1.3. Uma metodologia de autoria e implementação dos repositórios de dados é demonstrada na Seção 3.1.5. A Seção 3.1.4 propõe uma linguagem de Script capaz de prover a **manipulação** dos conteúdos dos repositórios.

3.1.2.2 Estruturação e Fragmentação de documentos

Como já foi citado anteriormente, cada repositório armazenará conteúdos de um domínio particular. Supõe-se que um determinado servidor contenha informações didáticas que são utilizadas para o ensino a distância. Vários repositórios serão necessários para armazenar, por exemplo, figuras, tabelas, fórmulas, gráficos, exemplos

e exercícios. A Figura 3.3 mostra como um documento destinado ao ensino a distância poderá compor uma combinação de dados desses diferentes repositórios.

A estruturação e fragmentação de documentos preocupa-se em como estes dados serão armazenados internamente. Determinadas estruturas deverão ser definidas e informações semânticas deverão ser adicionadas para que a manipulação e o compartilhamento de conteúdos tornem-se possíveis. Inicialmente, em um nível abstrato, uma aplicação particular deve ser refinada e os repositórios de dados devem ser definidos, conforme ilustração na Figura 3.2.

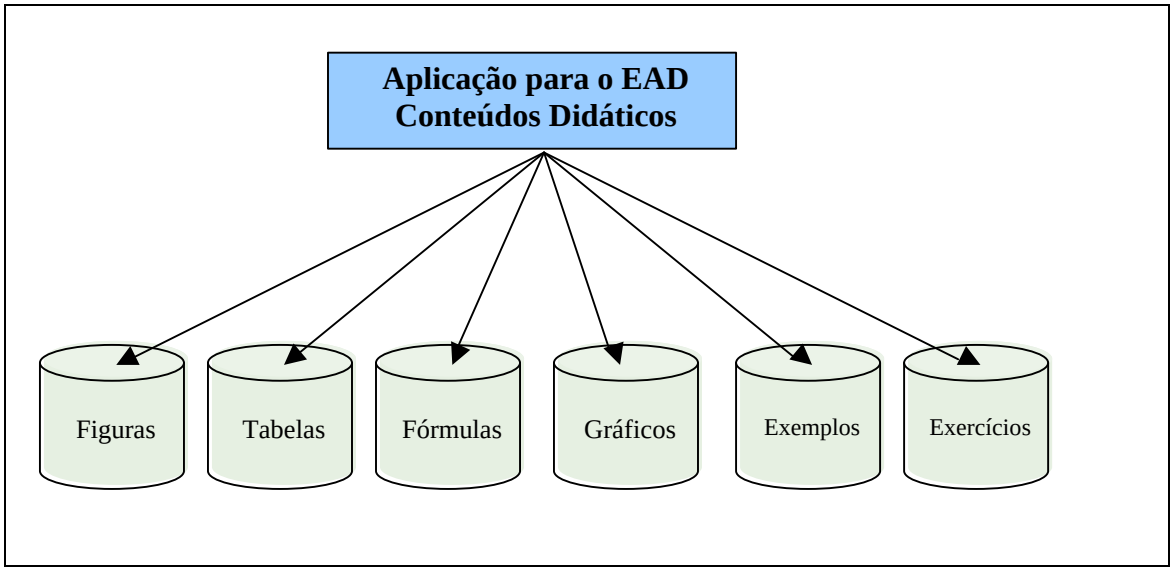


Figura 3.2 Arquitetura Web Semântica – Refinamento de uma aplicação.

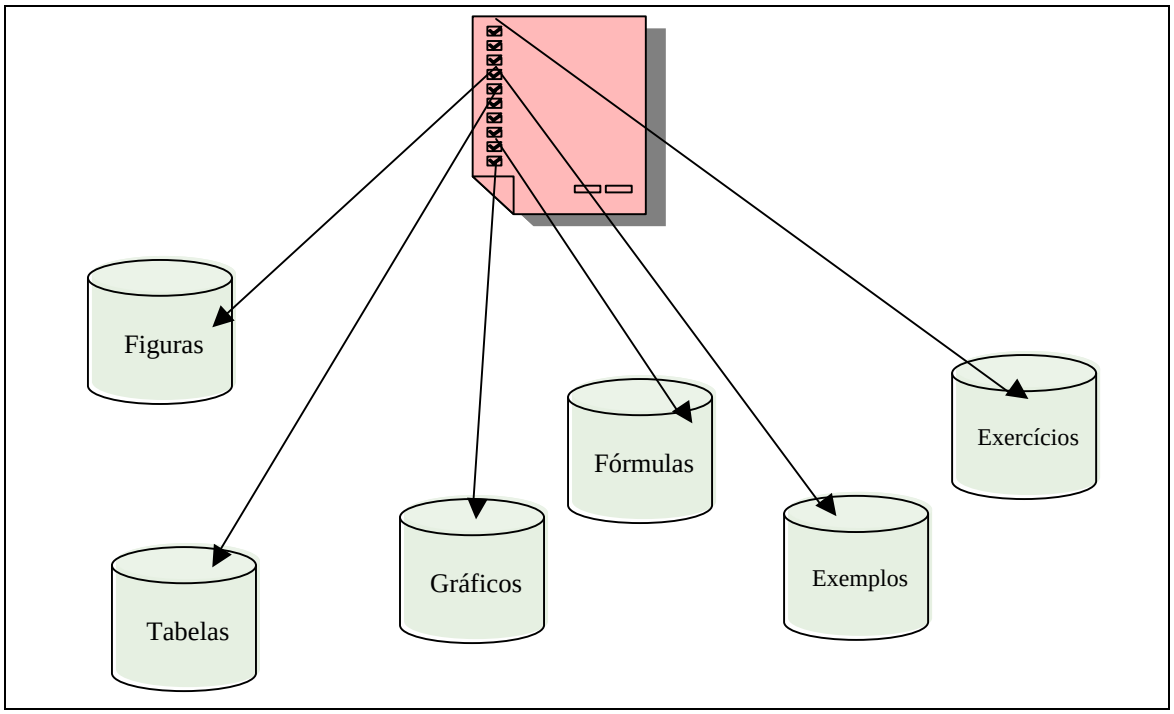


Figura 3.3 Arquitetura Web Semântica – Compartilhamento de repositórios de dados.

Cada repositório deverá conter uma estrutura específica que identifique o seu conteúdo. Esta estrutura deverá fornecer informações semânticas e fragmentar os dados de forma que estes possam ser manipulados independentemente. Assim, uma determinada aplicação poderá fazer uso de apenas um fragmento de um repositório. Portanto, cada repositório deverá ser refinado em um formato capaz de prover estas informações, conforme exposto na Figura 3.4.

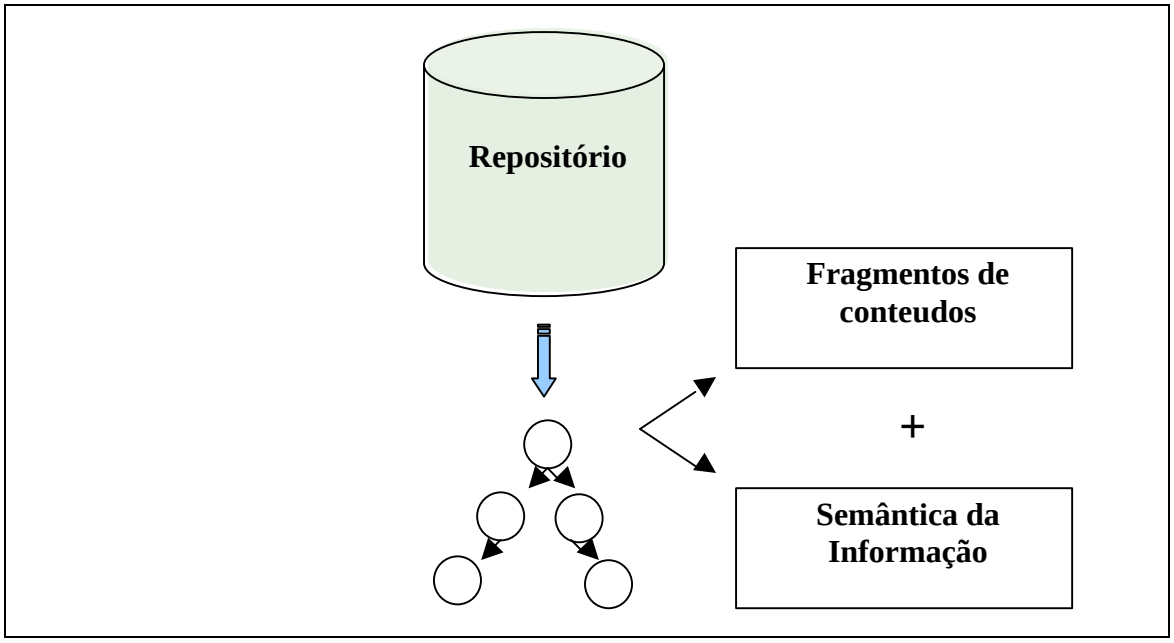


Figura 3.4 Arquitetura Web Semântica – Estrutura de um repositório.

Cada repositório deverá ser referenciado como uma lista de objetos. Cada objeto representará a informação armazenada neste repositório. Assim, um repositório de exercícios, por exemplo, armazena uma lista de exercícios, onde cada exercício é um objeto. A estrutura deste repositório será definida, então, partindo da estrutura destes objetos, no caso, exercícios. A Figura 3.5 ilustra esta composição de objetos.

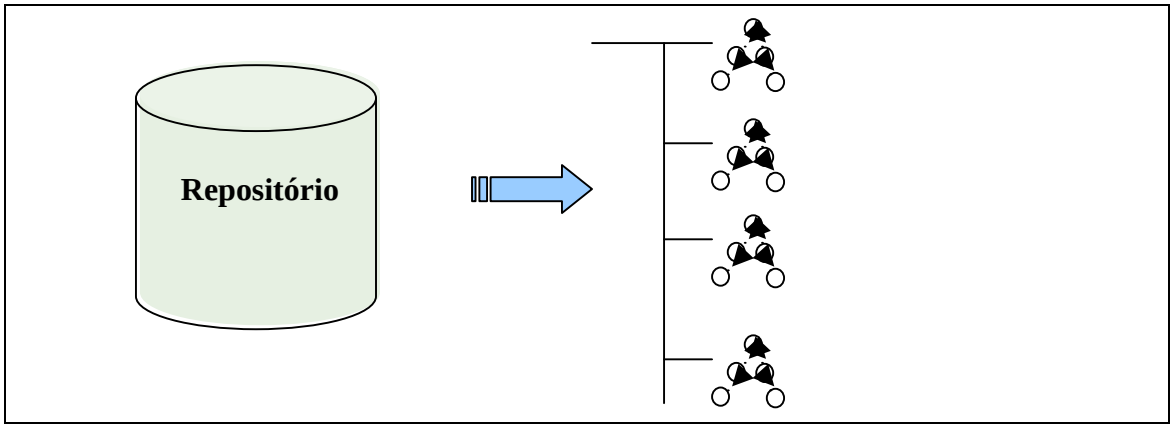


Figura 3.5 Arquitetura Web Semântica – Um repositório como uma lista de objetos.

A especificação da estrutura de um certo repositório de dados deve preocupar-se com as seguintes questões :

* Quais as informações que deverão ser armazenadas
* Como estas informações devem ser agrupadas em uma ordem hierárquica de contenção.

Por exemplo, um exercício poderá conter informações como título, descrição, resolução e resposta. A ordem hierárquica que estas informações devem ser agrupadas é ilustrada na Figura 3.6.

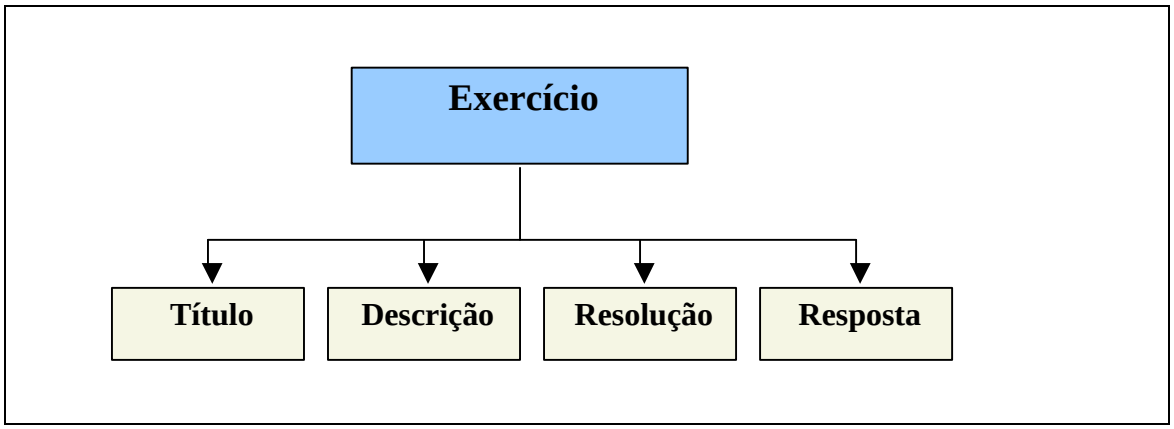


Figura 3.6 Arquitetura Web Semântica – Exemplo de estrutura de um exercício.

Em outras palavras, pode-se dizer que um exercício é composto pelos seguintes fragmentos :

* Título;
* Descrição;
* Resolução;
* Resposta.

Generalizando, um objeto é uma composição de um ou mais fragmentos de dados. Cada fragmento deverá ser capaz de ser identificado e manipulado independentemente por agentes de software para a produção de um documento composto. Finalmente, um documento apresentado ao cliente é a renderização particular de um conjunto de fragmentos armazenados em diferentes repositórios de dados.

3.1.2.3 A Reusabilidade da Informação

Um conceito de grande importância é o de “reutilização de documentos”. Por conteúdo reutilizável, entende-se um conjunto de dados que possa ter diferentes interpretações ao mesmo tempo, ou seja, por exemplo, vários formatos de visualização de uma mesma base de dados.

Todos os conteúdos devem ser produzidos e planejados de tal forma que estes possam ser reaproveitados de diferentes maneiras, garantindo que estes sejam editados e armazenados apenas uma vez. Supõe-se, em um documento didático, a existência de um exemplo, seja *“Exemplo 1.2”*. É importante a detecção de que este conteúdo será compartilhado por diferentes aplicações. Esse é o primeiro passo para a produção de documentos bem estruturados. Todos os outros conteúdos que farão uso do documento “*Exemplo 1.2”* utilizarão apenas uma *referência* para o mesmo. Esta estratégia repete-se para todos os conteúdos que deverão ser apresentados repetidamente em várias partes de um conjunto de informações. Supõe-se, agora, que, além da reutilização do *Exemplo1.2*, será necessário que este seja apresentado de maneiras diferentes, para cada aplicação, conforme o objetivo do documento que o está referenciando. Vários formatos de apresentação do mesmo conjunto de dados, no caso, o *“Exemplo 1.2”* poderão ser fornecidos.

Com todos os documentos, que possívelmente serão reutilizados e detectados, é necessária a detecção de quais desses conteúdos serão reutilizados um número de vezes grande o suficiente para que estes se tornem fragmentos reutilizáveis. Se o “*Exemplo 1.2”* fosse utilizado apenas 2 vezes no documento, talvez não fosse necessário utilizá-lo como um fragmento de conteúdo reutilizável. Questões como estas dependem muito da aplicação e do tipo de informação com que se está trabalhando. Também é necessária a definição, dentre os conteúdos que serão fragmentados, de quais necessitarão de uma interpretação diferente, como, por exemplo, um caso onde a aplicação de uma folha de estilos diferente torna-se-ia necessária.

Concluindo, com documentos estruturados, muitas vantagens são encontradas, como, por exemplo, fácil produção, manutenção e a utilização e compartilhamento de informações feito por diversas aplicações.

3.1.3 RCML – uma solução baseada em XML

3.1.3.1 Introdução

A linguagem XML será utilizada neste trabalho como tecnologia base para a modelagem proposta, pois é um formato que, atualmente, proporciona uma excelente estruturação dos dados na internet. Dependendo da aplicação, também será preservada a semântica da informação.

XML fornece estrutura ao documento através de *tags* bem planejadas, e também reutilização pois sua estrutura tem a capacidade de separar o conteúdo da formatação. Assim, será possível a existência de várias interpretações e/ou formatações da mesma base de dados.

RCML (*Reusable Marcup Language*), Linguagem de Marcação para reusabilidade de conteúdo, é o aplicativo XML proposto para a estruturação dos objetos contidos nos repositórios de dados. A Seção 3.1.3.2 mostra a estrutura desta linguagem e como esta pode ser utilizada para tornar as informações estruturadas e reutilizáveis.

3.1.3.2 A linguagem RCML

A RCML, *Reusable Marckup Language*, é capaz de agrupar os seguintes tipos de informações :Estilos, Recursos, Tabelas, Figuras, Fórmulas, Gráficos, Exemplos, Exercicios, Teoremas, Definicoes, Glossarios, Artigos, Arquivos, Bibliografias, Telas Gráficas, Applets, Flashs, Videos. Embora existam alguns objetos específicos para uma determinada ontologia (figuras e tabelas, por exemplo), o propósito da RCML é ser uma linguagem capaz de aplicar-se para qualquer domínio. Repositórios de dados para ontologias que não utilizem objetos como tabelas e figuras, devem utilizar objetos do tipo “Recursos”, cuja proposta é prover capacidades de generalização destas estruturas.

Para cada tipo de informação pertencente a esta linguagem, um conjunto de fragmentos foi definido. Conforme ilustrado na Figura 3.6, um objeto do tipo “Exercício” possui os fragmentos “Título”, “Descrição”, “Resolução” e “Resposta”. A estrutura RCML para descrever este objeto e seus fragmentos pode ser observada na Figura 3.7.

```
<Exercicio>
   <Titulo>     </Titulo>
   <Descricao>  </Descricao>
   <Resolucao>  </Resolucao>
   <Resposta>   </Resposta>
</Exercicio>
```

Figura 3.7 RCML – Estrutura RCML para o objeto “Exercicio”.

A descrição completa de todos os elementos suportados pela RCML e exemplos de sua utilização podem ser encontrados no Anexo V.

3.1.4 A Linguagem RCM Script

3.1.4.1 Introdução

Justificando-se na alta complexidade da linguagem XSL, e buscando prover forte capacidade de manipulação da informação de uma forma muito simples, este trabalho propõe uma linguagem de script para trabalhar juntamente com o modelo proposto, fornecendo capacidade de manipulação de dados aos agentes de softwares. A linguagem RCM *Script* foi projetada para, além de descrever transformações de conteúdos, descrever a reusabilidade da informação. Através desta linguagem reduz-se em grande parte a necessidade de utilização de folhas de estilos complexas, trazendo para junto dos dados toda a funcionalidade necessária de manipulação. Com isso, os documentos deixam de ser estáticos e passam a ser dinâmicos, pois, além de possuírem texto simples, possuem, também, comandos para gerar conteúdos. A sintaxe desta linguagem é totalmente baseada em XML, sendo, assim, formada por um conjunto de *tags*. A descrição completa de todos os comandos suportados pela RCM Script e exemplos de sua utilização podem ser encontrados no Anexo VI.

3.1.4.2 Principais funcionalidades :

A linguagem de script proposta possui as seguintes funcionalidades :

* Referências a fragmentos de conteúdos;
* Desvio Condicional;
* Laços Interativos;
* Geração de arquivos.

3.1.5 Metodologia RCM

A Metodologia RCM mostra uma estratégia para a implantação da arquitetura proposta na Seção 3.1.2. Todo e qualquer tipo de documento pode ser estruturado através da separação dos fragmentos reutilizáveis representados através da RCML. Para isto, um Gerenciador de conteúdos estruturados, capaz de manipular e gerenciar os dados e gerar a base RCML, é proposto. Partindo da hipótese da existência de uma ferramenta capaz de efetuar este gerenciamento das informações, então tem-se o seguinte esquema funcional exposto na Figura 3.8. A exemplificação desta metodologia, estratégias de implementação e implementação de uma ferramenta podem ser observados na Seção 4.1.2.

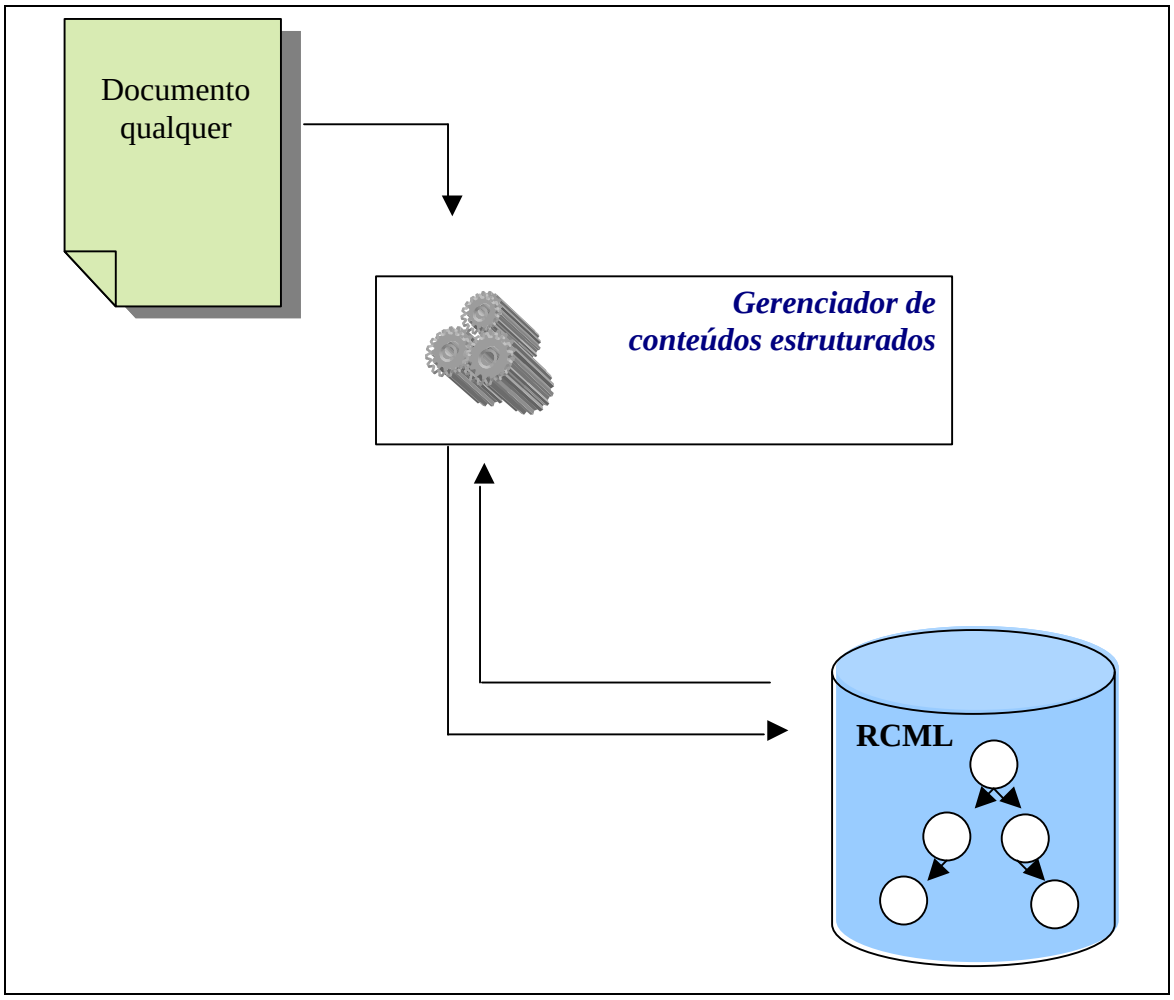


Figura 3.8 Metodologia RCM – Gerenciador de conteúdos estruturados.

O Gerenciador de conteúdos estruturados deverá possuir a capacidade de gerar dados RCML, manipular (editar, apagar e adicionar fragmentos) esses dados e permitir recursos de integração desses conteúdos (através da reusabilidade) como, por exemplo, referências.

Com a fragmentação dos conteúdos, vários documentos podem ser formados reutilizando informações em diferentes situações. Um documento que utilize esta estratégia terá que realizar referências aos fragmentos necessários, como pode ser observado na Figura 3.9.

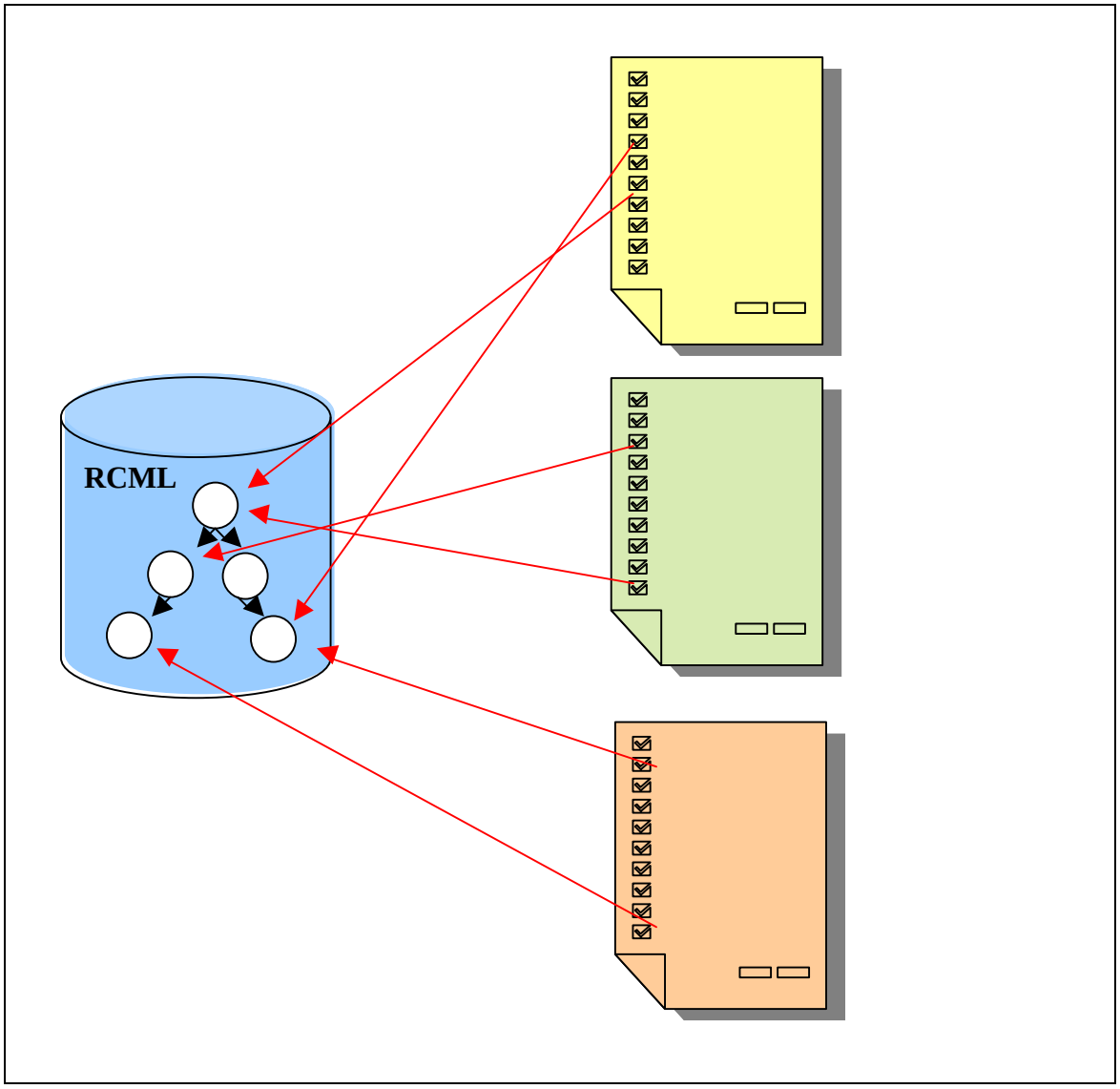


Figura 3.9 Metodologia RCM – Diferentes interpretações de conteúdos.

Assim, qualquer modificação será feita apenas na base de dados RCML, e todos os documentos que fazem referência para os dados alterados serão atualizados

automaticamente. Isso traz grandes vantagens para a produção de documentos, uma vez que a manutenção torna-se muito simples.

O conceito de conteúdo reutilizável é introduzido neste modelo como um fragmento que pode ser referenciado por diferentes documentos com diferentes interpretações, caso exista necessidade. Assim, um mesmo conteúdo poderá ser visto, por exemplo, em diversos formatos visuais simultâneamente, conforme ilustração da Figura 3.10.

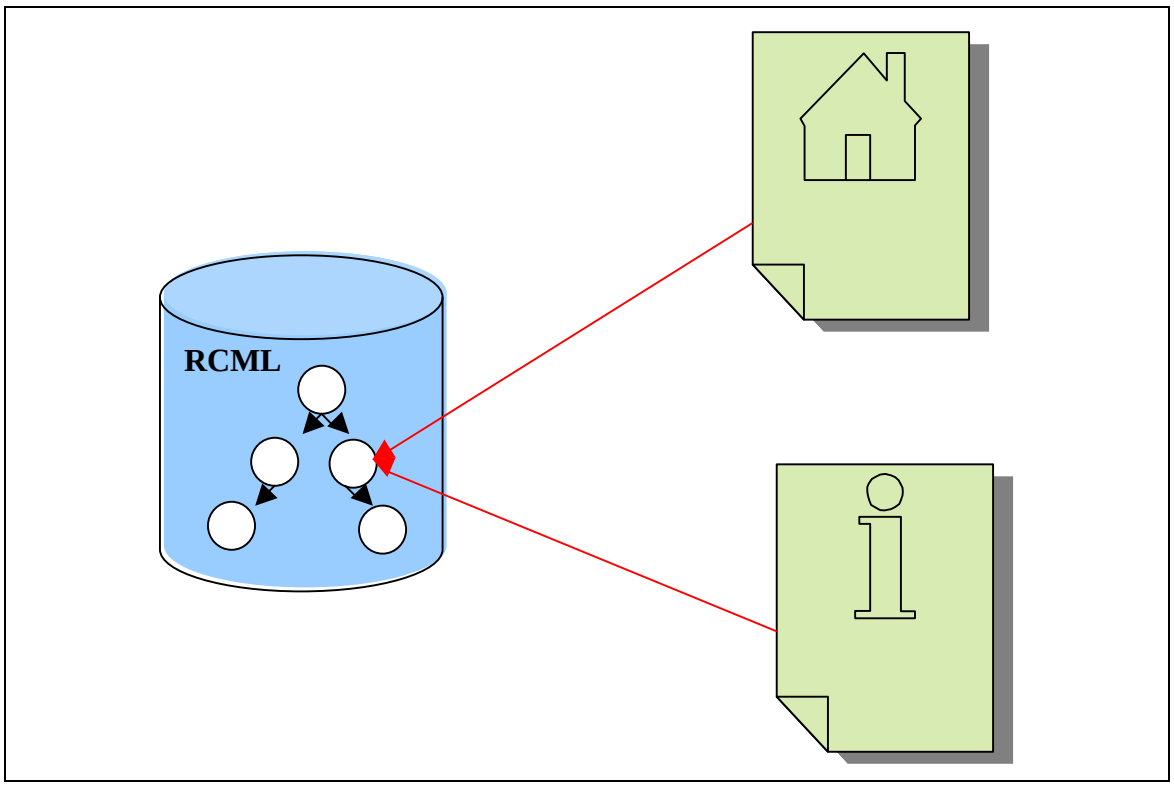


Figura 3.10 Metodologia RCM – Reutilizando fragmentos de conteúdos.

Para a obtenção de um processo completo que utilize esta estratégia, serão necessárias as seguintes ferramentas :

* Entrada de dados;
* Um Gerenciador de Conteúdos;
  a. Um Processador RCML;
  b. Um Interpretador de Código;
  c. Um Processador XSL;
  d. Um Formatador RCML;
* Um Processador de Saídas;
* Saída de dados.

Este processo é ilustrado na Figura 3.11. As ferramentas necessárias são detalhadas nas próximas seções.

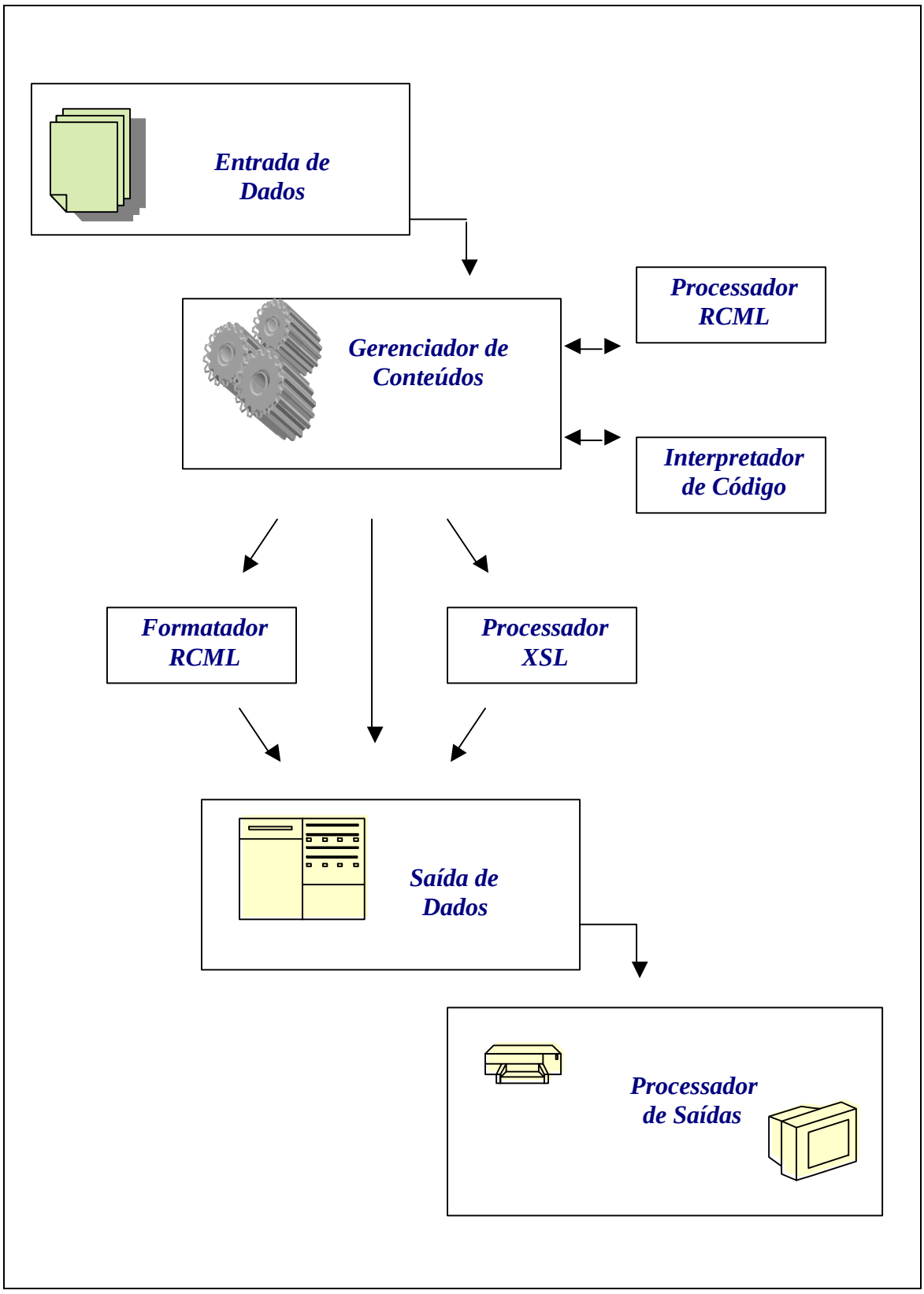


Figura 3.11 Metodologia RCM – Processo completo.

3.1.5.2 Entrada de Dados

Nesta etapa o usuário entra com os dados não estruturados através de uma interface que permita a entrada das informações. O resultado são os dados que serão gerenciados e estruturados pelo gerenciador de conteúdos. Alguma estrutura deve ser definida pelo usuário informando os dados em formato RCML Uma interface será necessária para possibilitar que o usuário informe tal estrutura. Recursos visuais que mostram dados agrupados hierarquicamente podem realizar esta tarefa permitindo a visualização gráfica da estrutura das informações. Assim, é na fase de entrada de dados que é definida a estrutura dos dados, ou seja, quais os fragmentos de conteúdo reutilizáveis. Finalmente, tais dados são passados para o gerenciador de conteúdos, que é a segunda fase deste processo. A Entrada de Dados é ilustrada na Figura 3.12.

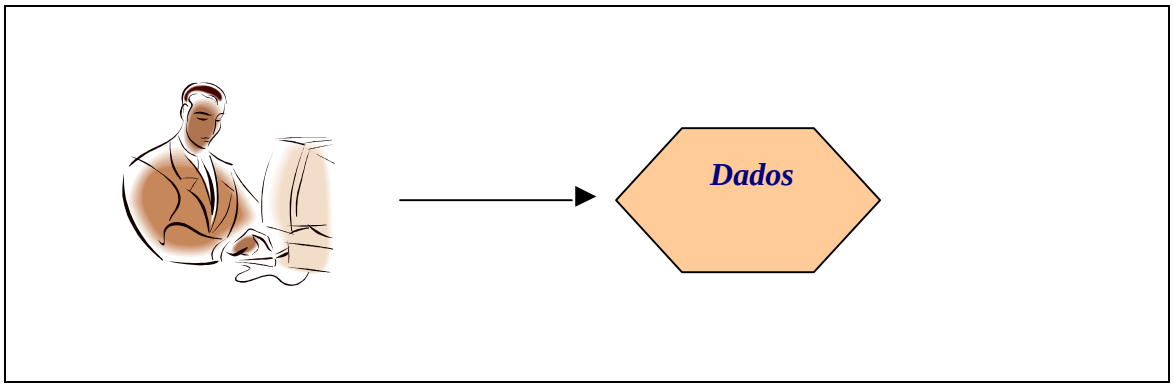


Figura 3.12 Metodologia RCM – Entrada de Dados.

3.1.5.3 Gerenciador de Conteúdos

Esta fase possui a tarefa de manipular as informações recebidas da fase de entrada de dados. Esta manipulação consiste em armazenar os dados em RCML (Processador RCML), Interpretar código para manipulação dos conteúdos (Interpretador de Código), formatar os dados RCML e gerar saídas em vários formatos (Formatador RCML) e processar as informações com o uso de folhas de estilos XSL (Processador XSL).

a. Processador RCML :

Esta ferramenta é capaz de, apartir das informações estruturadas em um determinado formato, gerar uma base de dados RCML. Assim, é tarefa do processador RCML a produção de código RCML dado um conjunto de informações fornecido na fase de entrada de dados. O processo inverso também faz-se necessário, uma vez que os documentos serão manipulados várias vezes. Portanto, para que se torne possível a importação de dados RCML para a estrutura definida pelo usuário, um mecanismo que

interprete código RCML já definido e gere a estrutura visual apropriada, será necessário. O processador RCML irá realizar esta importação, conforme Figura 3.13.

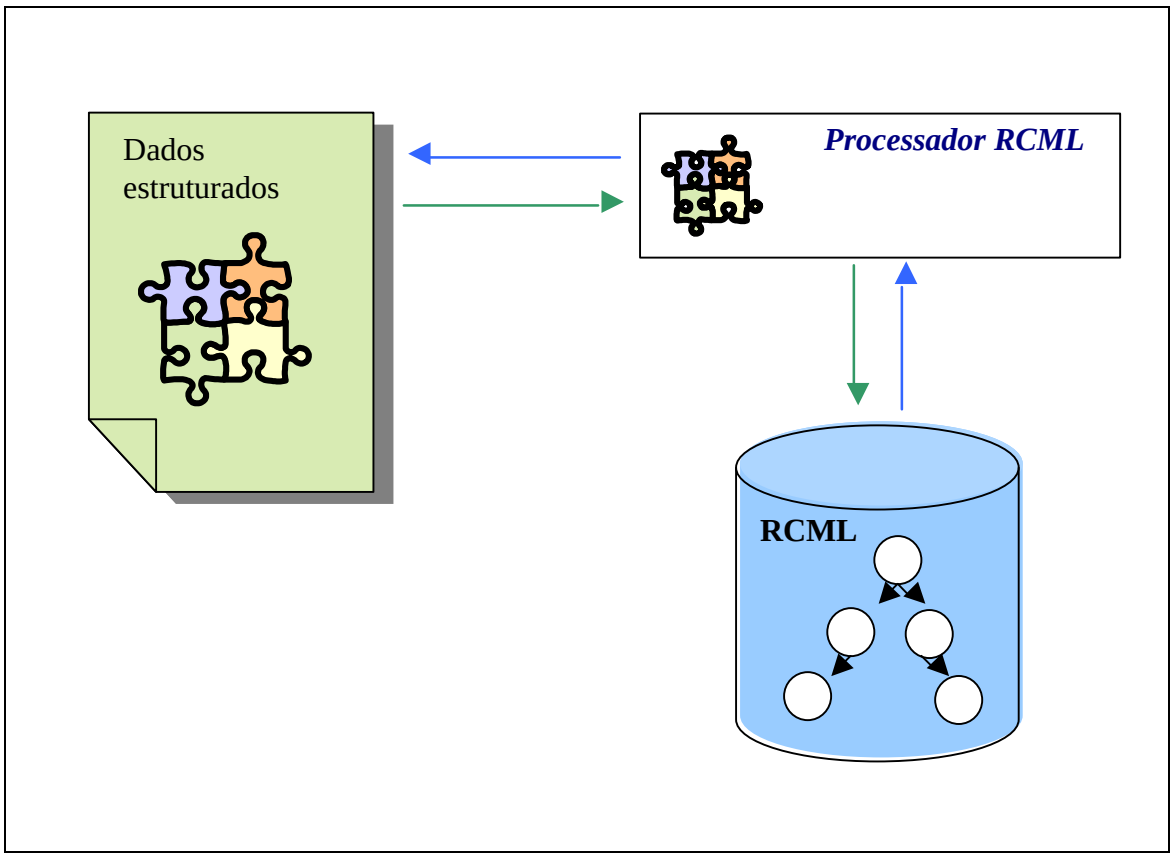


Figura 3.13 Metodologia RCM – Processador RCML.

b. Interpretador de Código :

Esta ferramenta é capaz de interpretar instruções RCM Script inseridas nos documentos estruturados cujo objetivo é a manipulação dos fragmentos de conteúdo. Com isso, documentos dinâmicos que façam uso da reutilização dos conteúdos podem ser desenvolvidos de uma forma muito simples. O Interpretador de código é capaz de realizar a verificação léxica e sintática de códigos expressados seguindo as regras RCM Script, e, caso não haja a detecção de algum erro, então o interpretador de código comunica-se com o Processador RCML solicitando tal manipulação. O processador RCML então realizará as tarefas solicitadas e, finalmente, os dados serão formatados para visualização ou pelo Processador XSL ou pelo Formatador RCML. O esquema de funcionamento do interpretador de código é mostrado na Figura 3.14.

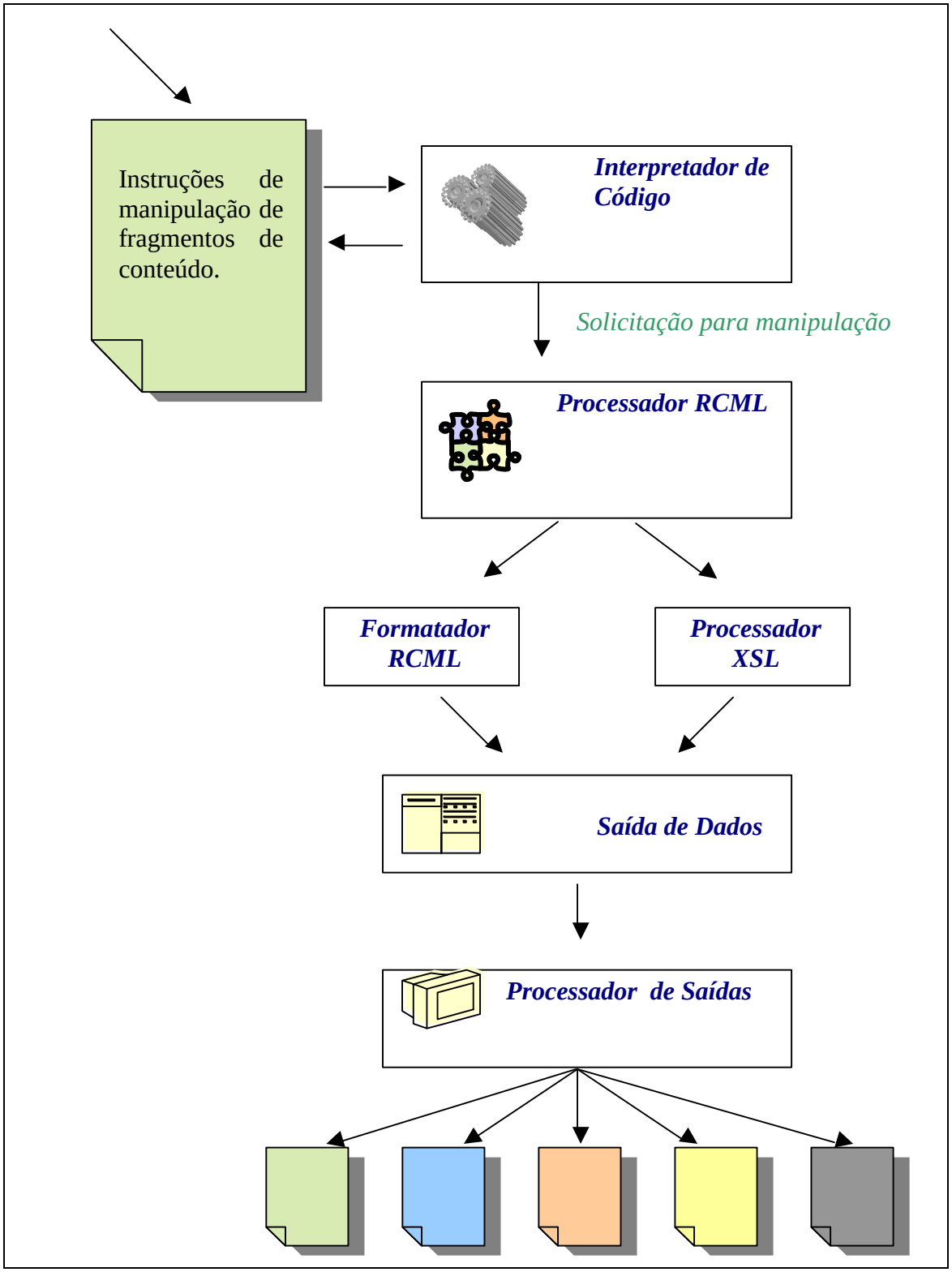


Figura 3.14 Metodologia RCM – Interpretador de Código.

c. Formatador RCML :

O Formatador RCML é uma ferramenta que interpreta, através de solicitações de dados ao Processador RCML, um conteúdo nesta linguagem e converte em uma saída desejada. Diferentes saídas podem representar uma mesma base de dados neste formato, como HTML, XML, DOC, PPT, e assim por diante. O funcionamento deste mecanismo ocorre através da solicitação de dados feita ao Processador RCML e da conversão destas informações em um formato suportado por este mecanismo. O Formatador RCML funciona como uma folha de estilos, mas realiza tarefas que XSL não é capaz (transformação em formato DOC, por exemplo). Outra vantagem da utilização desta ferramenta é que o uso de folha de estilos requer que o sistema operacional tenha suporte para XSL, o que ainda hoje não é muito comum. O Processador RCML gera uma saída que é enviada para o Processador de Saídas. Conforme o formato utilizado na conversão, diferentes mecanismo externos (*plug-ins*) serão utilizados para efetuar a visualização dos documentos. O Formatador RCML pode ser observado na Figura 3.15.

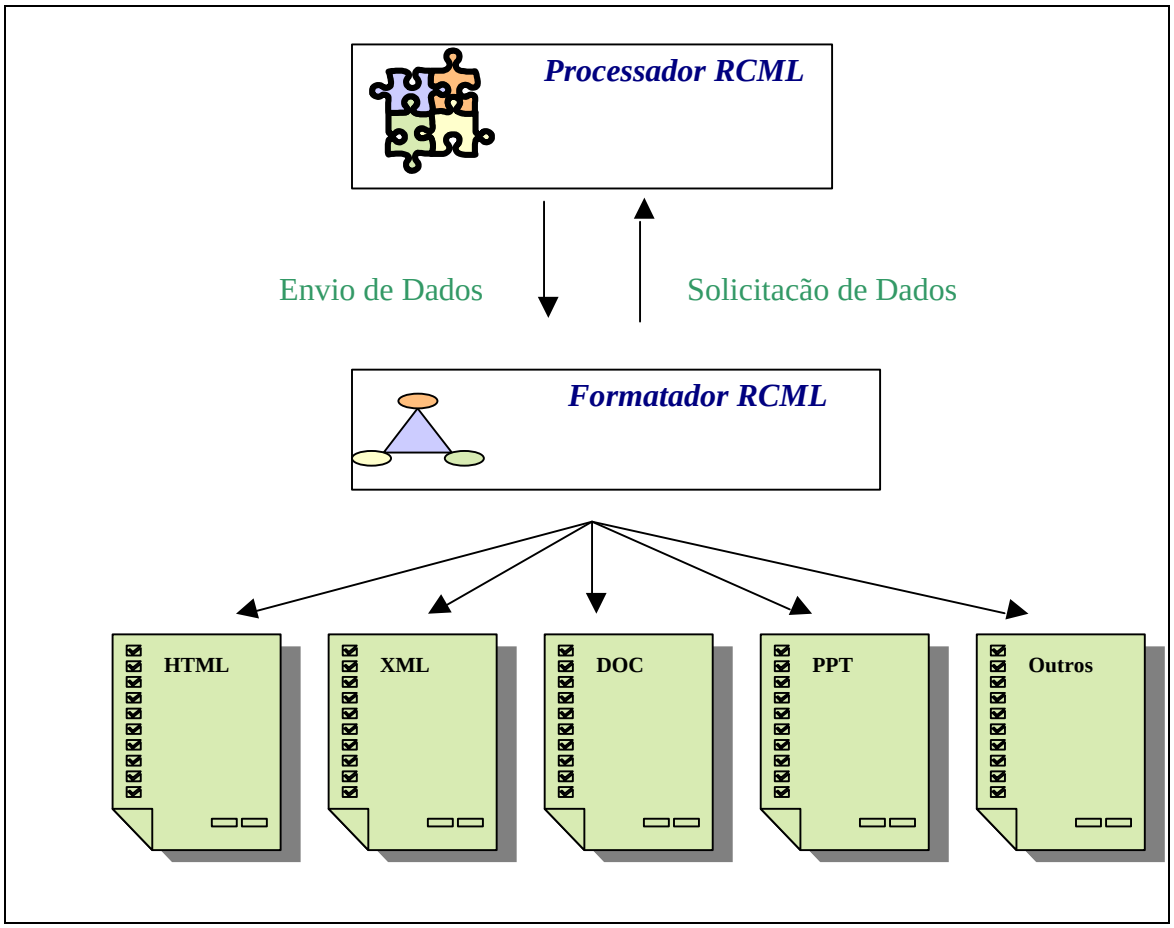


Figura 3.15 Metodologia RCM – Formatador RCML.

d. Processador XSL :

O Processador XSL é uma ferramenta capaz de aplicar uma folha de estilos externa a um determinado fragmento de dados RCML. Para tanto, é necessária a solicitação ao Processador RCML de um determinado fragmento de conteúdo. Dada a obtenção de um determinado conteúdo, então a aplicação a uma determinada folha de estilos pode ser realizada. Embora nem todos os formatos de saída (como DOC, por exemplo) podem ser gerados a partir de folhas de estilos, esta estratégia possui a grande vantagem de possibilitar a adição e/ou produção de novas folhas de estilos, independente do modelo e da ferramenta que está sendo utilizada para gerenciar os conteúdos. Portanto, torna-se ilimitado o número de formatações que podem ser realizadas através da utilização de folhas de estilos XSL, contando que o usuário tenha um conhecimento prévio desta tecnologia. O processador XSL recebe os dados do Processador RCML, através de solicitações de fragmentos de documentos, e gera as saídas formatadas conforme a especificação de uma determinada folha de estilos. A Figura 3.16 apresenta o Processador XSL.

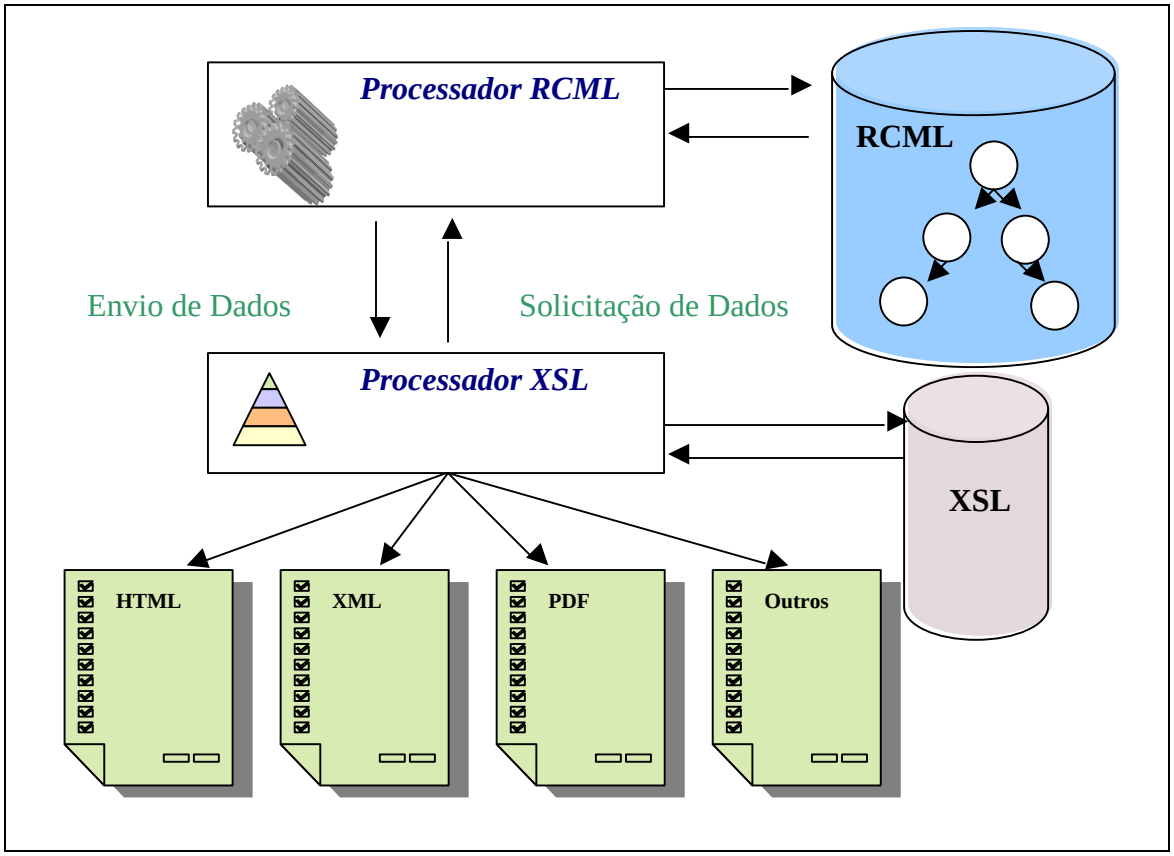


Figura 3.16 Metodologia RCM – Processador XSL.

3.1.5.4 Saída de Dados

A Saída de dados é o resultado obtido pelo Processador XSL ou pelo Formatador RCML. Os dados poderão estar em diversos formatos, como HTML, XML, MathML, SVG, PDF, DOC, PPT, etc, ou uma combinação dessas tecnologias. Esses dados são passados, então, para o Processador de Saídas. A Saída de Dados é ilustrada na Figura 3.17.

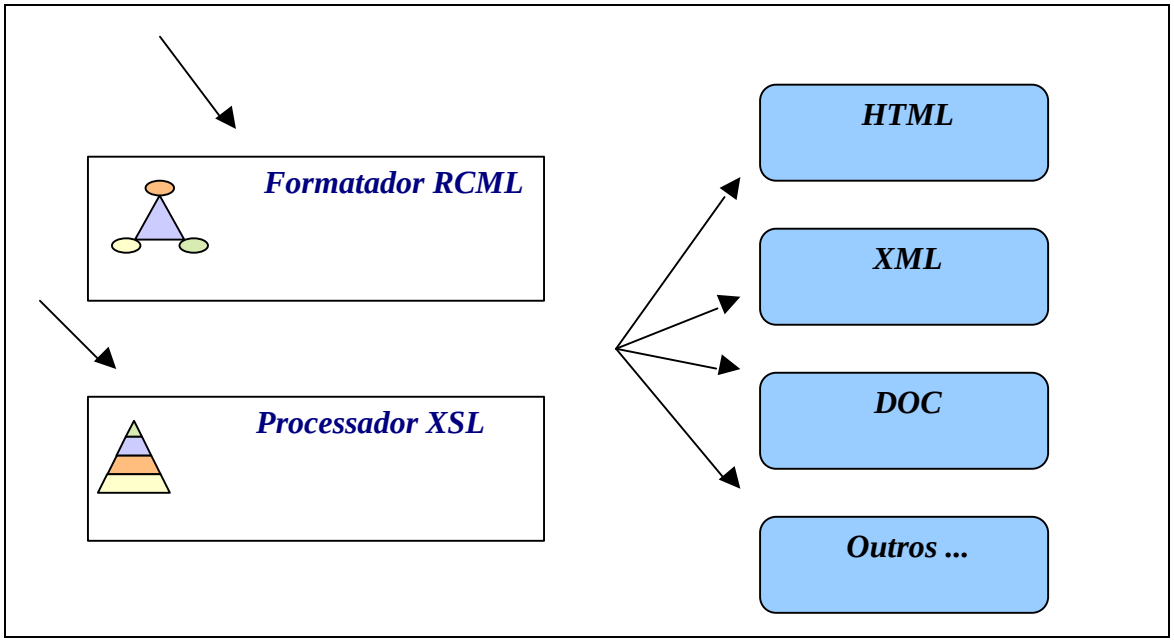


Figura 3.17 Metodologia RCM – Saída de Dados.

3.1.5.5 Processador de Saídas

Geralmente, tecnologias que estendem as funcionalidades da linguagem HTML, necessitam de mecanismos externos para viabilizar a visualização de seus conteúdos em um navegador web. Tais mecanismos são referenciados como *plugin´s*, . Os *plugin´s*, associados com diversas tencnologias, são utilizados até que os navegadores desenvolvam suporte nativo para essas tecnologias. Por exemplo, para visualizar conteúdo codificado em MathML em um navegados web, será necessária a existência de um *plugin* MathML. Atualmente, tem-se, dentre outros, o MathPlayer [MATHML 03] e o WebEQ [CLA03]. Já o *plugin* mais utilizado para a visualização de dados em SVG é o *Adobe SVG plugin* [ADO 03b]. Para a visualização de arquivos com extensão .DOC ou .PPT em navegadores, a Microsoft desenvolveu alguns *plug-in' s* específicos. Portanto, o processador de saídas depende da existência destas ferramentas intaladas no computador. Uma ilustração do Processador de saídas é apresentada na Figura 3.18.

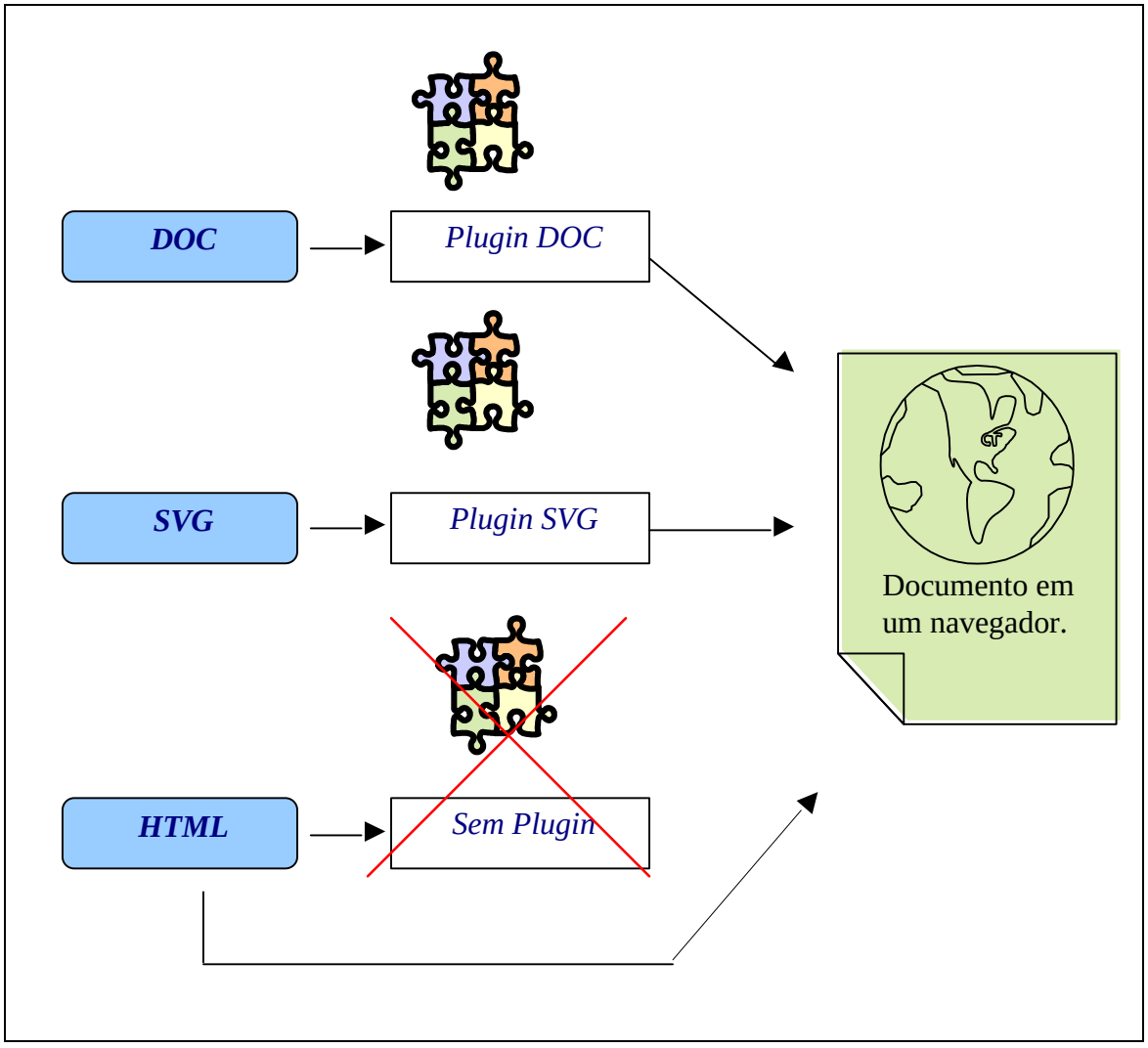


Figura 3.18 Metodologia RCM – Processador de Saídas.

3.2 Autoria de Conteúdos em formato MathML

3.2.1 Introdução

Há muito tempo, os desenvolvedores de *sites* ou páginas que necessitem conter textos matemáticos convivem com o problema de publicar notação matemática através de recursos nativos da Internet, [CLA03]. As fórmulas eram, e ainda são, inseridas, em grande parte, no formato GIF. Dependendo da quantidade, este modo de resolver o problema pode gerar uma grande dificuldade de gerenciamento, tornando, ainda, os sites muito "pesados", pela necessidade de transferência de uma enorme quantidade de figuras. Como o formato GIF representa uma figura estática, torna-se impossível a construção de fórmulas interativas e a realização de pesquisas que envolvam símbolos matemáticos. Outras técnicas também são utilizadas, como o formato PDF (*Portable Document Format*), da Adobe [ADO 03], mas que apresentam os mesmos problemas e, portanto, se mostram insatisfatórias quando utilizadas em aplicações complexas voltadas à educação. MathML (*Mathematical Markup Language*), desenvolvida para tratar conteúdo e apresentação matemática é uma tecnologia com enorme potencial para auxiliar na solução destes problemas por se tratar de uma linguagem nativa da Internet (totalmente especificada no modo texto), com amplas possibilidades de implementação de processos interativos.

Tecnologias baseadas em XML, como o MathML, ainda são muito recentes, o que resulta em uma grande carência de ferramentas e mecanismos para a manipulação de dados nestes formatos. Com certeza, grandes empresas de desenvolvimento de software estão com seus olhares voltados para essas tecnologias, [DES 03]. Mesmo assim, ainda existem poucos aplicativos, e, quando se trata de algo muito específico, a dificuldade é ainda maior.

Visando buscar uma solução de autoria baseada em MathML, esta seção mostra algumas estratégias de implementação para o desenvolvimento de aplicativos que façam uso desta linguagem, gerando dados neste formato. A intenção deste modelo é oferecer mecanismos capazes de converter e gerar dados matemáticos estruturados formatados em MathML. A metodologia proposta nesta seção será foi denominada "Metodologia MathTeX".

Para comprovar a potencialidade do modelo proposto, será necessária a existência de um interpretador MathTeX. Para tanto, desenvolveu-se o Editor de

Fórmulas, uma ferramenta que permite, a partir de entradas codificadas em LaTeX [WIL 95], a geração de código MathML para ser publicado através da web. O Capítulo 4 apresenta a descrição desta ferramenta, bem como estratégias de implemetação e exemplificações desta solução.

3.2.2 Metodologia MathTeX

3.2.2.1 Introdução

A proposta de produção de utilização do MathTeX será acompanhada, ou pelo menos hipotéticamente acompanhada, de um interpretador com suporte total para a linguagem LaTeX. Neste ponto, assume-se a existência de um processador MathTeX que reconheça toda a sintaxe LaTeX.

O modelo desta proposta segue em 4 passos : Entrada de Dados, Processamento de Código MathTeX, Saída de Dados , e Processamento de Saídas. Uma ilustração da Metodologia MathTeX é apresentada na Figura 3.19.

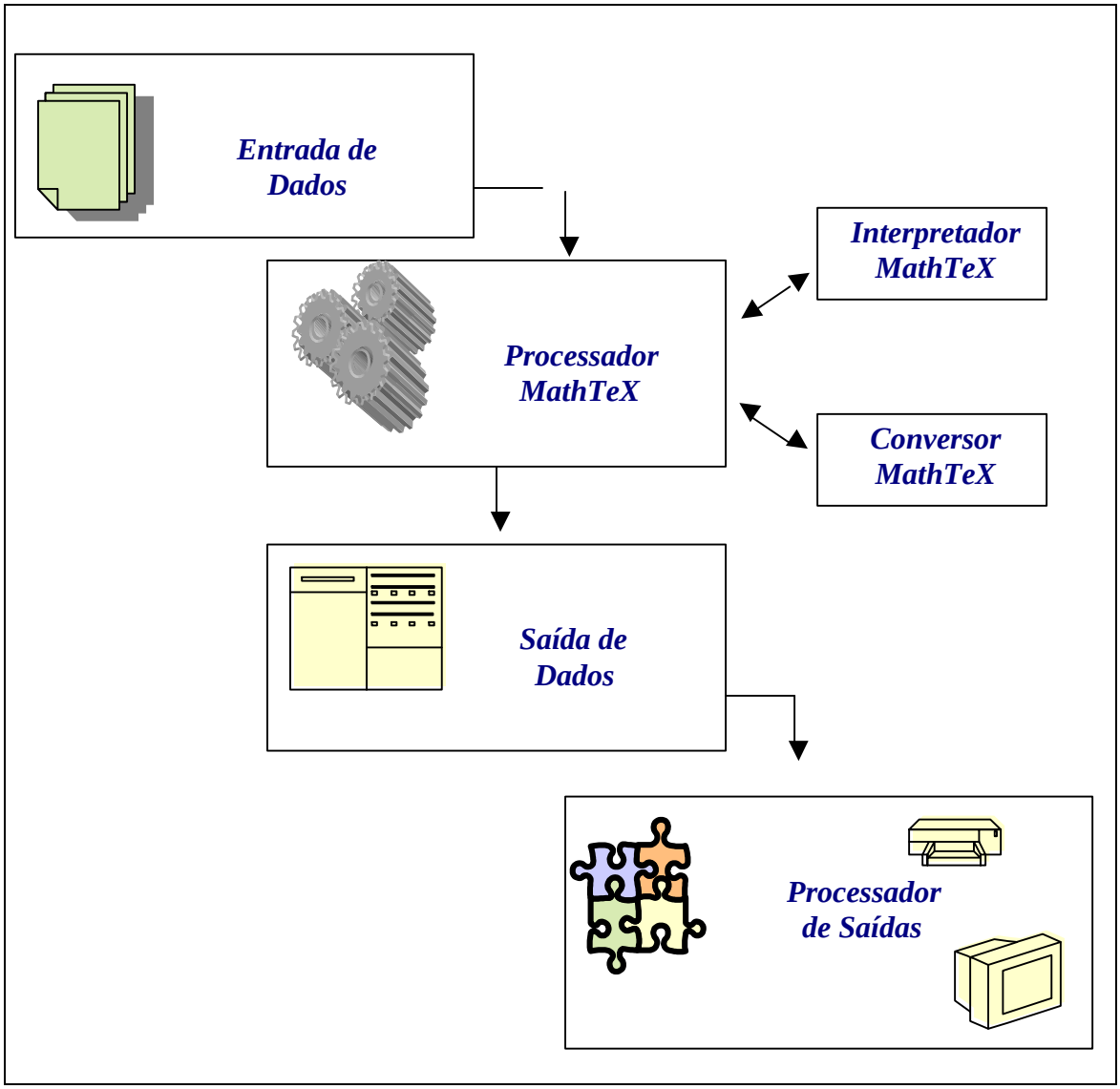


Figura 3.19 Metodologia MathTeX – Modelo Geral.

3.2.2.2 Entrada de Dados

O processo inicia-se através da entrada de dados. Nesta etapa, o usuário entra com o código LaTeX através de uma interface gráfica que deve ser construída para este propósito. O resultado desta etapa, o código fonte LaTeX, é passado para o Processador MathTeX. A ilustração deste processo é mostrada na Figura 3.20.

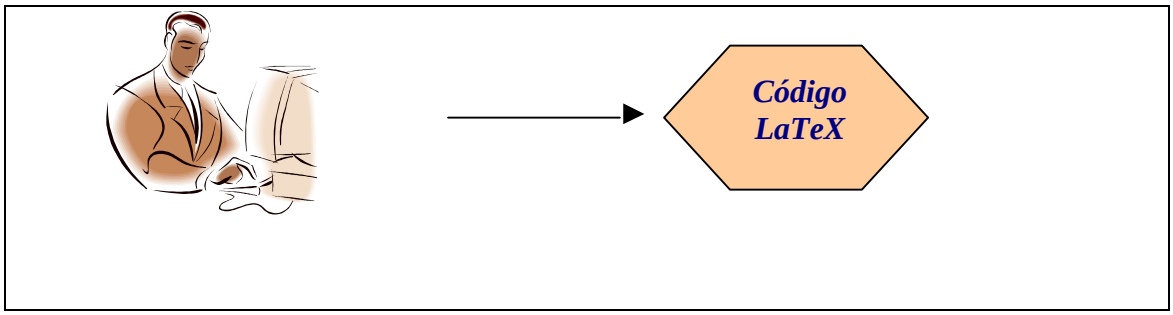


Figura 3.20 Metodologia MathTeX - Entrada de Dados.

3.2.2.3 Processador MathTeX

O processador MathTeX recebe, como entrada, um código fonte em formato LaTeX, e, após uma verificação de erros de programação, converte os dados em MathML. Para a realização deste processo, vários passos são necessários : análise léxica, análise sintática, conversão de dados. O processador de análise léxica e sintática, [COM 03], compõe um interpretador, que, a partir do código fonte, gerará uma árvore sintática. Um conversor terá como entrada esta árvore e efetuará uma conversão para MathML. Assim, a saída final do processador MathTeX é um código MathML válido. O Processador MathTex pode ser observado na Figura 3.21.

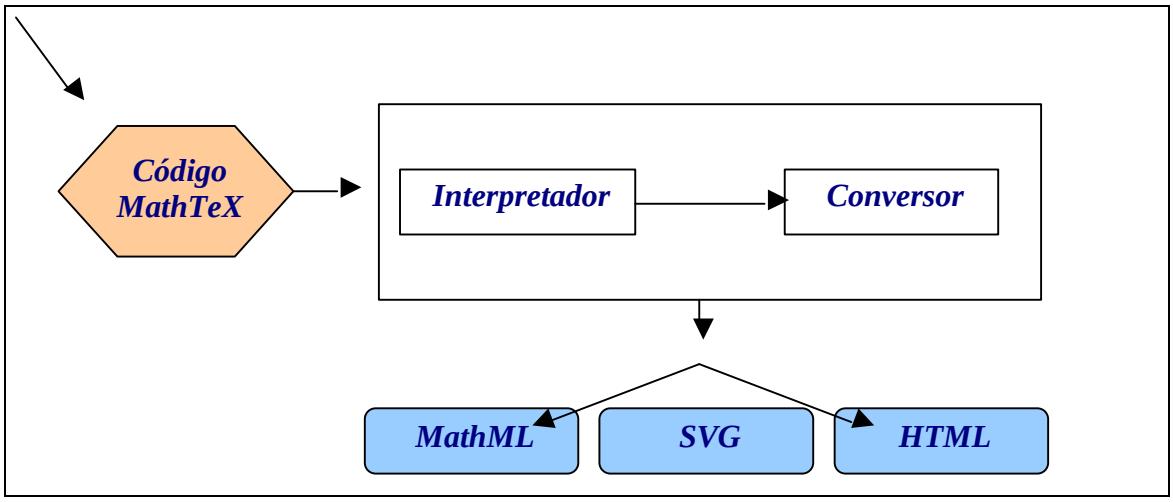


Figura 3.21 Metodologia MathTeX – Processador MathTeX.

3.2.2.4 Saída de Dados

A Saída de dados é o resultado obtido pelo Processador MathTeX. Os dados estarão em formato MathML. Uma extensão para este modelo seria a capacidade de exportação de dados em formato SVG, HTML, ou uma combinação dessas três tecnologias. Esses dados são passados, então, para o Processador de Saídas. A Saída de Dados é ilustrada na Figura 3.22.

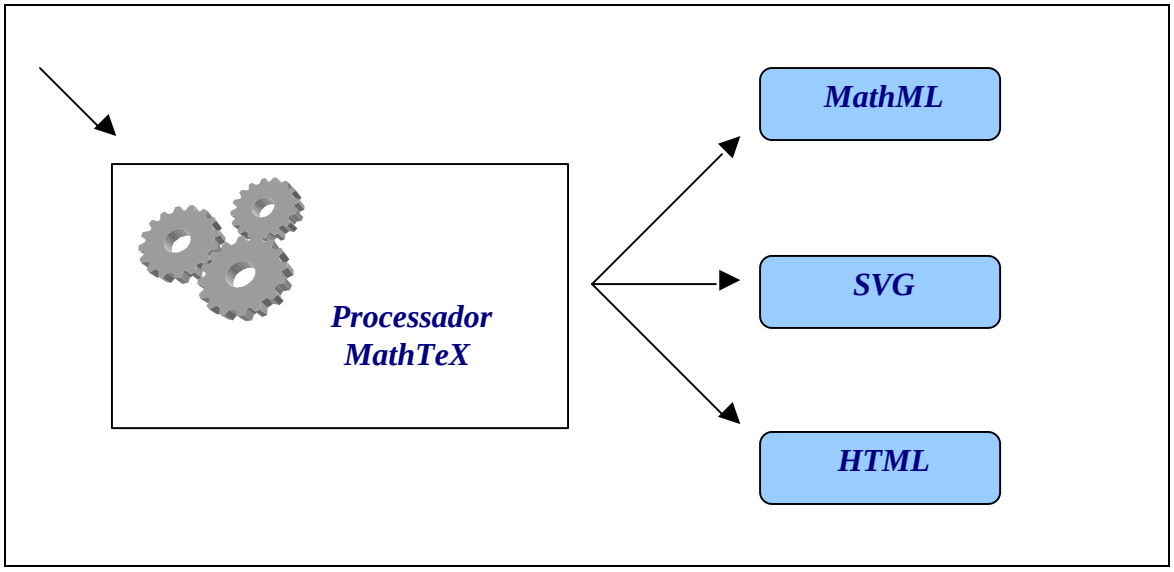


Figura 3.22 Metodologia MathTeX – Saída de Dados.

3.2.2.5 Processador de Saídas

Geralmente, tecnologias que estendem as funcionalidades da linguagem HTML, necessitam de mecanismos externos para viabilizar a visualização de seus conteúdos em um navegador web. Tais mecanismos são referenciados como *plugin´s*, . Os *plugin´s*, associados com diversas tencnologias, são utilizados até que os navegadores desenvolvam suporte nativo para essas tecnologias. Por exemplo, para visualizar conteúdo codificado em MathML em um navegados web, será necessária a existência de um *plugin* MathML. Atualmente, tem-se, dentre outros, o MathPlayer, [MATHML 03]e o WebEQ, [CLA03]0. Já o *plugin* mais utilizado para a visualização de dados em SVG é o *Adobe SVG plugin*, [ADO 03b]. O Processador de Saídas pode ser observado na Figura 3.23.

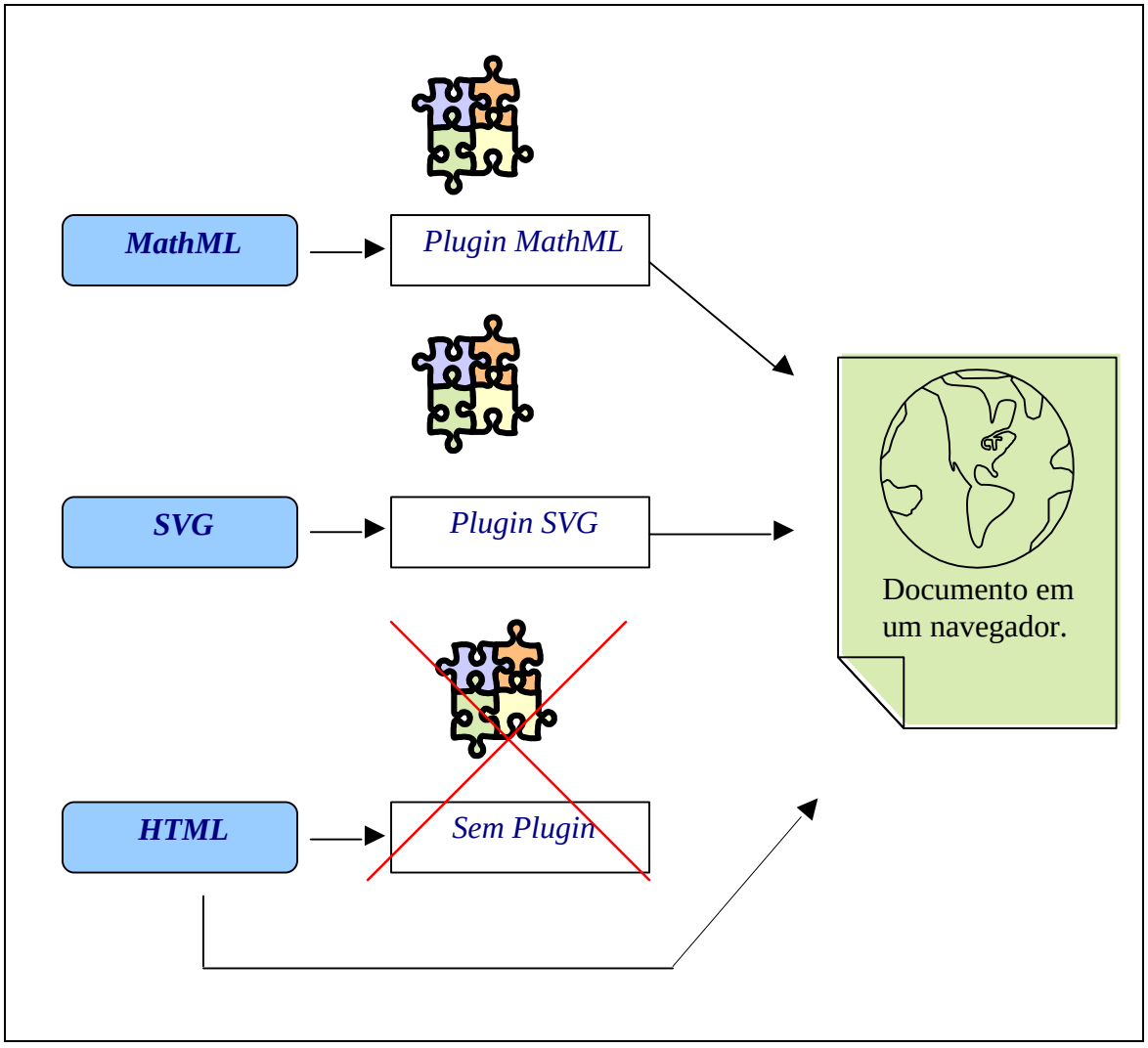


Figura 3.23 Metodologia MathTeX – Processador de Saídas.

3.3 Um Modelo de Gráficos Dinâmicos *DataDriven* em formato SVG

3.3.1 Introdução

A linguagem SVG, uma aplicação XML, possui um código muito complexo. Na maioria das aplicações, torna-se inviável a codificação manual deste documento. Já existem algumas ferramentas de suporte visual [W3C 03h], nas quais, o usuário cria as imagens SVG apenas com o mouse. Com isso, muito dos problemas de codificação podem ser resolvidos. Entretanto, uma vez que SVG é uma linguagem nova e flexível, não existe um número conhecido de aplicações desta tecnologia. Como XML, SVG possui utilidades diferentes que surgem a cada dia. É claro, os problemas mais comuns são resolvidos com a geração de imagens gráficas para a web. Mesmo assim, determinadas aplicações podem possuir particularidades diferentes e que não podem ser oferecidas pelas ferramentas de suporte "generalizado". Surge então a necessidade de aplicativos específicos para cada aplicação.

A técnica descrita no decorrer desta seção se refere à utilização do XML juntamente com a tecnologia XSL, fornecendo assim um mecanismo automatizado e independente para ser utilizado no desenvolvimento de ferramentas de suporte, especialmente de autoria, que fazem o gerenciamento e a produção de documentos Web contendo gráficos SVG. XML será utilizado para fornecer os dados de um determinado gráfico para um posterior mapeamento em SVG através de XSL. As folhas de estilos XSLT desenvolvidas para realizar as transformações em SVG podem ser encontradas no Anexo X.

Resumindo, procura-se mostrar uma metodologia que faz uso de três tecnologias, XML e XSL para gerar SVG,através dos seguintes passos:

* Geração de documentos XML;
* Processamento e transformação da base de dados XML em SVG.

A Figura 3.24 mostra a idéia principal desta estratégia.

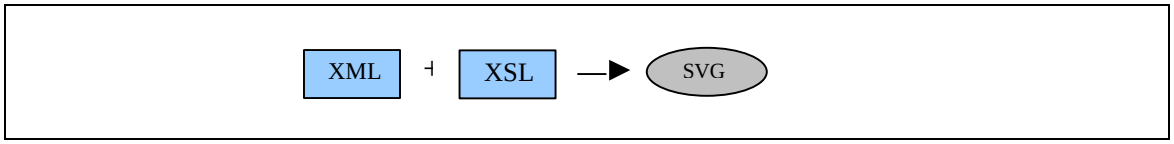


Figura 3.24 Fluxo geral.

A utilização das técnicas descritas nos dois itens acima possui, como aplicação principal, o desenvolvimento de ferramentas de suporte para a autoria de documentos Web. Já a utilização **apenas** do segundo item tem como principal aplicação a produção de documentos Web contendo gráficos SVG.

Esta seção mostrará a utilização da linguagem XML para representar os dados de gráficos estatísticos. Assim, uma linguagem de descrição de gráficos estatísticos será proposta, cuja denominação inicial será GraphML (*Graph Marckup Language*) – Linguagem de Marcação para Gráficos. Esta seção mostrará a estrutura desta linguagem e como está pode ser utilizada para a produção de conteúdo em formato SVG. O Capítulo 4 apresenta a descrição de uma ferramenta proposta para validar esta metodologia, bem como estratégias de implementação e exemplificações desta solução.

3.3.2 A linguagem GraphML

A linguagem GraphML, sigla de *Graph Marckup Language* foi desenvolvida com a intenção de descrever dados de gráficos estatísticos. Inicialmente, foram propostas apenas as descrições para quatro tipos de gráficos : gráficos de pizza, barras, pontos e linhas. Novas descrições de gráficos poderão ser adicionadas em futuros trabalhos através da utilização da metodologia aqui proposta. A DTD para esta linguagem pode ser encontrada no Anexo III, e a descrição completa de todos os seus elementos pode ser encontrada no Anexo VII.

Para exemplificar, supõe-se um gráfico de barras que represente o número de habitantes dos países Brasil e Estados Unidos. Dados os números dos habitantes 10000 e 15000 para cada país, respectivamente, então obtém-se o código GraphML correspondente ilustrado na Figura 3.25.

```
<Grafico Tipo = “Barras”>
   <Item Titulo=”Brasil”>
      <Valor> 10000 </Valor>
   </Item>
   <Item Titulo=”Brasil”>
      <Valor> 15000 </Valor>
   </Item>
</Grafico>
```

Figura 3.25 GraphML – Exemplo de utilização.

3.3.3 Metodologia GraphML

3.3.3.1 Introdução

Gráficos dirigidos aos dados (*DataDriven)* são representados de forma dinâmica pois sua renderização é feita somente quando alguma solicitação de visualização é realizada. Para isto, torna-se necessária a descrição dos dados em alguma linguagem. GraphML irá descrever dados de gráficos estatísticos seguindo o modelo da Figura 3.26.

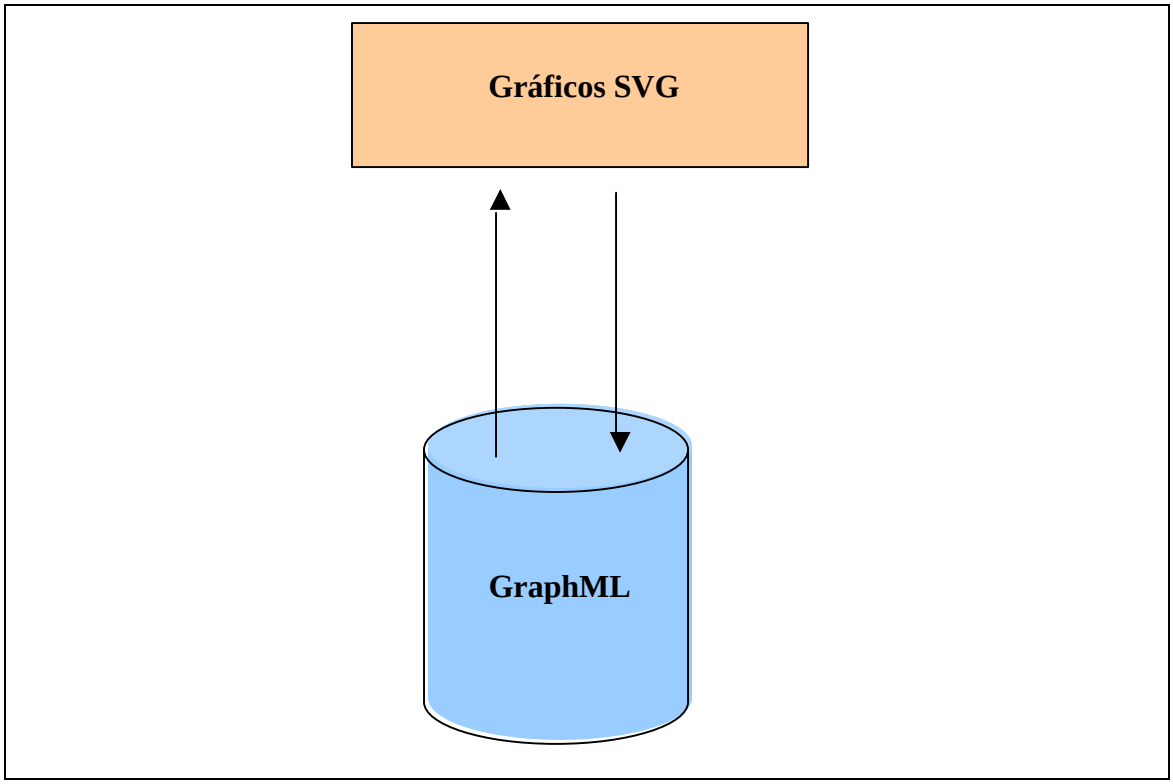


Figura 3.26 Metodologia GraphML – Gráficos DataDriven

Cada base GraphML representa um conjunto de dados para serem representados em um tipo de gráfico estatístico específico, como por exemplo, um gráfico de barras. Para a realização do processo de renderização do conteúdo, foi escolhida a tecnologia XSL por ser um mecanismo transformador de documentos XML. Enquanto XML estrutura os dados de um determinado gráfico estatístico através da GraphML, a XSL transforma-os em formato SVG através de folhas de estilos específicas para cada gráfico. Este processo pode ser observado através da Figura 3.27.

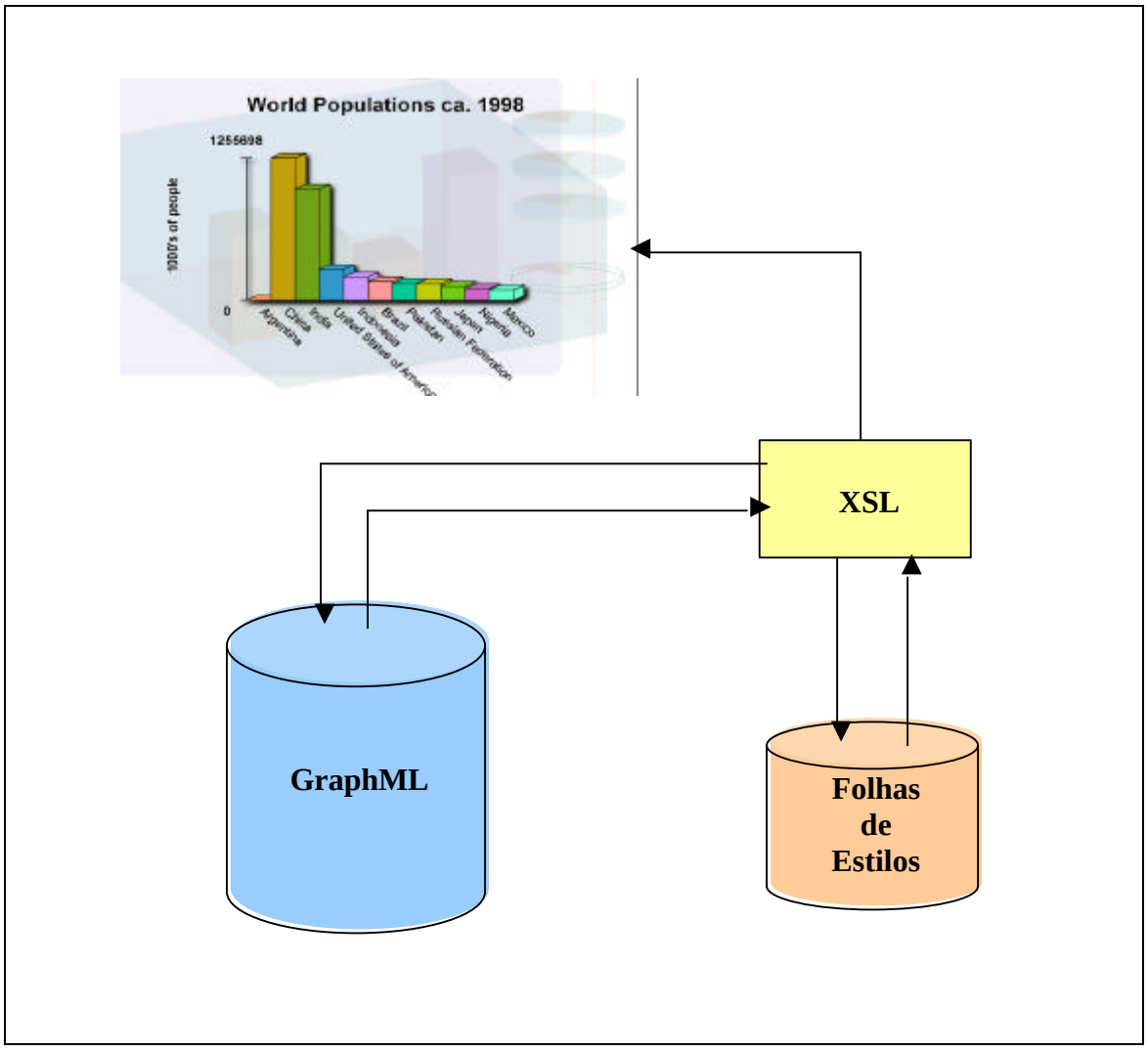


Figura 3.27 Metodologia GraphML – Processo geral.

Para a obtenção de um processo completo que utilize esta estratégia, será necessário :

* Entrada de Dados
* Um Gerenciador de Dados GraphML
  a. Um Processador GraphML
  b. Um Processador XSL
* Folhas de estilos específicas
* Um Processador de Saídas
* Saída dos Dados

Este processo é ilustrado na Figura 3.28. As ferramentas necessárias são detalhadas nas próximas seções.

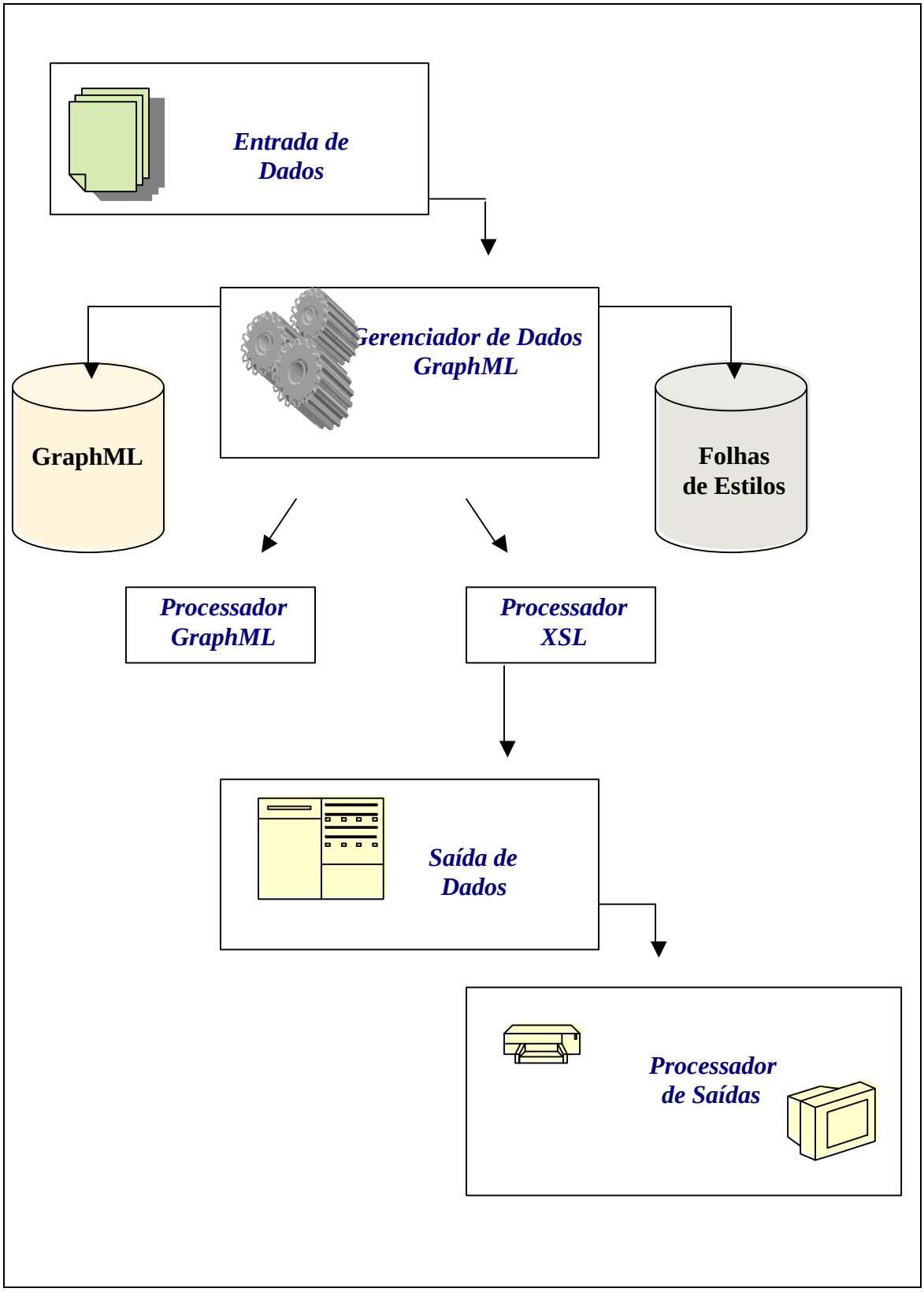


Figura 3.28 Metodologia GraphML – Modelo de Implementação.

3.3.3.2 Entrada de Dados

Nesta etapa, o usuário fornece a base de dados de um determinado gráfico e informa o tipo de renderização desejada. Assim, o Gerenciador de Dados GraphML irá se encarregar de gerar uma base de dados em formato GraphML adequada. A Figura 3.29 apresenta este processo.

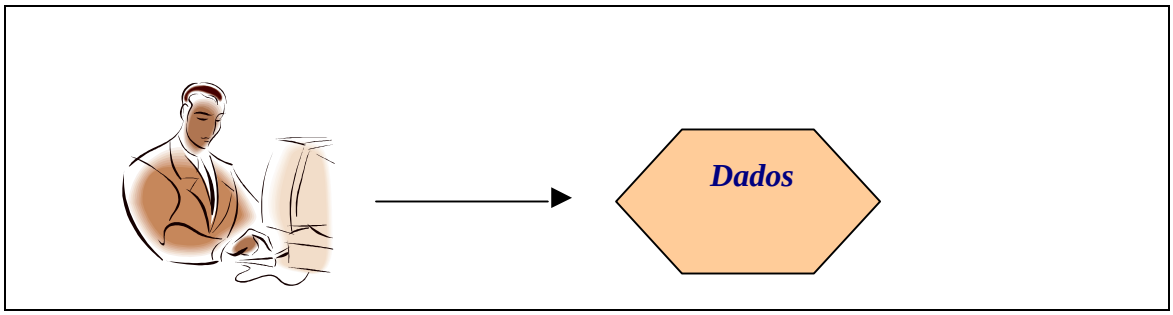


Figura 3.29 Metodologia GraphML – Entrada de Dados.

3.3.3.3 Gerenciador de Dados GraphML

Nesta etapa do processo, os dados fornecidos pelo usuário são armazenados em uma base de dados em formato GraphML. Para tanto, um Processador GraphML faz-se necessário uma vez que a manipulação e geração de dados neste formato tornam-se necessárias. O Gerenciador de Dados GraphML, além de fazer uso de um Processador GraphML, utiliza, também, um processador XSL para a renderização desses dados em SVG.

a. Processador GraphML :

O Processador GraphML é uma ferramenta capaz de, apartir das informações estruturadas em um determinado formato , gerar uma base de dados GraphML. Assim, é tarefa do processador a produção de código GraphML, e a conversão da entrada de dados para o código equivalente. O processo inverso também faz-se necessário, uma vez que os documentos serão manipulados várias vezes. Portanto, para que torne-se possível a importação de dados XML para a estrutura definida pelo usuário, um mecanismo que interprete código XML já definido e gere a estrutura apropriada, será necessário. O processador GraphML irá realizar esta importação. A Figura 3.30 apresenta o Processador GraphML.

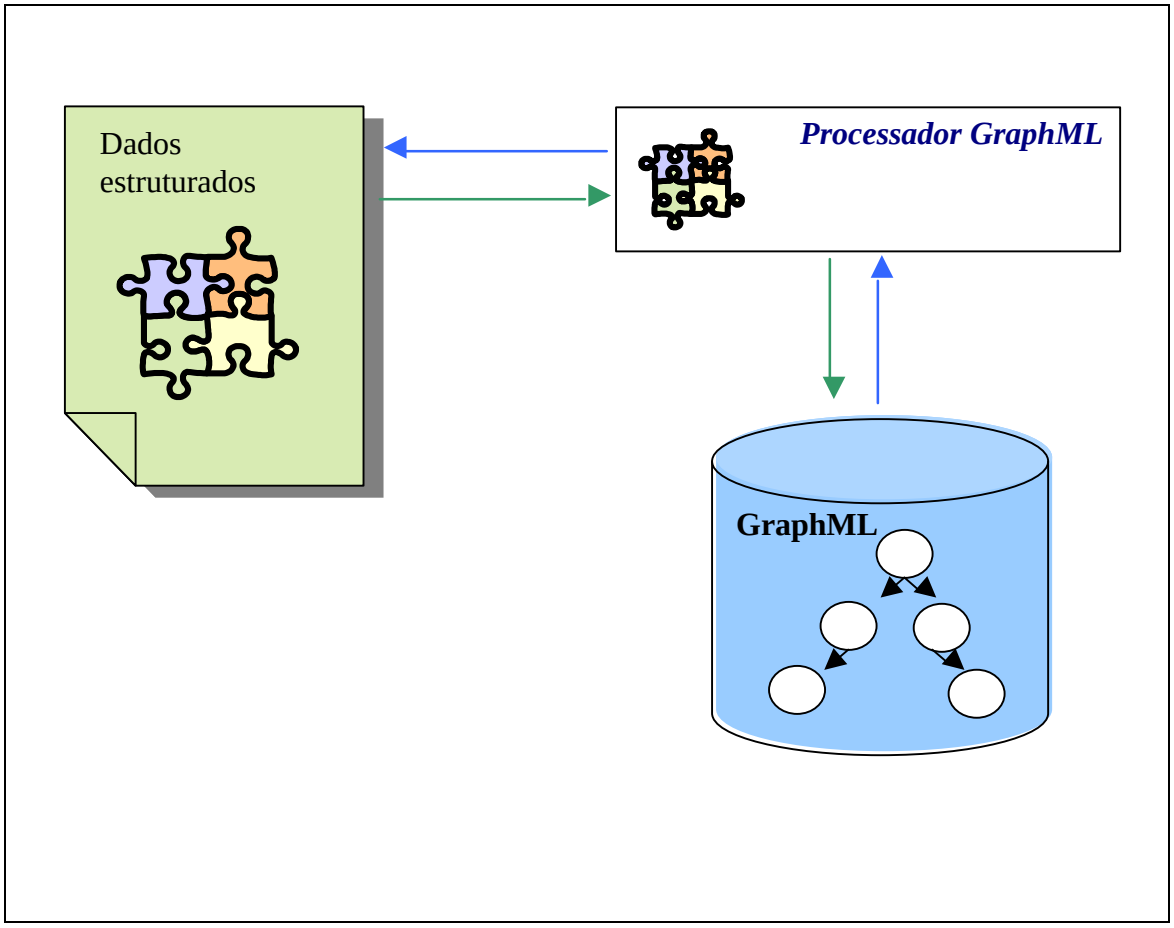


Figura 3.30 Metodologia GraphML – Processador GraphML.

b. Processador XSL :

O processador XSL é uma ferramenta capaz de aplicar uma folha de estilos externa a um determinado fragmento de dados XML. Para tanto, é necessária a solicitação ao Processador GraphML de um determinado fragmento de conteúdo. Dada a obtenção de um determinado conteúdo, então a aplicação a uma determinada folha de estilos pode ser realizada. Embora nem todos os formatos de saída (como DOC, por exemplo) podem ser gerados a partir de folhas de estilos, esta estratégia possui a grande vantagem de possibilitar a adição e/ou produção de novas folhas de estilos, independente da ferramenta que está sendo utilizada para gerenciar os conteúdos. Portanto, torna-se ilimitado o número de formatações que podem ser realizadas através da utilização de folhas de estilos XSL, contando que o usuário tenha um conhecimento prévio desta tecnologia. O processador XSL recebe os dados do processador GraphML, através de solicitações de fragmentos de documentos, e gera as saídas formatadas conforme a especificação de uma determinada folha de estilos. O Processador XSL pode ser visualizado através da Figura 3.31.

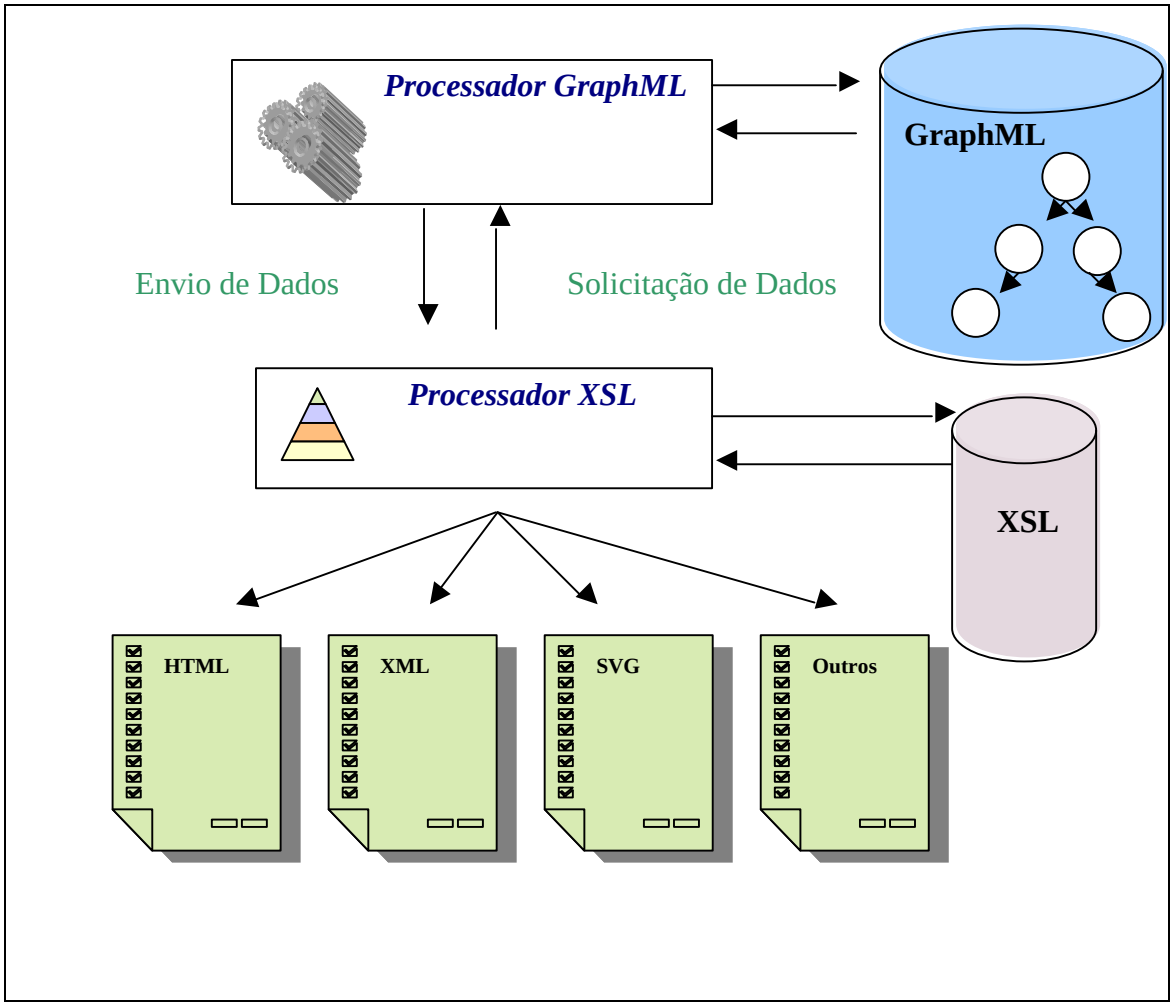


Figura 3.31 Metodologia GraphML – Processador XSL.

3.3.3.4 Folhas de Estilos específicas

Cada gráfico estatístico definido necessitará de uma folha de estilos para sua renderização. O processador XSL fará uso destas folhas de estilos para realizar o processo de transformação de dados GraphML em SVG. A Figura 3.32 ilustra este processo.

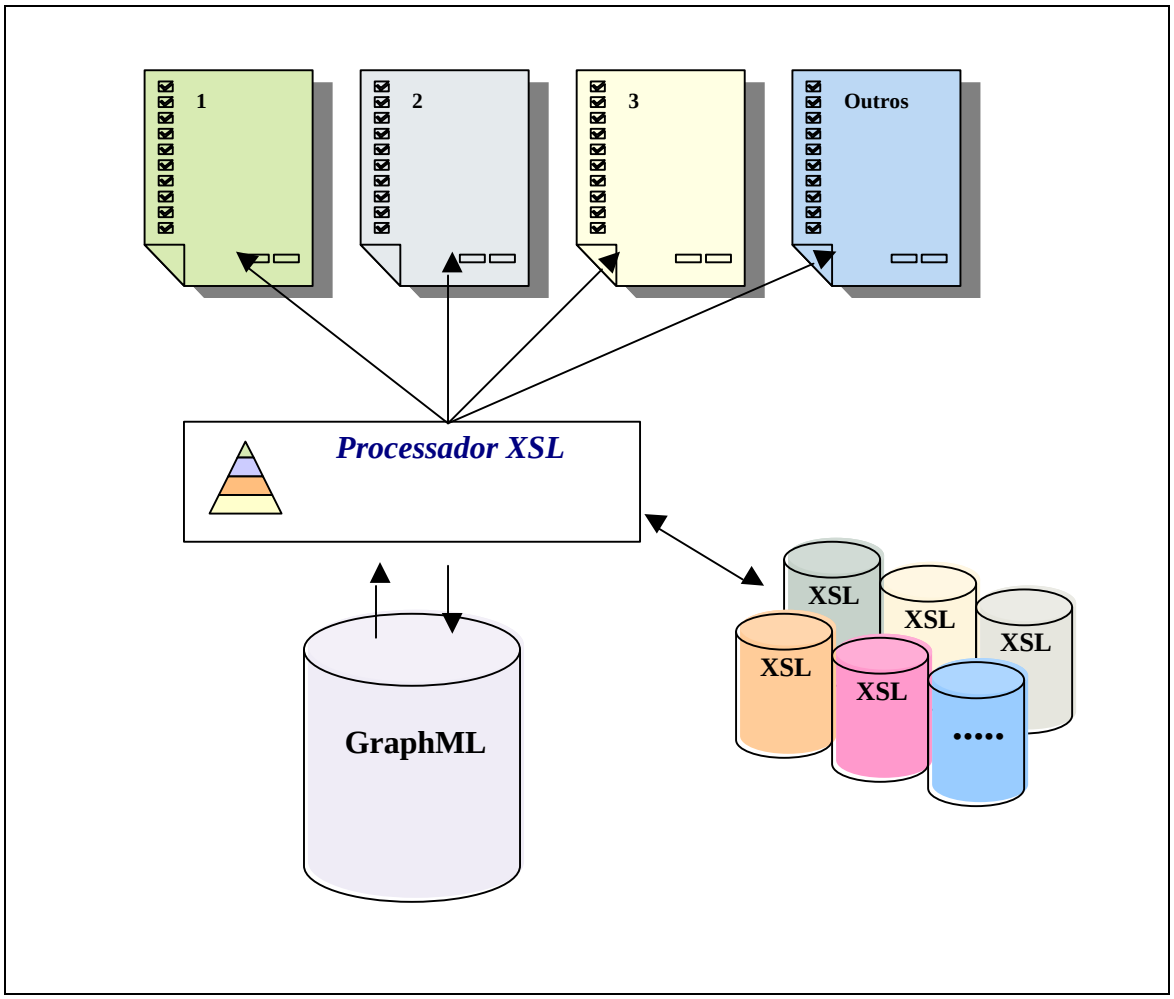


Figura 3.32 Metodologia GraphML – Folhas de estilos específicas.

3.3.3.5 Saída de Dados

A saída de dados é o resultado do processamento do processador XSL, dada uma base de dados em GraphML gerando conteúdo em formato SVG. A Saída de dados é apresentada na Figura 3.33.

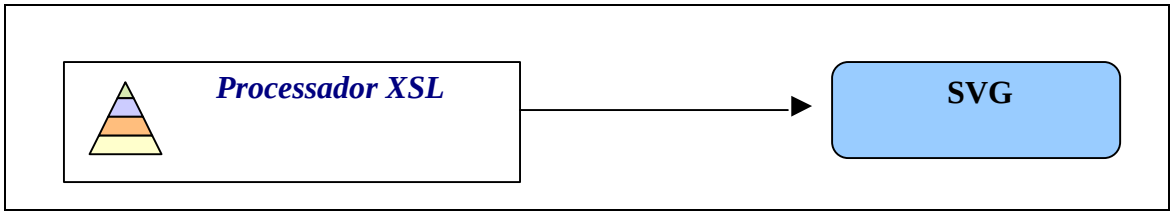


Figura 3.33 Metodologia GraphML – Saída de Dados.

3.3.3.6 Processador de Saídas

Conforme já foi mencionado, geralmente, tecnologias que estendem as funcionalidades da linguagem HTML necessitam de mecanismos externos para viabilizar a visualização de seus conteúdos em um navegador web. Tais mecanismos são referenciados como *plugin´s*, . Os *plugin´s*, associados com diversas tecnologias, são utilizados até que os navegadores desenvolvam suporte nativo para essas tecnologias. Para visualizar conteúdo codificado em SVG é necessário o uso do *plugin Adobe SVG plugin* [ADO 03b]. O Processador de Saídas pode ser observado na Figura 3.34.

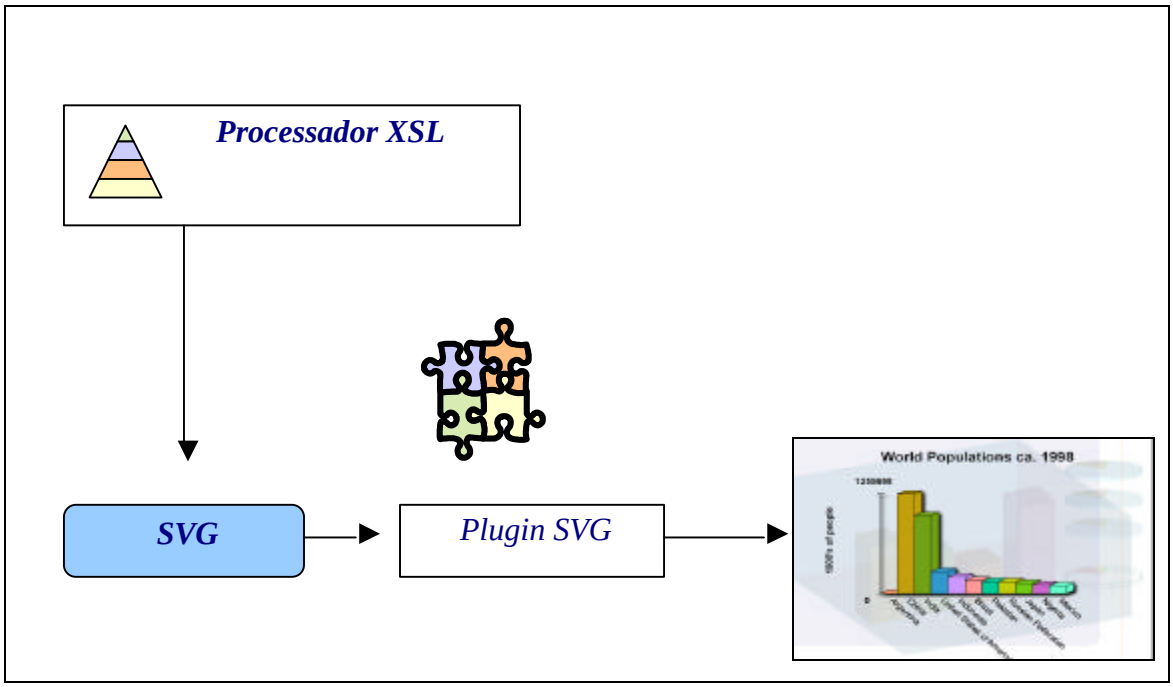


Figura 3.34 Metodologia GraphML – Processador de Saídas.

3.3.4 Proposta de um modelo GraphML reutilizável

Uma possível expansão para o modelo GraphML seria a modelagem de dados de uma forma genérica permitindo a reutilização das bases de dados. Enquanto que cada base GraphML é específica para um gráfico específico, onde armazena-se os dados dos gráficos e informações indicando o tipo de gráfico estatístico a ser renderizado, uma nova linguagem poderia ser desenvolvida de forma a separar os dados das informações de renderização, da seguinte maneira :

Supõe-se uma linguagem que descreva apenas os dados dos gráficos, podendo ser denominada “DataML”. Assim, GraphML descreveria apenas informações referentes à renderização dos gráficos. Este processo é apresentado através da Figura 3.35.

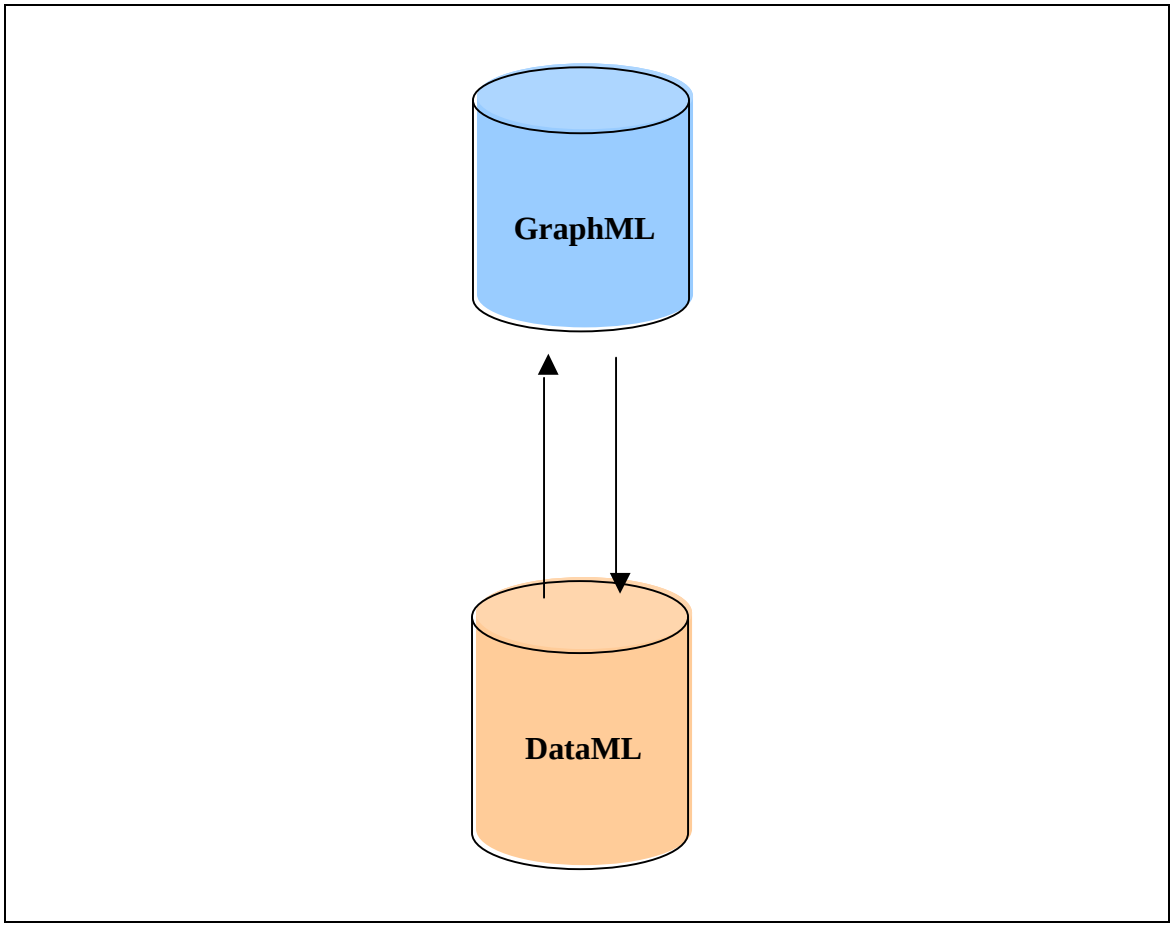


Figura 3.35 Metodologia GraphML Reutilizável – Linguagem DataML.

Dessa forma, uma mesma base de dados (DataML) poderia ser reutilizada com diferentes renderizações através do compartilhamento de várias bases de dados GraphML. Por exemplo, uma mesma base de dados poderia ser representada por diferentes tipos de gráficos, como barras, pizza, etc. Outra forma de reutilização poderia

fazer uso da mesma base de dados para um mesmo tipo de gráficos, mas com especificações diferentes como intervalos, limites, etc. A Figura 3.36 apresenta o compartilhamento de dados em DataML.

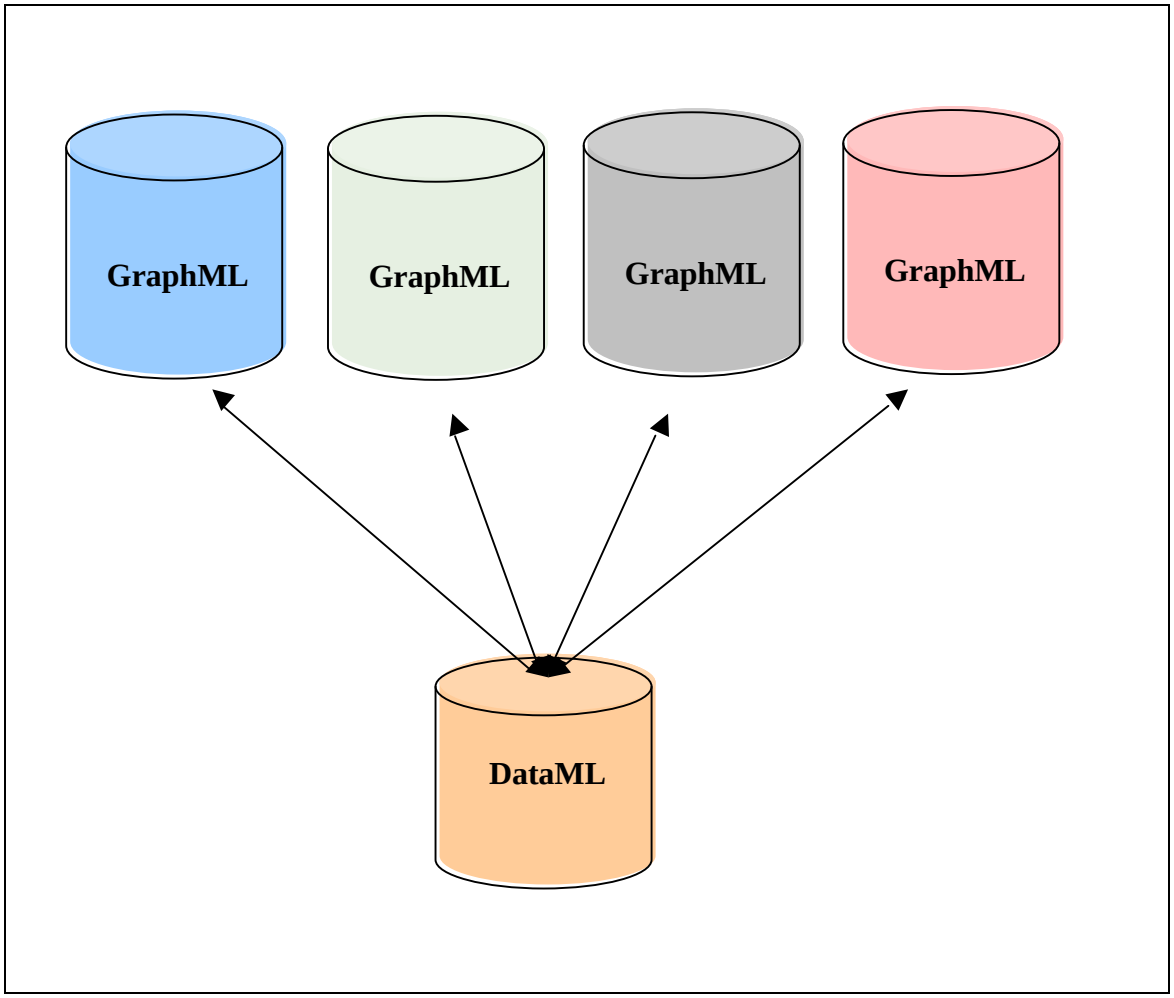


Figura 3.36 Metodologia GraphML Reutilizável – Compartilhando dados DataML.

O resultado final desta estratégia seria a utilização de vários gráficos diferentes dirigidos à mesma base de dados. A manutenção dos documentos baseados neste modelo torna-se-ia muito simples uma vez que bastaria alterar os dados em DataML que todos os gráficos que compartilham estas informações automaticamente seriam atualizados. Para completar este modelo, uma folha de estilos genérica, para qualquer tipo de gráfico estatístico, tornaria-se necessária. O modelo completo, resultando na reutilização de conteúdos gráficos, como acima citado, é mostrado na Figura 3.37.

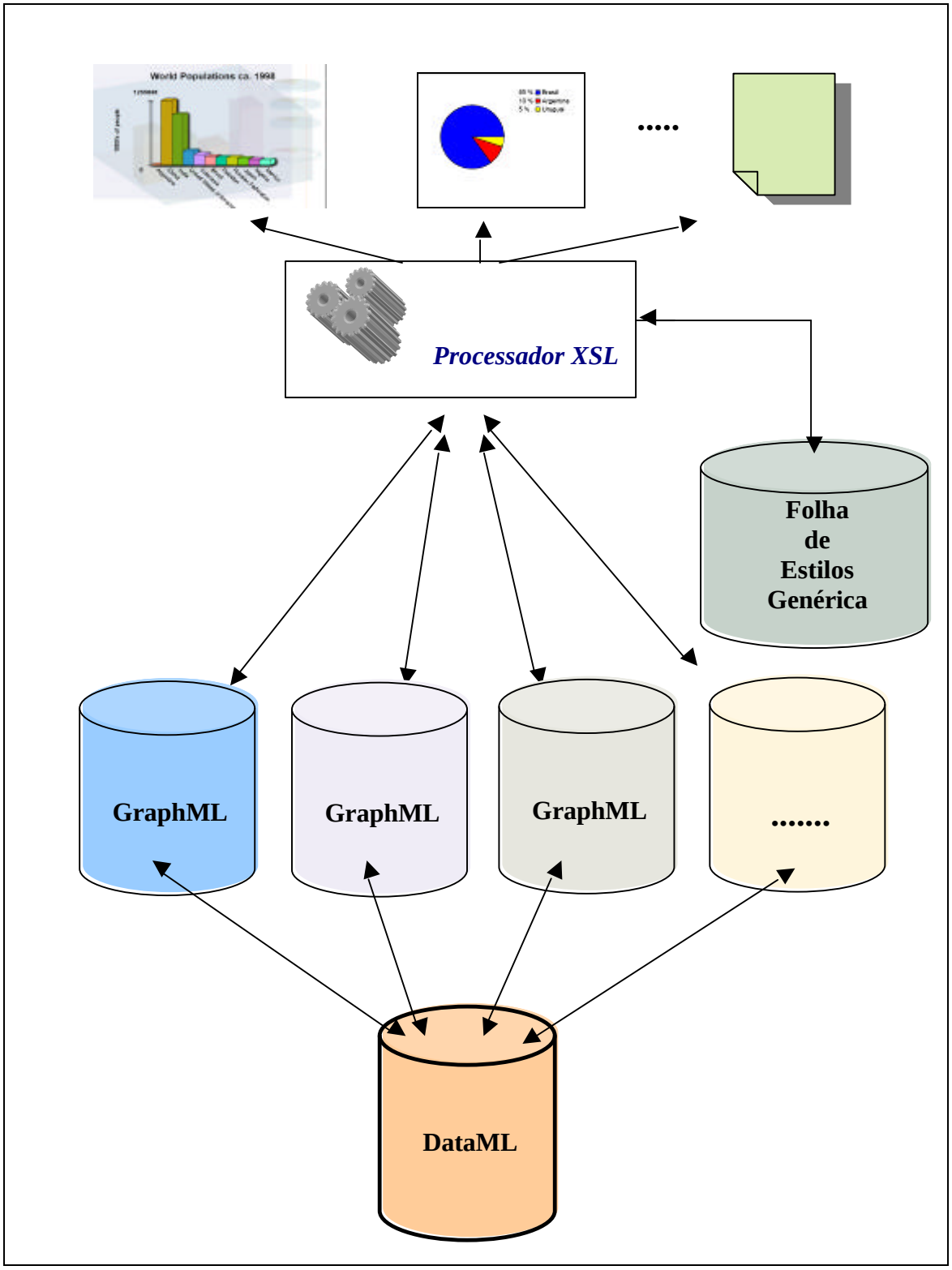


Figura 3.37 Metodologia GraphML Reutilizável – Processo Geral.

Muitas aplicações poderiam tirar grande proveito dos objetos gráficos reutilizáveis aqui modelados, uma vez que o problema de reutilização de conteúdos é emergente [WS 03]. A Metodologia GraphML Reutilizável não foi desenvolvida, sendo apenas uma proposta para trabalhos futuros.

3.4 Integração das metodologias propostas

O framework desenvolvido propõe uma estratégia de integração de diversos tipos de conteúdos. Através desta integração, outros softwares de autoria desses diferentes conteúdos assumem a posição de *plug-ins,* onde uma ferramenta gerenciadora de conteúdos unificaria todos os documentos. Na Seção 4.1.5 e Seção 4.1.7, serão descritas duas ferramentas de autoria que servem como *plug-ins* para o software Gerenciador de Hipermídias, que será descrito na Seção 4.1.3. Enquanto que o Editor de Fórmulas e o Editor de Gráficos fornecem a capacidade de gerar fórmulas e gráficos, respectivamente, o Gerenciador de Hipermídias oferece a facilidade de gerenciar, agrupar e integrar esses documentos dentro de um contexto maior. A integração das metodologias propostas é apresentada na Figura 3.38.

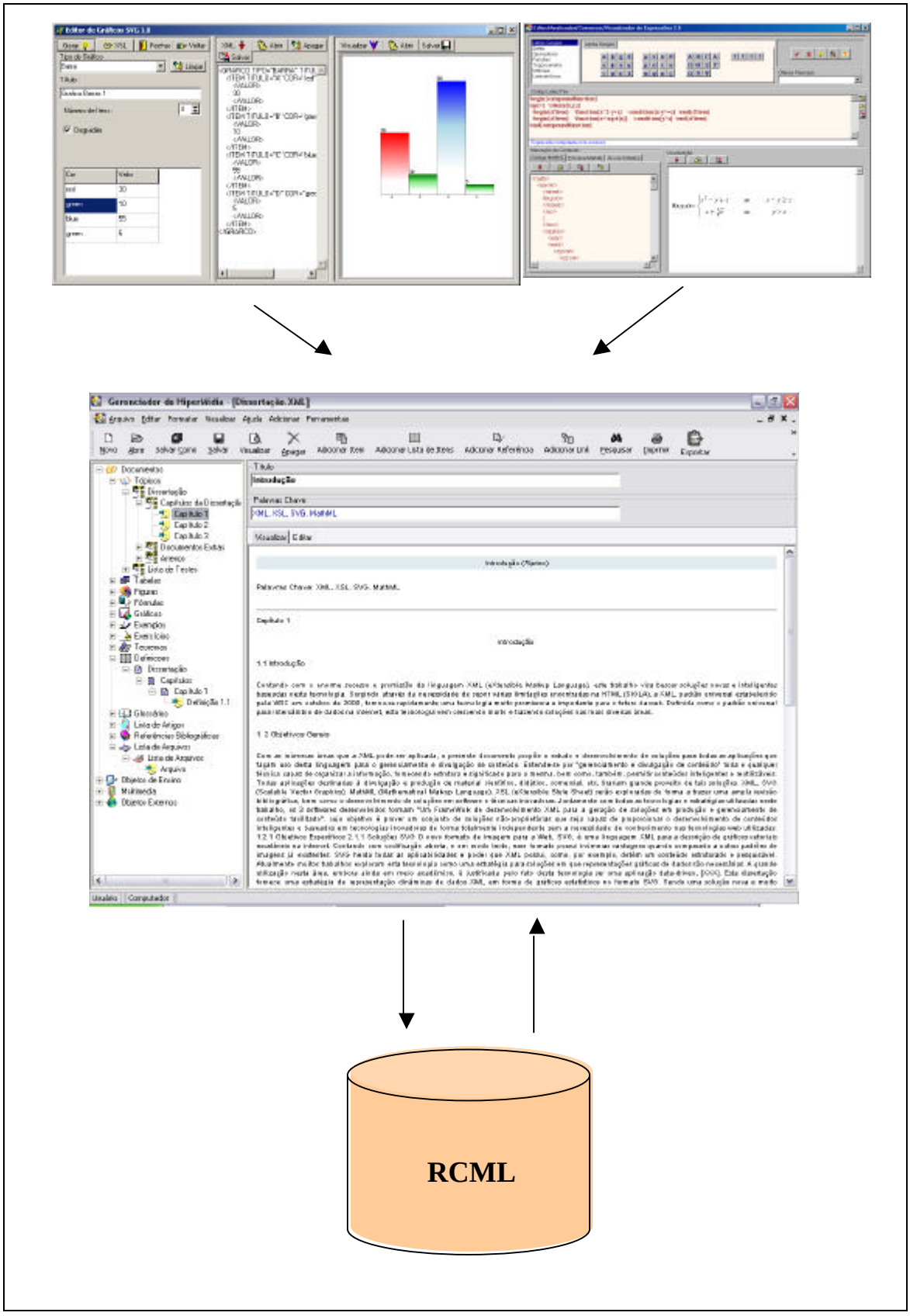


Figura 3.38 Integração de softwares *plug-in's*.

3.5 Conclusão

Este capítulo apresentou todas as metodologias propostas neste trabalho. A Metodologia RCM, Metodologia MathTeX e Metodologia GraphML, foram apresentadas em detalhes. As linguagens RCML, RCM Script e GraphML foram propostas. O Capítulo 4 apresenta estratégias de implementação e implantação destas metodologias e mostra os softwares que foram desenvolvidos para validar o modelo proposto : Gerenciador de Hipermídias, Editor de Gráficos e Editor de Fórmulas.

Capítulo 4

Implementação do Modelo Proposto

Este capítulo mostrará algumas soluções e estratégias de implementação que viabilizam a implantação das metodologias propostas. As ferramentas desenvolvidas para a validação das metodologias propostas também serão apresentadas.

4.1.1 Uma estratégia para implementação de bibliotecas geradoras de código XML : Modelo XMLMaker

4.1.1.1 Introdução

Para a obtenção de um mecanismo completo para a geração de conteúdos em formato XML, como, por exemplo, conteúdo GraphML ou RCML, será necessário fornecer uma estratégia de implementação para a geração de código XML.

Esta estratégia torna-se especialmente útil no desenvolvimento de ferramentas de autoria com suporte XML. O objetivo principal é fornecer uma hierarquia de classes geradoras de código nesta linguagem.

Uma biblioteca de classes geradoras de código XML irá propor uma estrutura padrão para o desenvolvimento de mecanismos geradores de código XML, tornando, assim, muito mais fácil o desenvolvimento de editores específicos. Uma biblioteca geradora que segue este padrão terá todal reusabilidade de código podendo ser utilizada para o desenvolvimento de inúmeras aplicações. O modeo descrito abaixo baseia-se totalmente no paradígma de orientação à objetos [WIN 93], através do qual procura-se descrever toda a hierarquia de elementos de uma determinada linguagem.

4.1.1.2 A Estrutura Hierárquica

Uma vez que a estruturação de conteúdos de documentos XML possui hierarquia em forma de árvore, torna-se mais fácil estruturar documentos genéricos e extrair dados para públicos-alvo diferentes. A interpretação e caminhamento nessa estrutura são feitos através de bibliotecas específicas para esta tarefa, como, por exemplo, o DOM (*Document Object Model*) [W3C 03c].

A definição da estrutura hierárquica de um documento XML é feita através de uma DTD. A Figura 4.1 mostra um exemplo de código GraphML e a Figura 4.2 ilustra sua hierarquia.

```
<Grafico Tipo="Pizza">

     <Item Titulo="Brasil" Cor="azul">

       <Valor>

        85

        </Valor>

    /Item>

    <Item Titulo="Argentina" Cor="vermelho" >

        <Valor>

        10

        </Valor>

    </Item>

    <Item Titulo="Uruguai" Cor="amarelo" >

        <Valor>

        75

        </Valor>

        </Item>

</Grafico>
```

Figura 4.1 Gerando XML – Exemplo de código GraphML.

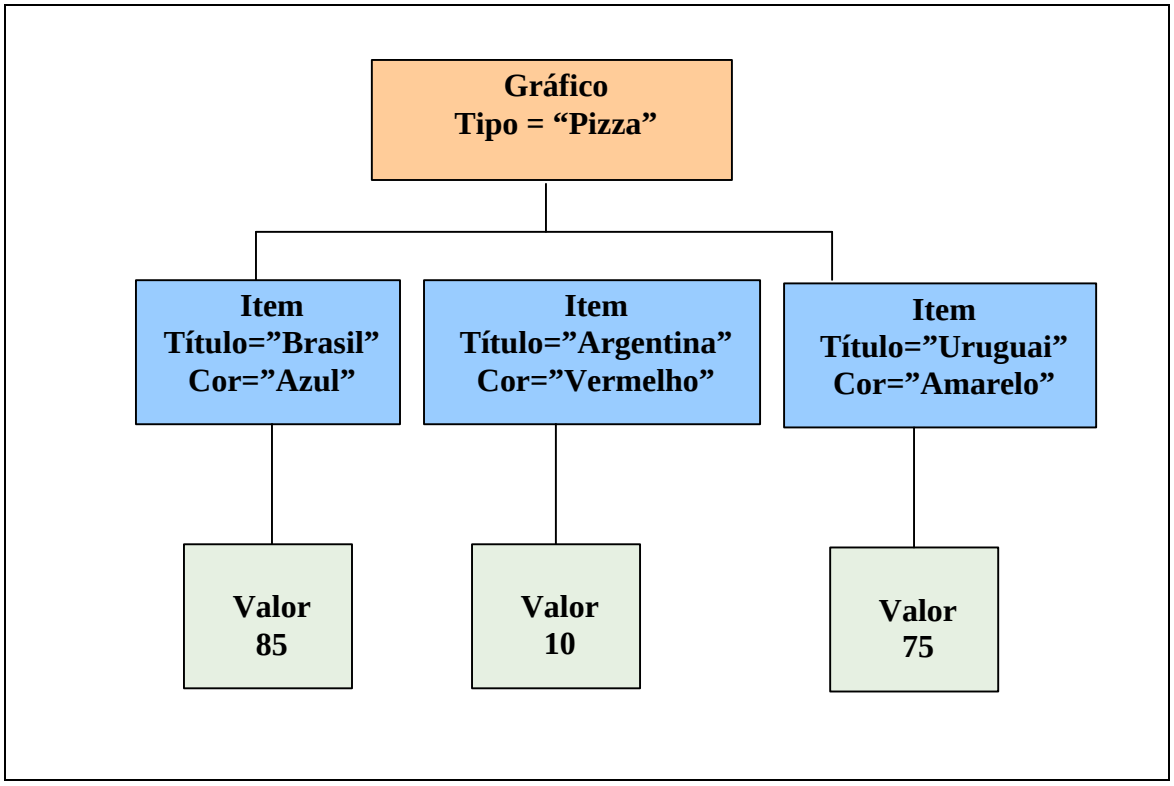


Figura 4.2 Gerando XML – Estrutura hierárquica de um documento GraphML.

Conforme apresentado na Figura 4.2, o documennto GraphML, Figura 4.1, é mapeado na forma de uma árvore, com cada elemento representando um nó. Os nós são blocos de construção básicos de todas as árvores, e a árvore XML contém **nós elementos** (Item, por exemplo), **nós atributos** (Descricao, por exemplo), e **nós textos** (Valor, por exemplo). O elemento “raíz” do exemplo acima é o nó elemento Grafico, que corresponde ao elemento de documento XML.

4.1.1.3 Estratégia de Implementação

A estratégia proposta faz com que a partir da estrutura acima sejam criadas bibliotecas através do paradígma de orientação a objetos seguindo a mesma ordem hierárquica.. Assim, para cada nó elemento de uma determinada estrutura XML, cria-se uma classe, e para cada nó atributo cria-se um atributo para a classe criada.

Em um primeiro passo, a **fase descritiva**, apenas observa-se cada ocorrência de elementos e atributos, criando-se as classes e os atributos correspondentes. Após esta fase, torna-se necessário fornecer a essas classes a ordem hierárquica. A hierarquia utilizada é de agregação, ou seja, objetos que **contém** objetos, da mesma forma que elementos (XML) contém elementos. Assim, para cada elemento que contiver

subelementos, agrega-se, para cada subelemento, um objeto da classe correspondente. Com isso, tem-se uma biblioteca de classes com a mesma estrutura hierárquica de um determinado documento XML.

É importante ressaltar que, para a obtenção dessas bibliotecas, será necessário fazer somente o exposto acima. Supõe-se que todos os processos de interpretação de documentos, geração de documentos e utilização do DOM já estão implementados em classes bases das quais as que estão sendo criadas irão herdar.

Esta estratégia é capaz de agilizar o desenvolvimento de editores XML específicos. Por exemplo, ao se criar um padrão XML (ontologia), seria interessante e necessário o desenvolvimento de um editor que criasse e gerenciasse esse código. Com o modelo proposto, o desenvolvedor irá preocupar-se apenas com a interface gráfica deste editor, pois todo o gerenciamento do código XML para esta situação será criado automaticamente.

Um código que utiliza esta estrategia para a produção do documento GraphML, exposto na Figura 4.1, é mostrado na Figura 4.3.

```
App.Begin_Grafico(“Pizza”)
  App.Grafico.Begin_Item(“Brasil”,”blue”)
    App.Grafico.Item.Begin_Valor
      App.Grafico.Item.Valor.Write(85)
    App.Grafico.Item.End_Valor
  App.Grafico.End_Item
  App.Grafico.Begin_Item(“Argentina”,”red”)
    App.Grafico.Item.Begin_Valor
      App.Grafico.Item.Valor.Write(10)
    App.Grafico.Item.End_Valor
  App.Grafico.End_Item
  App.Grafico.Begin_Item(“Uruguai”,”yellow”)
  App.Grafico.Item.Begin_Valor
```

```
    App.Grafico.Item.Valor.Write(75)

  App.Grafico.Item.End_Valor

  App.Grafico.End_Item

App.End_Grafico
```

Figura 4.3 Gerando XML – Gerando código GraphML.

4.1.2 Estratégias de implementação RCM

4.1.2.1 Introdução

O propósito desta seção é fornecer estratégias de implementação para a produção de biliotecas com suporte à metodologia RCM. O desenvolvimento destas bibliotecas será independente dos softwares que poderão utiliza-las. Isso traz muitas possibilidades de novas aplicações serem desenvolvidas com objetivos diferentes. As bibliotecas aqui referenciadas possibilitarão o desenvolvimento de qualquer ferramenta capaz de manipular e estruturar dados segundo a modelagem proposta. Serão mostradas, detalhadamente, todas as estratégias de implementação que deverão ser utilizadas para a obtenção de bons resultados com a implantação destas bibliotecas em determinados projetos de desenvolvimento de software.

Qualquer ferramenta que vise a utilização de algum mecanismo capaz de manipular e estruturar dados (segundo a metodologia RCM), pode ser desenvolvida utilizando os seguintes recursos, que foram desenvolvidos neste trabalho, conforme Capítulo 3.

a. Processador RCML :

Como descrito na Seção 3.1.5.3, o processador RCML tem, como entrada, um conjunto de dados estruturados neste formato e armazena-o em uma base RCML. Para a realização destre processo, o Processador RCML utiliza os recursos oferecidos pelo DOM (*Document Object Model*) [W3C 03], através da estratégia descrita na Seção 4.1.1. O Processador RCML possui as seguintes funcionalidades : importar dados estruturados; gerar dados em RCML, importar informações RCML e gerar saídas em um outro formato estruturado.

b. Interpretador de código :

Na Seção 3.1.5.3 observou-se que um interpretador de código deverá ser utilizado para a verificação léxica, sintática e semântica de código RCM Script. Neste trabalho, foi desenvolvido um interpretador que se divide em duas fases : Análise e Execução de Código. A fase de Análise e subdividida em Análise Léxica, Análise Sintática e Análise Semântica.

c. Processador XSL :

Para a manipulação de dados XML através do uso de XSL, o Processador XSL utiliza recursos do Processador RCML através de solicitação de informações. Tendo em vista o modelo de implementação aqui referenciado, o Processador XSL é visto como um método da classe que implementa o Processador RCML.

d. Formatador RCML :

Esta ferramenta funciona em colaboração com o Processador RCML, cujo objetivo é o percurso em uma base de dados nesta linguagem e a conversão destes dados em um determinado formato de saída.

Todos os itens acima citados executarão a parte funcional de um software desenvolvido para fins de manipulação de dados RCML. Restará apenas a necessidade de implementação de uma interface gráfica para acesso às bibliotecas. Várias interfaces gráficas, com diferentes aplicações, poderão fazer uso destas bibliotecas ao mesmo tempo. O Modelo de Implementação pode ser visualizado na Figura 4.4.

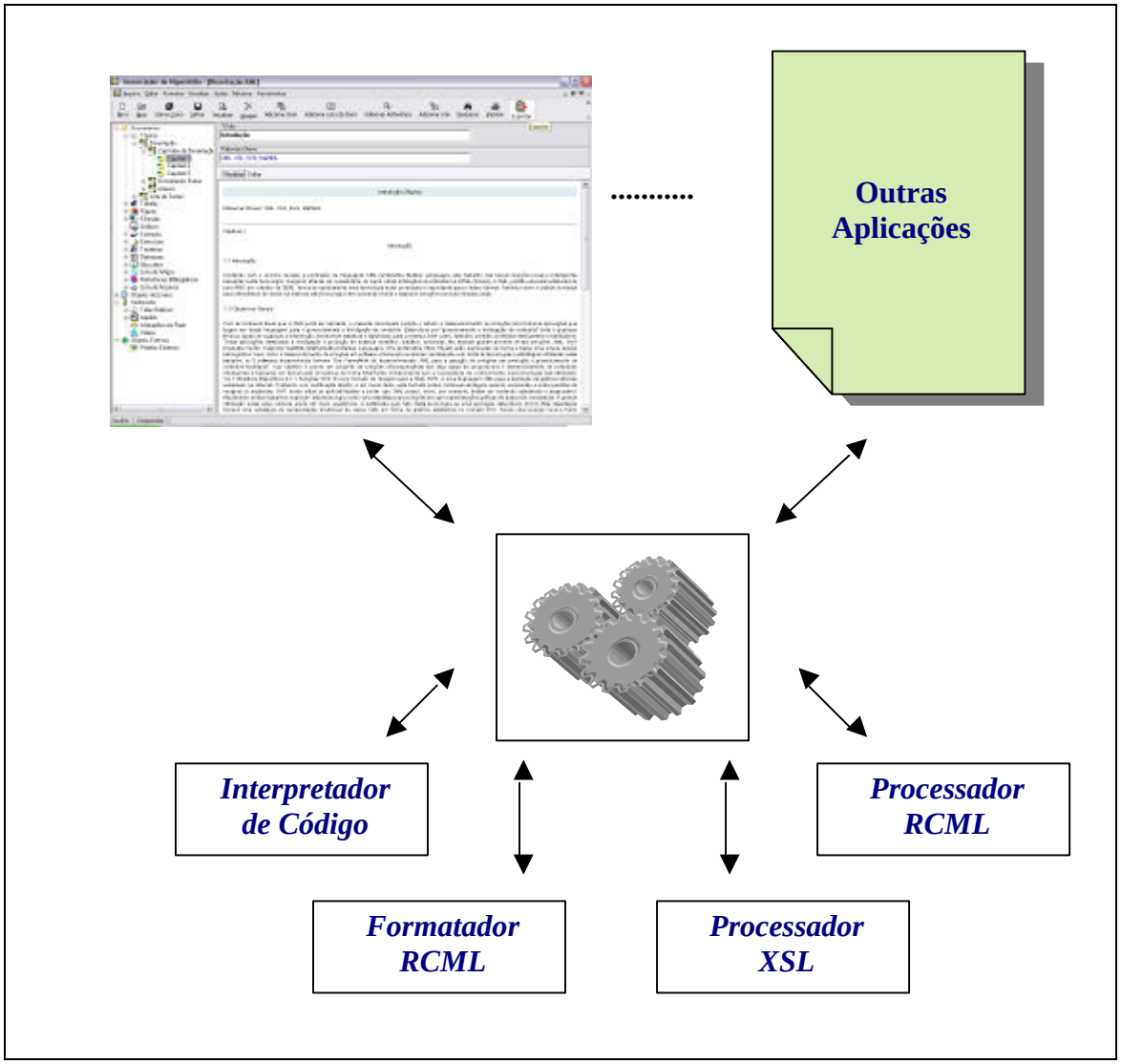


Figura 4.4 Estratégias de Implementação RCM - Modelo de Implementação RCM.

4.1.2.2 Processador RCML

Herdando toda a funcionalidade provida pela *api* DOM [W3C 03c], através da estratégia descrita na Seção 4.1.1, o Processador RCML pode ser implementado como uma classe chamada TGerenciador, que, juntamente com uma hierarquia de classes desenvolvidas para toda a linguagem RCML (Anexo V), é capaz de gerar, importar e manipular dados nesta linguagem. Alguns métodos desta classe constituem outras ferramentas como o Formatador RCML e o Processador XSL.

A classe TGerenciador pode ser assim denominada por realizar a produção e *gerenciamento* de qualquer tipo de conteúdo em formato RCML.

A proposta da classe TGerenciador é a obtenção de uma biblioteca totalmente independente da aplicação, podendo, assim, ser utilizada em qualquer ferramenta que necessite o gerenciamento de dados em formato RCML. Seguem, na Tabela 4.1, os passos para a realização deste procedimento. Supõe-se a utilização do ambiente de programação Delphi 6.0.

Ordem	Ação	Descrição
1	**Var Gerenciador : TGerenciador**	**Declaração do objeto Gerenciador**
2	**Gerenciador := TGerenciador.Create**	**Instanciação do objeto Gerenciador**
3	**Gerenciador.....**	**Utilização dos métodos**

Tabela 4.1 Estratégias de Implementação RCM – Processador RCML.

A Figura 4.5 apresenta um trecho de código utilizado para a realização do processo acima referenciado.

```
....

Gerenciador := TGerenciador.Create(TreeView);

Gerenciador.Begin_Projeto;

Gerenciador.Projeto.Begin_Objetos_Dados;

Gerenciador.Projeto.Objetos_Dados.Begin_Lista_Recursos;

Gerenciador.Projeto.Objetos_Dados.Lista_Recursos[0].Begin_Recurso;

Gerenciador.Projeto.Objetos_Dados.Lista_Recursos[0].Titulo := ‘Teste’;

Gerenciador.Projeto.Objetos_Dados.Lista_Recursos[0].Texto := ‘Texto’;

Gerenciador.Filename := ‘c:\base.xml’;

Gerenciador.Save;

.....
```

Figura 4.5 Estratégias de Implementação RCM – Processador RCML.

4.1.2.3 Interpretador de Código :

Este processo se divide em duas fases : Análise e Execução de código. Para isto, um método da classe TGerenciador deve ser desenvolvido. Alguns outros métodos também podem ser adicionados para prover a execução do código RCM Script. Juntos, todos os métodos desenvolvidos formam o Interpretador de Código, com todas as capacidades necessárias para a análise e execução de um determinado trecho de código

em RCM Script. O método que implementa o Interpretador de Código deve retornar os dados resultantes da execução das instruções. Por exemplo, para uma referência a outros fragmentos (comando REF), o resultado da invocação deste método é um texto formado pela concatenação de todas as referências expandidas. Seguem, na Tabela 4.2, os passos para a realização deste procedimento. Supõe-se a utilização do ambiente de programação Delphi 6.0. O método "Convert" será referenciado como o método que implementa o Interpretador de Código.

Ordem	Ação	Descrição
1	**Var Gerenciador : TGerenciador**	**Declaração do objeto Gerenciador**
2	**Gerenciador:TGerenciador.Create**	**Instanciação do objeto Gerenciador**
3	**Gerenciador.Convert(Codigo)**	**Invocação do Interpretador de Código**

Tabela 4.2 Estratégias de Implementação RCM – Interpretador de Código.

A Figura 4.6 apresenta um trecho de código utilizado para a realização do processo acima referenciado.

```
....

Gerenciador := TGerenciador.Create(TreeView);

Result.Add(Gerenciador.Convert(Memo.Lines.Text).SavetoFile('Result.xml
'))

....
```

Figura 4.6 Estratégias de Implementação RCM – Interpretador de Código.

4.1.2.4 Processador XSL.

Outro método da classe TGerenciador deve ser desenvolvido para realizar a tarefa de aplicar uma certa folha de estilos XSL a um determinado fragmento RCML que esta sendo manipulado pelo Processador RCML através da classe Tgerenciador. Este procedimento é realizado através da utilização de mecanismos providos pala api DOM, através do modelo descrito na Seção 4.1.1. O resultdo deste método é um documento transformado pela folha de estilos. Seguem, na Tabela 4.3, os passos para a realização deste procedimento. Supõe-se a utilização do ambiente de programação Delphi 6.0.

Ordem	Ação	Descrição
1	**Var Gerenciador : TGerenciador**	**Declaração do objeto Gerenciador**
2	**Gerenciador:TGerenciador.Create**	**Instanciação do objeto Gerenciador**
3	**Gerenciador.ProcessXSL**	**Invocação do Interpretador de Código**

Tabela 4.3 Estratégias de Implementação RCM – Processador XSL.

A Figura 4.7 apresenta um trecho de código utilizado para a realização do processo acima referenciado.

```
....

Gerenciador := TGerenciador.Create(TreeView);

Gerenciador.ProcessXSL(Folha.XSL).SavetoFile('Result.txt')

.....
```

Figura 4.7 Estratégias de Implementação RCM – Processador XSL.

4.1.2.5 Formatador RCML

O Formatador RCML é outro método integrado com o Processador RCML, podendo ser chamado de "Format". Para cada tipo de formatação é necessário um método separado. Assim, para a formatação dos dados em HTML, pode ser desenvolvido um método chamado "TOHTML". A função do Formatador RCML é realizar, além da função de uma folha de estilos (transformar dados XML), também conversões de dados em formatos complexos, como DOC, PPT. O procedimento desta ferramenta consiste, basicamente, em percorrer a hierarquia de dados RCML e, através de solicitações do Processador RCML, obter dados e transformá-los em um formato desejado. Por exemplo, para a conversão de RCML em DOC, é necessária a utilização de alguma api capaz de gerar dados em formato DOC. O resultado deste método é o documento formatado. Seguem, na Tabela 4.4, os passos para a realização deste procedimento. Supõe-se a utilização do ambiente de programação Delphi 6.0.

Ordem	Ação	Descrição
1	**Var Gerenciador : TGerenciador**	**Declaração do objeto Gerenciador**
2	**Gerenciador:TGerenciador.Create**	**Instanciação do objeto Gerenciador**
3	**Gerenciador.Format**	**Invocação do Formatador XML**

Tabela 4.4 Estratégias de Implementação RCM – Formatador RCML.

A Figura 4.8 apresenta um trecho de código utilizado para a realização do processo acima referenciado.

```
....

Gerenciador := TGerenciador.Create(TreeView);

Gerenciador.Format(ftHTML).SavetoFile('Result.HTML')

....
```

Figura 4.8 Estratégias de Implementação RCM – Formatador RCML.

4.1.3 O Gerenciador de Hipermídia

Nesta seção será demonstrado o Gerenciador de Hipermídia, uma ferramenta que possui suporte total à Metodologia RCM.

4.1.3.1 Introdução

O Gerenciador de Hipermídias é uma ferramenta que faz uso de todos os recursos propostos neste trabalho para a geração e manipulação de conteúdos estruturados para a web, baseando-se no padrão RCML (Anexo V). Contando com a capacidade que XML fornece aos documentos de separar conteúdo de formatação, esta ferramenta vem prover tal facilidade. Através de um mecanismo de classificar diferentes tipos de conteúdos, torna-se claro o fácil gerenciamento das informações, bem como o seu reaproveitamento, tornando o gerenciamento e atualização de documentos muito simples.

O principal objetivo da ferramenta é a produção de documentos destinados à web, mas, entretanto, vários mecanismos de conversões e exportações para outros formatos de documentos, como DOC e PPT, poderão ser incluídos em suas funcionalidades. A Figura 4.9 ilustra o modelo geral do Gerenciador de Hipermídias.

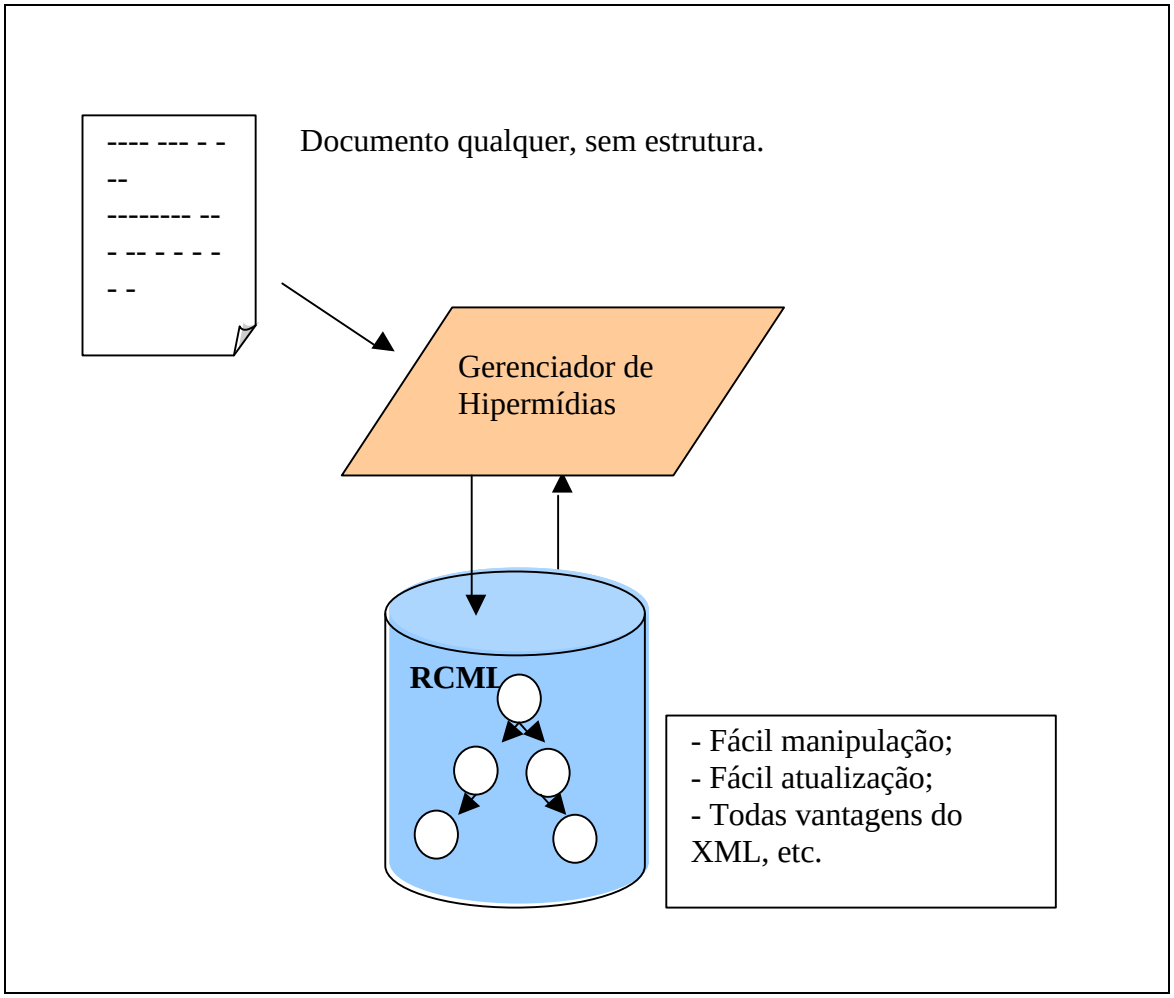


Figura 4.9 Gerenciador de Hipermídias – Modelo geral.

Todo o conteúdo gerenciado por este software é armazenado em um repositório de dados RCML, possibilitando futuros intercâmbios com outras aplicações, bem como a importação de todas as vantagens que XML provê.

Mecanismos de pesquisa através de palavras-chave também são suportados permitindo a busca de qualquer tipo de conteúdo.

Baseadas em XML, as informações podem ser manipuladas através de XSL para diferentes propostas, como, por exemplo, visualização, transformação, filtragem dos dados, etc. O gerenciador de Hipermídias incorpora suporte total à linguagem RCM Script.

Através de uma interface intuitiva que representa cada nó em sua forma hierárquica (árvore), este aplicativo oculta a manipulação de arquivos, ou seja, torna-se transparente a existência de diferentes arquivos de dados para qualquer tamanho de

documento. Assim, a produção e o gerenciamento de *links* tornam-se claros e independentes de arquivos. A proposta desta ferramenta é que todos os dados sejam orientados aos “nós” na árvore XML, e não orientados aos arquivos, como acontece em qualquer editor HTML existente.

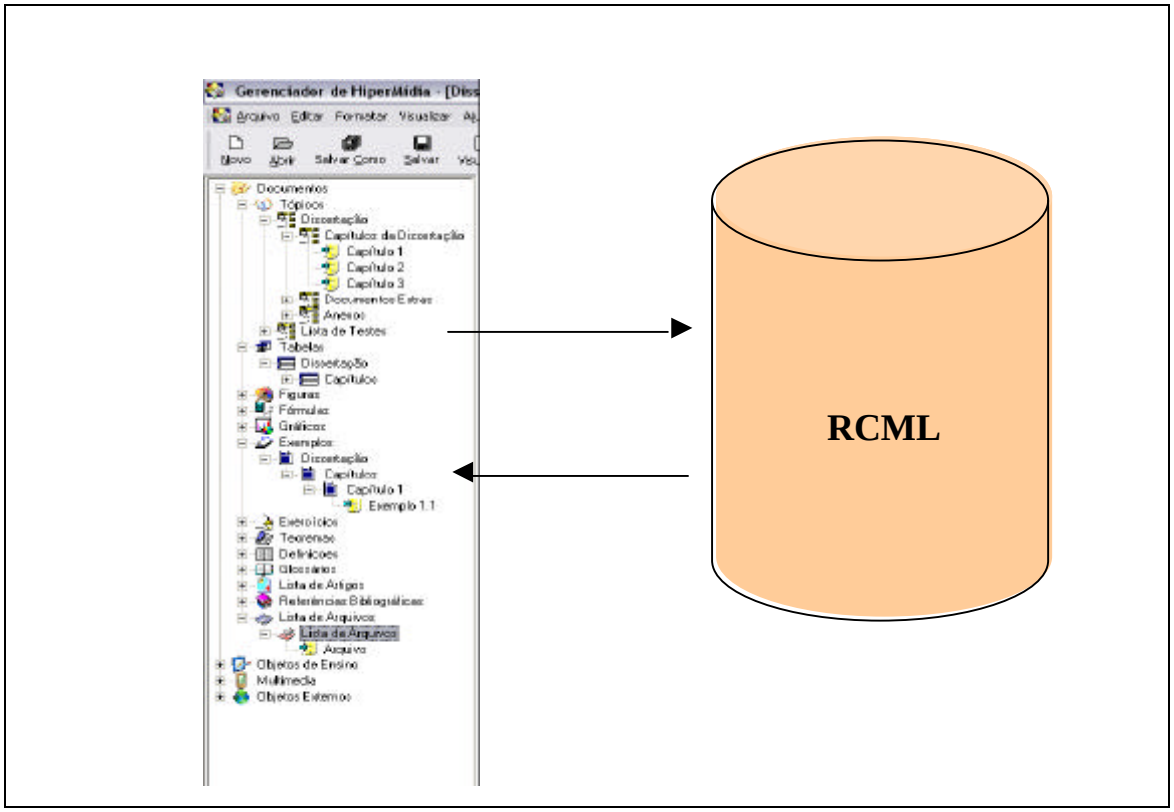


Figura 4.10 Gerenciador de Hipermídias – Estrutura Hierárquica do conteúdos.

A Figura 4.10 mostra como esta ferramenta representa visualmente a estrutura de dados RCML. Através desta interface é realizada a fase de Entrada de Dados da Metodologia RCM, conforme explicado na Seção 3.1.5.2.

O Gerenciador de Hipermídias não é um editor HTML, mas muito mais que isso. A proposta desta ferramenta é ser um editor e gerenciador de conteúdos, podendo ser utilizado para a produção de vários formatos, como HTML, XML, PDF, CHS. Além de suas capacidades de edição de documentos, poderosos mecanismos de manipulação e atualização tornam-se evidentes através da importação de todas as facilidades que XML oferece e da utilização de XSL e RCM Script.

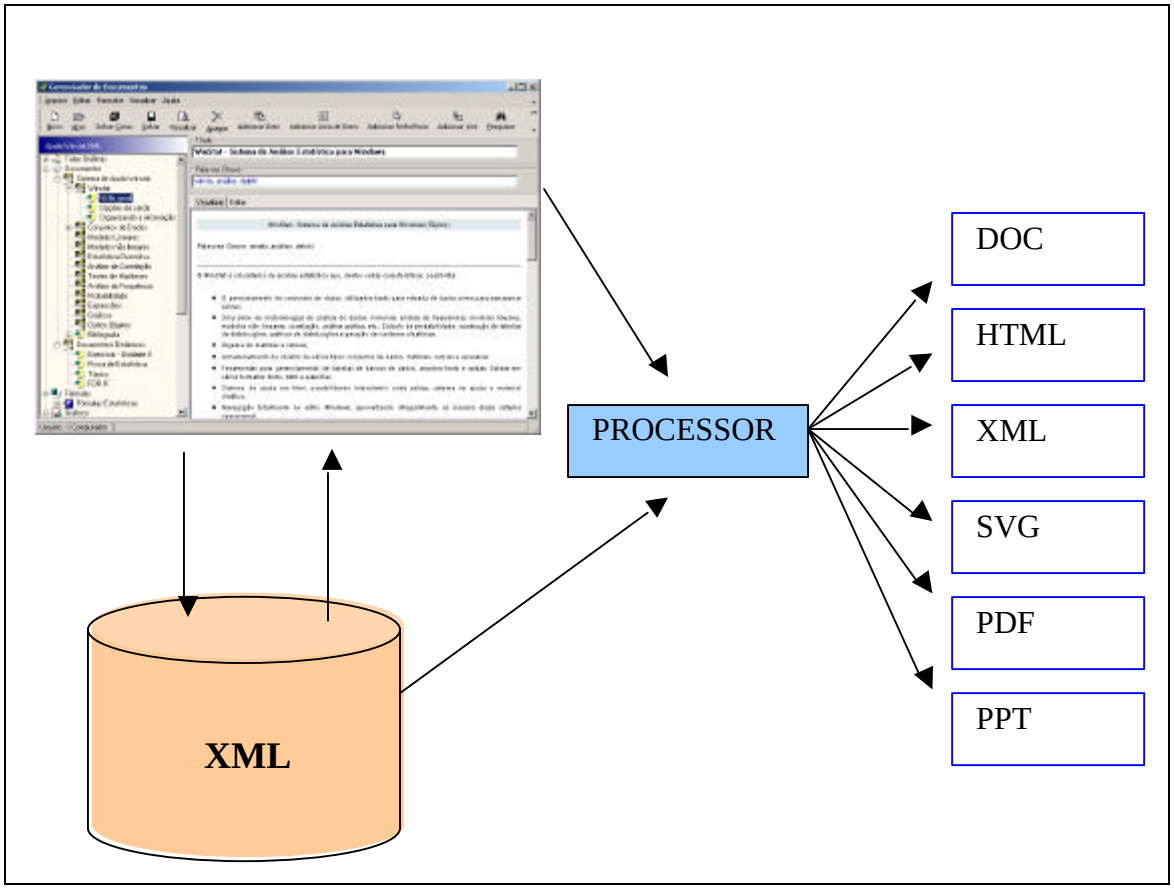


Figura 4.11 Gerenciador de Hipermídias – Modelo de exportação estática.

O Gerenciador de Hipermídias propõe dois tipos de exportação e/ou manipulação : estática e dinâmica. Na **exportação estática**, Figura 4.11, os dados são processados pela ferramenta e o resultado é gerado para uma determinada aplicação específica. Neste caso, se houver a necessidade de atualização dos dados, estes terão que ser gerados novamente e substituirão os anteriores. Já numa **exportação dinâmica**, os dados não são processados pela ferramenta, mas sim convertidos em uma base XML que será manipulada dinâmicamente através de folhas de estilos XSL. Quando houver a necessidade de alteração desses dados, bastará modificar-se diretamente esta base XML, através do software, sem a necessidade de substituição de conteúdos. A proposta da exportação dinâmica exigirá uma folha de estilos capaz de interpretar códigos RCM Script. Esta funcionalidade não foi implementada, deixando, assim, extensões para futuros trabalhos. A Figura 4.12 ilustra o modelo de exportação dinâmica.

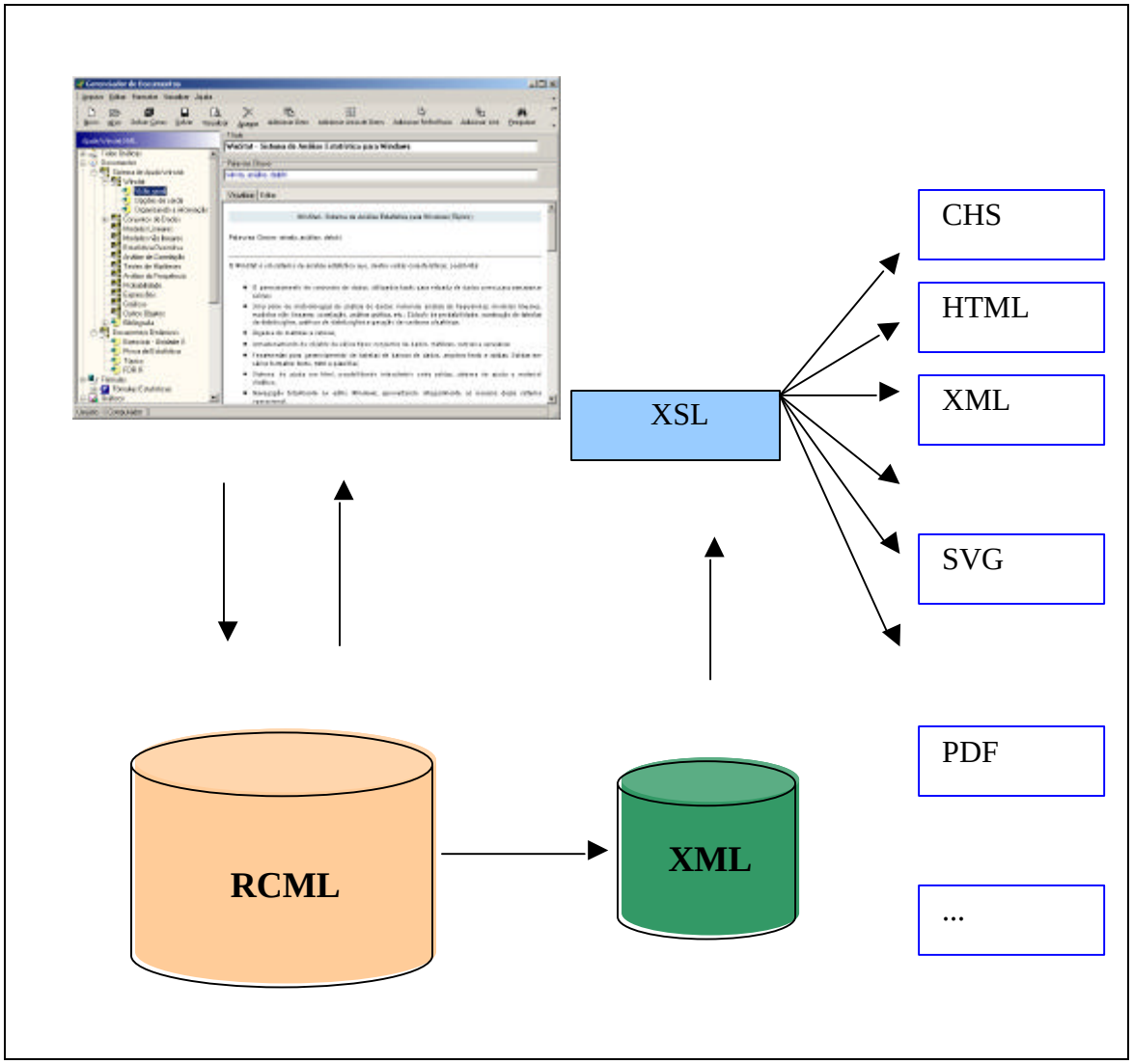


Figura 4.12 Gerenciador de Hipermídias – Modelo de exportação dinâmica.

O software proposto possui os seguintes componentes :

* Interface gráfica amigável, que representa os dados RCML em sua forma hierárquica permitindo a edição e visualização dos documentos;
* Poderosos mecanismos de busca de determinados fragmentos de documentos;
* Um analisador léxico e sintático de código RCML para a manipulação da informação;
* Um analisador léxico e sintático de código LaTeX para a manipulação de conteúdos matemáticos;
* Um analisador conversor de dados em LaTeX para MathML;

* Uma anialisador léxico, sintático e semântico para a linguagem RCM Script;
* Uma máquina virtual para a execução de código RCM Script;
* Mecanismos geradores e manipuladores de exportação e conversão de dados.
* Uma linguagem específica para a definição de modelos de exportação e conversão de dados.

A interface deste aplicativo pode ser dividida em :

* Janela Principal
* Janela Novo Documento
* Janela dos documentos
* Janela de Pesquisas
* Janela Adicionar Referências
* Janela Adicionar Links
* Janela Gerenciador de Exportações
* Janela Assistente de Exportação

4.1.3.2 Objetivos

* Reaproveitamento do conteúdo através de módulos reutilizáveis;
* Manipulação e geração dinâmica de documentos através de uma linguagem de script própria (RCM Script);
* Exportação para diversos formatos, como PDF, XML,HTML, CHS, SVG, DOC, PPT, etc.
* Prover a separação do conteúdo da formatação;
* Gerenciamento dinâmico e automático de diversas representações do mesmo conteúdo;
* Prover mecanismos para a geração de telas gráficas para a visualização dos conteúdos com suporte a todos os recursos HTML e suas extensões, como java, flash, etc
* Fácil gerenciamento e atualização de links, tornando a existência de arquivos transparentes para o usuário;

* Oferecer o máximo possível as funcionalidades de manipulação do XSL de uma forma muito mais simples e intuitiva.
* Permitir mecanismos de buscas através de palavras-chave para qualquer tipo de conteúdo

4.1.3.3 Interface Gráfica

* Janela Principal

O Gerenciador de Hipermídias pode manipular vários arquivos ao mesmo tempo dentro da mesma janela. Portanto, a janela principal deste aplicativo é um *container* para todos os arquivos que estiverem abertos ao mesmo tempo.

A Figura 4.13 mostra a janela sem nenhum arquivo aberto. Já a Figura 4.14 mostra a janela principal com vários arquivos abertos ao mesmo tempo.

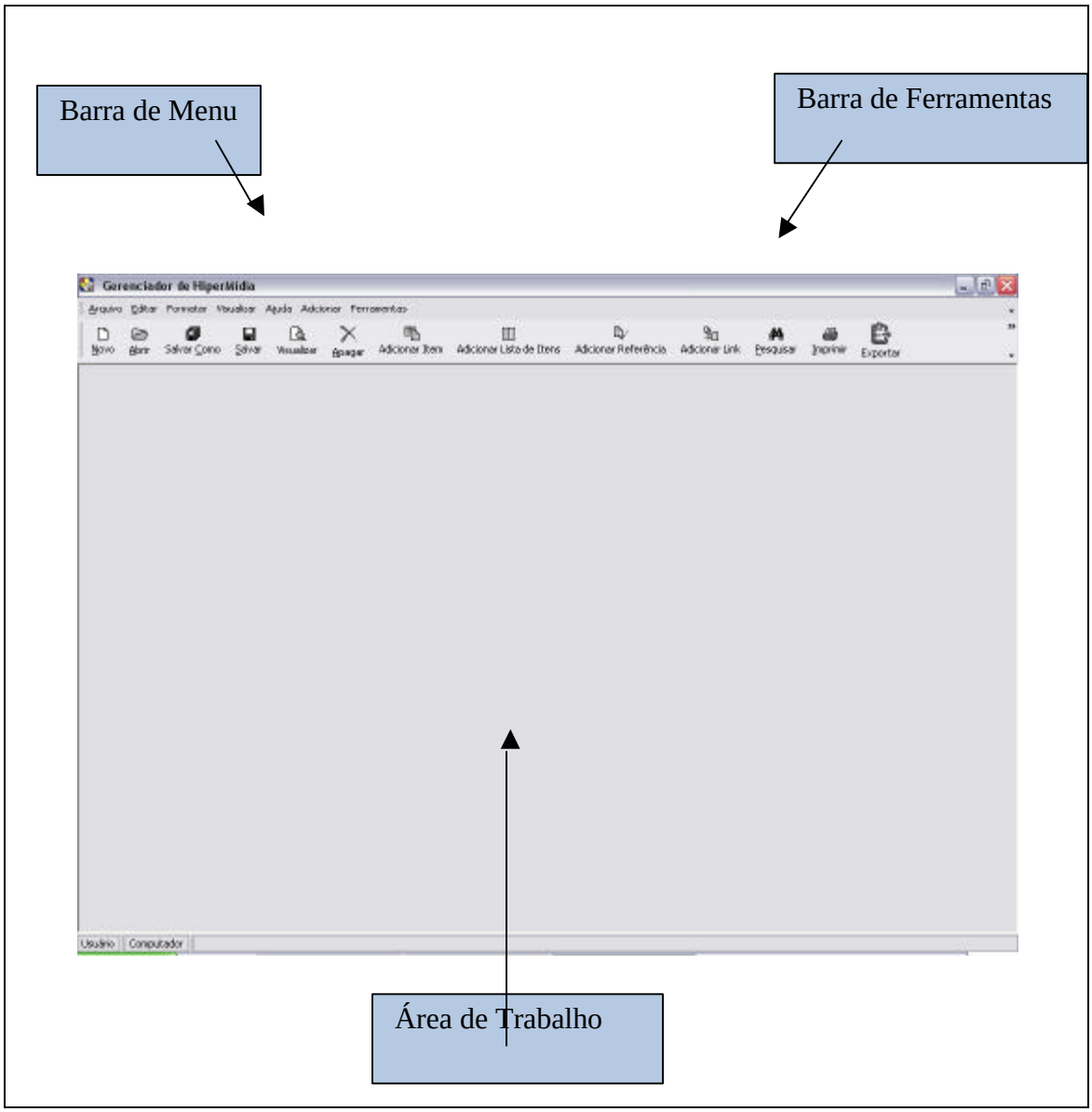


Figura 4.13 Gerenciador de Hipermídias – Interface Principal.

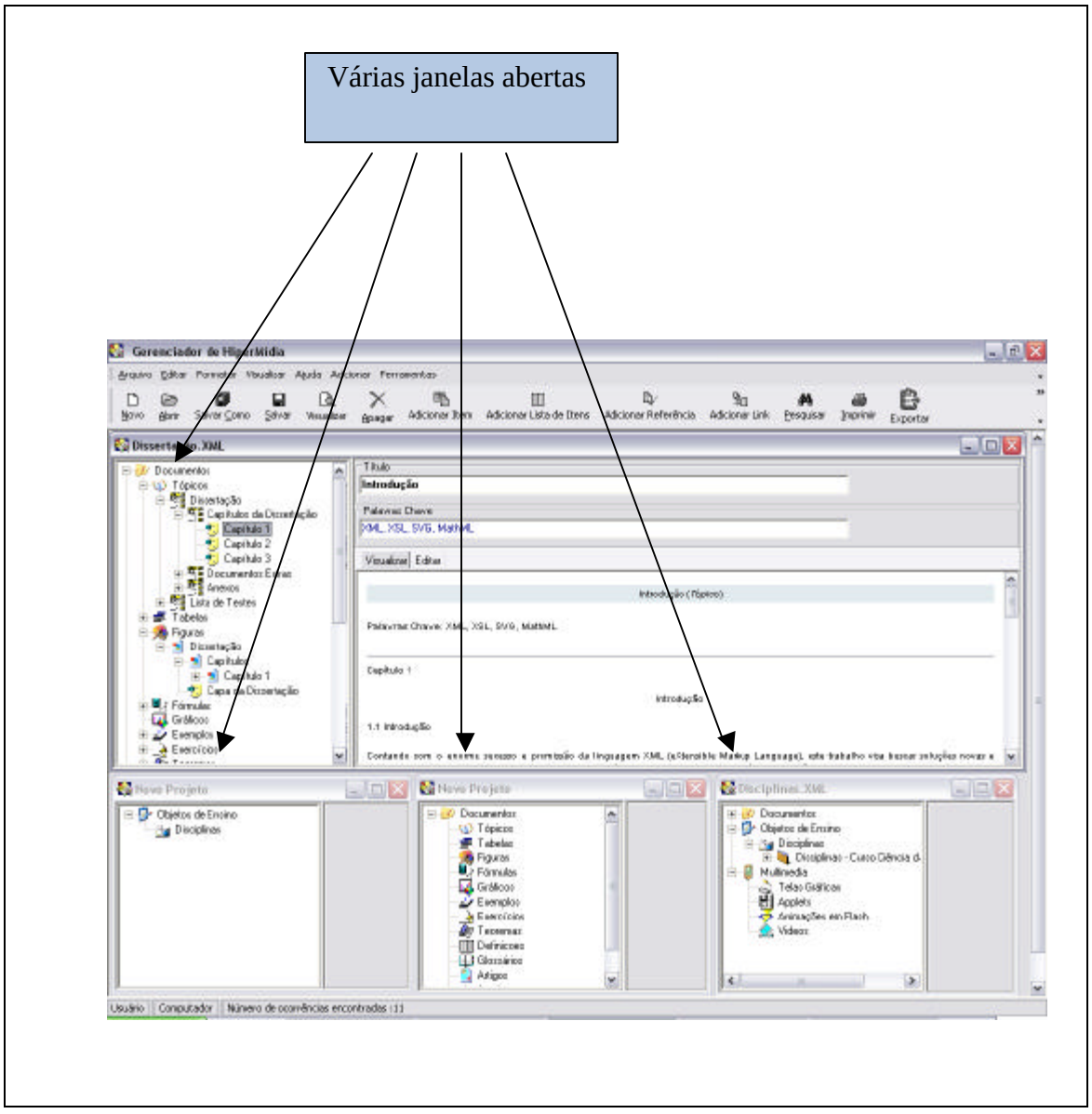


Figura 4.14 Gerenciador de Hipermídias – Várias janelas abertas.

* Janela Novo Documento

Para criar um novo projeto de conteúdos, é necessário utilizar o botão “Novo”. O gerenciador de conteúdos suporta vários tipos de conteúdos, que podem ser gerados separadamente conforme as seguintes opções que são disponibilizadas através da janela novo documento :

a. Documento Simples :

Possui apenas nós para armazenar Objetos de Dados e Multimedia.

b. Documento Completo :

Trabalha com todos os tipos de documentos suportados pelo Gerenciador de Conteúdos.

c. Objetos de Dados :

Trabalha apenas com os documentos relacionados com o objeto “ObjetosdeDados”.

d. Objetos de Estilos :

Trabalha apenas com os documentos relacionados com o objeto “ObjetosdeEstilos”.

e. Multimedia :

Trabalha apenas com os documentos relacionados com o objeto “Multimedia”.

f. Objetos Externos :

Trabalha apenas com os documentos relacionados com o objeto “Objetos Externos”.

A Figura 4.15 apresenta esta janela.

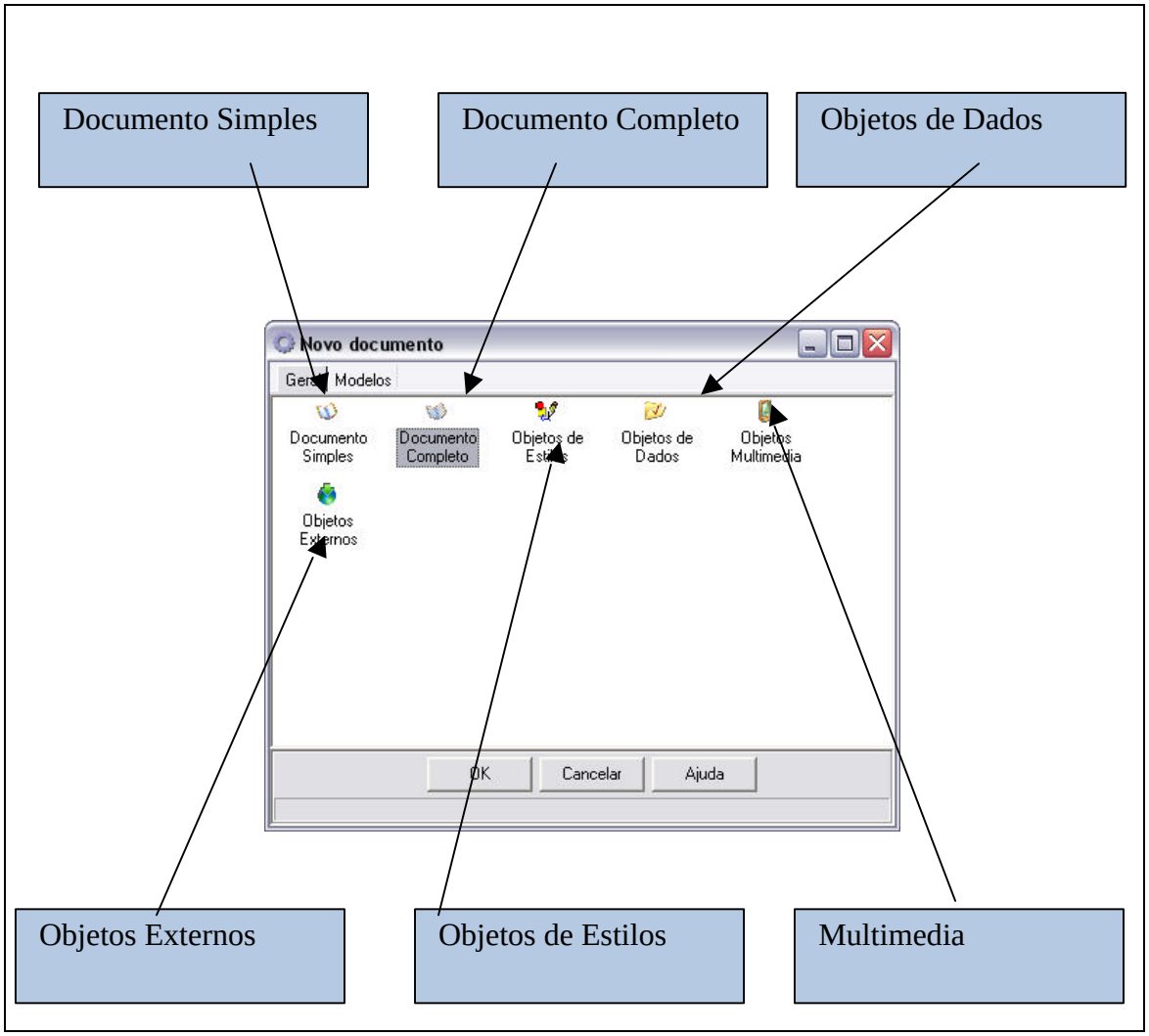


Figura 4.15 Gerenciador de Hipermídias – Novo Documento Geral.

* Janela de Documentos

Como explicado acima, a janela principal é o elemento *containter* para uma ou mais janelas de documentos. Cada janela de documento representa um conjunto de dados específicos formando um “Projeto Hipermídia”. Através desta janela é possível manipular os dados, pesquisá-los e várias outras funcionalidades que serão citadas abaixo. A Janela de Documentos pode ser observada na Figura 4.16.

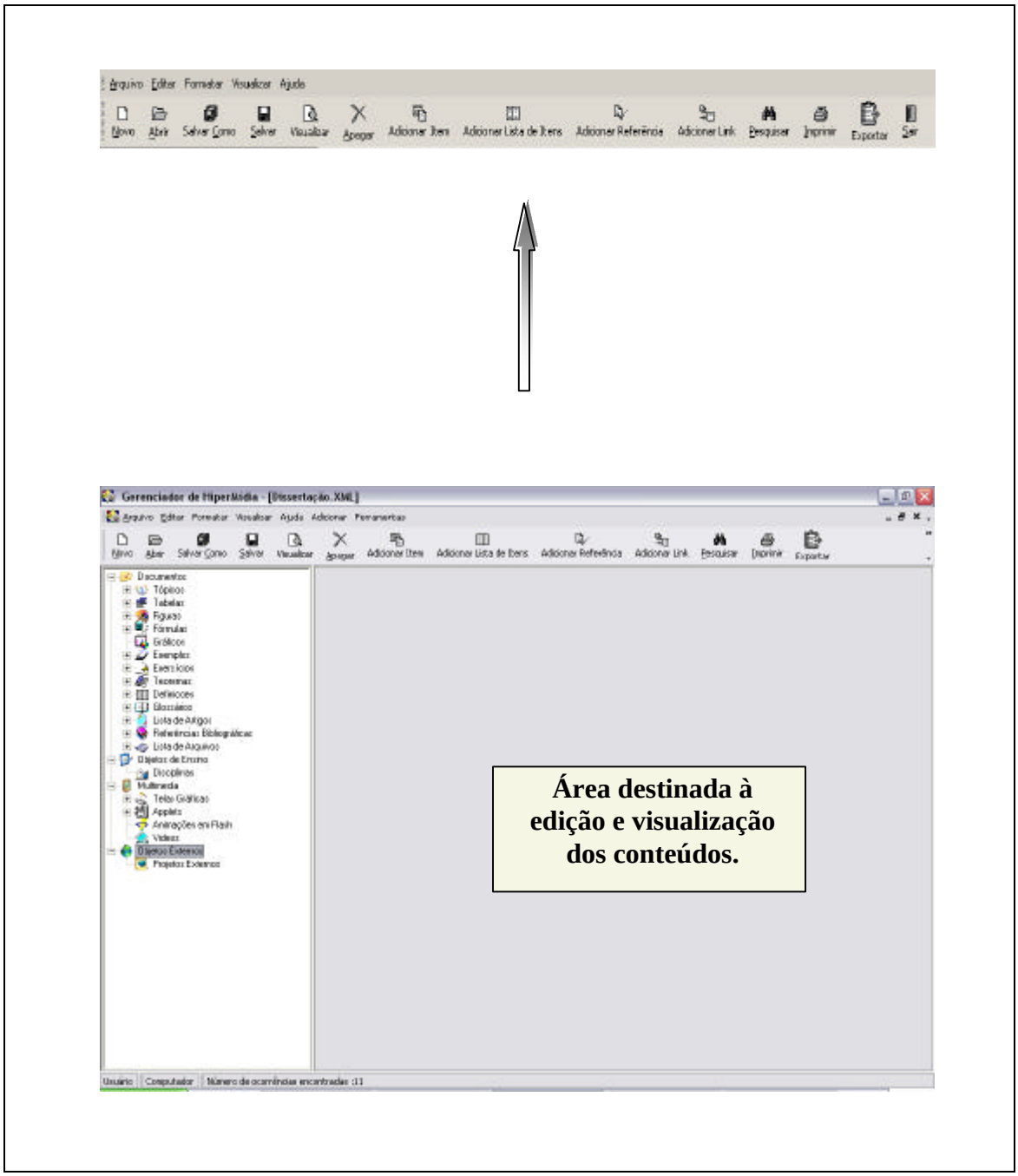


Figura 4.16 Gerenciador de Hipermídias – Janela de documentos.

Para cada tipo de conteúdo, uma interface própria para sua edição e visualização foi desenvolvida. Será demonstrada aqui, apenas a tela de edição (Figura 4.18) e visualização de documentos (Figura 4.17).

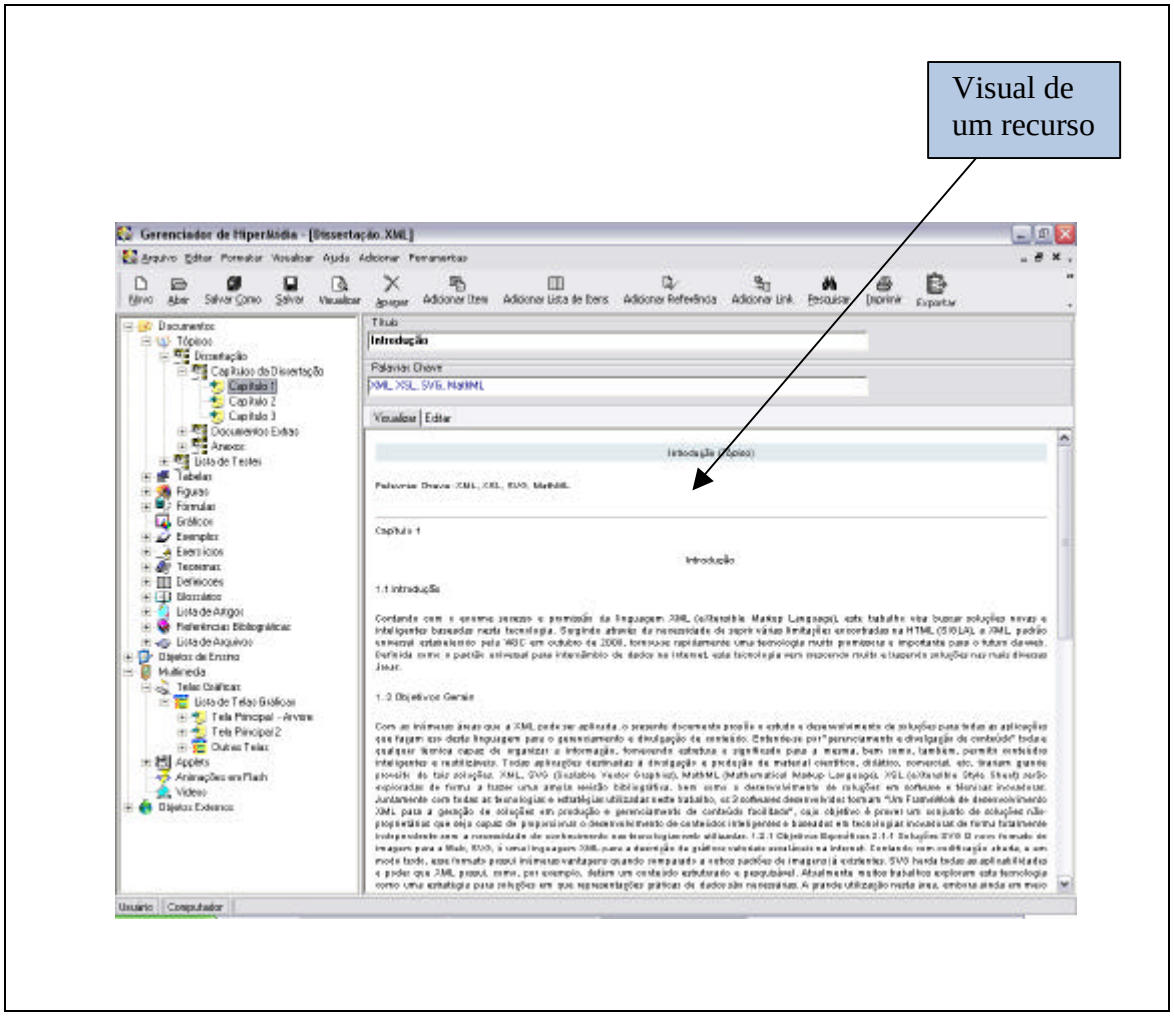


Figura 4.17 Gerenciador de Hipermídias – Visualizando conteúdos.

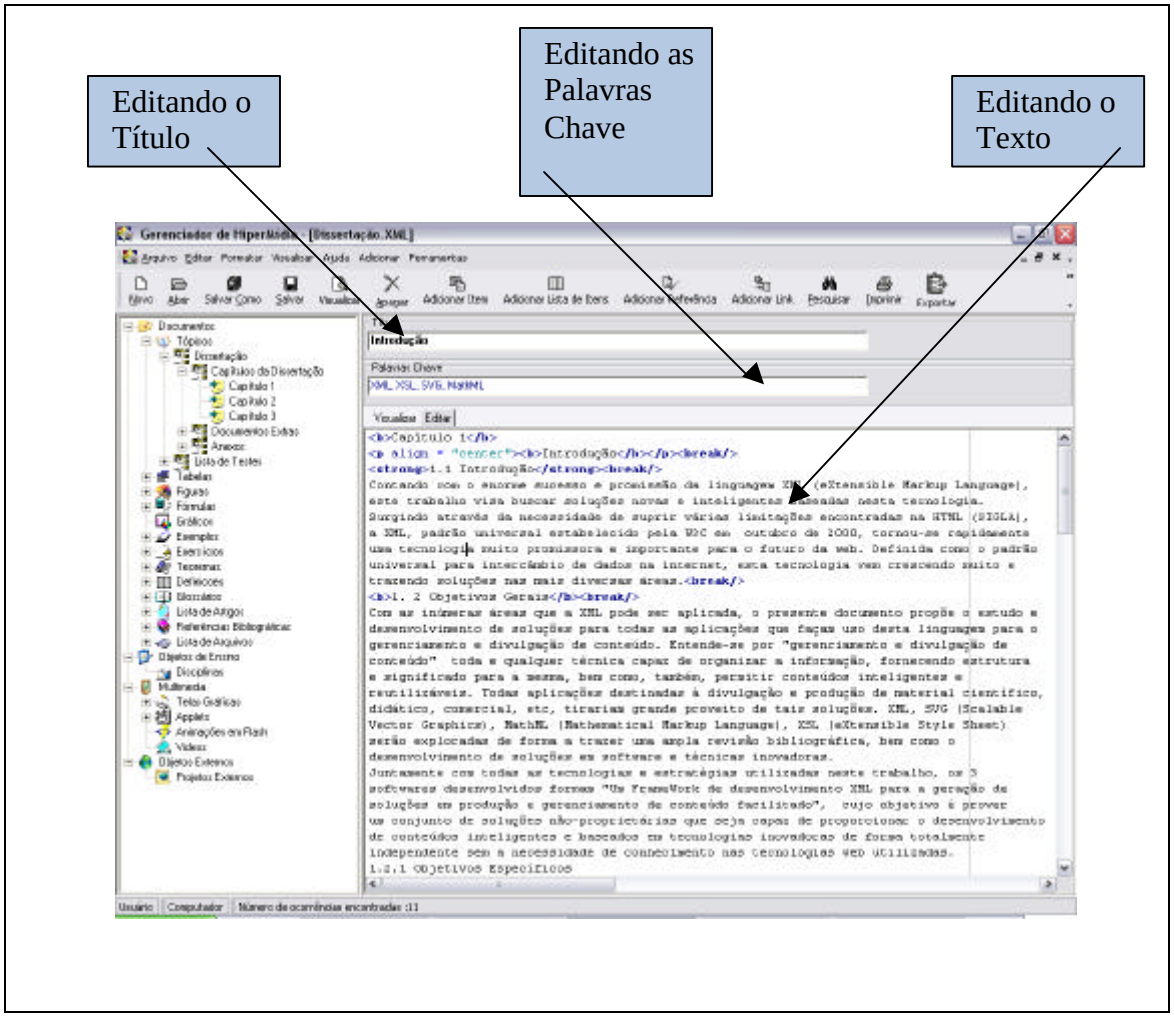


Figura 4.18 Gerenciador de Hipermídias – Editando conteúdos.

* Janela de Pesquisas

Poderosos mecanismos de pesquisa foram desenvolvidos para prover manipulações e buscas complexas de dados. É possível buscar documentos através de pesquisas por palavras chave e nome. Assim, uma listagem é mostrada com todos os resultados da pesquisa realizada. Três tipos de consulta podem ser realizados : “Iniciando com”, “Contendo Texto” e “Exatamente Igual” A Figura 4.19 ilustra a Janela de Pesquisas.

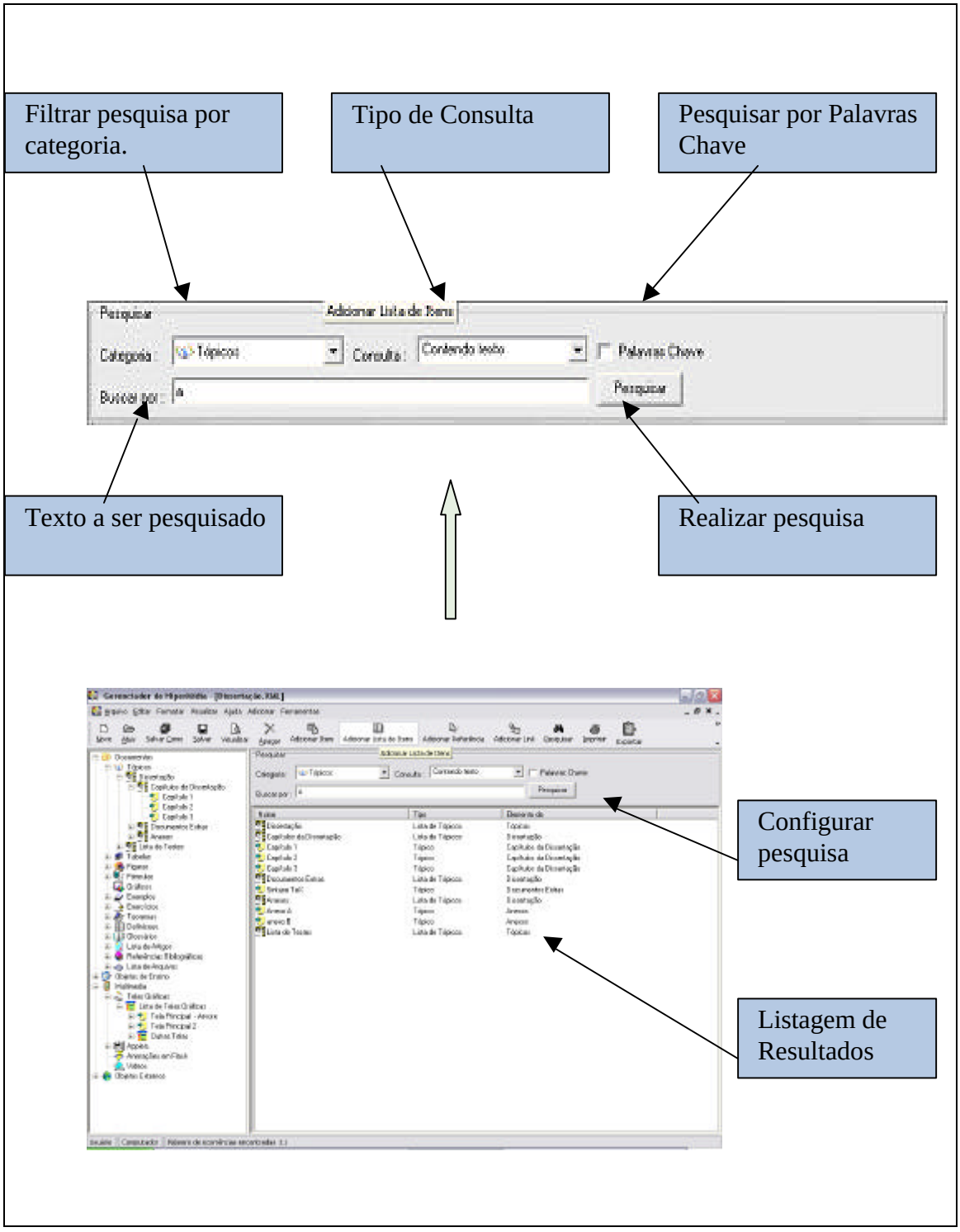


Figura 4.19 Gerenciador de Hipermídias – Janela de Pesquisas.

* Janela Adicionar Referências

A adição de referências em um determinado documento, pode ser realizada com o seguinte código : “<REF TIPO=”.....” ID=”....”/>”. Para facilitar a edição deste e de outros códigos, uma interface gráfica foi desenvolvida para gerar código automáticamente. Com isso, o usuário faz uma pesquisa buscando encontrar o item desejado, e o código fazendo referência a este objeto é então gerado pelo software. Segue na Figura 4.20 a ilustração desta janela.

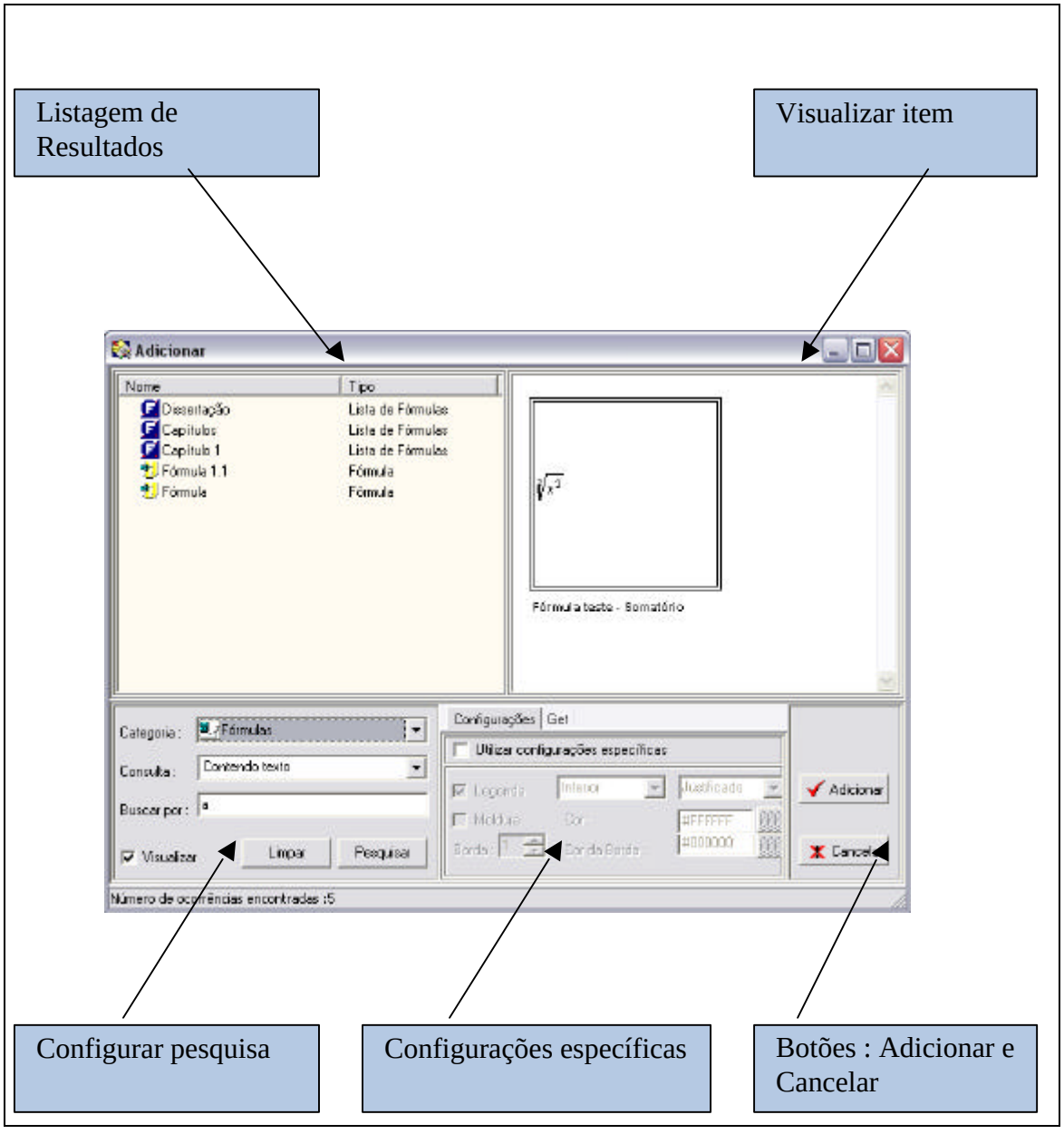


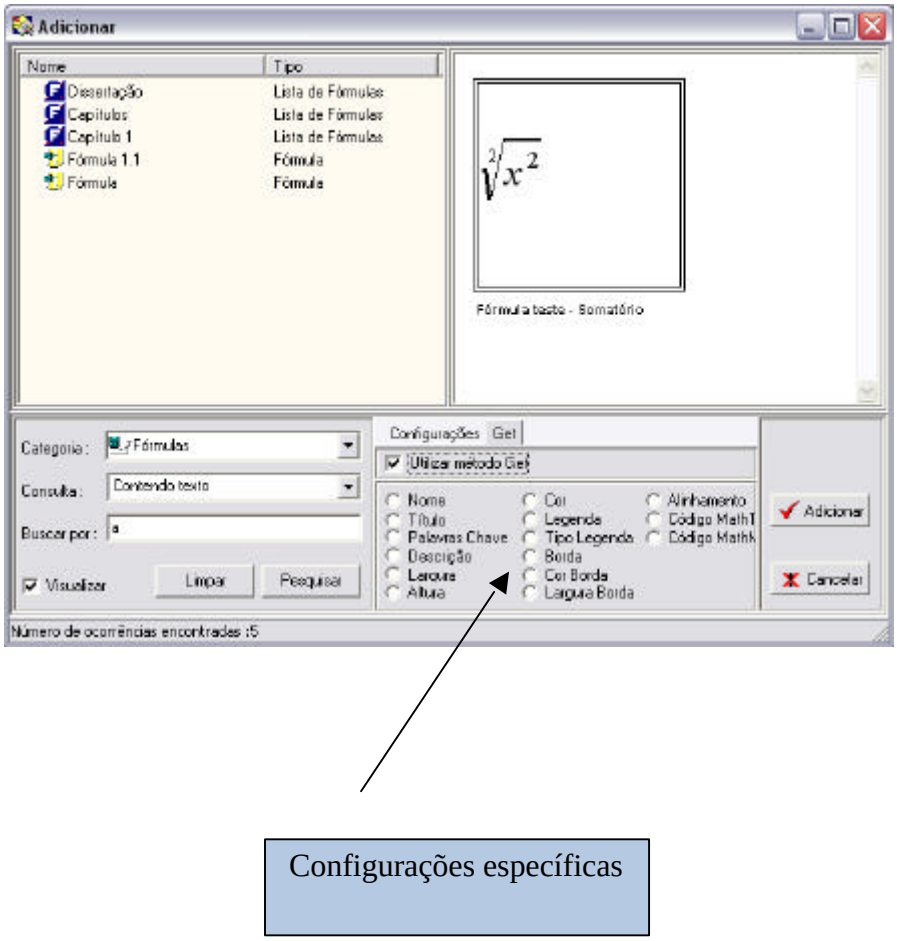
Figura 4.20 Gerenciador de Hipermídias – Janela Adicionar Referências.

Através da figura acima, é possível observar que existe um mecanismo de pesquisa para buscar o elemento desejado. Uma listagem de resultados de pesquisa é

mostrada e uma visualização ao lado torna fácil a adição de referências para qualquer fragmento de conteúdo.

Esta tela permite também a utilização do método “GET” para buscar alguma informação específica do elemento escolhido, bastando clicar no item desejado. A Figura 4.21 mostra a adição de uma referência para o “Nome” da fórmula referenciado como “Área”:

<REF TIPO=”Fórmula” ID=”Área” Get=”Nome”/>



Configurações específicas

Figura 4.21 Gerenciador de Hipermídias – Adicionando Referências com configurações específicas.

* Janela Gerenciador de Exportações

O gerenciador de Hipermídias possui uma interface específica para a produção e gerenciamento de modelos de exportação de dados. Assim, torna-se infinito o número de modelos de exportações que a ferramenta pode realizar, pois estes são definidos através de uma linguagem (RCM Script), onde se fornece total flexibilidade. Enquanto que em softwares comerciais conhecidos é oferecido um número finito de tipos de exportações, o Gerenciador de Hipermídias é capaz de interpretar uma linguagem de definição de modelos de exportação, tornando-se capaz de exportar dados (em diferentes formatos) seguindo qualquer tipo de modelo previamente definido. Por exemplo, é possível definir a exportação de um determinado fragmento RCML em um conjunto de tabelas HTML ou um conjunto de links HTML. A estrutura e o conteúdo dos arquivos exportados pelo Gerenciador de Hipermídias seguem os modelos que podem ser implementados através da linguagem RCM Script. A Janela gerenciador de exportações é similar à janela principal do software e pode ser observada na Figura 4.22.

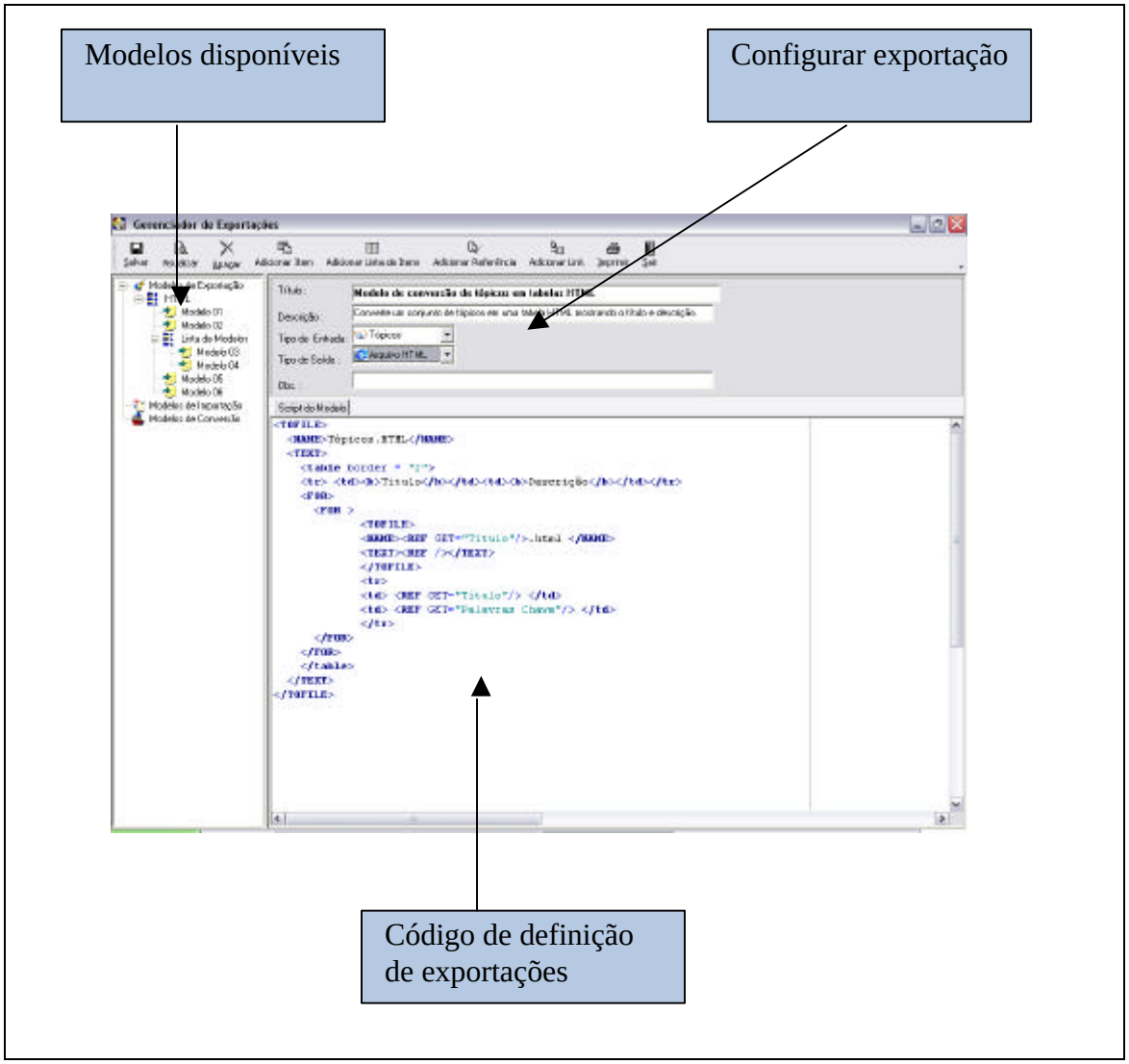


Figura 4.22 Gerenciador de Hipermídias – Janela Gerenciador de Exportações.

* Janela Assistente de Exportações

Após a produção de diferentes modelos de exportação, pode-se, através de uma interface amigável (assistente), exportar os dados seguindo algum modelo definido. Para tanto, deve-se selecionar o objeto que deseja-se exportar e chamar o assistente de exportação. Então, surgirá uma janela mostrando os modelos disponíveis, conforme pode ser observado na Figura 4.23.

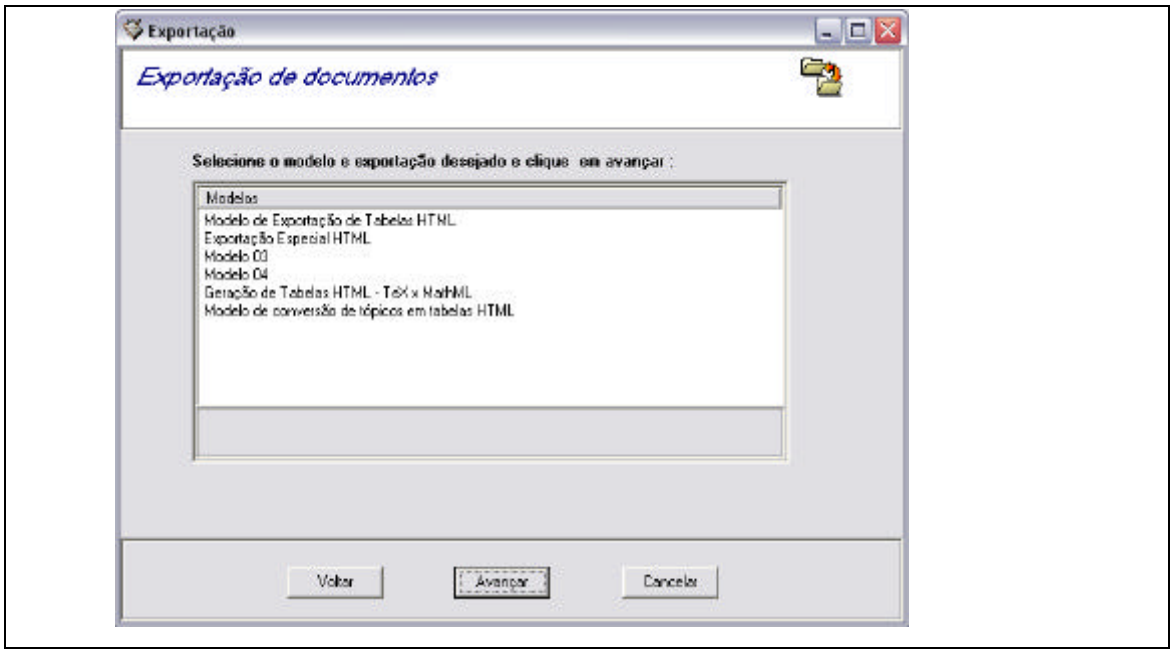


Figura 4.23 Gerenciador de Hipermídias – Janela Assistente de Exportações 1.

O próximo passo é, através deste assistente, configurar as entradas e saídas da exportação, como, por exemplo, especificar o diretório de saída. A Figura 4.24 mostra esta janela.

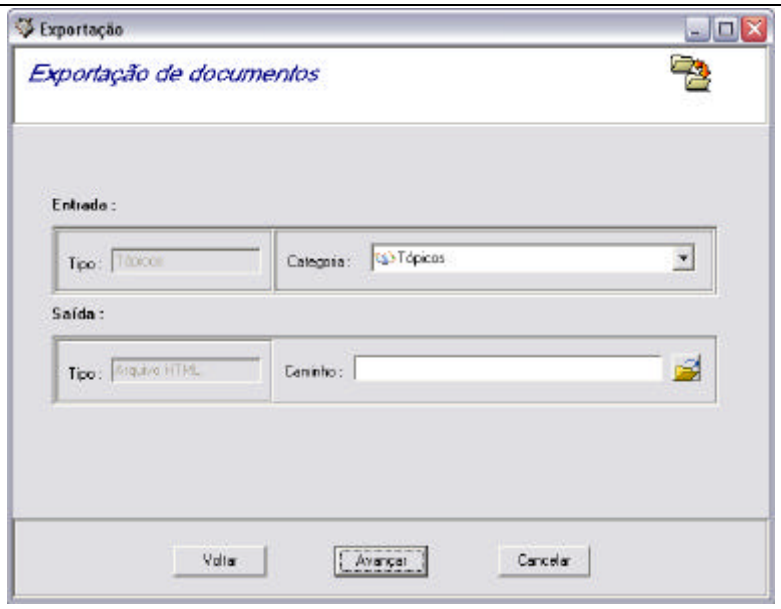


Figura 4.24 Gerenciador de Hipermídias – Janela Assistente de Exportações 2.

Assim, o processo é realizado e os dados são exportados gerando a tela exposta na Figura 4.25.

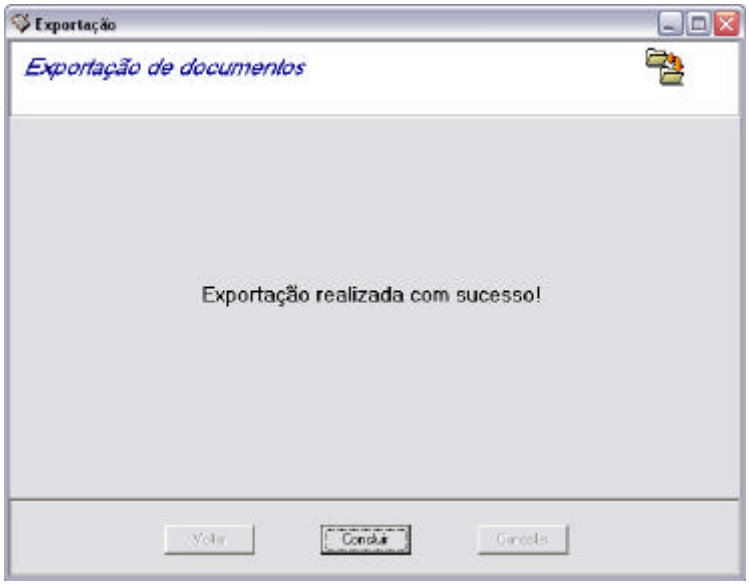


Figura 4.25 Gerenciador de Hipermídias – Janela Assistente de Exportações 3.

4.1.4 Estratégias de implementação MathTeX.

4.1.4.1 Introdução

O propósito desta seção é fornecer estratégias de implementação para a produção de bibliotecas com suporte à Metodologia MathTeX. O desenvolvimento dessas bibliotecas é independente de qualquer software que poderá utiliza-las. Isso traz muitas possibilidades de novas aplicações serem desenvolvidas com diferentes objetivos. Serão mostradas, detalhadamente, todas as estratégias de implementação que devem ser utilizadas para a obtenção de bons resultados com a implantação destas bibliotecas em determinados projetos de desenvolvimento de software.

Qualquer ferramenta que vise a utilização de algum mecanismo capaz de interpretar e converter código LaTeX, e, também, que vise gerar conteúdo matemático para a web (MathML), pode ser desenvolvida utilizando os seguintes recursos, que foram desenvolvidos neste trabalho :

a. Processador MathTeX :

Como descrito anteriormente, o processador MathTeX tem, como entrada, um código fonte na linguagem LaTeX e gera, como saída, um código MathML equivalente. Para a realização deste processo, as ferramentas abaixo deverão ser utilizadas.

b. Interpretador de código LaTeX :

Um interpretador de código LaTeX é utilizado para o reconhecimento desta sintaxe e verificação sintática. Este interpretador divide-se em duas fases : Análise Léxica e Análise Sintática. Maiores informações sobre a implementação de interpretadores podem ser obtidas em [COM 03]

c. Conversor de Código :

Um conversor de código LaTeX deve ser construído para interpretar uma estrutura sintática obtida através do Interpretador de código e, finalmente, gerar o código MathML correspondente, através de um gerador de código MathML.

d. Gerador de Código MathML :

Uma biblioteca capaz de gerar e interpretar código MathML é necessária para a conclusão do processo. Esta biblioteca é capaz de gerar qualquer tipo de conteúdo neste formato, podendo, assim, ser aplicada para diferentes propósitos. Estratégias para o desenvolvimento de geradores de código XML foram definidas na Seção 4.1.1.

Todos os itens acima citados executam a parte funcional de um software desenvolvido para esse fim. Restará apenas a necessidade de implementação de uma interface gráfica para acesso às bibliotecas. Várias interfaces gráficas, com diferentes aplicações, poderão fazer uso destas bibliotecas. O Modelo de Implementação pode ser visualizado na Figura 4.26.

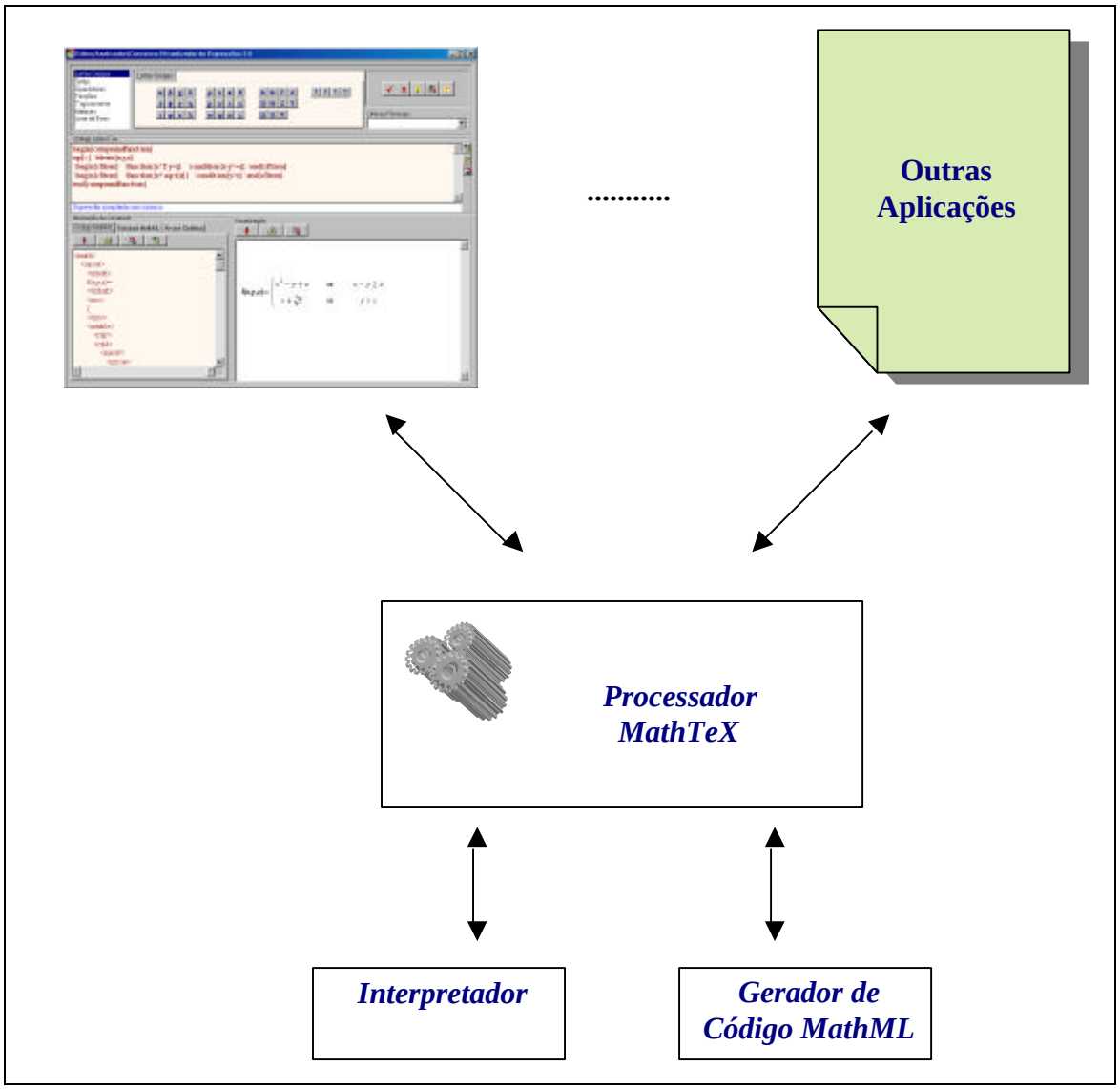


Figura 4.26 Estratégias de implementação MathTeX – Modelo de Implementação.

4.1.4.2 Interpretador de Código MathTeX :

Este processo se divide em duas fases : Análise Léxica, e Análise Sintática. Para isto, duas classes devem ser desenvolvidas : TScanner e TParser. A classe TScanner possui, dentre outros, um método chamado AnaliseLexica. Este método deve retornar uma listagem de *tokens*, [COM 03]. Esta listagem é passada para o processo de análise sintática, que é realizada através do método AnaliseSintatica, devendo ser implementado na classe TParser. Este método retorna uma árvore sintática do código

fonte, caso este esteja sintaticamente correto. Seguem na Tabela 4.5 os passos para a realização deste procedimento. Supõe-se a utilização do ambiente de programação Delphi 6.0.

Ordem	Ação	Descrição
1	**Var Scanner : TScanner**	**Declaração do objeto Scanner**
2	**Var Parser : Tparser**	**Declaração do objeto Tparser**
3	**Scanner := Tscanner.Create**	**Instanciação do objeto Scanner**
4	**Parser := Tparser.Create**	**Instanciação do objeto Parser**
5	**Scanner.AnaliseLexica**	**Realização da análise léxica**
6	**Parser.AnaliseSintatica**	**Realização da análise sintática**

Tabela 4.5 Estratégias de implementação MathTeX – Interpretador de Código.

A Figura 4.27 apresenta um trecho de código que poderá ser utilizado para a realização do processo acima referenciado.

```
....

Lexica    := TScanner.Create;

Sintatica := TParser.Create;

Lexica.AnaliseLexica(Memo2.Text);

If Lexica.ListError.Count = 0 then

   begin

   Sintatica.AnaliseSintatica(Lexica.Text);

   if Sintatica.ListError.Count = 0 then

     begin

....
```

Figura 4.27 Estratégias de implementação MathTeX – Interpretador de Código.

4.1.4.3 Conversor de Código

Juntamente com o processador de análise sintática, o conversor de código deve ser desenvolvido permitindo, através da interpretação de uma estrutura sintática, a produção de conteúdo em formato MathML. Dois métodos são adicionados na classe Tparser : ToMathMLPres e ToMathMLCont, para a obtenção de codificação de apresentação e codificação de conteúdo MathML, respectivamente. Assim, após a análise sintática de um código LaTeX, este método poderá ser utilizado para a

realização do processo conversor. Um gerador de código MathML é utilizado por esse conversor para a geração dos dados de saída, conforme modelo da Seção 4.1.1. Seguem, na Tabela 4.6, os passos para a realização deste procedimento. Supõe-se a utilização do ambiente de programação Delphi 6.0.

Ordem	Ação	Descrição
1	**Var Scanner : TScanner**	**Declaração do objeto Scanner**
2	**Var Parser : Tparser**	**Declaração do objeto Tparser**
3	**Scanner := Tscanner.Create**	**Instanciação do objeto Scanner**
4	**Parser := Tparser.Create**	**Instanciação do objeto Parser**
5	**Scanner.AnaliseLexica**	**Realização da análise léxica**
6	**Parser.AnaliseSintatica**	**Realização da análise sintática**
7	**Parser.ToMathMLCont ou**	**Conversão de MathTeX para MathML (conteúdo)**
	Parser.ToMathMLPress	**Conversão de MathTeX para MathML (apresentação)**

Tabela 4.6 Estratégias de implementação MathTeX – Conversor de Código.

A Figura 4.28 ilustra um trecho de código que poderá ser utilizado para a realização do processo acima referenciado.

```
  ....
    Lexica    := TScanner.Create;
    Sintatica := TParser.Create;
    Lexica.AnaliseLexica(Memo2.Text);
    If Lexica.ListError.Count = 0 then
      begin
      Sintatica.AnaliseSintatica(Lexica.Text);
      if Sintatica.ListError.Count = 0 then
        begin
        Sintatica.ToMathMLCont;
        Sintatica.ToMathMLPres;
....
```

Figura 4.28 Modelo MathTeX – Conversor de Código.

4.1.4.4 Gerador de Código MathML.

Uma biblioteca deve ser desenvolvida seguindo a estratégia exposta na Seção 4.1.1 para realizar a produção de qualquer tipo de conteúdo em formato MathML. Esta biblioteca é utilizada pelo Conversor de Código para a geração dos dados em MathML. Seguem, na Tabela 4.7, os passos para a realização deste procedimento. Supõe-se a utilização do ambiente de programação Delphi 6.0.

Ordem	Ação	Descrição
1	**Var MathML : TMathML**	**Declaração do objeto MathML**
2	**MathML : TmathML.Create**	**Instanciação do objeto MathML**
3	**MathML.....**	**Utilização dos métodos**

Tabela 4.7 Estratégias de implementação MathTeX – Gerador de Código MathML.

A Figura 4.29 mostra um trecho de código que poderá ser utilizado para a realização do processo acima referenciado.

```
MathML.semantics.Begin_apply;

   MathML.semantics.Begin_root;

      MathML.semantics.Begin_cn;

          MathML.write('4');

      MathML.semantics.End_cn;

MathML.semantics.End_apply;
```

Figura 4.29 Estratégias de implementação MathTeX – Gerador de Código MathML.

4.1.5 O Editor de Fórmulas

Nesta seção será demonstrado o Editor de Fórmulas, uma ferramenta proposta para validar a Metodologia MathTeX.

4.1.5.1 Introdução

O Editor de Fórmulas é uma ferramenta que faz uso de todos os recursos acima descritos para a geração de conteúdos matemáticos para a web. Através de uma interface amigável é possível a geração de código em LaTeX, conversão em MathML e visualização do resultado, da mesma forma que este será mostrado em um navegador web.

A interface deste aplicativo pode ser dividida em quatro partes :

* Paleta de Símbolos
* Editor de Código LaTeX
* Editor de Código MathML
* Visualizador de conteúdo MathML

Este software possui as seguintes funcionalidades :

* Uma paleta “amigável” para a geração de dados matemáticos em LaTeX;
* Um analisador léxico e sintático para a lingugem LaTeX;
* Um conversor de código LaTeX em MathML;
* Um visualizador de conteúdos MathML.

A Figura 4.30 apresenta a tela principal do Editor de Fórmulas.

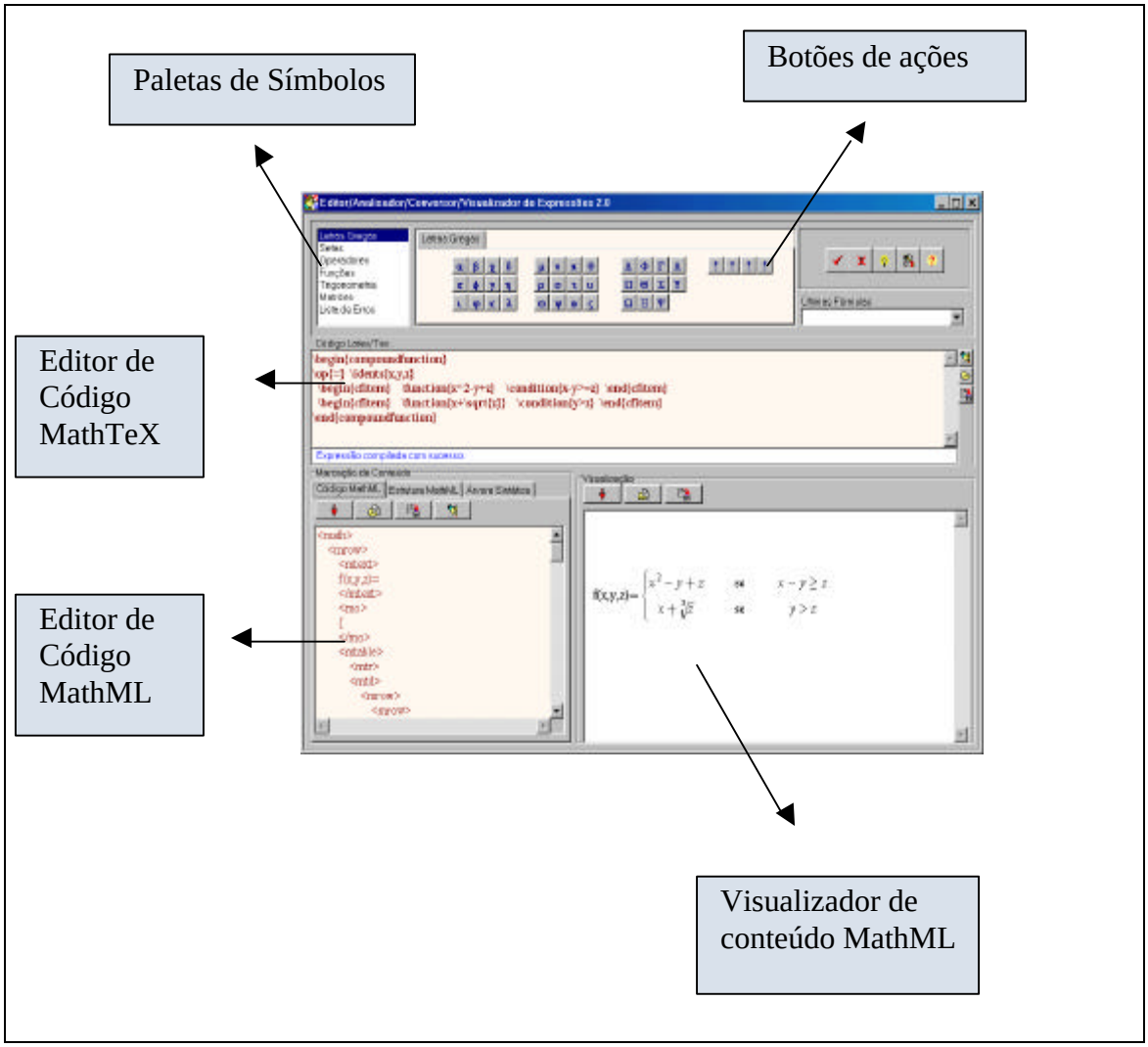


Figura 4.30 Editor de Fórmulas – Tela Principal.

4.1.5.2 Objetivos

Os principais objetivos desta ferramenta são:

* Gerar código LaTeX através do uso de paletas de símbolos matemáticos;
* .Analisar, do ponto de vista sintático, expressões codificadas em LaTeX;
* Mapear expressões descritas em LaTeX para MathML.

4.1.5.3 Interface Gráfica

* Paleta de Símbolos

O Editor de fórmulas possui uma paleta com grande parte dos símbolos matemáticos contidos na sintaxe de codificação LaTeX. Assim é possível, de uma forma muito intuitiva, a produção de código neste formato. Esta paleta viabiliza a obtenção dos recursos que a linguagem LaTeX oferece, sem a necessidade de seu conhecimento completo, através da utilização da interface gráfica da ferramenta.

As paletas existentes são ilustradas na Figura 4.31.

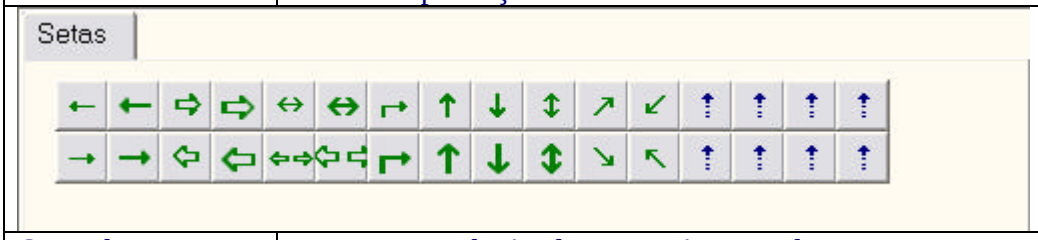
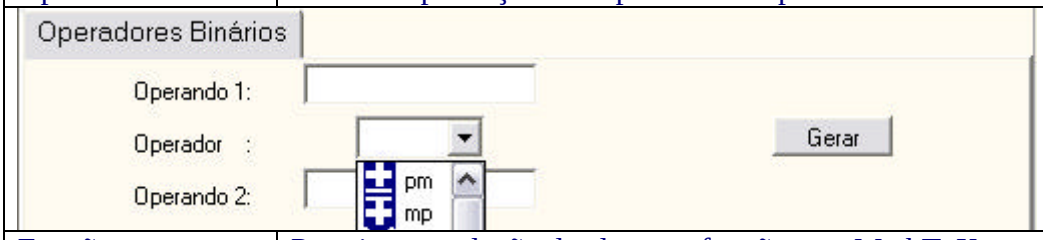
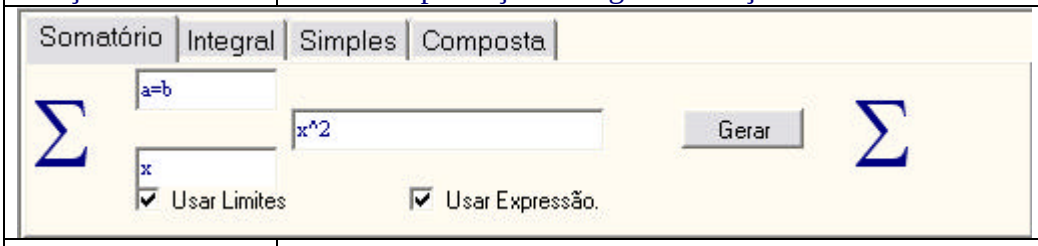
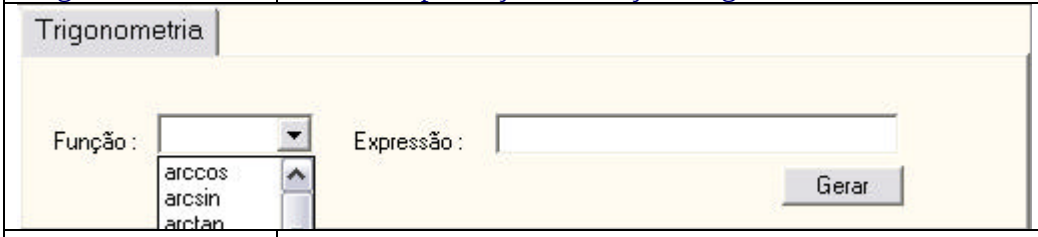
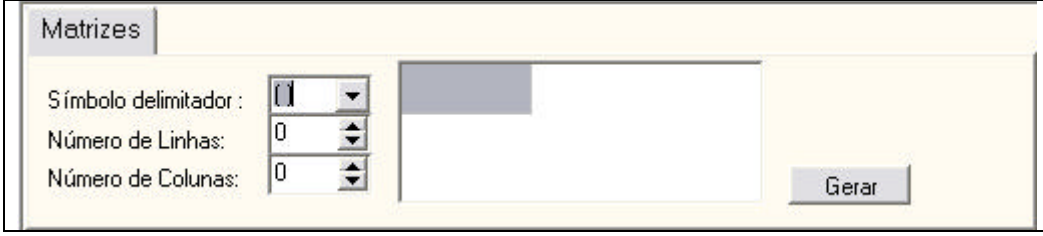
Letras Gregas	Permite a produção de letras gregas em LaTeX
	
Setas	Permite a produção de setas em LaTeX
	
Operadores	Permite a produção de expressões simples em LaTeX
	
Funções	Permite a produção de algumas funções em MathTeX
	
Trigonometria	Permite a produção de funções trigonométricas em LaTeX
	
Matrizes	Permite a produção de matrizes em LaTeX
	

Figura 4.31 Editor de Fórmulas – Paletas de Símbolos.

* Editor de Código LaTeX

O editor de código LaTeX permite que o usuário crie e altere código neste formato. A Figura 4.32 apresenta este editor.

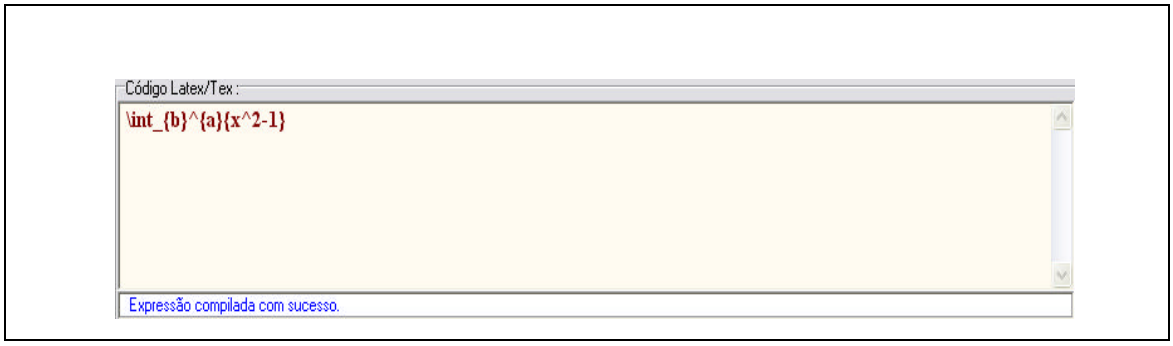


Figura 4.32 Editor de Fórmulas – Editor de Código MathTeX/LaTeX.

* Editor de Código MathML

Após a interpretação de dados em LaTeX e a conversão em MathML, o conteúdo é gerado em um editor de código MathML, onde pode-se alterar o resultado obtido pelo conversor. Este editor pode ser observado através da Figura 4.33.

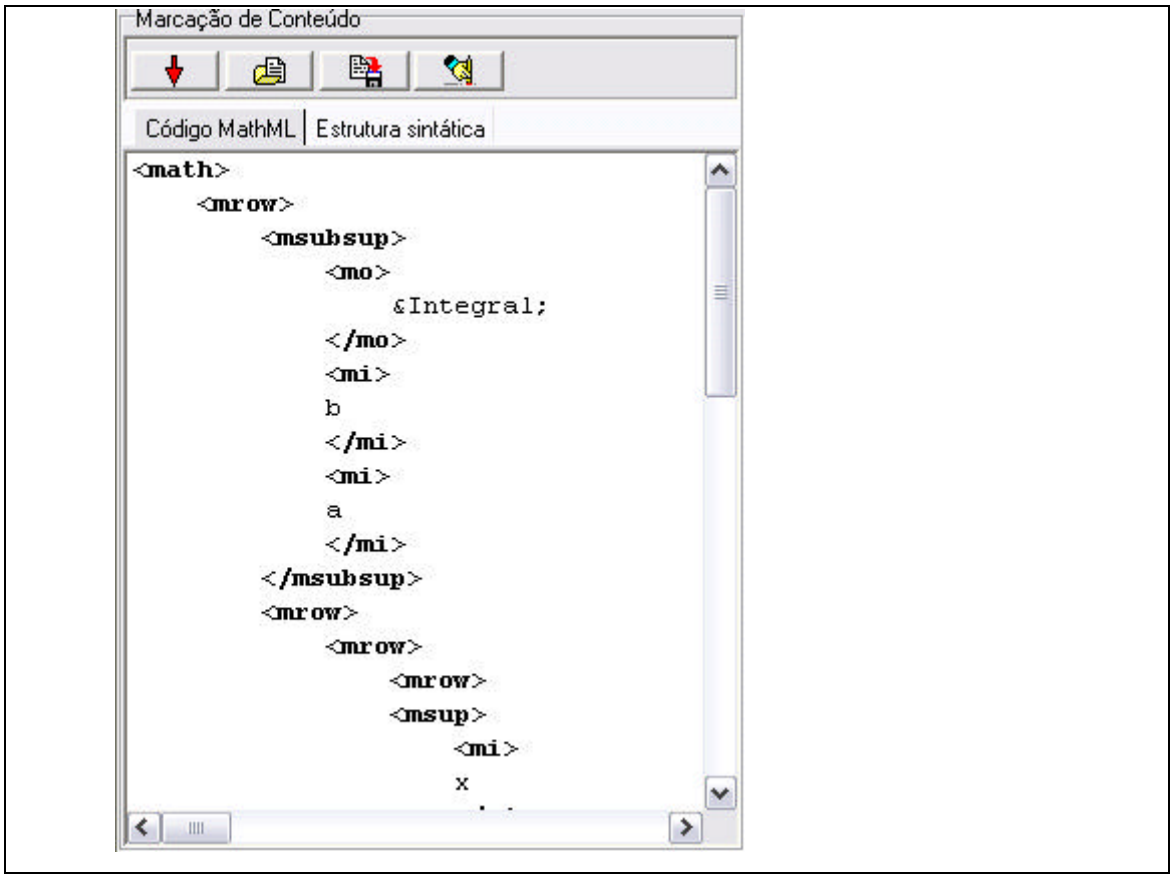


Figura 4.33 Editor de Fórmulas – Editor de Código MathML.

* Visualizador de conteúdo MathML

A renderização do conteúdo MathML é visualizada no “Visualizador de conteúdo MathML”, conforme pode ser observado na Figura 4.34.

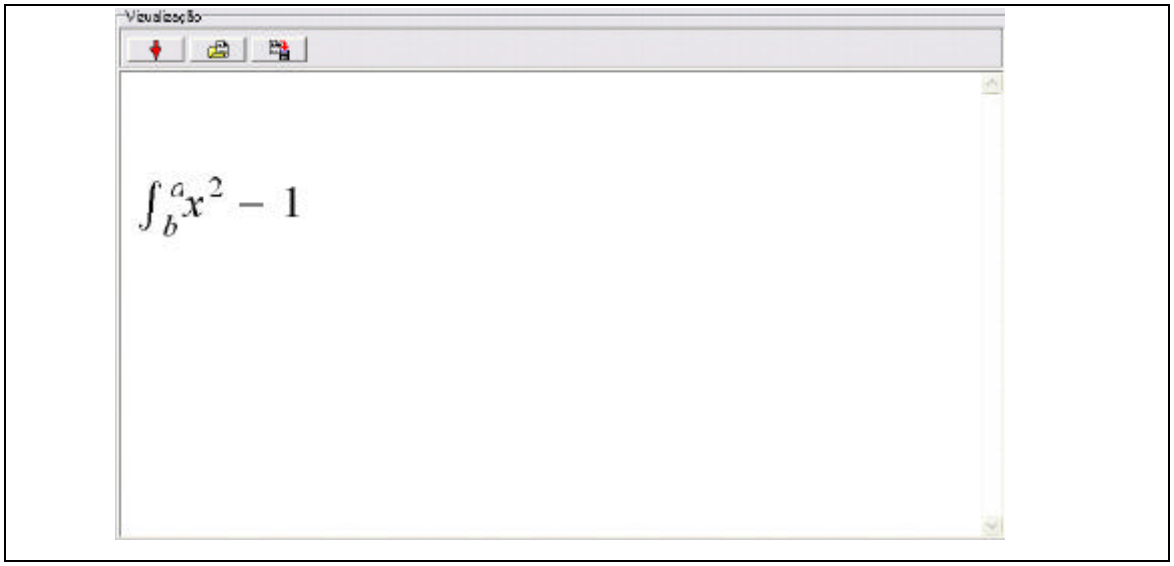


Figura 4.34 Editor de Fórmulas – Visualizador de conteúdo MathML.

4.1.6 Estratégias de Implementação GraphML

4.1.6.1 Introdução

O propósito desta seção é fornecer estratégias de implementação para a produção de bibliotecas com suporte à Metodologia GraphML. O desenvolvimento destas bibliotecas é independente de qualquer software que as utilize. As bibliotecas aqui referenciadas possibilitam o desenvolvimento de qualquer ferramenta capaz de manipular, gerar e exportar dados segundo a modelagem GraphML. Serão mostradas, detalhadamente, todas as estratégias de implementação que deverão ser utilizadas para a obtenção de bons resultados com a implantação das bibliotecas em determinados projetos de desenvolvimento de software.

Qualquer ferramenta que vise a utilização de algum mecanismo capaz de manipular e estruturar dados GraphML e realizar a produção de conteúdos em SVG, poderá ser desenvolvida utilizando os seguintes recursos, que foram desenvolvidos neste trabalho :

a. Processador GraphML :

Como descrito anteriormente, o Processador GraphML tem, como entrada, um conjunto de dados estruturados em um determinado formato, e armazena-o em uma base GraphML. Para a realização deste processo, o Processador GraphML utiliza os recursos oferecidos pelo DOM (*Document Object Model*), uma *api* capaz de manipular qualquer linguagem descrita em XML. O Processador GraphML possui as seguintes funções : importar dados estruturados; gerar dados em GraphML e importar informações GraphML gerando saídas em um outro formato estruturado.

b. Processador XSL :

Para a manipulação de dados XML, através do uso de XSL, o Processador XSL utiliza recursos do Processador GraphML através de solicitação de informações. Tendo em vista o modelo de implementação aqui referenciado, o Processador XSL é visto como um método da classe que implementa o Processador GraphML.

Todos os itens acima citados executam a parte funcional de um software desenvolvido para esse fim. Resta apenas a necessidade de implementação de uma interface gráfica para acesso às bibliotecas. Várias interfaces gráficas, com diferentes aplicações, podem fazer uso destas bibliotecas ao mesmo tempo. O Modelo de Implementação GraphML pode ser visualizado na Figura 4.35.

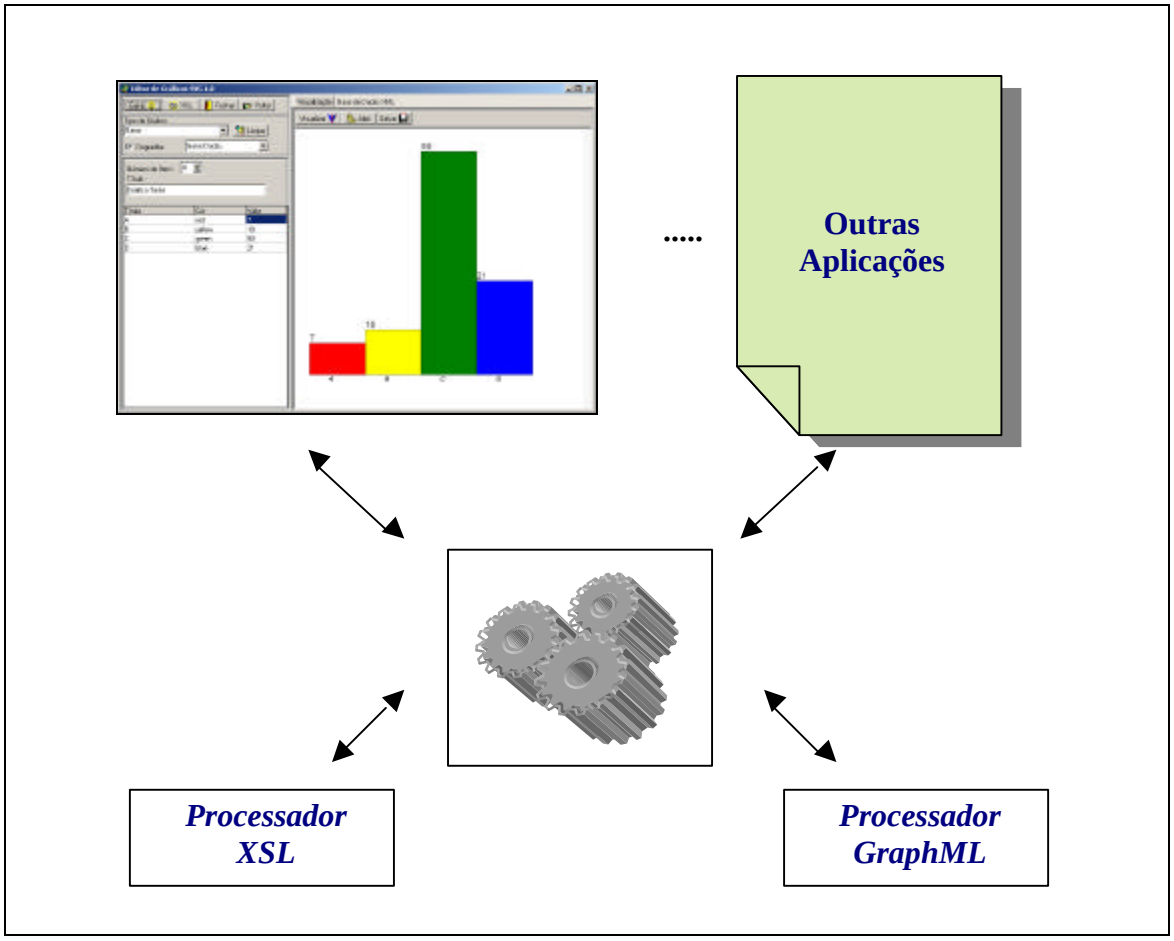


Figura 4.35 Estratégias de Implementação GraphML – Modelo de Implementação GraphML.

4.1.6.2 Processador GraphML

Herdando toda a funcionalidade provida pela api DOM, o Processador GraphML pode ser implementado como uma classe chamada “TgraphML”, que, juntamente com uma hierarquia de classes desenvolvidas para toda a linguagem GraphML (Anexo VII), é capaz de gerar, importar e manipular dados XML na linguagem anteriormente definida. Alguns métodos desta classe constituem outras ferramentas como o Processador XSL.

A classe TGraphML deve ser desenvolvida para realizar a produção de qualquer tipo de conteúdo em formato GraphML. A proposta da classe TGraphML é ser uma biblioteca totalmente independente da aplicação, podendo, assim, ser utilizada em qualquer ferramenta que necessite a obtenção e manipulação de dados neste formato. Seguem, na Tabela 4.8, os passos para a realização deste procedimento. Supõe-se a utilização do ambiente de programação Delphi 6.0.

Ordem	Ação	Descrição
1	Var GraphML : TGraphML	Declaração do objeto GraphML
2	GraphML := TgraphML.Create	Instanciação do objeto GraphML
3	GraphML.....	Utilização dos métodos

Tabela 4.8 Estratégias de Implementação GraphML – Processador GraphML.

A Figura 4.36 mostra um trecho de código que poderá ser utilizado para a realização do processo acima referenciado.

```
....

GraphML.Begin_Grafico(“Pizza”)
GraphML.Grafico.Begin_Item(“Brasil”,”blue”)

    GraphML.Grafico.Item.Begin_Valor

      GraphML.Grafico.Item.Valor.Write(85)

    GraphML.Grafico.Item.End_Valor

  GraphML.Grafico.End_Item

  GraphML.Grafico.Begin_Item(“Argentina”,”red”)

    GraphML.Grafico.Item.Begin_Valor

      GraphML.Grafico.Item.Valor.Write(10)

    GraphML.Grafico.Item.End_Valor

  GraphML.Grafico.End_Item

  GraphML.Grafico.Begin_Item(“Uruguai”,”yellow”)

  GraphML.Grafico.Item.Begin_Valor

    GraphML.Grafico.Item.Valor.Write(85)

  GraphML.Grafico.Item.End_Valor

  GraphML.Grafico.End_Item

GraphML.End_Grafico

GraphML.Filename := ‘c:\pizza.xml’;

GraphML.Save;

.....
```

Figura 4.36 Estratégias de Implementação GraphML – Processador GraphML.

4.1.6.3 Processador XSL.

Outro método da classe TGraphML deve ser desenvolvido, o "ApplyXSL". Este método possui a tarefa de aplicar uma certa folha de estilos XSL ao documento XML, que está sendo manipulado pelo Processador GraphML através da classe TGraphML. Este procedimento é realizado através da utilização de mecanismos providos pela api DOM. O resultado deste método é um documento transformado pela folha de estilos. As folhas de estilos desenvolvidas para a renderização em gráficos de barras, pizza, pontos e linhas em formato SVG podem ser encontradas no Anexo X. Seguem na Tabela 4.9 os passos para a realização deste procedimento. Supõe-se a utilização do ambiente de programação Delphi 6.0.

Ordem	Ação	Descrição
1	**Var GraphML : TGraphML**	**Declaração do objeto GraphML**
2	**GraphML:=TGraphML.Create**	**Instanciação do objeto GraphML**
3	**GraphML.ApplayXSL**	**Invocação do Interpretador de Código**

Tabela 4.9 Estratégias de Implementação GraphML – Processador XSL.

A Figura 4.37 mostra um trecho de código que poderá ser utilizado para a realização do processo acima referenciado.

```
....

GraphML := TGraphML.Create;

GraphML.ApplyXSL(Folha.XSL).SavetoFile('Result.txt')

....
```

Figura 4.37 Estratégias de Implementação GraphML – Processador XSL.

4.1.7 O Editor de Gráficos

Nesta seção será demonstrado o Editor de Gráficos, uma ferramenta proposta para validar a Metodologia GraphML.

4.1.7.1 Introdução

O Editor de Gráficos é uma ferramenta de suporte SVG que foi criada através da metodologia GraphML. Através da observação deste aplicativo pode-se comprovar a eficiência desta metodologia. Além de prover mecanismos para a geração de gráficos a partir de uma base de dados em GraphML, este aplicativo detém a *proposta* de uma funcionalidade capaz de gerar folhas de estilos específicas para qualquer base de dados XML, independente do GraphML. Assim, a expansão de suas aplicações torna-se evidente. O modelo da ferramenta é exposto na Figura 4.38.

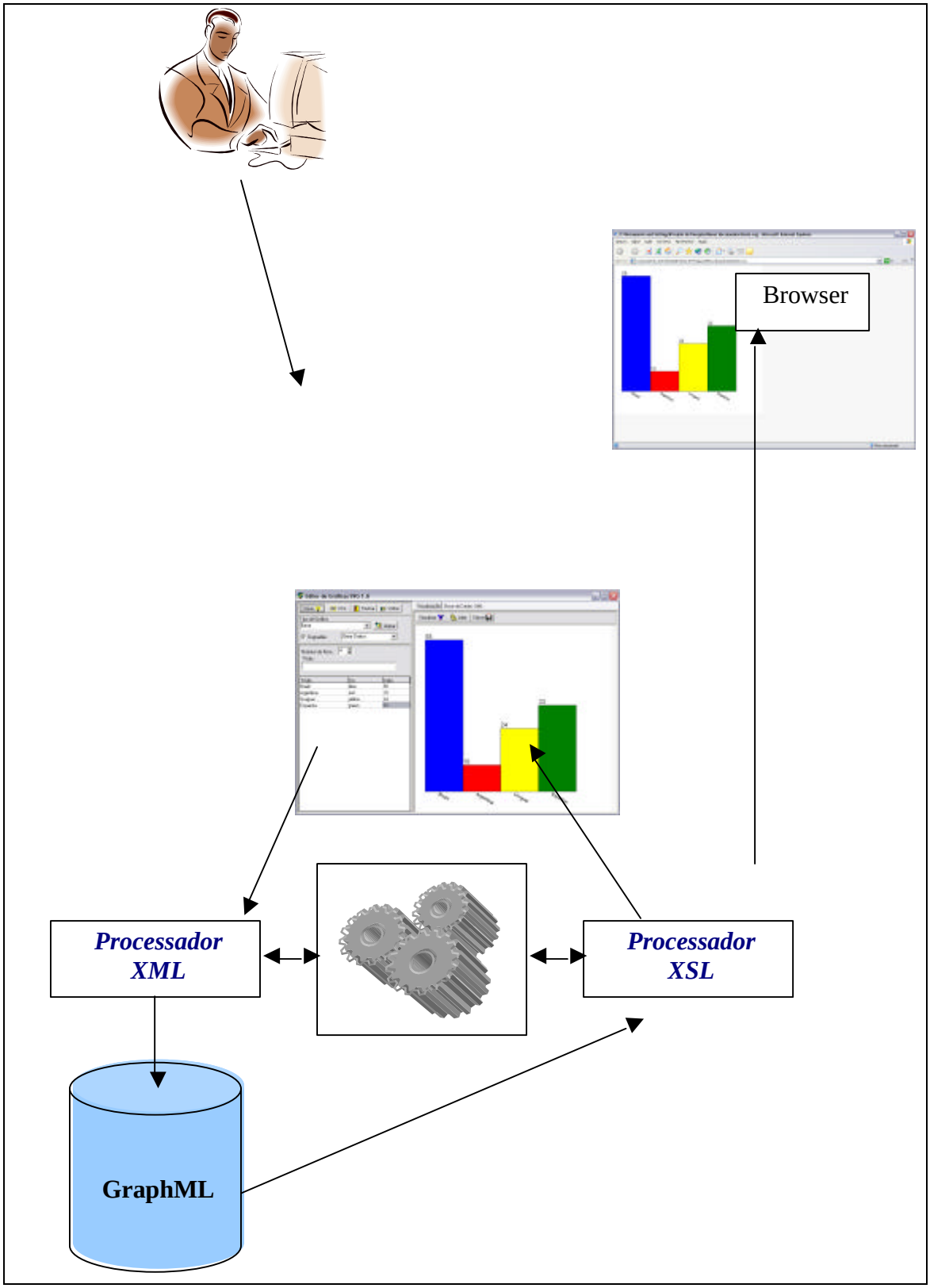


Figura 4.38 Editor de Gráficos – Modelo Geral.

A interface deste aplicativo pode ser dividida em :

* Janela Principal
* Janela Gerar Dados
* Janela Importar Dados
* Janela Visualizar Dados
* Janela Editar GraphML

4.1.7.2 Objetivos

* Gerar dados em GraphML através de uma interface amigável.
* Exportar dados em formato SVG.
* Manipular dados em GraphML.
* Visualizar gráficos GraphML em SVG.

4.1.7.3 Interface Gráfica

* Janela Principal

A janela principal do Editor de Gráficos possui todas as outras janelas acima descritas. A ilustração desta janela pode ser observada na Figura 4.39.

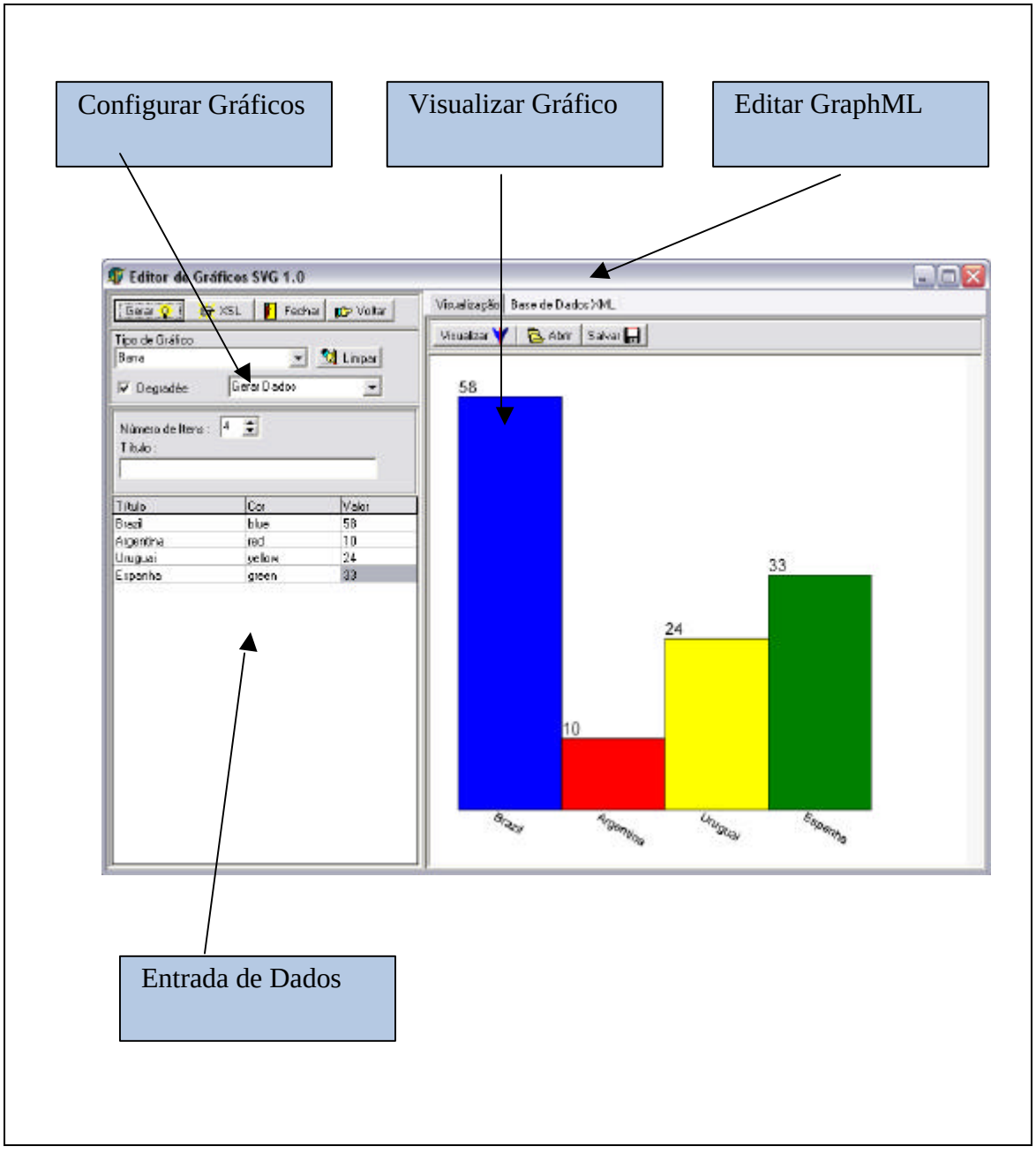


Figura 4.39 Editor de Gráficos – Interface Principal.

* Janela Gerar Dados

O Editor de Gráficos possui uma interface amigável para a produção de conteúdo em formato GraphML. A Figura 4.40 ilustra esta funcionalidade.

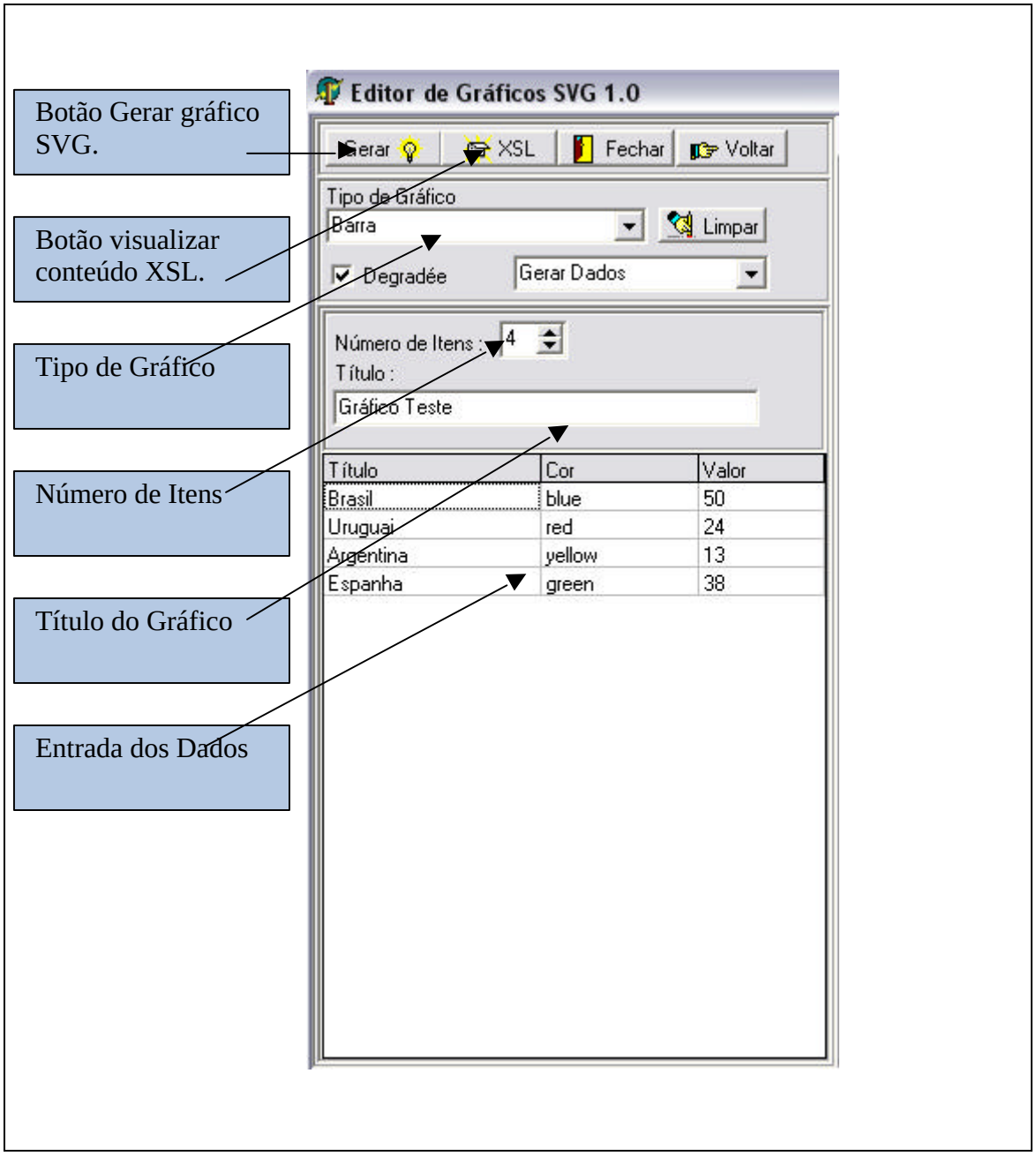


Figura 4.40 Editor de Gráficos – Janela Gerar Dados.

* Janela Visualizar Dados

O Editor de Gráficos é capaz de visualizar os resultados em SVG antes de efetuar a exportação dos dados. A janela que realiza esta tarefa é apresentada na Figura 4.41.

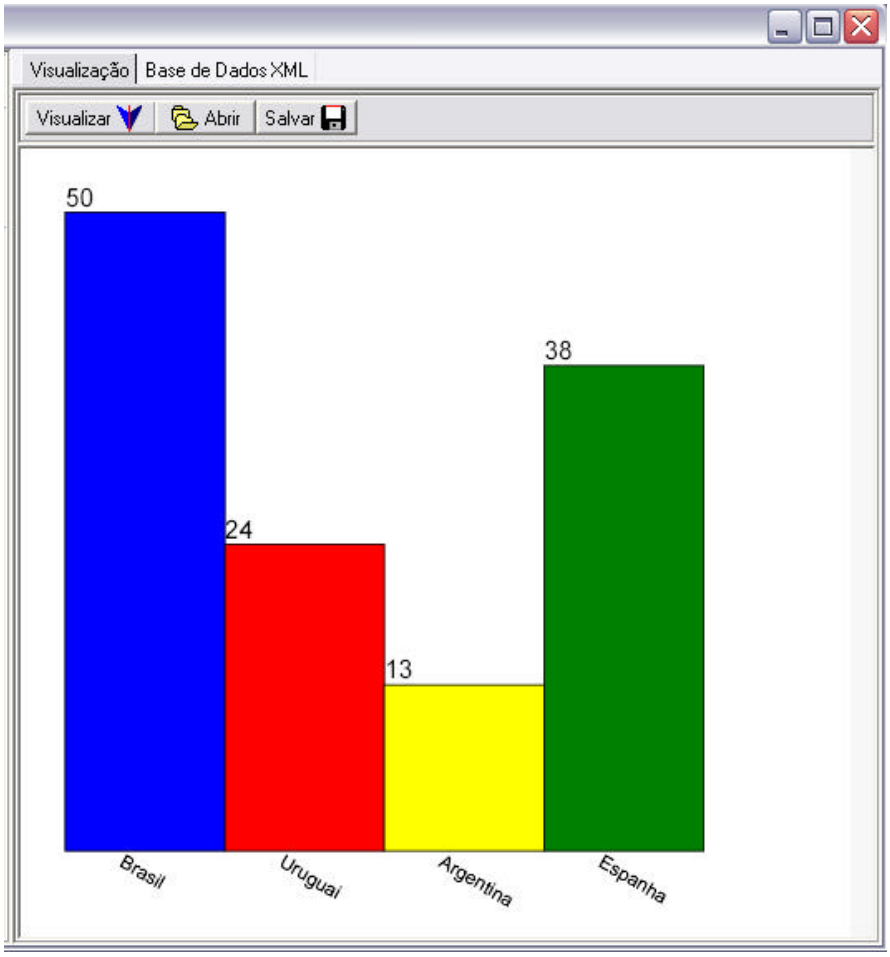


Figura 4.41 Editor de Gráficos – Janela Visualizar Dados.

* Janela Editar GraphML

Na tela exposta na Figura 4.42, o gráfico é gerado estaticamente a partir da interface provida pelo Editor de Gráficos, gerando, automaticamente, o código GraphML equivalente. Este editor oferece suporte para a edição deste código, conforme a Figura 4.42.

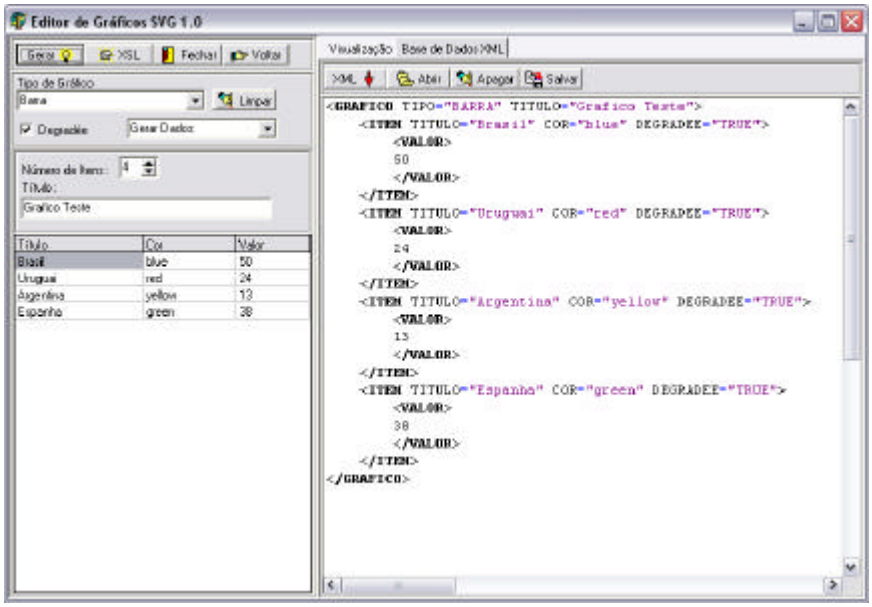


Figura 4.42 Editor de Gráficos – Janela Editar GraphML.

4.2 Conclusão

Este capítulo apresentou todas as estratégias de implementação e implantação das metodologias propostas e demonstrou os softwares desenvolvidos. O próximo capítulo, Capítulo 5, apresenta as conclusões deste trabalho.

Capítulo 5

CONCLUSÕES

5.1 Introdução

Este trabalho, através das soluções propostas, trouxe uma nova estratégia para a comunidade da web que busca resolver o problema da semântica dos dados na Internet. Através do framework proposto, muitos desenvolvedores de conteúdo para a divulgação na web poderão desenvolver ferramentas de autoria e gerenciamento da informação utilizando recursos altamente sofisticados de reutilização de objetos de dados. A idéia descrita no presente trabalho fornece várias possibilidades para o desenvolvimento de soluções novas, bem como, a continuidade do framework desenvolvido. O Gerenciador de Hipermdias e, também, a sua estratégia de gerenciamento da informação, é uma solução, até a data deste trabalho, inovadora. Já o Editor de Fórmulas fornece uma solução adicional para o gerenciamento de fórmulas MathML. O Editor de Gráficos é uma ferramenta que facilita a produção de gráficos em formato SVG. Essas ferramentas formam um pacote de autoria de conteúdo semântico para a web podendo ser utilizados em diversas aplicações, como, por exemplo, a educação a distância. Já as metodologias de desenvolvimento propostas provêm estratégias para a implementação de ferramentas de autoria que seguem as estratégias dos softwares desenvolvidos neste trabalho. Assim, as contribuições desta dissertação trazem, além de soluções prontas, estratégias para a produção de novas soluções para domínios específicos, ou seja, a alta capacidade de generalização é claramente evidente.

5.2 A respeito da Metodologia RCM

* A arquitetura para a web semântica proposta mostrou-se uma colaboração muito importante para a comunidade científica, provendo soluções de organização, estruturação e gerenciamento da informação disponível na Internet;
* A linguagem RCML viabilizou a estruturação, semântica e reusabilidade da informação. Assim, mostrou-se uma ontologia que pode ser aplicada em diferentes domínios, provendo uma solução baseada em XML para a web semântica.
* RCM Script proporcionou a realização de tarefas que as atuais linguagens de manipulação de metadados (XSLT,por exemplo) não são capazes de realizar. Esta linguagem tornou-se uma estratégia para trabalhar de forma colaborativa com XSL ou Xquery, oferecendo excelentes mecanismos de manipulação e reusabilidade de conteúdo.
* A Metodologia RCM proporcionou o desenvolvimento do Gerenciador de Hipermídias, mostrando como esta estratégia poderá ser utilizada pela comunidade científica para o desenvolvimento de ferramentas de autoria e gerenciamento de metadados;
* O Gerenciador de Hipermídia tornou-se uma ferramenta capaz de agilizar o processo de autoria e gerenciamento de documentos para a web. Muitas instituições poderão fazer uso desta ferramenta automatizando em grande parte o processo de produção de conteúdos.
* O framework proposto neste trabalho encontrou uma solução prática para a produção, gerenciamento, reutilização e semântica de conteúdos para a web. Forneceu uma colaboração muito importante para a comunidade científica da área “Semantic Web”, que busca constantemente soluções como estas propostas neste trabalho.

5.3 A respeito da Metodologia MathTeX

* Utilizando MathML, recursos poderosos para o tratamento de fórmulas matemáticas mostram-se uma tecnologia adequada , em particular para a produção de textos matemáticos a serem distribuídos eletronicamente.
* Embora possuindo um grande número de vantagens sobre diversas técnicas de publicação matemática, MathML possui algumas desvantagens no que diz respeito ao suporte. Sendo uma linguagem recente, o número de ferramentas gerenciadoras e de navegadores com suporte MathML é pequeno. Com isso, ainda é necessária a utilização de mecanismos externos, como *plug-ins*, para a visualização de fórmulas nesse formato em navegadores como o Internet Explorer. Também, devido à inexistência de aplicativos gerenciadores de código MathML, faz-se necessária a produção própria dessas ferramentas quando as mesmas tornam-se importantes.
* A falta de suporte nesta área, bem como a existência de muitos pontos ainda não explorados, é um fator de grande motivação para a pesquisa e produção de novos mecanismos capazes de suprir deficiências encontradas atualmente e fornecer soluções inovadoras.
* MathTeX fornece um mecanismo eficiente capaz de automatizar o processo de autoria de documentos Web, especialmente no que se refere a conteúdos matemáticos.

5.4 A respeito da Metodologia GraphML

* Desenvolvedores de ferramentas se beneficiarão muito com esta estratégia, ganhando tempo e eficiência;
* A eficiência desta estratégia foi confirmada no desenvolvimento do Editor de Gráficos, uma ferramenta que agilizou muito a produção de documentos eletrônicos destinados ao ensino / aprendizagem à distância;
* Juntamente com XML, SVG é uma tendência evidente para o desenvolvimento de soluções para a publicação de documentos Web;
* Apesar de complexo, XSL é muito poderoso, sendo impossível enfatizar aqui todas as suas vantagens. Esta tecnologia mostrou-se muito importante e eficaz na sua utilização juntamente com XML e SVG;
* O Editor de Gráficos é uma ferramenta muito eficiente e oferece novas sugestões de continuidade e pesquisa, como, por exemplo a geração automática de gráficos através de expressões matemáticas codificadas numa sintaxe qualquer.

5.5 Considerações Finais

Como pode ser observado na Seção 1.4, vários trabalhos foram descritos mostrando seus principais objetivos e soluções propostas. A Tabela 5.1 mostra uma comparação destes trabalhos com as três metodologias propostas nesta dissertação. Os items numerados de 1 a 7, mostrados na Tabela 5.1, estão definidos na Seção 1.3.2

Itens / Trabalhos	1	2	3	4	5	6	7
[SAN 03]	X	X					
[GUS 01]	X	X					X
[MUN 02]	X	X					
[CHA 03]	X	X	X		X		
[GEE 03]	X	X					
RCM	X	X	X	X	X	X	X
MathTeX	X	X	X	X	X	X	X
GraphML	X	X	X	X	X	X	X

Tabela 5.1 Comparação entre os trabalhos correlatos e as metodologias propostas.

Anexo I

DTD RCML – *Reusable Marckup Language*

```
<?xml encoding=”UTF-8”?>

<!ENTITY % NOHNAME “NOHNAME CDATA #REQUIRED”>

<ELEMENT PROJETO (TELASGRAFICAS,DOCUMENTOS,FORMULAS,GRAFICOS,TABELAS,

FIGURAS,EXEMPLOS,EXERCICIOS,TEOREMAS,DEFINICOES,GLOSSARIOS,BIBLIOGRAFI
AS,APPLETS,FLASHS,VIDEOS)>

<!ATTLIST  %NOHNAME; NOME CDATA #REQUIRED>

<!ELEMENT  RECURSOS (LISTASDERECURSOS*)>

<!ATTLIST RECURSOS %NOHNAME;>

<!ELEMENT LISTADERECURSOS (RECURSO | LISTADERECURSOS)*>

<!ATTLIST LISTADERECURSOS %NOHNAME;>

<!ELEMENT RECURSO  (TITULO,PALAVRASCHAVE,TEXTO>

<!ATTLIST RECURSOS %NOHNAME;>

<!ELEMENT  TELASGRAFICAS (LISTASTELASGRAFICAS*)>

<!ATTLIST TELASGRAFICAS %NOHNAME;>

<!ELEMENT LISTATELASGRAFICAS (TELAGRAFICA | LISTATELASGRAFICAS)*>

<!ATTLIST LISTATELASGRAFICAS %NOHNAME;>

<!ELEMENT TELAGRAFICA  (TITULO,PALAVRASCHAVE,DESCRICAO,TIPO,

LINHAS,COLUNAS,BORDA,VALORBORDA,FRAMESPACING,SCROLLING,NORESIZE) >

<!ATTLIST TELAGRAFICA %NOHNAME;>

<!ELEMENT TITULO  (#PCDATA) >

<!ATTLIST TITULO %NOHNAME;>

<!ELEMENT  PALAVRASCHAVE  (#PCDATA) >

<!ATTLIST PALAVRASCHAVE %NOHNAME;>

<!ELEMENT DESCRICAO  (#PCDATA) >

<!ATTLIST DESCRICAO %NOHNAME;>
```

```
<!ELEMENT TIPO  (#PCDATA) >

<!ATTLIST TIPO %NOHNAME;>

<!ELEMENT LINHAS  (#PCDATA) >

<!ATTLIST LINHAS %NOHNAME;>

<!ELEMENT COLUNAS  (#PCDATA) >

<!ATTLIST COLUNAS %NOHNAME;>

<!ELEMENT BORDA  (#PCDATA) >

<!ATTLIST BORDA %NOHNAME;>

<!ELEMENT BORDA  (#PCDATA) >

<!ELEMENT FRAMESPACING  (#PCDATA) >

<!ATTLIST FRAMESPACING %NOHNAME;>

<!ELEMENT SCROLLING  (#PCDATA) >

<!ATTLIST SCROLLING %NOHNAME;>

<!ELEMENT NORESIZE  (#PCDATA) >

<!ATTLIST NORESIZE %NOHNAME;>

<!ELEMENT DOCUMENTOS (LISTATOPICOS)* >

<!ATTLIST DOCUMENTOS %NOHNAME;>

<!ELEMENT TEXTO (#PCDATA) >

<!ATTLIST TEXTO %NOHNAME;>

<!ELEMENT FORMULAS (LISTAFORMULAS)* >

<!ATTLIST  FORMULAS %NOHNAME;>

<!ELEMENT LISTAFORMULAS (LISTAFORMULAS | FORMULA)* >

<!ATTLIST  LISTAFORMULAS %NOHNAME;>

<!ELEMENT FORMULA (TITULO,PALAVRASCHAVE,DESCRICAO,LATEX,MATHML)>

<!ATTLIST FORMULA %NOHNAME;>

<!ELEMENT LARGURA (#PCDATA)>

<!ATTLIST LARGURA  %NOHNAME;>

<!ELEMENT ALTURA (#PCDATA)>
```

```
<!ATTLIST ALTURA  %NOHNAME;>

<!ELEMENT COR (#PCDATA)

<!ATTLIST COR %NOHNAME;>

<!ELEMENT LEGENDA (#PCDATA)>

<!ATTLIST LEGENDA  %NOHNAME;>

<!ELEMENT TIPOLEGENDA (#PCDATA)>

<!ATTLIST  TIPOLEGENDA %NOHNAME;>

<!ELEMENT BORDA (#PCDATA)>

<!ATTLIST BORDA %NOHNAME;>

<!ELEMENT CORBORDA (#PCDATA)>

<!ATTLIST  CORBORDA %NOHNAME;>

<!ELEMENT VALORBORDA (#PCDATA) >

<!ATTLIST VALORBORDA %NOHNAME;>

<!ELEMENT ALINHAMENTO (#PCDATA) >

<!ATTLIST ALINHAMENTO  %NOHNAME;>

<!ELEMENT MATHTEX (#PCDATA) >

<!ATTLIST  MATHTEX %NOHNAME;>

<!ELEMENT CODIGO (#PCDATA) >

<!ATTLIST  CODIGO %NOHNAME;>

<!ELEMENT GRAFICOS (LISTAGRAFICOS)*>

<!ATTLIST GRAFICOS %NOHNAME;>

<!ELEMENT LISTAGRAFICOS (LISTAGRAFICOS,GRAFICO)*>

<!ATTLIST LISTAGRAFICOS %NOHNAME;>

<!ELEMENT GRAFICO (TITULO,PALAVRASCHAVE,DESCRICAO,CODIGO,GRAPHML) >

<!ATTLIST  GRAFICO %NOHNAME;>

<!ELEMENT GRAPHML (#PCDATA) >

<!ATTLIST GRAPHML %NOHNAME;>

<!ELEMENT TABELAS (LISTATABELAS)* >
```

```
<!ATTLIST TABELAS %NOHNAME;>

<!ELEMENT LISTATABELAS (LISTATABELAS | TABELA)*>

<!ATTLIST LISTATABELAS %NOHNAME;>

<!ELEMENT TABELA (TITULO,PALAVRASCHAVE,DESCRICAO,LINHAS,COLUNAS,

TEXTO,LINHA) >

<!ATTLIST TABELA %NOHNAME;>

<!ELEMENT LINHA (COLUNA,ALTURA)>

<!ATTLIST  LINHA %NOHNAME;>

<!ELEMENT COLUNA (LARGURA,TEXTO)

<!ATTLIST COLUNA %NOHNAME;>

<!ELEMENT FIGURAS (LISTAFIGURAS)* >

<!ATTLIST FIGURAS %NOHNAME;>

<!ELEMENT LISTAFIGURAS (LISTAFIGURA | FIGURA)*>

<!ATTLIST LISTAFIGURAS %NOHNAME;>

<!ELEMENT FIGURA (TITULO,DESCRICAO,PALAVRASCHAVE,ARQUIVO)>

<!ATTLIST FIGURA %NOHNAME;>

<!ELEMENT ARQUIVO (#PCDATA) >

<!ATTLIST ARQUIVO %NOHNAME;>

<!ELEMENT EXEMPLOS (LISTAEXEMPLOS)* >

<!ATTLIST  EXEMPLOS %NOHNAME;>

<!ELEMENT LISTAEXEMPLOS (LISTAEXEMPLOS | EXEMPLO)* >

<!ATTLIST LISTAEXEMPLOS %NOHNAME;>

<!ELEMENT EXEMPLO (TITULO,PALAVRASCHAVE,TEXTO)>

<!ATTLIST  EXEMPLO %NOHNAME;>

<!ELEMENT EXERCICIOS (LISTAEXERCICIOS)* >

<!ATTLIST  EXERCICIOS %NOHNAME;>

<!ELEMENT LISTAEXERCICIOS (LISTAEXERCICIOS | EXERCICIO)* >

<!ATTLIST LISTAEXERCICIOS EXEMPLO %NOHNAME;>
```

```
<!ELEMENT EXERCICIO (TITULO,PALAVRASCHAVE,DESCRICAO,RESOLUCAO,
RESPOSTA) >
<!ATTLIST  EXERCICIO EXEMPLO %NOHNAME;>
<!ELEMENT RESOLUCAO (#PCDATA)>
<!ATTLIST  RESOLUCAO %NOHNAME;>
<!ELEMENT RESPOSTA (#PCDATA) >
<!ATTLIST  RESPOSTA %NOHNAME;>
<!ELEMENT TEOREMAS (LISTATEOREMAS)* >
<!ATTLIST  TEOREMAS %NOHNAME;>
<!ELEMENT LISTATEOREMAS (LISTATEOREMAS | TEOREMA)* >
<!ATTLIST  LISTATEOREMAS %NOHNAME;>
<!ELEMENT TEOREMA (TITULO,PALAVRASCHAVE,DESCRICAO,RESOLUCAO,
RESPOSTA) >
<!ATTLIST  TEOREMA %NOHNAME;>
<!ELEMENT DEFINICOES (LISTADEFINICOES)* >
<!ATTLIST  DEFINICOES %NOHNAME;>
<!ELEMENT LISTADEFINICOES (LISTADEFINICOES | DEFINICAO)* >
<!ATTLIST  LISTADEFINICOES %NOHNAME;>
<1ELEMENT DEFINICAO (TITULO,PALAVRASCHAVE,TEXTO) >
<!ATTLIST  DEFINICAO %NOHNAME;>
<!ELEMENT GLOSSARIOS (LISTAGLOSSARIOS)* >
<!ATTLIST  GLOSSARIOS %NOHNAME;>
<!ELEMENT LISTAGLOSSARIOS (GLOSSARIO)* >
<!ATTLIST  LISTAGLOSSARIOS %NOHNAME;>
<!ELEMENT GLOSSARIO (LETRA)* >
<!ATTLIST  GLOSSARIO %NOHNAME;>
<!ELEMENT LETRA (TOPICO)* >
<!ATTLIST LETRA %NOHNAME;>
```

```
<!ELEMENT BIBLIOGRAFIAS (LISTABIBLIOGRAFIAS)* >

<!ATTLIST BIBLIOGRAFIAS %NOHNAME;>

<!ELEMENT LISTABIBLIOGRAFIAS (LISTABIBLIOGRAFIAS | BIBLIOGRAFIA)* >

<!ATTLIST  LISTABIBLIOGRAFIA %NOHNAME;>

<!ELEMENT BIBLIOGRAFIA (ABREVIATURA,AUTORES,TITULO,WEB,OBS,

PALAVRASCHAVE) >

<!ATTLIST BIBLIOGRAFIA %NOHNAME;>

<!ELEMENT ABREVIATURA (#PCDATA) >

<!ATTLIST  ABREVIATURA %NOHNAME;>

<!ELEMENT AUTORES (#PCDATA) >

<!ATTLIST  AUTORES %NOHNAME;>

<!ELEMENT WEB (#PCDATA) >

<!ATTLIST  WEB %NOHNAME;>

<!ELEMENT OBS (#PCDATA) >

<!ATTLIST  OBS %NOHNAME;>

<!ELEMENT APPLETS (LISTAAPPLETS)* >

<!ATTLIST  APPLETS %NOHNAME;>

<!ELEMENT LISTAAPPLETS (LISTAAPPLETS,APPLET)* >

<!ATTLIST  LISTAAPPLET %NOHNAME;>

<!ELEMENT APPLET (TITULO,PALAVRASCHAVE,DESCRICAO,PATH,BASE,

LARGURA,ALTURA,ALINHAMENTO,(PARAMETRO)*,TEXTO) >

<!ATTLIST  APPLET %NOHNAME;>

<!ELEMENT PATH (#PCDATA) >

<!ATTLIST  PATH %NOHNAME;>

<!ELEMENT BASE (#PCDATA) >

<!ATTLIST  BASE %NOHNAME;>

<!ELEMENT PARAMETRO >

<!ATTLIST  %NOHNAME; NOME   CDATA  #REQUIRED
```

```
  VALOR  CDATA  #REQUIRED>
<!ELEMENT FLASHS (LISTAFLASHS)* >
<!ATTLIST  FLASHS %NOHNAME;>
<!ELEMENT LISTAFLASHS (LISTAFLASHS,FLASH)* >
<!ATTLIST  LISTAFLASHS %NOHNAME;>
<!ELEMENT FLASH (TITULO,PALAVRASCHAVE,DESCRICAO,ARQUIVO,LARGURA,
ALTURA,ALINHAMENTO,SCALE,COR,LOOP,AUTOPLAY,QUALITY,(PARAMETRO)*,
TEXTO)>
<!ATTLIST  FLASH %NOHNAME;>
<!ELEMENT SCALE (#PCDATA) >
<!ATTLIST  SCALE %NOHNAME;>
<!ELEMENT LOOP (#PCDATA) >
<!ATTLIST  LOOP %NOHNAME;>
<!ELEMENT AUTOPLAY (#PCDATA) >
<!ATTLIST  AUTOPLAY %NOHNAME;>
<!ELEMENT QUALITY (#PCDATA) >
<!ATTLIST  QUALITY %NOHNAME;>
<!ELEMENT VIDEOS (LISTAVIDEOS)* >
<!ATTLIST  VIDEOS %NOHNAME;>
<!ELEMENT LISTAVIDEOS (LISTAVIDEOS | VIDEO)* >
<!ATTLIST LISTAVIDEOS %NOHNAME;>
<!ELEMENT VIDEO (TITULO,PALAVRASCHAVE,DESCRICAO,ARQUIVO,LARGURA,
ALTURA,ALINHAMENTO) >
<!ATTLIST VIDEO %NOHNAME;>
```

Figura I.1 DTD RCML.

Anexo II

Uma breve descrição sobre a linguagem LaTex

II.1 Introdução

O sistema de produção de documentos LaTeX é uma versão especial do programa LaTeX. LaTeX é um programa sofisticado destinado para a produção de documentos de alta qualidade, principalmente com conteúdo matemático. LaTeX adiciona ao TeX uma coleção de macros que simplificam o documento permitindo que o usuário concentre-se na estrutura do documento [WIL 95].

O ambiente matemático do LaTeX permite uma descrição de expressões preservando o seu conteúdo. Serão exemplificados aqui alguns exemplos de utilização e de sintaxe.

II.2 Sintaxe

II.2.1 *Sintaxe Básica*

A sintaxe básica do LaTeX é a seguinte :

```
\NOME_COMANDO {ARG1 }{ARG2 }....{ARGN}
```

Figura II.1 LaTeX – Sintaxe Básica.

Os comandos em LaTeX iniciam com uma barra vertical, "\", seguido do nome do comando e, se existir, de uma lista de argumentos, cujos valores estão entre chaves ("{}").

II.2.2 *Algumas Funções LaTeX*

Nesta seção será mostrada uma lista de funções utilizadas na linguagem LaTeX.

* Função Somatório

a. Definição :

A função somatório é descrita através do comando "sum" seguido de três argumentos : limite inferior (INF), limite superior (SUP), e expressão (EXP). Os operadores "_" e "^" representam os dados em uma posição inferior e superior, respectivamente.

b. Sintaxe :

```
\sum _{INF}^{SUP}{EXP}
```

Figura II.2 LaTeX – Função Somatório.

c. Exemplo :

Para a expressão $\sum_{i=1}^{n} x_i^2 - \sqrt{x_i - n}$, o código LaTex correspondente será: `\sum_{i=1}^{n}{x_{i}^{2}-\sqrt{xi-n}}.`

Figura II.3 LaTeX – Exemplo Função Somatório.

* Função Integral

a. Definição :

A função integral é descrita através do comando "int" seguido de três argumentos: limite inferior (INF), limite superior (SUP), e expressão (EXP). Os operadores "_" e "^" representam os dados em uma posição inferior e superior, respectivamente.

b. Sintaxe :

```
\int _{INF}^{SUP}{EXP}
```

Figura II.4 LaTeX – Função Integral.

c. Exemplo :

Seja a função $\int_a^b x^9 - 9$, o código LaTex correspondente seria o seguinte : `\int_{a}^{b}{x^9-9}}.`

Figura II.5 LaTeX – Exemplo Função Integral.

* Funções Trigonométricas

a. Definição :

A função seno é descrita através do comando "sin" seguido de um argumento: expressão (EXP).

b. Sintaxe:

```
\sin{EXP}
```

Figura II.6 LaTeX – Funções Trigonométricas.

c. Exemplo :

Para a expressão $sen(x^2-2)$, o código LaTex correspondente seria o seguinte : \sin{x^2-2}.

Figura II.7 LaTeX – Exemplo Funções Trigonométricas.

* Frações

a. Definição :

Em LaTeX, uma fração é descrita através do comando “frac” seguido de dois argumentos : numerador (NUM) e denominador(DEN).

b. Sintaxe :

```
\frac{NUM}{DEN}
```

Figura II.8 LaTeX – Frações.

c. Exemplo :

Seja a fração $\frac{\sqrt{x-x^2}-x^2}{x^3-3x^2+1}$, o código LaTex correspondente seria o seguinte : \frac{\sqrt{x-x^2}-x^2}{x^2-3*x^2+1}. (o asterisco para indicação de produto não faz parte do Látex. Pode ser necessário para uma marcação de conteúdo, não de apresentação)

Figura II.9 LaTeX – Exemplo Frações.

* Raíz

a. Definição :

Uma raíz é definida em LaTeX através do comando “sqrt” seguido de dois argumentos :

- o índice da raíz que é opcional. Caso a raíz seja quadrada, este argumento não é necessário;

- a expressão.

b. Sintaxe :

```
\sqrt{EXP}
```

Figura II.10 LaTeX – Raiz.

c. Exemplo :

Seja a função $\sqrt[3]{x^2-y}$, o código LaTex correspondente seria o seguinte : \sqrt{3}{x^2-y}.

Figura II.11 LaTeX – Exemplo Raiz.

II.2.3 *Alguns Símbolos*

A linguagem LaTeX possui um conjunto de comandos para representar símbolos matemáticos bem como letras gregas. Abaixo serão mostrados alguns deles.

* Alpha
  - **a.** ***Símbolo*:** α
  - **b.** ***Código*:** \alpha
* Beta
  - **a.** ***Símbolo*:** β
  - **b.** ***Código*:** \beta
* Lambda
  - **a.** ***Símbolo*:** λ
  - **b.** ***Código*:** \lambda
* Pi
  - **a.** ***Símbolo*:** π
  - **b.** **Código:** \pi
* Delimitadores (“{}“,”()”)
  - **a.** ***Símbolo*:** ”{”,”}”,”(”,”)”
  - **b.** ***Código*:** \left{,\right},\left(,\right)
* Setas
  - **a.** ***Símbolo*:** $\leftarrow$
  - **b.** ***Código*:** \leftarrow
* Setas Duplas
  - **a.** ***Símbolo*:** $\Leftarrow$
  - **b.** ***Código*:** \Leftarrow

II.2.4 *Definindo elementos mistos através do comando “array”*

* Definindo uma matriz

Para a definição da matriz $\begin{pmatrix} a & b & c \\ d & e & f \\ 1 & 2 & 3 \end{pmatrix}$ em LaTex será necessário a composição dos seguintes elementos : \left, \right e \begin{array} da seguinte forma:

```
\left( \begin{array}{

   a  b  c  \\

   d  e  f  \\

   1  2  3  \\

 }\end{array}\right)
```

Figura II.12 LaTeX – Definindo uma matriz.

* Definindo uma função composta

Para a definição da função composta $f(x) = \begin{cases} x^2 - 1, x > 0 \\ x^2 - 2, x < 0 \\ x^2 - 3, x = 0 \end{cases}$, será necessária a composição dos seguintes elementos : \left,\begin{array} da seguinte forma:

```
f(x)=\left{\begin{array}x^2-1,x>0  \\x^2-2,x<0 \\  x^2-3,x=0
\end{array}
```

Figura II.13 LaTeX – Definindo uma função composta.

II.3 Vantagens da Linguagem LaTeX

* Torna-se fácil para a comunidade de usuários LaTeX;
* Possui uma estrutura que define a sintaxe e semântica da fórmula;
* Possui um riquíssimo conjunto de definições, símbolos, etc.

II.4 Devantagens da Linguagem LaTex

* Softwares de suporte tendem a ser complicados;
* Possui algumas limitações de pesquisa e autoria;
* A audiência é limitada para os usuários de LaTeX;
* É uma linguagem difícil para muitos usuários;
* Não é integrada com a Web.

Anexo III

DTD GraphML

```
<!ELEMENT Grafico (Item)*>

<!ATTLIST Grafico

   Tipo #PCDATA

   Titulo #PCDATA

>

<!ELEMENT Item (Valor)>

<!ATTLIST Item

    Descricao #PCDATA

    Titulo  #PCDATA

    Cor    #PCDATA

   Degradee #PCDATA >

<!ELEMENT Valor (#PCDATA |(EX,EY) )>

<!ELEMENT EX (#PCDDATA)>

<!ELEMENT EY (#PCDDATA)>
```

Figura III.1 DTD GraphML.

Anexo IV

O DOM - *Document Object Model*

IV.1 Introdução

O DOM (*Document Object Model*) é uma API (*Application programming Interface*) para a manipulação de documentos HTML válidos e documentos XML bem formados. Ele define a estrutura lógica de documentos e a forma no qual um documento é acessado e manipulado. Na especificação do DOM, [W3C 03c], o termo "documento" é usado num sentido amplo. Crescentemente, XML está sendo usado como uma forma de representar muitos tipos diferentes de informações que podem ser armazenadas em diversos sistemas, e, mais que isto, está sendo tradicionalmente visto como dados ao contrário de documentos. Ainda assim, XML representa estes dados como documentos, e o DOM pode ser usado para gerenciar estes dados.

Com o *Document Object Model*, programadores podem construir documentos, navegar em sua estrutura, e adicionar, modificar, ou apagar elementos e conteúdo. Qualquer dado encontrado em um documento HTML ou XML pode ser acessado, alterado, apagado ou adicionado através do uso do *Document Object Model*, com poucas exceções – em particular, as Interfaces DOM para os subconjuntos internos e externos XML que ainda não foram especificadas.

Como uma especificação W3C [W3C 03m], um importante objetivo do DOM é oferecer uma interface de programação padrão que possa ser usada em uma ampla variedade de ambientes e aplicações. O DOM é destinado para ser usado com qualquer linguagem de programação.

IV.2 O que é o Document Object Model

O termo *Document Object Model* foi aplicado a navegadores da web por algum tempo. Os objetos como janela, documento e histórico foram considerados parte do modelo orientado a objetos do navegador. Em qualquer aplicação de desenvolvimento para a web, no entanto, poderá ser observada uma grande variação, no que se refere à implementação, entre diferentes navegadores. Objetivando criar um meio mais padronizado de acessar e manipular estrutura de documentos na web, a W3C produziu especificações resultando no que hoje é chamado de DOM.

O DOM é uma definição neutra de plataforma e de linguagem, ou seja, as interfaces são definidas para os diferentes objetos englobando o DOM, mas nenhuma característica específica da implementação é fornecida, e ele poderá ser utilizado em qualquer linguagem de programação. Isto permite, por exemplo, que depósitos de dados legados sejam acessados ao implementar o DOM como um fino invólucro ao redor das funções de acesso a esses dados legados. Os objetos no DOM permitem que o programador leia, pesquise, modifique, adicione e exclua um objeto de documento. Através desta interface define-se uma funcionalidade padrão para navegação e manipulação de conteúdo e estrutura dos documentos XML e HTML.

IV.3 Porque usar o DOM

Baseado no que foi exposto acima, observa-se que o DOM é uma boa estratégia a ser utilizada quando o objetivo é a manipulação de dados em formato XML. Usar o DOM tem diversas vantagens em relação a outros mecanismos disponíveis para geração de documentos XML :

* Garantia de boa formação e gramática adequada;
* Abstração do conteúdo da gramática;
* Simplificação da manipulação interna de documentos;
* Reflexão das estruturas típicas dos bancos de dados relacional e hierárquico.

Resumindo, quando lemos e manipulamos documentos XML, usando o DOM, garantiremos o maior nível de interoperabilidade entre várias plataformas. No entanto, usar o DOM não é necessariamente a melhor estratégia, especialmente quando trabalhamos com grandes arquivos. Para evitar o *overheard* de carregar o documento inteiro na memória, grandes arquivos XML podem ser processados usando-se um analisador orientado a eventos como o SAX [SAX 03].

Anexo V

Linguagem RCML

Segue abaixo uma breve descrição dos elementos da linguagem RCML. A DTD completa pode ser encontrada no Anexo I.

* Projeto

a. Definição

Este elemento possui a função de agrupar todos os outros definidos na DTD da linguagem RCML. Na terminologia XML, este elemento é definido como o "elemento de documento" (document element).

b. Estrutura

O elemento projeto contém os seguintes fragmentos : "ObjetosdeEstilos", "ObjetosdeDados", "ObjetosMultimedia", "ObjetosExternos". A Figura V.1 mostra sua estrutura hierárquica.

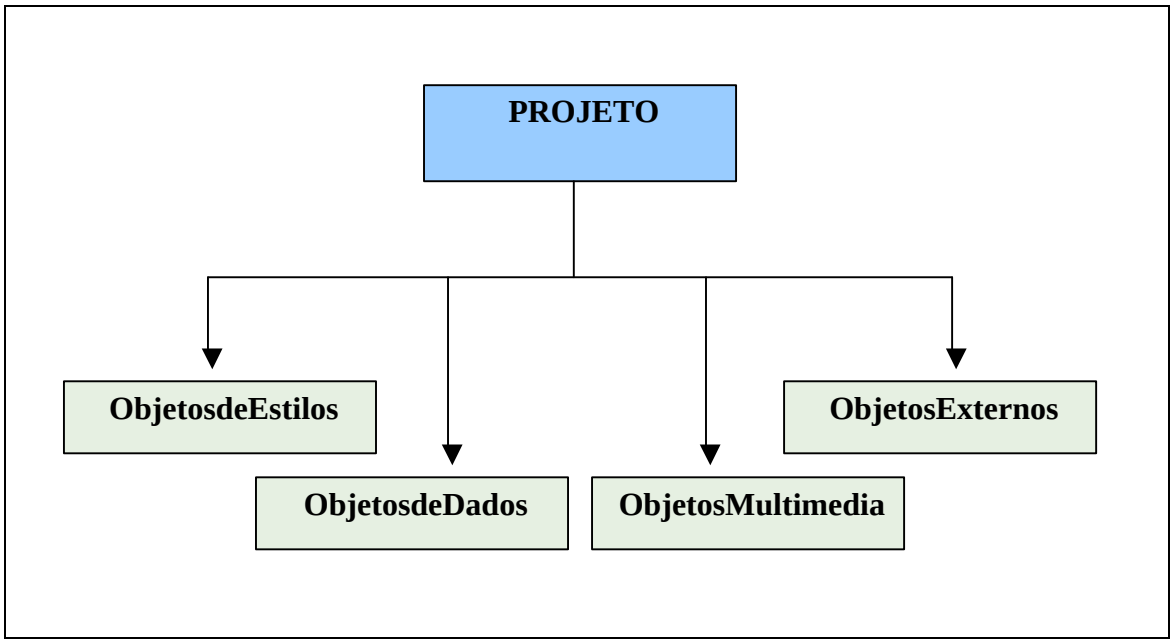


Figura V.1RCML – Estrutura do elemento "Projeto".

c. Sintaxe

```
<!ELEMENT Projeto( ObjetosdeEstilos, ObjetosdeDados,
ObjetosMultimedia, ObjetosExternos)>

<!ATTLIST  Projeto %Nohname;>
```

Figura V.2RCML – Sintaxe do elemento “Projeto”.

d. Exemplo

```
<Projeto NohName = “Dissertação”>

    <ObjetosdeEstilos ...>

        ....

    </ObjetosdeEstilos>

    <ObjetosdeDados ...>

    ......

    </ObjetosdeDados>

    <ObjetosExternos ...>

    .......

    </ObjetosExternos>

</Projeto>
```

Figura V.3RCML – Exemplo de utilização do elemento “Projeto”.

* ObjetosdeEstilos

a. Definição

O elemento “ObjetosdeEstilos” armazena informações referentes à formatação dos dados. Embora o suporte inicial seja apenas HTML, sua proposta é guardar renderizações em outros formatos como, por exemplo, DOC (Microsoft Word), XML, etc.

b. Estrutura

Este elemento possui a função de agrupar o fragmento “EstilosHTML”. A Figura V.4 mostra sua estrutura hierárquica.

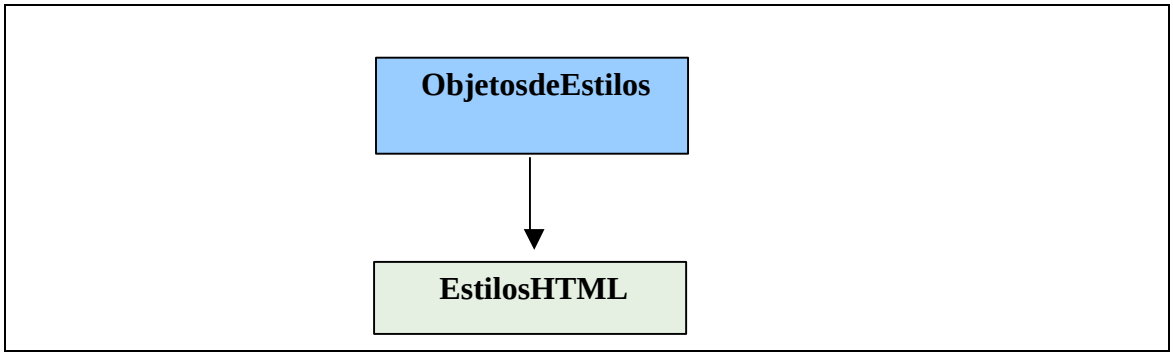


Figura V.4RCML – Estrutura do elemento “ObjetosdeEstilos”.

c. Sintaxe

```
<!ELEMENT ObjetosdeEstilos(EstilosHTML)>

<!ATTLIST  ObjetosdeEstilos %Nohname;>
```

Figura V.5RCML – Sintaxe do elemento “ObjetosdeEstilos”.

d. Exemplo

```
<ObjetosdeEstilos Nohname=”ObjetosdeEstilos”>

   <EstilosHTML> ... </EstilosHTML>

</ObjetosdeEstilos>
```

Figura V.6RCML – Exemplo de utilização do elemento “ObjetosdeEstilos”.

* EstilosHTML

a. Definição

Este elemento armazena renderizações HTML para dados de diferentes repositórios.

b. Estrutura

O objeto “EstilosHTML” contém o fragmento “ListadeEstilos”. A Figura V.7 mostra sua estrutura hierárquica.

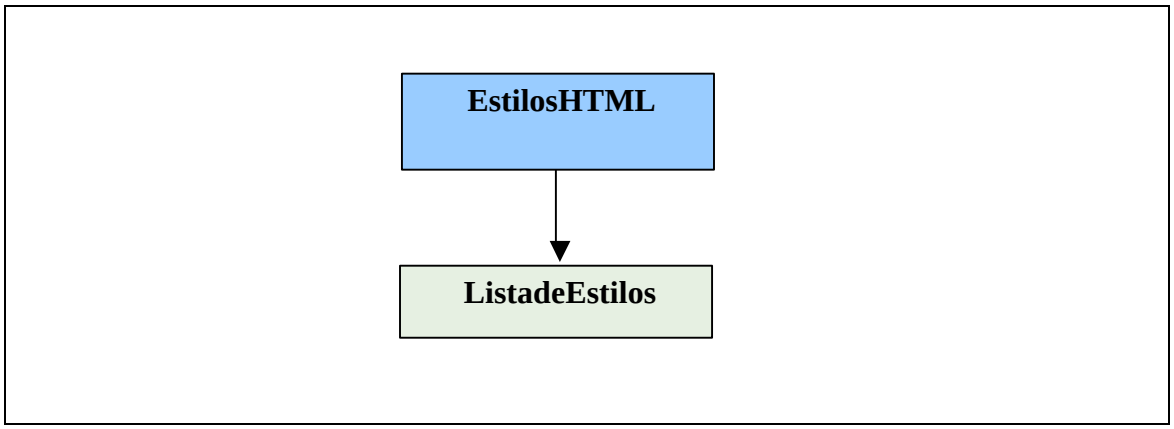


Figura V.7 RCML – Estrutura do elemento “EstilosHTML”.

c. Sintaxe

```
<!ELEMENT EstilosHTML (ListadeEstilos)*>)

<!ATTLIST  EstilosHTML %Nohname;>
```

Figura V.8 RCML – Sintaxe do elemento “EstilosHTML”.

d. Exemplo :

```
<EstilosHTML  NohName = “Estilos”>

   <ListadeEstilos> ... </ListadeEstilos>

   <Estilos> ... </ListadeEstilos>

</EstilosHTML>
```

Figura V.9 RCML – Exemplo de utilização do elemento “EstilosHTML”.

* ListadeEstilos

a. Definição

O elemento “ListadeEstilos” serve para agrupar vários elementos “Estilo”. Este objeto armazena uma listagem de estilos. Pode também agrupar elementos do mesmo tipo, ou seja, “ListadeEstilos”.

b. Estrutura

Este elemento contém os fragmentos “Titulo”, “Obs”, “ListadeEstilos” e “Estilo”. A Figura V.10 mostra sua estrutura hierárquica.

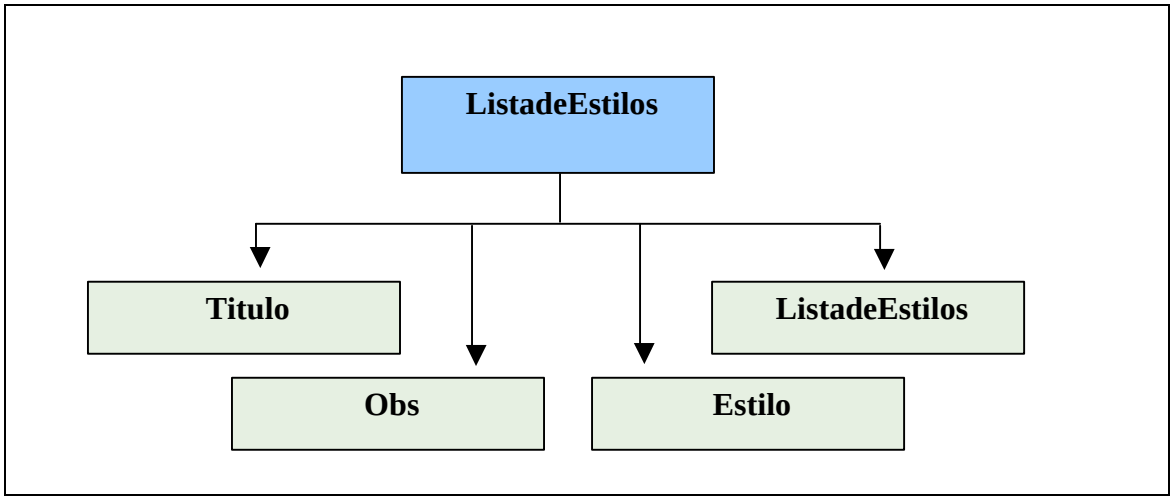


Figura V.10 RCML – Estrutura do elemento “ListadeEstilos”.

c. Sintaxe

```
<!ELEMENT ListadeEstilos (Titulo, Observacao,( Estilo|
ListadeEstilos)*>)

<!ATTLIST  ListadeEstilos %Nohname;>
```

Figura V.11 RCML – Sintaxe do elemento “ListadeEstilos”.

d. Exemplo

```
<ListadeEstilos  NohName = “ListadeEstilos”>

 <Titulo> ... </Titulo> <Obs> ... </Obs>

 <Estilo> ... </Estilo>

 <ListadeEstilos> ... </ListadeEstilos>

 <Estilo> ... </Estilo>

</ListadeEstilos>
```

Figura V.12 RCML – Exemplo de utilização do elemento “ListadeEstilos”.

* Estilo

a. Definição

O objeto “Estilo” armazena uma determinada renderização inicialmente apenas em formato HTML.

b. Estrutura

Este elemento contém os fragmentos “Titulo”, “Descricao”, “PalavrasChave” e “Texto”. A Figura V.13 mostra sua estrutura hierárquica.

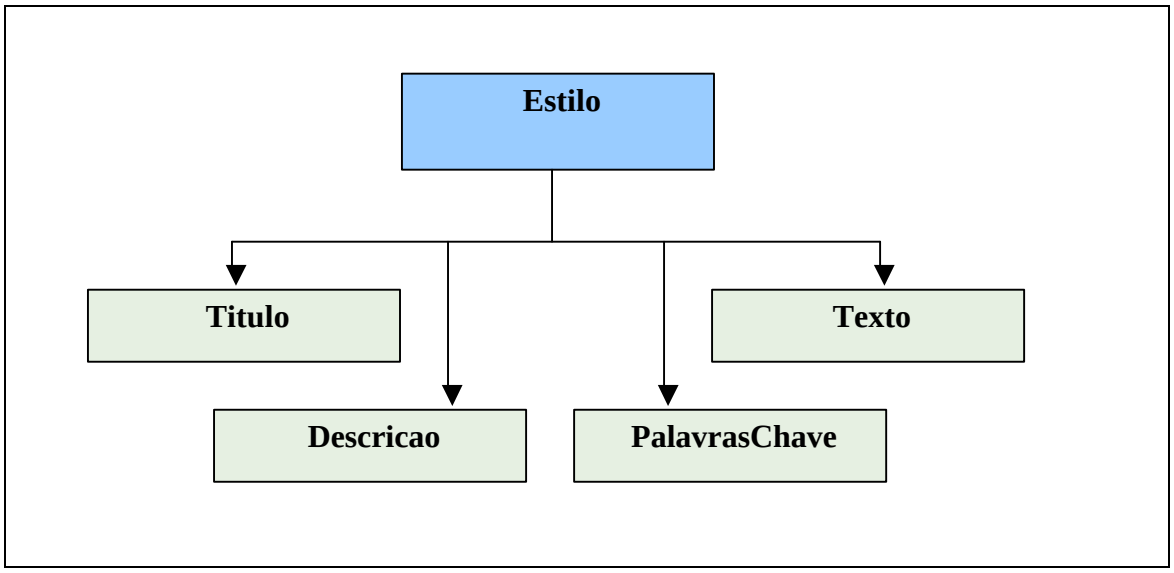


Figura V.13 RCML – Estrutura do elemento “Estilo”.

c. Sintaxe

```
<!ELEMENT Estilo (Titulo,Descricao,PalavrasChave,Texto)>)

<!ATTLIST  Estilo %Nohname;>
```

Figura V.14 RCML – Sintaxe do elemento “Estilo”.

d. Exemplo

```
<Estilo  NohName = “Estilos”>

   <Titulo> ... </Titulo>

   <Descricao> ... </Descricao>

   <PalavrasChave> ... </PalavrasChave>

   <Texto> ... </Texto>

</Estilo>
```

Figura V.15 RCML – Exemplo de utilização do elemento “Estilo”.

* ObjetosdeDados

a. Definição

O elemento “ObjetosdeDados” armazena informações referentes aos dados de um determinado repositório.

b. Estrutura

Este elemento possui a função de agrupar os fragmentos “Recursos”, “Tabelas”, “Figuras”, Formulas”, “Graficos”, “Exemplos”, “Exercicios”, “Teoremas”, “Definicoes”, “Glossarios”, “Artigos”, “Arquivos”, “Bibliografias”. A Figura V.16 mostra sua estrutura hierárquica.

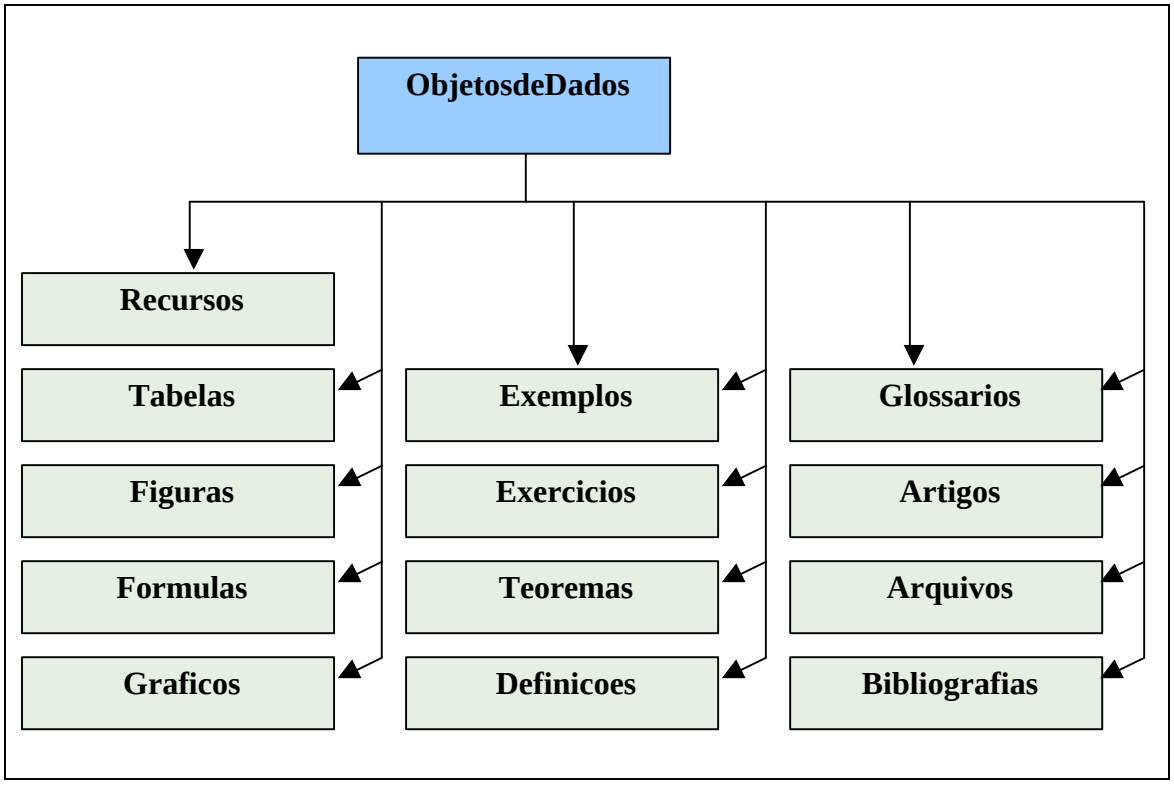


Figura V.16 RCML – Estrutura do elemento “ObjetosdeDados”.

c. Sintaxe

```
<!ELEMENT ObjetosdeDados(Recursos,Tabelas,Figuras,Formulas, Graficos,
Exemplos,Exercicios,Teoremas,Definicoes,Glossarios,Artigos,Arquivos,
Bibliografias)>

<!ATTLIST  ObjetosdeDados %Nohname;>
```

Figura V.17 RCML – Sintaxe do elemento “ObjetosdeDados”.

d. Exemplo

```
<ObjetosdeDados Nohname=”ObjetosdeDados”>

   <Recursos>   ... </Recursos>

   <Tabelas>    ... </Tabelas>

   <Figuras>    ... </Figuras>

   <Formulas>   ... </Formulas>

   <Graficos>   ... </Graficos>

   <Exemplos>   ... </Exemplos>

   <Exercicios> ... </Exercicios>

   <Teoremas>   ... </Teoremas>

   <Definicoes> ... </Definicoes>

   <Glossarios> ... </Glossarios>

   <Artigos>    ... </Artigos>

   <Arquivos>   ... </Arquivos>

   <Bibliografias> ... </Bibliografias>

</ObjetosdeEstilos>
```

Figura V.18 RCML – Exemplo de utilização do elemento “ObjetosdeDados”.

* Recursos

a. Definição

O elemento “Recursos” serve para agrupar vários elementos “ListadeRecursos”.

b. Estrutura

Este elemento contém o fragmento “ListadeRecursos”. A Figura V.19 mostra sua estrutura hierárquica.

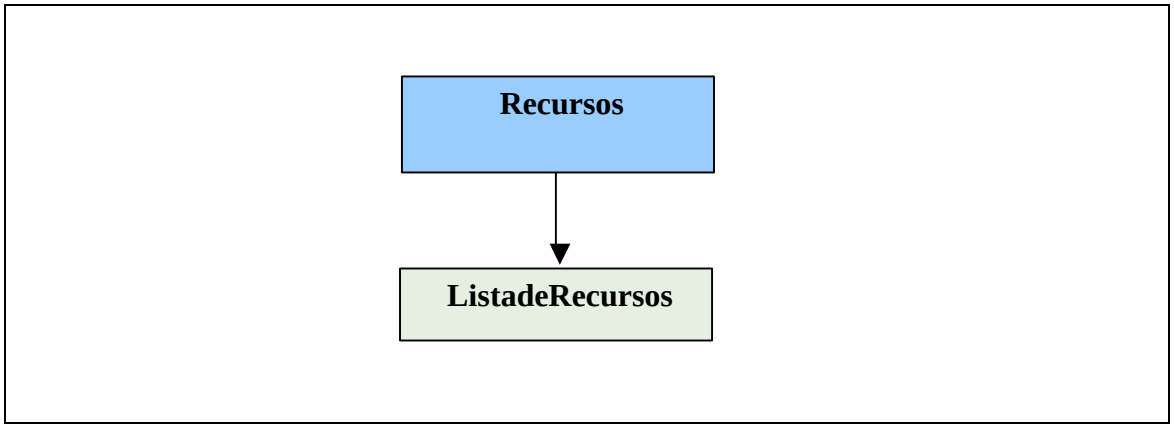


Figura V.19 RCML – Estrutura do elemento “Recursos”.

c. Sintaxe

```
<!ELEMENT Recursos (ListadeRecursos)*>)

<!ATTLIST  Recursos %Nohname;>
```

Figura V.20 RCML – Sintaxe do elemento “Recursos”.

d. Exemplo

```
<Recursos  NohName = “Recursos”>

 <ListadeRecursos> ... </ListadeRecursos>

 <ListadeRecursos> ... </ListadeRecursos>

 <ListadeRecursos> ... </ListadeRecursos>

</Recursos>
```

Figura V.21 RCML – Exemplo de utilização do elemento “Recursos”.

* ListadeRecursos

a. Definição

O elemento “ListadeRecursos” serve para agrupar vários elementos “Recurso”. Este objeto armazena uma listagem de recursos. Pode também agrupar elementos do mesmo tipo, ou seja, “ListadeRecursos”.

b. Estrutura

Este elemento contém os fragmentos “Titulo”, “Obs”, “ListadeRecursos” e “Recurso”. A Figura V.22 mostra sua estrutura hierárquica.

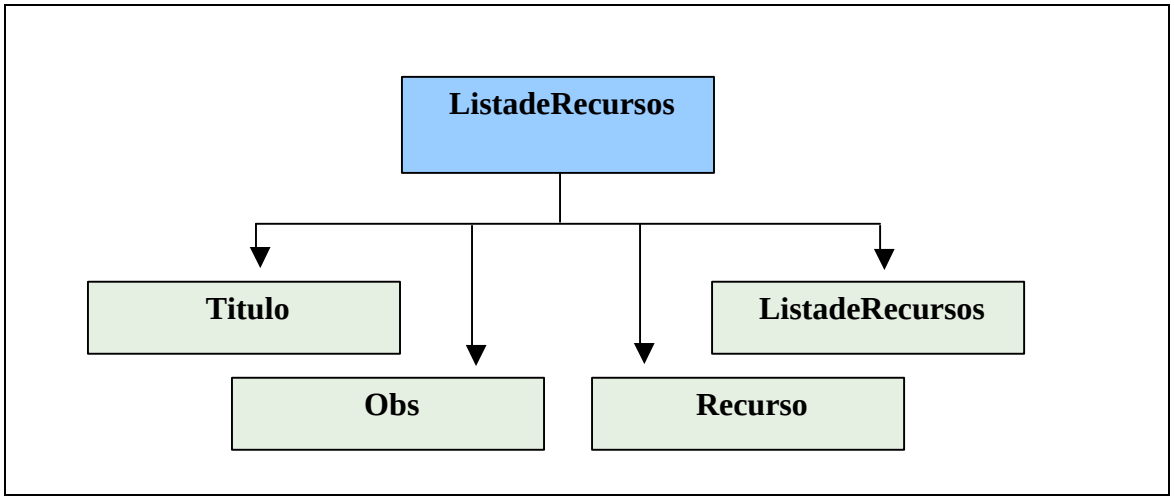


Figura V.22 RCML – Estrutura do elemento “ListadeRecursos”.

c. Sintaxe

```
<!ELEMENT ListadeRecursos (Titulo, Observacao,(Recurso |
ListadeRecursos)*>)

<!ATTLIST  ListadeRecursos %Nohname;>
```

Figura V.23 RCML – Sintaxe do elemento “ListadeRecursos”.

d. Exemplo

```
<ListadeRecursos  NohName = “Recursos”>

 <Titulo> ... </Titulo>

 <Obs> ... </Obs>

 <Recurso> ... </Recurso> <Recurso> ... </Recurso>

 <ListadeRecursos> ... </ListadeRecursos></ListadeRecursos>
```

Figura V.24 RCML – Exemplo de utilização do elemento “ListadeRecursos”.

* Recurso

a. Definição

O elemento “Recurso” serve para armazenar qualquer tipo de conteúdo independente do domínio de aplicação. Seu objetivo é adicionar a capacidade de generalização à linguagem RCML.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “PalavrasChave” e “Texto”. A Figura V.25 mostra sua estrutura hierárquica.

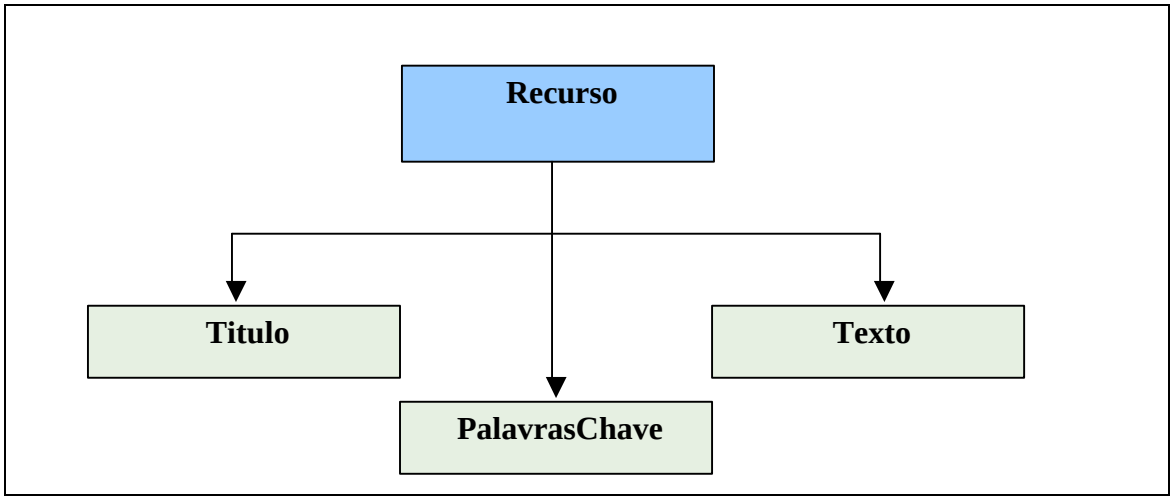


Figura V.25 RCML – Estrutura do elemento “Recurso”.

c. Sintaxe

```
<!ELEMENT Recurso (Titulo,PalavrasChave,Texto)>)

<!ATTLIST  Recurso %Nohname;>
```

Figura V.26 RCML – Sintaxe do elemento “Recurso”.

d. Exemplo

```
<Recurso  NohName = “Tópicos”>

   <Titulo> ... </Titulo>

   <PalavrasChave> ... </PalavrasChave>

   <Texto> ... </Texto>

</Recurso>
```

Figura V.27 RCML – Exemplo de utilização do elemento “Recurso”.

* Tabelas

a. Definição

O elemento “Tabelas” serve para agrupar vários elementos do tipo “ListadeTabelas”.

b. Estrutura

Este objeto contém o fragmento “ListadeTabelas”. A Figura V.28 mostra sua estrutura hierárquica.

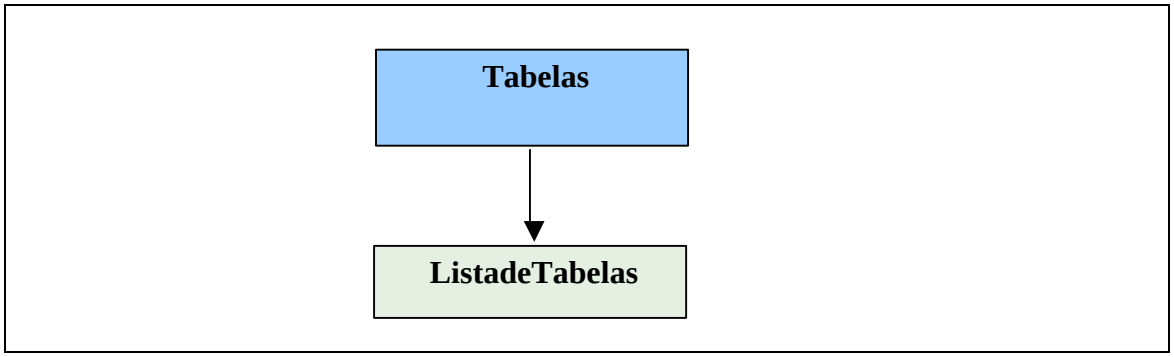


Figura V.28 RCML – Estrutura do elemento “Tabelas”.

c. Sintaxe

```
<!ELEMENT Tabelas (ListadeTabelas)*>)

<!ATTLIST  Tabelas %Nohname;>
```

Figura V.29 RCML - Sintaxe do elemento “Tabelas”.

d. Exemplo

```
<Tabelas  NohName = “Tabelas”>

  <ListadeTabelas ... > .... </ListadeTabelas>

  <ListadeTabelas ... > .... </ListadeTabelas>

  <ListadeTabelas ... > .... </ListadeTabelas>

</Tabelas>
```

Figura V.30 RCML - Exemplo de utilização do elemento “Tabelas”.

* ListadeTabelas

a. Definição

O elemento “ListadeTabelas” serve para agrupar vários elementos do tipo “Tabela”. Este elemento armazena uma listagem de tabelas. Pode também agrupar elementos do mesmo tipo, ou seja, “ListadeTabelas”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “Tabela” e “ListadeTabelas”. A Figura V.31 mostra sua estrutura hierárquica.

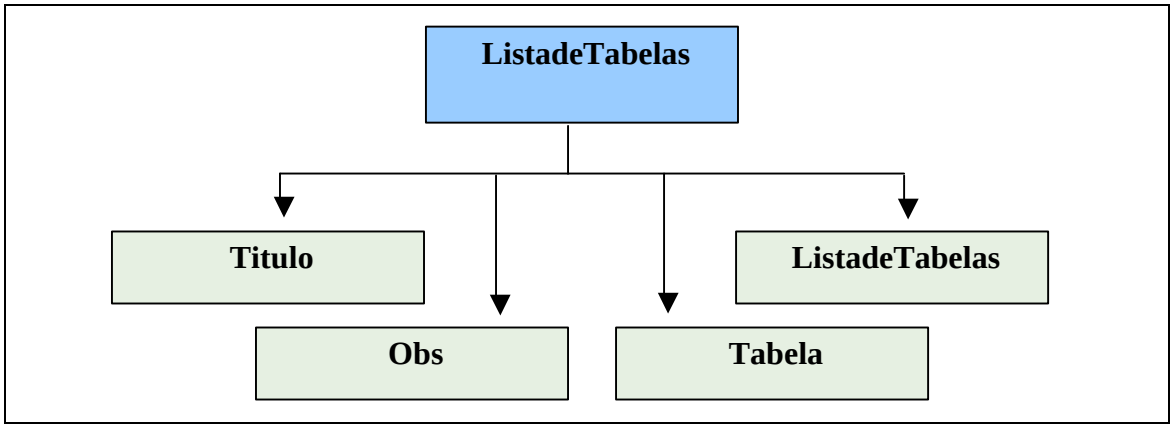


Figura V.31 RCML – Estrutura do elemento “ListadeTabelas”.

c. Sintaxe

```
<!ELEMENT ListadeTabelas (Titulo, Observacao,(Tabela |
ListadeTabelas)* >

<!ATTLIST ListadeTabelas %Nohname;>
```

Figura V.32 RCML - Sintaxe do elemento “ListadeTabelas”.

d. Exemplo

```
<ListadeTabelas NohName = “Tabelas do Capítulo 1”>

  <Titulo> Teste </Titulo>

  <Obs> .... </Obs>

  <Tabela ....>   ...  </Tabela>

  <Tabela ....>  ...  </Tabela>

  <ListadeTabelas ....>  ...  </ListadeTabelas>

</ListadeTabelas>
```

Figura V.33 RCML - Exemplo de utilização do elemento “ListadeTabelas”.

* Tabela

a. Definição

Um elemento do tipo “Tabela” estrutura todas as linhas e colunas de uma determinada tabela.

b. Estrutura

Este objeto possui os fragmentos “Titulo”, “PalavrasChave”, “Descricao” e “Linha”. A Figura V.34 mostra sua estrutura hierárquica.

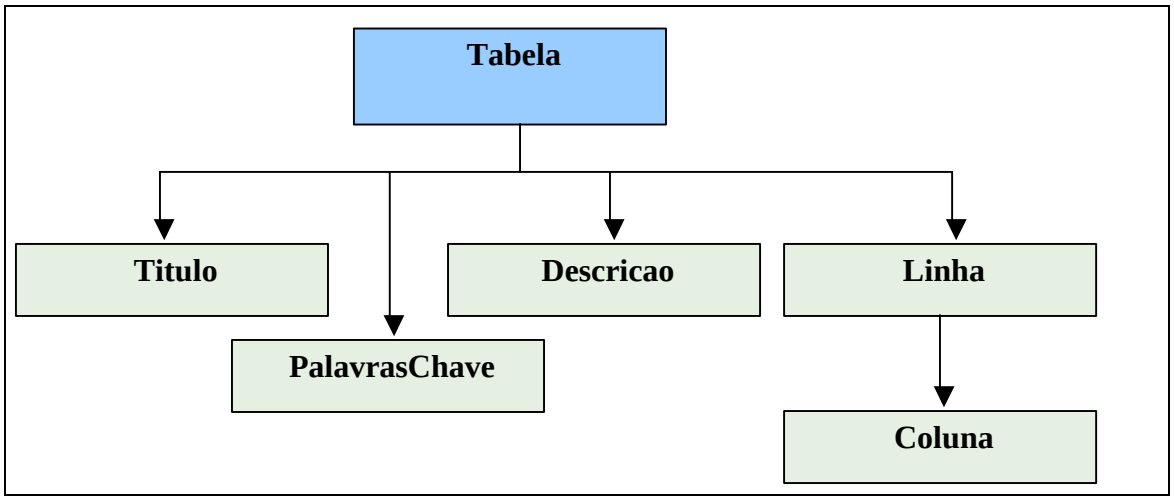


Figura V.34 RCML – Estrutura do elemento “Tabela”.

c. Sintaxe

```
<!ELEMENT Tabela (Linha)*,  Texto >

<!ELEMENT Linha (Coluna)+>

<!ELEMENT Coluna (Texto)>

<!ATTLIST Tabela %Nohname;>
```

Figura V.35 RCML - Sintaxe do elemento “Tabela”.

d. Exemplo

```
<Tabela NohName = “Tabela 1.1”>

  <Titulo> Teste </Titulo>  <PalavrasChave> .... </PalavrasChave>

  <Descricao> Esta tabela descreve........ </Descricao>

   <Linha> <Coluna> ..... </Coluna>  </Linha>

  <Linha>  <Coluna> ..... </Coluna>  </Linha></Tabela>
```

Figura V.36 RCML – Exemplo de utilização do elemento “Tabela”.

* Figuras

a. Definição

O elemento “Figuras” serve para agrupar vários elementos do tipo “ListadeFiguras”.

b. Estrutura

O objeto “Figuras” contém o fragmento “ListadeFiguras”. A Figura V.37 mostra sua estrutura hierárquica.

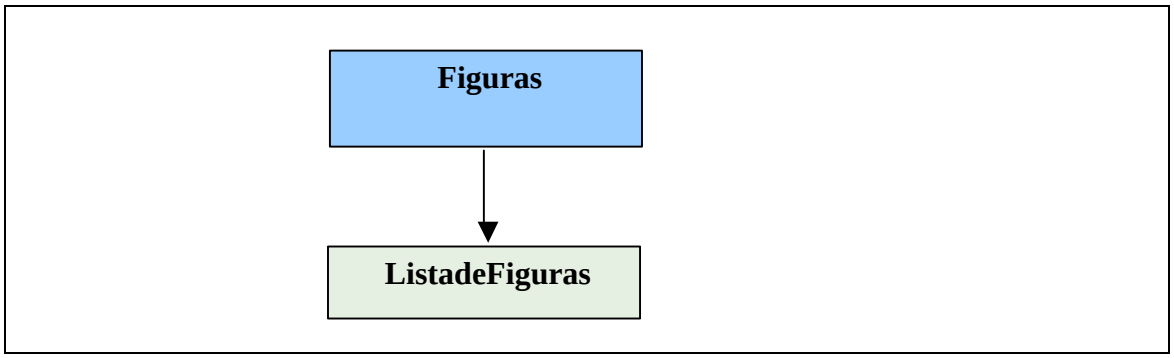


Figura V.37 RCML – Estrutura do elemento “Figuras”.

c. Sintaxe

```
<!ELEMENT Figuras (ListadeFiguras)*>)

<!ATTLIST  Figuras %Nohname;>
```

Figura V.38 RCML – Sintaxe do elemento “Figuras”.

d. Exemplo

```
<Figuras  NohName = “Figuras”>

  <ListadeFiguras ... > .... </ListadeFiguras>

  <ListadeFiguras ... > .... </ListadeFiguras>

  <ListadeFiguras ... > .... </ListadeFiguras>

</Figuras>
```

Figura V.39 RCML – Exemplo de utilização do elemento “Figuras”.

* ListadeFiguras

a. Definição

O elemento “ListadeFiguras” serve para agrupar vários elementos do tipo “Figura”. Este elemento armazena uma listagem de figuras. Pode também agrupar elementos do mesmo tipo, ou seja, “ListadeFiguras”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “ListadeFiguras” e “Figura”. A Figura V.40 mostra sua estrutura hierárquica.

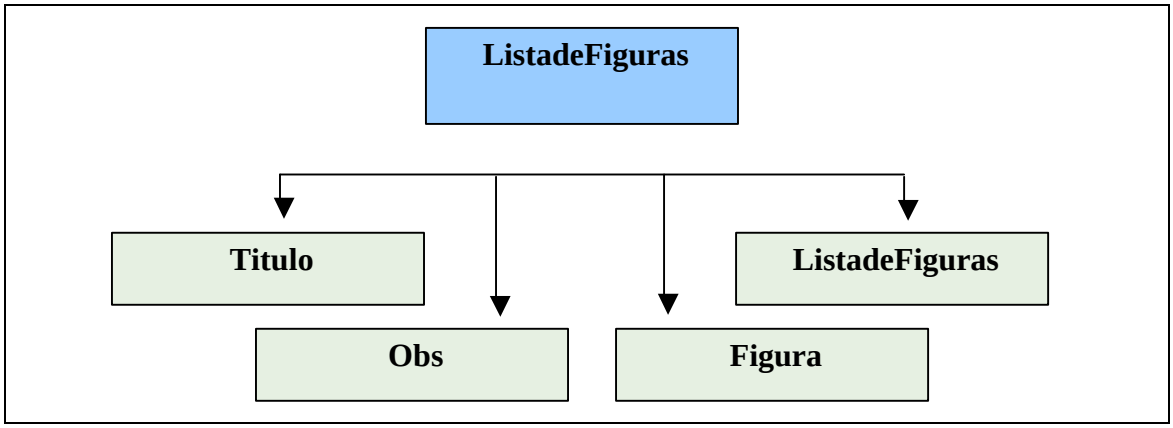


Figura V.40 RCML – Estrutura do elemento “ListadeFigura”.

c. Sintaxe

```
<!ELEMENT ListadeFiguras (Titulo, Observacao,(Figura |
ListadeFiguras)* >

<!ATTLIST ListadeFiguras %Nohname;>
```

Figura V.41 RCML – Sintaxe do elemento “ListadeFiguras”.

d. Exemplo

```
<ListadeFiguras NohName = “Figuras do capítulo 3”>

  <Titulo> .... </Titulo>  <Obs> ..... </Obs>

  <Figura ... >  ...  </Figura>

  <Figura ... >  ...  </Figura>

  <Figuras ... >  ...  </Figuras>

</ListadeFiguras>
```

Figura V.42 RCML – Exemplo de utilização do elemento “ListadeFiguras”.

* Figura

a. Definição

Um elemento do tipo “Figura” estrutura informações referentes à apresentação de uma figura, bem como o caminho e nome do arquivo que armazena a imagem.

b. Estrutura

Este objeto possui os fragmentos “Titulo”, “PalavrasChave”, “Descricao” e “Arquivo”. A Figura V.43 mostra sua estrutura hierárquica.

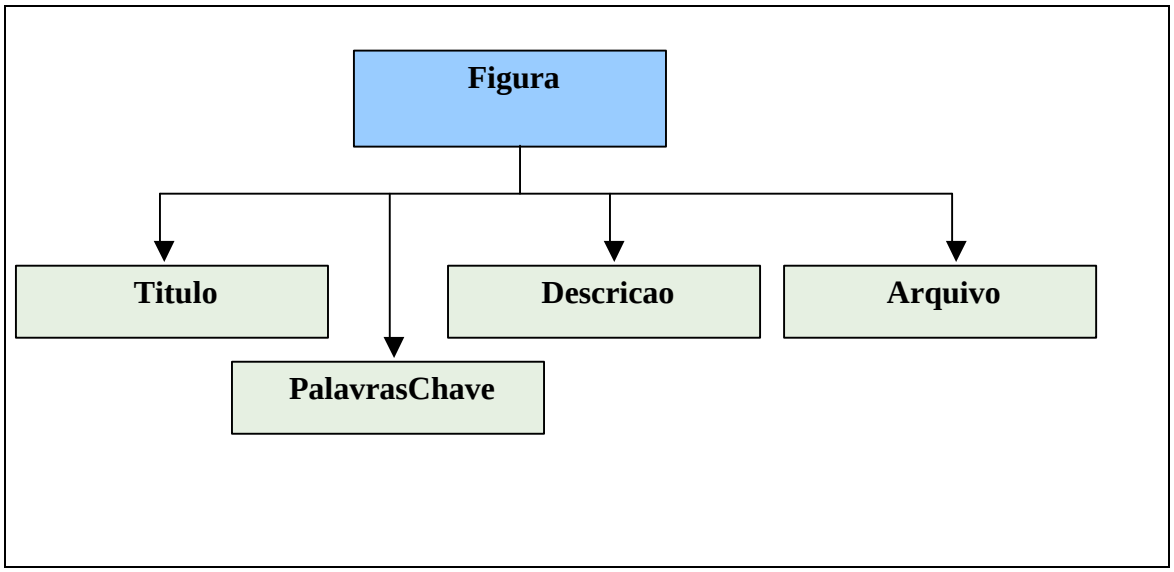


Figura V.43 RCML – Estrutura do elemento “Figura”.

c. Sintaxe

```
<!ELEMENT  Figura (Titulo, Descricao, PalavrasChave, Arquivo) >

<!ATTLIST Figura %Nohname;>
```

Figura V.44 RCML – Sintaxe do elemento “Figura”.

d. Exemplo

```
<Figura NohName = “Figura 1.1”>

  <Titulo> ... </Titulo>

  <Descricao> Esta figura é .... </Descricao>

  <PalavrasChave> .... </PalavrasChave>

  <Arquivo> c:\dados\figura1.jpeg </Arquivo>

  .....</Figura>
```

Figura V.45 RCML – Exemplo de utilização do elemento “Figura”.

* Formulas

a. Definição :

O elemento “Formulas” serve para agrupar vários elementos do tipo “ListadeFormulas”.

b. Estrutura

Este objeto contém o fragmento “ListadeFormulas”. A Figura V.46 mostra sua estrutura hierárquica.

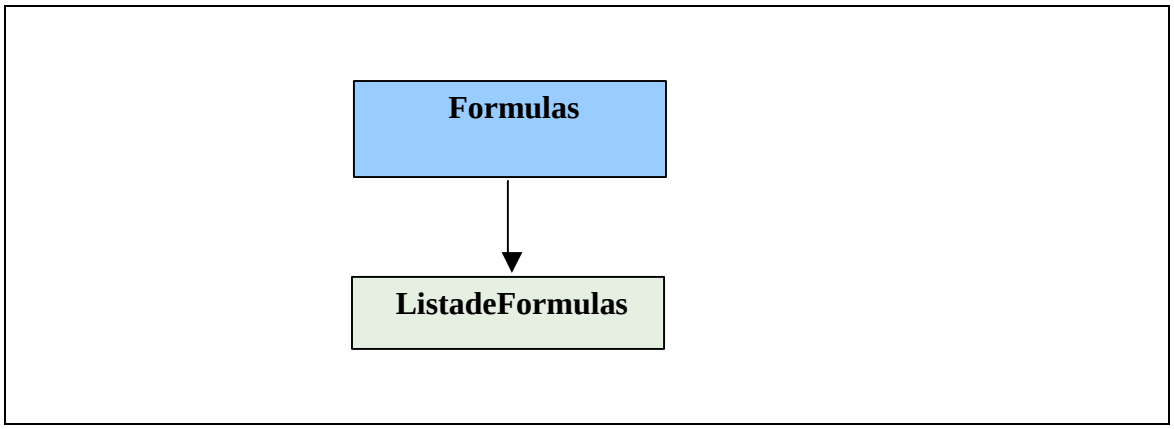


Figura V.46 RCML – Estrutura do elemento “Formulas”.

c. Sintaxe

```
<!ELEMENT Formulas (ListadeFormulas)*>)

<!ATTLIST  Formulas %Nohname;>
```

Figura V.47 RCML – Sintaxe do elemento “Formulas”.

d. Exemplo

```
<Formulas  NohName = “formulas”>

  <ListadeFormulas ... > .... </ListadeFormulas>

  <ListadeFormulas ... > .... </ListadeFormulas>

  <ListadeFormulas ... > .... </ListadeFormulas>

</Formulas>
```

Figura V.48 RCML – Exemplo de utilização do elemento “Formulas”.

* ListadeFormulas

a. Definição

O elemento “ListadeFormulas” serve para agrupar vários elementos do tipo “Formula”. Este elemento armazena uma listagem de fórmulas. Pode também, agrupar elementos do mesmo tipo, ou seja, “ListadeFormulas”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “ListadeFormulas” e “Formula”. A Figura V.49 mostra sua estrutura hierárquica.

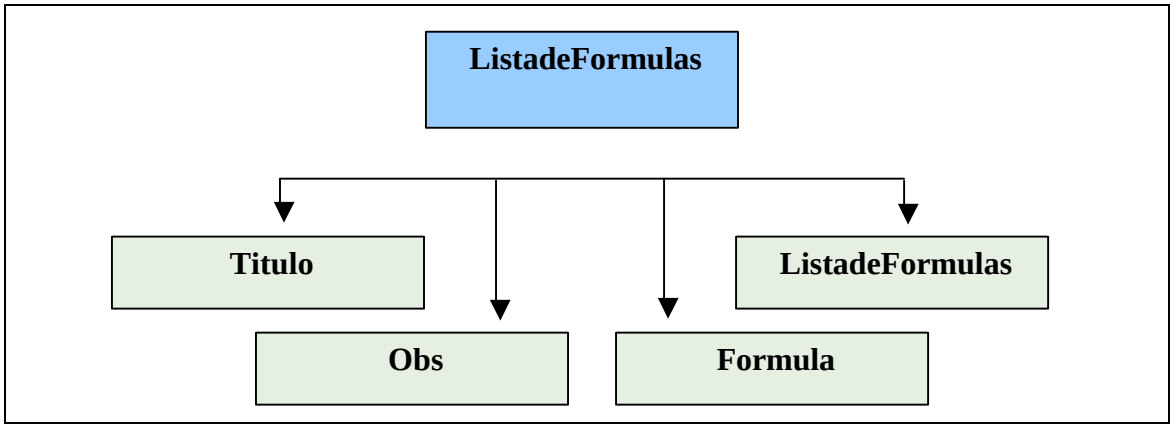


Figura V.49 RCML – Estrutura do elemento “ListadeFormulas”.

c. Sintaxe

```
<!ELEMENT ListadeFormulas (Titulo, Observacao,(Formula |
ListadeFormulas)* >

<!ATTLIST ListadeFormulas %Nohname;>
```

Figura V.50 RCML – Sintaxe do elemento “ListadeFormulas”.

d. Exemplo

```
<ListadeFormulas NohName = “Fórmulas do Capítulo 1”>

  <Titulo> ... </Titulo>

  <Observacao> ... </Observacao>

  <Formula... >  ...  </Formula>

  <Formula... >  ...  </Formula>

  <ListadeFormulas ... >  ...  </ListadeFormulas>

</ListadeFormulas>
```

Figura V.51 RCML – Exemplo de utilização do elemento “ListadeFormulas”.

* Formula

a. Definição

Um elemento do tipo “Formula” estrutura informações referentes à apresentação de uma formula, bem como os códigos LaTeX e MathML correspondentes.

b. Estrutura

Este objeto possui os fragmentos “Titulo”, “PalavrasChave”, “Descricao”, “LaTeX” e “MathML”. A Figura V.52 mostra sua estrutura hierárquica.

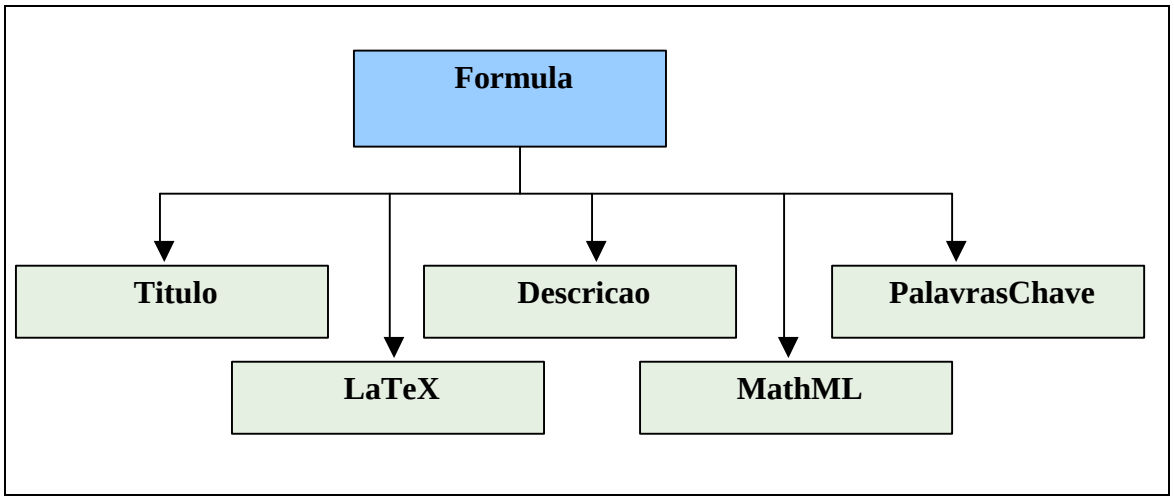


Figura V.52 RCML – Estrutura do elemento “Formula”.

c. Sintaxe

```
<!ELEMENT Formula (Titulo, PalavrasChave, Descricao, LaTeX, MathML)>

<!ATTLIST Formula %Nohname;>
```

Figura V.53 RCML – Sintaxe do elemento “Formula”.

d. Exemplo

```
<Formula NohName = “Formula 1.1”>

  <Titulo> .... </Titulo>

  <PalavrasChave> ... </PalavrasChave>

  <Descricao> ... </Descricao>

  <LaTeX> ... </LaTeX>

  <MathML> ... </MathML>

</Formula>
```

Figura V.54 RCML – Exemplo de utilização do elemento “Formula”.

* Graficos

a. Definição

O elemento “Graficos” serve para agrupar vários elementos do tipo “ListadeGraficos”.

b. Estrutura

Este objeto contém o fragmento “ListadeGraficos”. A Figura V.55 mostra sua estrutura hierárquica.

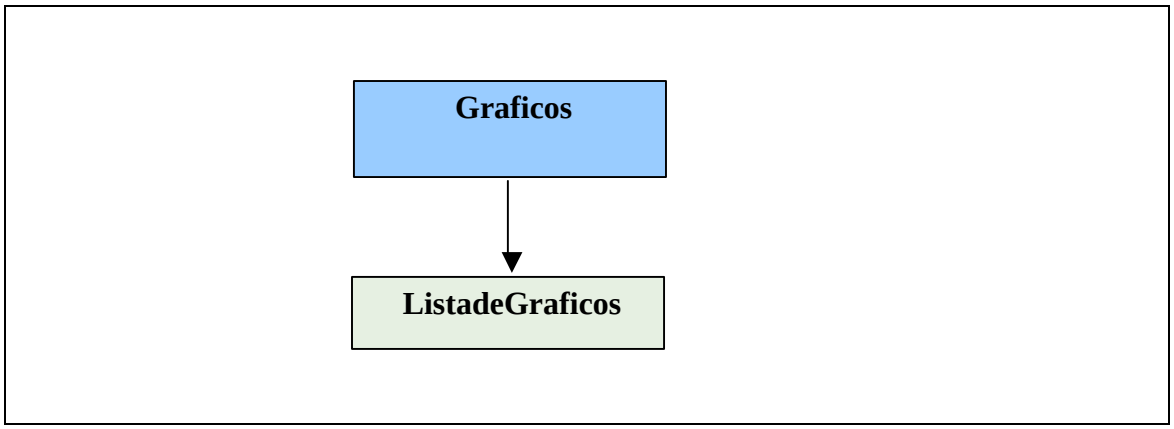


Figura V.55 RCML – Estrutura do elemento “Graficos”.

c. Sintaxe

```
<!ELEMENT Graficos (ListadeGraficos)*>)

<!ATTLIST  Graficos %Nohname;>
```

Figura V.56 RCML – Sintaxe do elemento “Graficos”.

d. Exemplo

```
<Graficos  NohName = “Gráficos”>

  <ListadeGraficos ... > .... </ListadeGraficos>

  <ListadeGraficos ... > .... </ListadeGraficos>

  <ListadeGraficos ... > .... </ListadeGraficos>

</Graficos>
```

Figura V.57 RCML – Exemplo de utilização do elemento “Graficos”.

* ListadeGraficos

a. Definição

O elemento “ListadeGraficos” serve para agrupar vários elementos do tipo “Grafico”. Este elemento armazena uma listagem de gráficos. Pode também agrupar elementos do mesmo tipo, ou seja, “ListadeGraficos”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “ListadeFormulas” e “Formula”. A Figura V.58 mostra sua estrutura hierárquica.

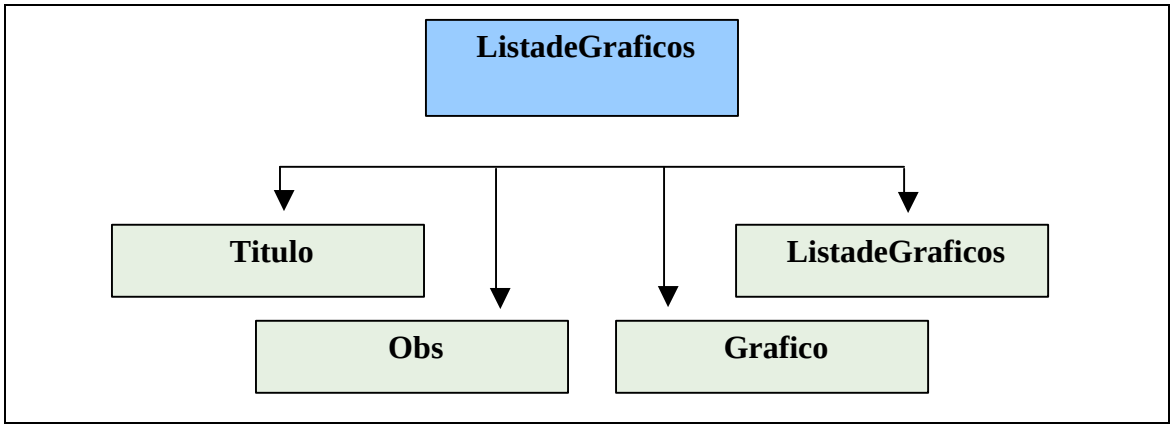


Figura V.58 RCML – Estrutura do elemento “ListadeGraficos”.

c. Sintaxe

```
<!ELEMENT ListadeGraficos (Titulo, Observacao,(Grafico |
ListadeGraficos)* >

<!ATTLIST ListadeGraficos %Nohname;>
```

Figura V.59 RCML – Sintaxe do elemento “ListadeGraficos”.

d. Exemplo

```
<ListadeGraficos NohName = “Graficos do Capítulo 1”>

  <Titulo> ... </Titulo>

  <Observacao> ... </Observacao>

  <Grafico ...>  ...  </Grafico>

  <ListadeGraficos ...> ...  </ListadeGraficos>

</ListadeGraficos>
```

Figura V.60 RCML – Exemplo de utilização do elemento “ListadeGraficos”.

* Grafico

a. Definição

Um elemento do tipo “Grafico” estrutura informações referentes a um gráfico em formato SVG.

b. Estrutura

Este objeto possui os fragmentos “Titulo”, “PalavrasChave”, “Descricao”, e “SVG”. A Figura V.61 mostra sua estrutura hierárquica.

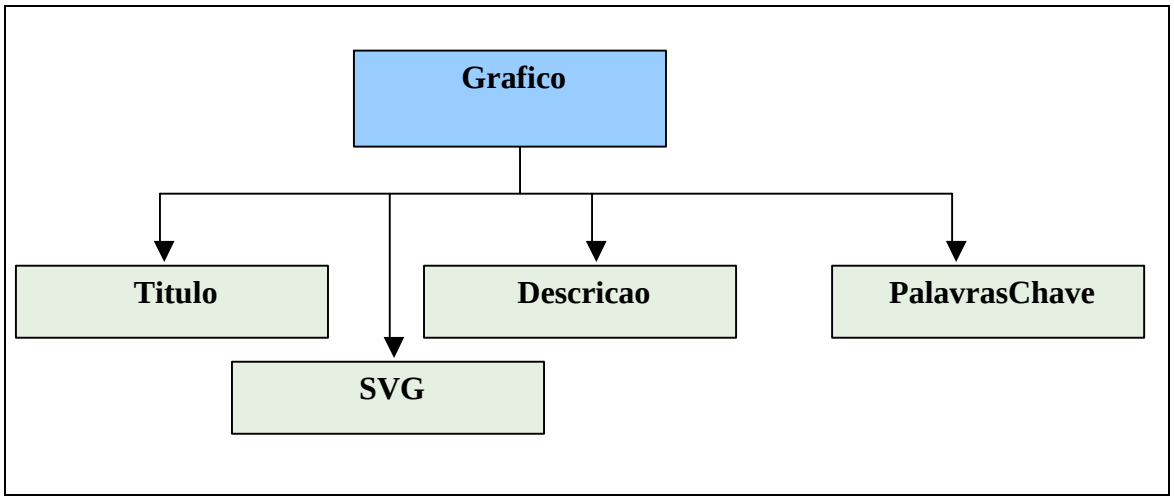


Figura V.61 RCML – Estrutura do elemento “Grafico”.

c. Sintaxe

```
<!ELEMENT Grafico (Titulo, PalavrasChave, Descricao,SVG )>

<!ATTLIST Grafico %Nohname;>
```

Figura V.62 RCML – Sintaxe do elemento “Grafico”.

d. Exemplo

```
<Grafico NohName = “Grafico 1.1”>

  <Titulo> ... </Titulo>

  <PalavrasChave> ... </PalavrasChave>

  <Descricao> ... </Descricao>

  <SVG> ... </SVG>

</Grafico>
```

Figura V.63 RCML – Exemplo de utilização do elemento “Grafico”.

* Exemplos

a. Definição

O elemento “Exemplos” serve para agrupar vários elementos do tipo “ListadeExemplos”.

b. Estrutura

Este objeto contém o fragmento “ListadeExemplos”. A Figura V.64 mostra sua estrutura hierárquica.

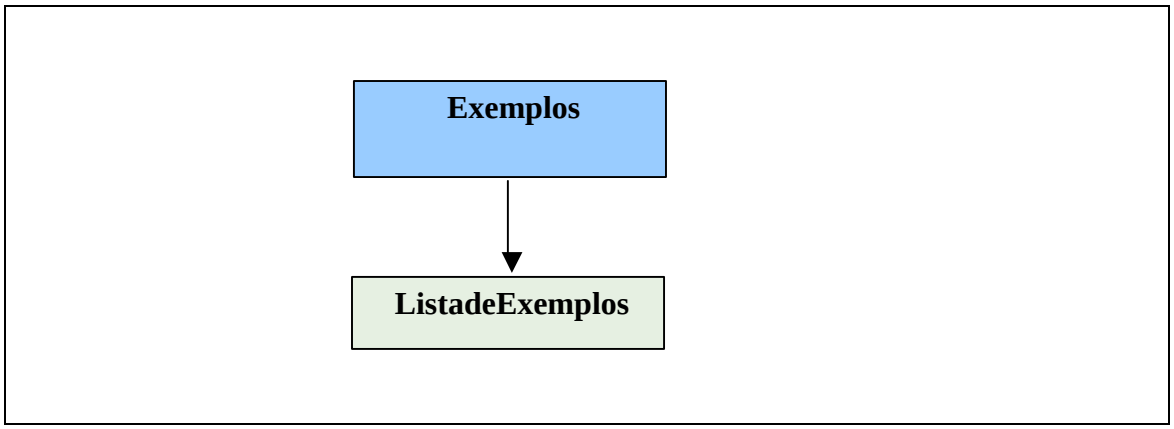


Figura V.64 RCML – Estrutura do elemento “Exemplos”.

c. Sintaxe

```
<!ELEMENT Exemplos (ListadeExemplos)*>)

<!ATTLIST  Exemplos %Nohname;>
```

Figura V.65 RCML – Sintaxe do elemento “Exemplos”.

d. Exemplo

```
<Exemplos  NohName = “Exemplos”>

  <ListadeExemplos ... > .... </ListadeExemplos>

  <ListadeExemplos ... > .... </ListadeExemplos>

  <ListadeExemplos ... > .... </ListadeExemplos>

</Exemplos>
```

Figura V.66 RCML – Exemplo de utilização do elemento “Exemplos”.

* ListadeExemplos

a. Definição

O elemento “ListadeExemplos” serve para agrupar vários elementos do tipo “Exemplo”. Este elemento armazena uma listagem de exemplos. Pode também agrupar elementos do mesmo tipo, ou seja, “ListadeExemplos”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “ListadeExemplos” e “Exemplo”. A Figura V.67 mostra sua estrutura hierárquica.

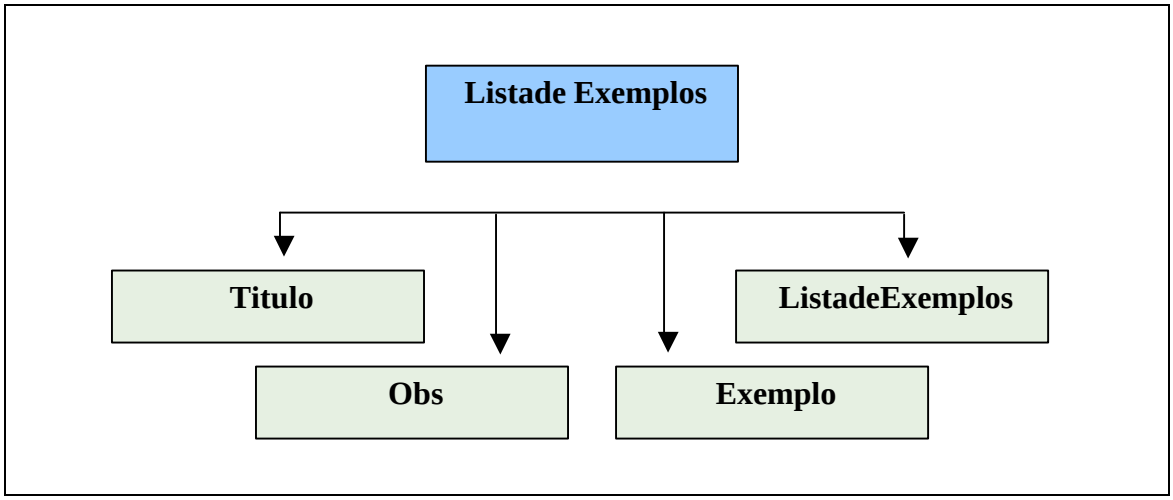


Figura V.67 RCML – Estrutura do elemento “ListadeExemplos”.

c. Sintaxe

```
<!ELEMENT ListadeExemplos (Titulo, Observacao,(Exemplo |
ListadeExemplos)* >

<!ATTLIST ListadeExemplos %Nohname;>
```

Figura V.68 RCML – Sintaxe do elemento “ListadeExemplos”.

d. Exemplo

```
<ListadeExemplos NohName = “Exemplos do capítulo 1”>

  <Titulo> ... </Titulo>  <Obs> ... </Obs>

  <Exemplo ... > ...   </Exemplo>

  <Exemplo ... > ...   </Exemplo>

  <ListadeExemplos... > ...  </ListadeExemplos></ListadeExemplos>
```

Figura V.69 RCML – Exemplo de utilização do elemento “ListadeExemplos”.

* Exemplo

a. Definição

Um elemento do tipo “Exemplo” estrutura informações referentes a exemplos de um determinado assunto.

b. Estrutura

Este objeto possui os fragmentos “Titulo”, “PalavrasChave” e “Texto”. A Figura V.70 mostra sua estrutura hierárquica.

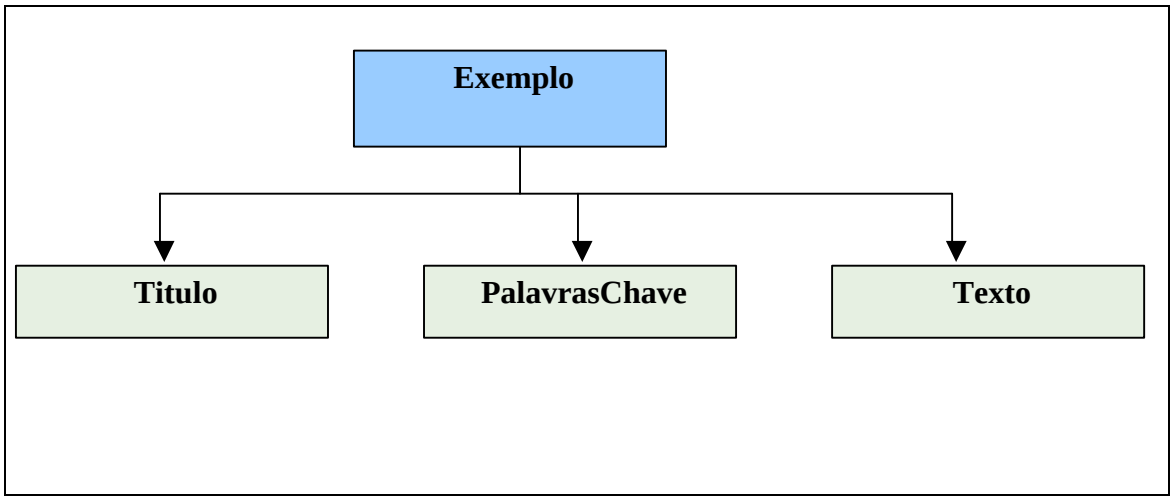


Figura V.70 RCML – Estrutura do elemento “Exemplo”.

c. Sintaxe

```
<!ELEMENT Exemplo (Titulo, PalavrasChave, Texto) >

<!ATTLIST Exemplo %Nohname;>
```

Figura V.71 RCML – Sintaxe do elemento “Exemplo”.

d. Exemplo

```
<Exemplo NohName = “Exemplo “1.1”>

<Titulo> ... </Titulo>

<Observacao> ... </Observacao>

<Texto> ... </Texto>

</Exemplo>
```

Figura V.72 RCML – Exemplo de utilização do elemento “Exemplo”.

* Exercicios

a. Definição :

O elemento “Exercicios” serve para agrupar vários elementos do tipo “ListadeExercicios”.

b. Estrutura

Este objeto contém o fragmento “ListadeExercicios”. A Figura V.73 mostra sua estrutura hierárquica.

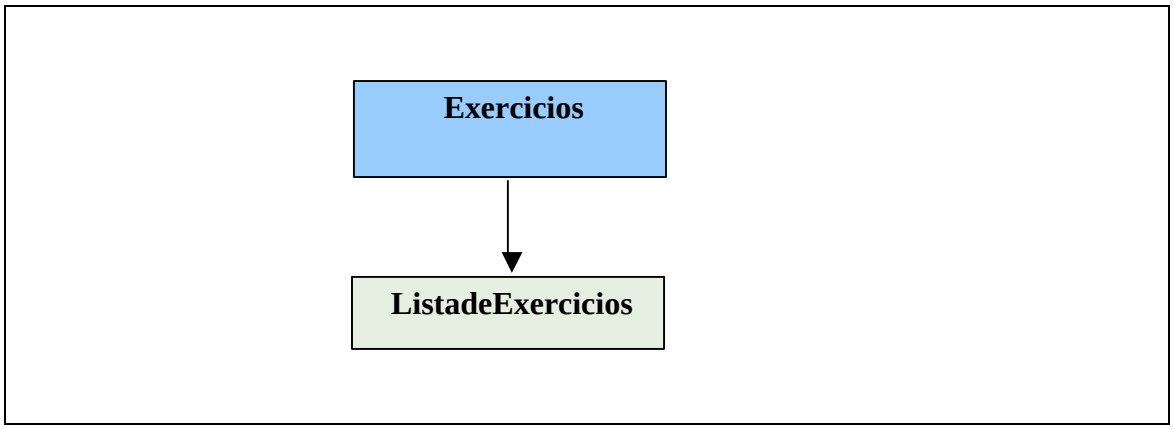


Figura V.73 RCML – Estrutura do elemento “Exercicios”.

c. Sintaxe :

```
<!ELEMENT Exercicios (ListadeExercicios)*>)

<!ATTLIST  Exercicios %Nohname;>
```

Figura V.74 RCML – Sintaxe do elemento “Exercicios”.

d. Exemplo :

```
<Exercicios NohName = “Exercicios”>

  <ListadeExercicios ... > .... </ListadeExercicios>

  <ListadeExercicios... > ....  </ListadeExercicios>

  <ListadeExercicios ... > ....  </ListadeExercicios>

</Exercicios>
```

Figura V.75 RCML – Exemplo de utilização do elemento “Exercicios”.

* ListadeExercicios

a. Definição :

O elemento “ListadeExercicios” serve para agrupar vários elementos do tipo “Exercicio”. Este elemento armazena uma listagem de exercicios. Pode também agrupar elementos do mesmo tipo, ou seja, “Exercicios”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “ListadeExercicios” e “Exercicio”. A Figura V.76 mostra sua estrutura hierárquica.

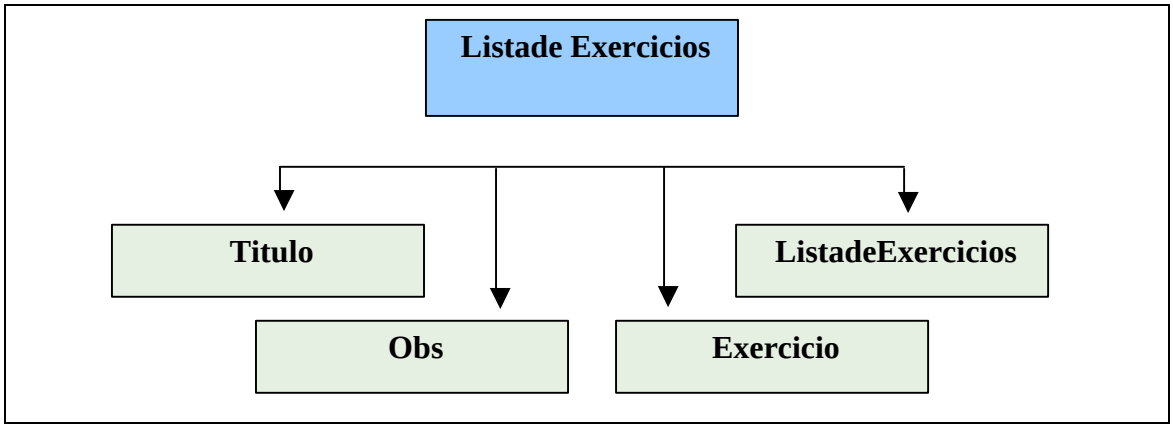


Figura V.76 RCML – Estrutura do elemento “ListadeExercicios”.

c. Sintaxe :

```
<!ELEMENT ListadeExercicios (Titulo, Observacao,(Exercicio |
ListadeExercicios)* >

<!ATTLIST ListadeExercicios %Nohname;>
```

Figura V.77 RCML – Sintaxe do elemento “ListadeExercicios”.

d. Exemplo :

```
<ListadeExercicios NohName =  “Exercícios do Capítulo 1”>

  <Titulo> ... </Titulo>  <Obs> ... </Obs>

  <Exercicio ... >  ...  </Exercicio>

  <Exercicio ... >  ...  </Exercicio>

  <ListadeExercicios ... >  ...  </ListadeExercicios>

</ListadeExercicios>
```

Figura V.78 RCML – Exemplo de utilização do elemen7to “ListadeExercicios”.

* Exercicio

a. Definição

Um elemento do tipo “Exercicio” estrutura informações referentes à um exercício de um determinado assunto. Outros dados, como resolução e resposta também são preservados.

b. Estrutura

Este objeto possui os fragmentos “Titulo”, “PalavrasChave”, “Descricao”, “Resolucao” e “Resposta”. A Figura V.79 mostra sua estrutura hierárquica.

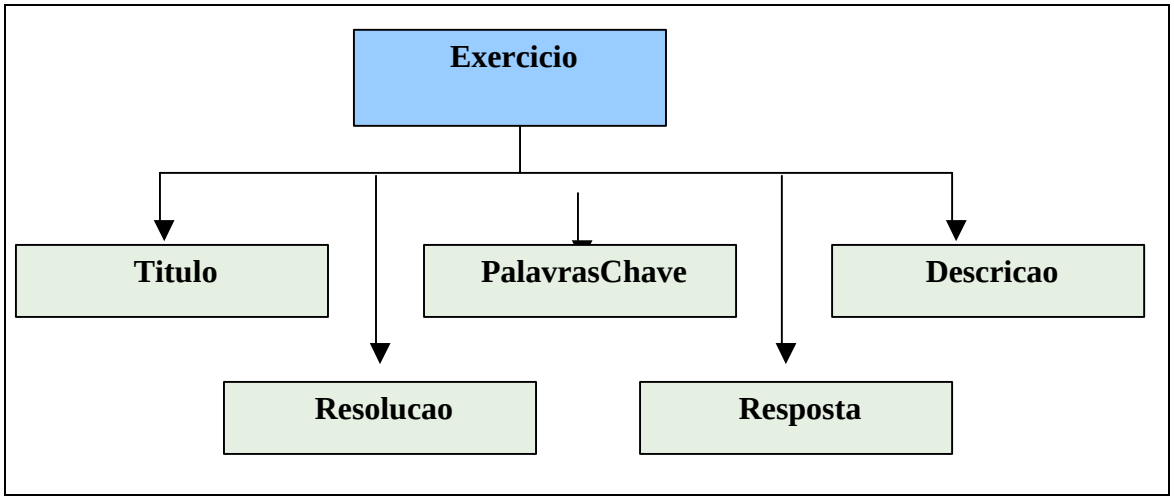


Figura V.79 RCML – Estrutura do elemento “Exercicio”.

c. Sintaxe

```
<!ELEMENT Exercicio (Titulo, PalavrasChave, Descricao, Resolucao,
Resposta)>

<!ATTLIST Exercicio %Nohname;>
```

Figura V.80 RCML – Sintaxe do elemento “Exercicio”.

d. Exemplo :

```
<Exercicio NohName = “Exercicio 1.1”>

  <Titulo> ... </Titulo>  <PalavrasChave> ... </PalavrasChave>

  <Descricao> ... </Descricao>

  <Resolucao> ... </Resolucao>  <Resposta> ... </Resposta></Exercicio>
```

Figura V.81 RCML – Exemplo de utilização do elemento “Exercicio”.

* Teoremas

a. Definição :

O elemento “Teoremas” serve para agrupar vários elementos do tipo “ListadeTeoremas”.

b. Estrutura

Este objeto contém o fragmento “ListadeTeoremas”. A Figura V.82 mostra sua estrutura hierárquica.

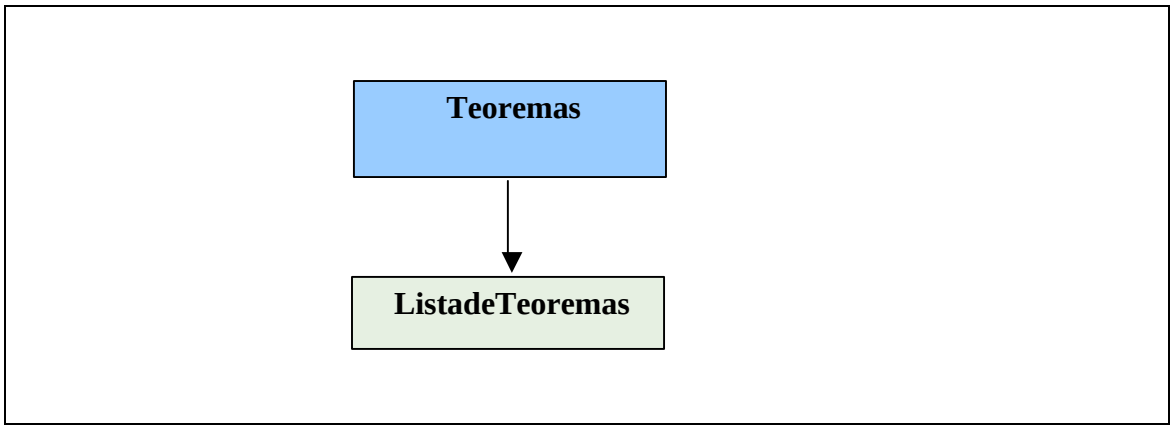


Figura V.82 RCML – Estrutura do elemento “Teoremas”.

c. Sintaxe :

```
<!ELEMENT Teoremas (ListadeTeoremas)*>)

<!ATTLIST  Teoremas %Nohname;>
```

Figura V.83 RCML – Sintaxe do elemento “Teoremas”.

d. Exemplo :

```
<Teoremas NohName = “Teoremas”>

  <ListadeTeoremas ... > .... </ListadeTeoremas>

  <ListadeTeoremas... > ....  </ListadeTeoremas>

  <ListadeTeoremas ... > ....  </ListadeTeoremas>

</Teoremas>
```

Figura V.84 RCML - Exemplo de utilização do elemento “Teoremas”.

* ListadeTeoremas

a. Definição :

O elemento “ListadeTeoremas” serve para agrupar vários elementos do tipo “Teorema”. Este elemento armazena uma listagem de teoremas. Pode também agrupar elementos do mesmo tipo, ou seja, “ListadeTeoremas”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “ListadeTeoremas” e “Teorema”. A Figura V.85 mostra sua estrutura hierárquica.

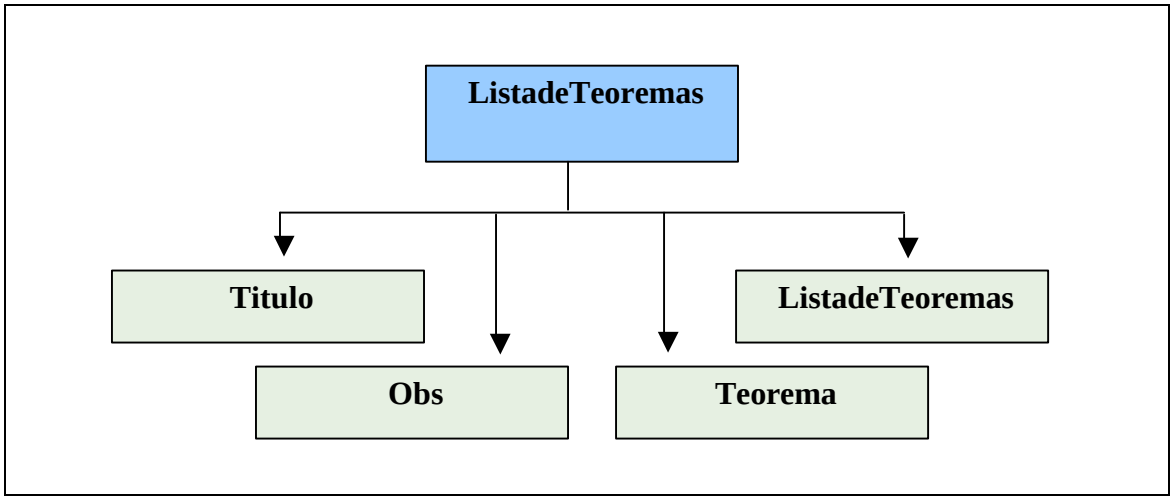


Figura V.85 RCML – Estrutura do elemento “ListadeTeoremas”.

c. Sintaxe :

```
<!ELEMENT ListadeTeoremas (Titulo, Observacao, (Teorema |
ListadeTeoremas)*)>)

<!ATTLIST  ListadeTeoremas %Nohname;>
```

Figura V.86 RCML – Sintaxe do elemento “ListadeTeoremas”.

d. Exemplo :

```
<ListadeTeoremas NohName = “Teoremas Gerais”>

  <Titulo> .... </Titulo>  <Obs> ... </Obs>

  <Teorema ... >   ...  </Teorema>

  <Teorema ... >  ...  </Teorema>

  <ListadeTeoremas... >  ...  </ListadeTeoremas></ListadeTeoremas>
```

Figura V.87 RCML– Exemplo de utilização do elemento “ListadeTeoremas”.

* Teorema

a. Definição :

Um elemento do tipo “Teorema” estrutura informações referentes a um teorema de um determinado assunto. Outros dados, como resolução, também são preservados.

b. Estrutura

Este objeto possui os fragmentos “Titulo”, “PalavrasChave”, “Descricao”, “Resolucao” e “Resposta”. A Figura V.88 mostra sua estrutura hierárquica.

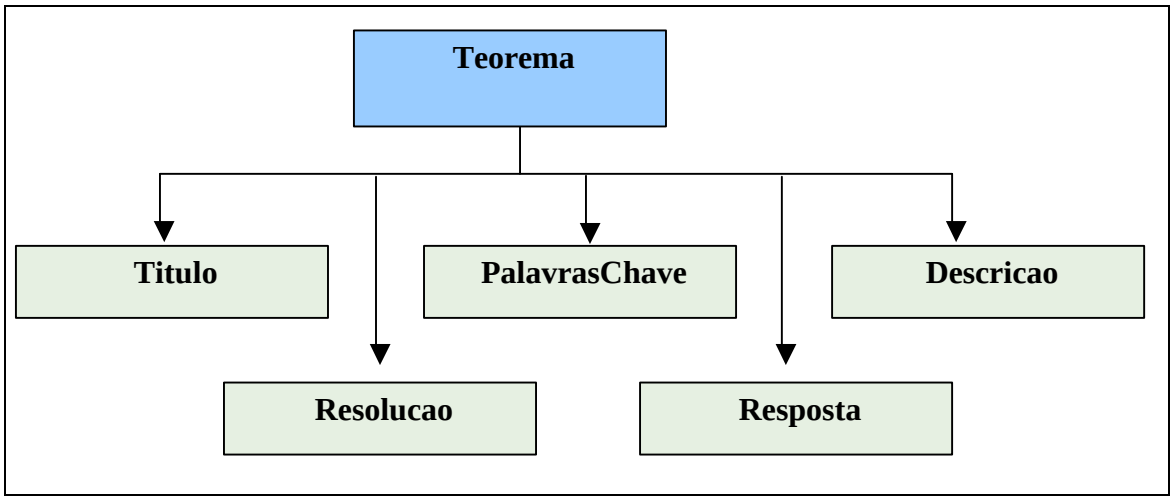


Figura V.88 RCML – Estrutura do elemento “Teorema”.

c. Sintaxe :

```
<!ELEMENT Teorema (Titulo, PalavrasChave, Descricao, Resolucao,
Resposta)>

<!ATTLIST Teorema %Nohname;>
```

Figura V.89 RCML – Sintaxe do elemento “Teorema”.

d. Exemplo :

```
<Teorema NohName = “Teorema 1.1”>

  <Titulo> ... </Titulo>  <PalavrasChave> ... </PalavrasChave>

  <Descricao> ... </Descricao>

  <Resolucao> ... </Resolucao>

  <Resposta> ... </Resposta>

</Teorema>
```

Figura V.90 RCML – Exemplo de utilização do elemento “Teorema”.

* Definicoes

a. Definição :

O elemento “Definicoes” serve para agrupar vários elementos do tipo “ListadeDefinicoes”.

b. Estrutura

Este objeto contém o fragmento “ListadeDefinicoes”. A Figura V.91 mostra sua estrutura hierárquica.

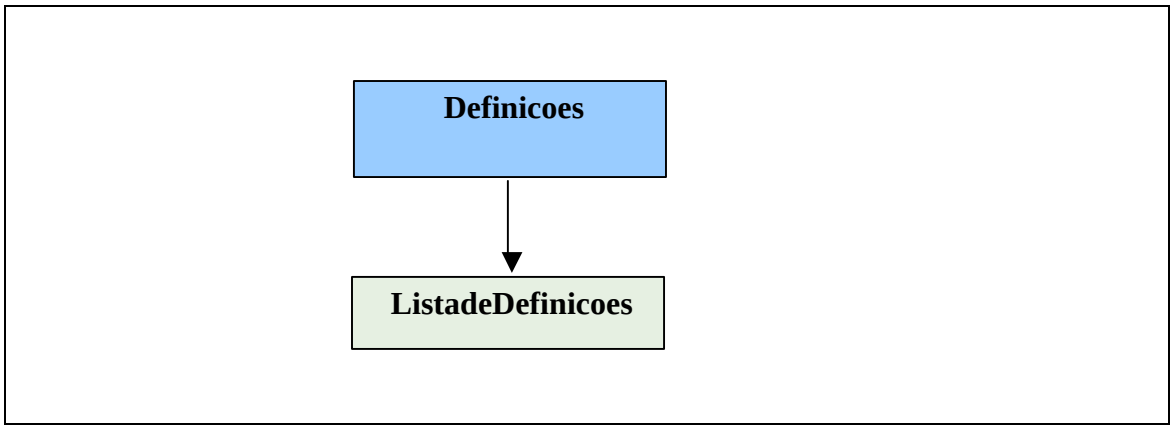


Figura V.91 RCML – Estrutura do elemento “Definicoes”.

c. Sintaxe :

```
<!ELEMENT Definicoes (ListadeDefinicoes)*>)

<!ATTLIST  Definicoes %Nohname;>
```

Figura V.92 RCML – Sintaxe do elemento “Definições”.

d. Exemplo :

```
<Definicoes NohName = “Definicoes”>

  <ListadeDefinicoes ...> .... </ListadeDefinicoes>

  <ListadeDefinicoes... > ....  </ListadeDefinicoes>

  <ListadeDefinicoes ... > ....  </ListadeDefinicoes>

</Definicoes>
```

Figura V.93 RCML – Exemplo de utilização do elemento “Definicoes”.

* ListadeDefinicoes

a. Definição :

O elemento “ListadeDefinicoes” serve para agrupar vários elementos do tipo “Definicao”. Este elemento armazena uma listagem de definições. Pode também agrupar elementos do mesmo tipo, ou seja, “ListadeDefinicoes”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “ListadeDefinicoes” e “Definicao”. A Figura V.94 mostra sua estrutura hierárquica.

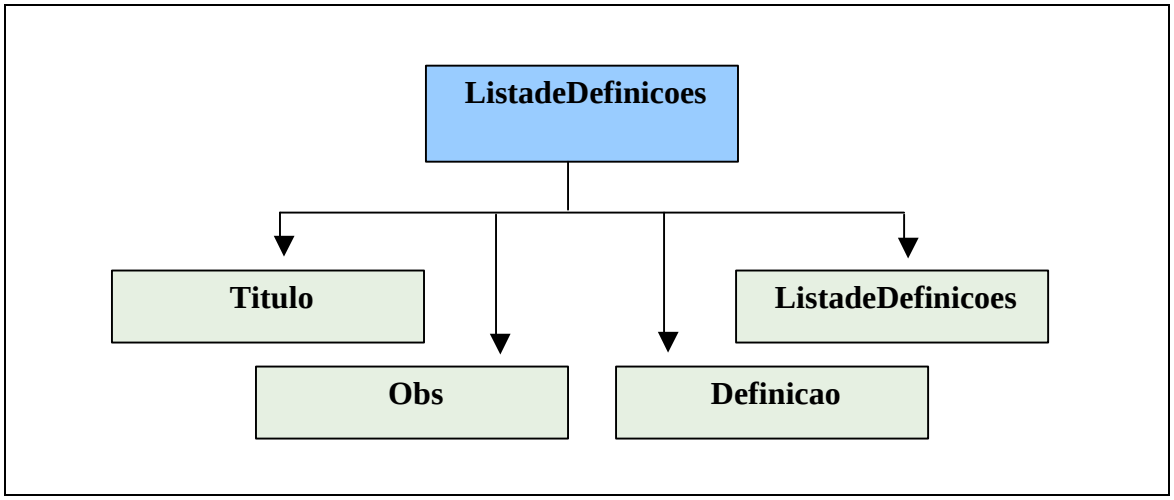


Figura V.94 RCML – Estrutura do elemento “ListadeDefinicoes”.

c. Sintaxe :

```
<!ELEMENT ListadeDefinicoes (Titulo, Observacao, (Definicao |
ListadeDefinicoes)*)>)

<!ATTLIST  ListadeDefinicoes %Nohname;>
```

Figura V.95 RCML – Sintaxe do elemento “ListadeDefinicoes”.

d. Exemplo :

```
<ListadeDefinicoes NohName = “Definicoes Gerais”>

  <Titulo> .... </Titulo>  <Obs> ... </Obs>

  < Definicao... >   ....  </Definicao >  <Definicao... >  ...
</Definicao>

  <ListadeDefinicoes... >  ...  ListadeDefinicoes></ListadeDefinicoes>
```

Figura V.96 RCML – Exemplo de utilização do elemento “ListadeDefinicoes”.

* Definicao

a. Definição :

Um elemento do tipo “Definicao” estrutura informações referentes à uma definição de um determinado assunto.

b. Estrutura

Este objeto possui os fragmentos “Titulo”, “PalavrasChave” e “Texto”. A Figura V.97 mostra sua estrutura hierárquica.

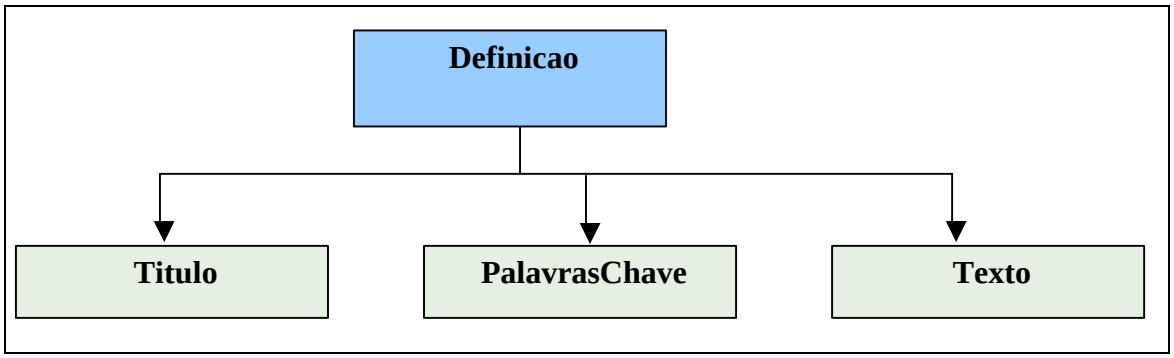


Figura V.97 RCML – Estrutura do elemento “Definicao”.

c. Sintaxe :

```
<!ELEMENT Definicao (Titulo, PalavrasChave, Texto)>

<!ATTLIST Definicao %Nohname;>
```

Figura V.98 RCML – Sintaxe do elemento “Definicao”.

d. Exemplo :

```
<Definicao NohName = “Definição 1.1”>

  <Titulo> ... </Titulo>

  <PalavrasChave> ... </PalavrasChave>

  <Texto> ... </Texto>

</Definicao>
```

Figura V.99 RCML– Exemplo de utilização do elemento “Definicao”.

* Glossarios

a. Definição :

O elemento “Glossarios” serve para agrupar vários elementos do tipo “ListadeGlossarios”.

b. Estrutura

Este objeto contém o fragmento “ListadeGlossarios”. A Figura V.100 mostra sua estrutura hierárquica.

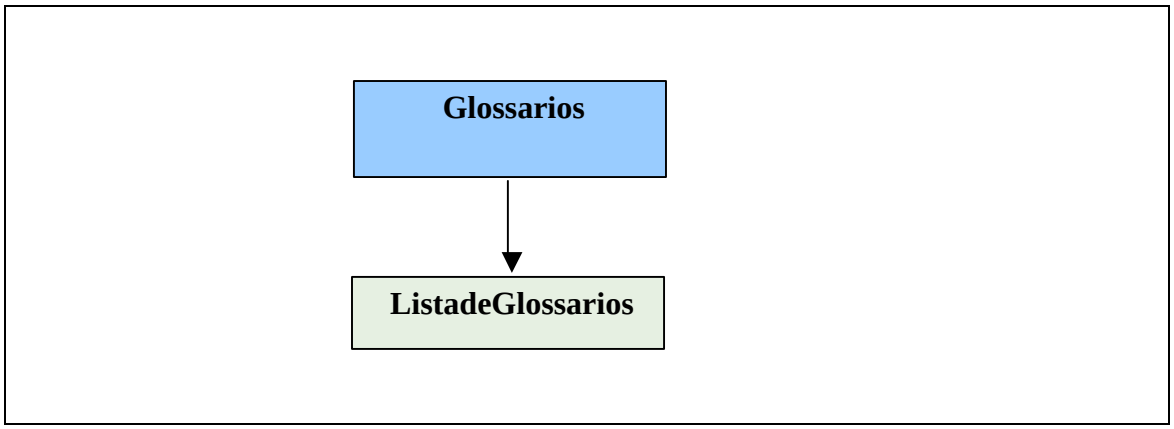


Figura V.100 RCML – Estrutura do elemento “Glossarios”.

c. Sintaxe :

```
<!ELEMENT Glossarios (ListadeGlossarios)*>)

<!ATTLIST  Glossarios %Nohname;>
```

Figura V.101 RCML – Sintaxe do elemento “Glossarios”.

d. Exemplo :

```
<Glossarios NohName = “Glossarios”>

  <ListadeGlossarios ...> .... </ListadeGlossarios>

  <ListadeGlossarios... > .... </ListadeGlossarios>

  <ListadeGlossarios ...> .... </ListadeGlossarios>

</Glossarios >
```

Figura V.102 RCML – Exemplo de utilização do elemento “Glossarios”.

* ListadeGlossarios

a. Definição :

O elemento “ListadeGlossarios” serve para agrupar vários elementos do tipo “Glossario”. Este elemento armazena uma listagem de Glossarios. Pode também, agrupar elementos do mesmo tipo, ou seja, “ListadeGlossarios”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “ListadeGlossarios” e “Glossario”. A Figura V.103 mostra sua estrutura hierárquica.

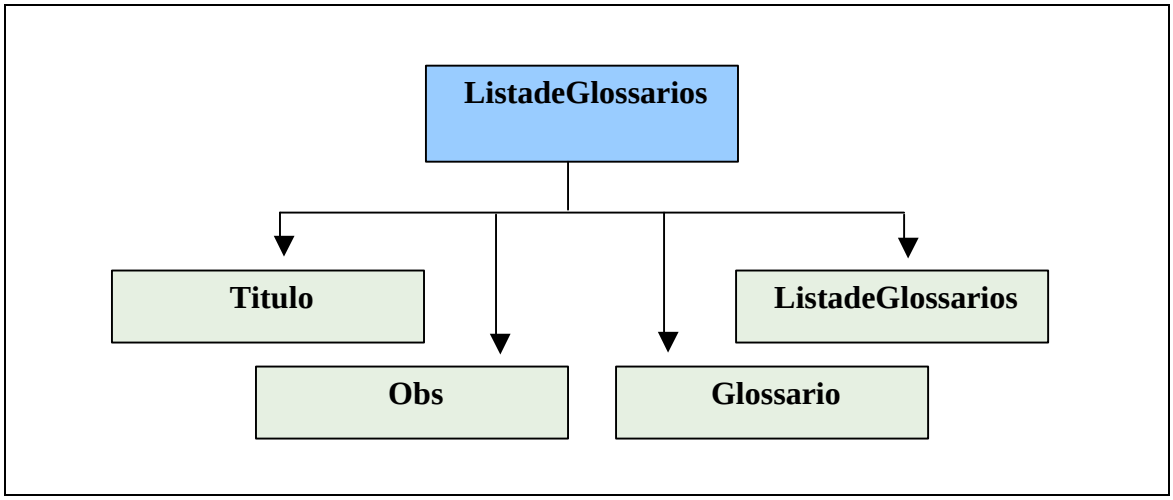


Figura V.103 RCML – Estrutura do elemento “ListadeGlossarios”.

c. Sintaxe :

```
<!ELEMENT ListadeGlossarios (Titulo, Observacao, (Glossario|
ListadeGlossarios)*)>)

<!ATTLIST  ListadeGlossarios %Nohname;>
```

Figura V.104 RCML – Sintaxe do elemento “ListadeGlossarios”.

d. Exemplo :

```
<ListadeGlossarios NohName = “Glossarios Gerais”>

  <Titulo> .... </Titulo>  <Obs> ... </Obs>

  < Glossario... >   ....  </ Glossario >

  < Glossario... >  ...  </ Glossario >

  <ListadeGlossarios... >...</ListadeGlossarios></ListadeGlossarios>
```

Figura V.105 RCML – Exemplo de utilização do elemento “ListadeGlossarios”.

* Glossario

a. Definição :

Um elemento do tipo “Glossario” estrutura informações referentes a um glossário de um determinado assunto..

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs” e “Letra”. A Figura V.106 mostra sua estrutura hierárquica.

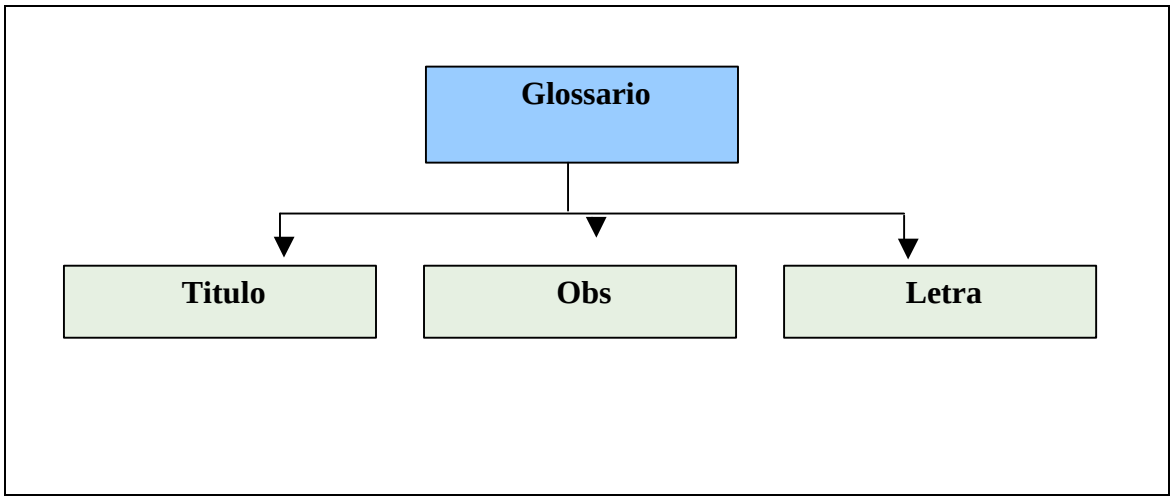


Figura V.106 RCML – Estrutura do elemento “Glossario”.

c. Sintaxe :

```
<!ELEMENT Glossario (Titulo, Observacao, (Letra)*)>

<!ATTLIST Glossario %Nohname;>
```

Figura V.107 RCML– Sintaxe do elemento “Glossario”.

d. Exemplo :

```
<Glossario NohName = “Glossário de Informática”>

  <Titulo> .. </Titulo>

  <Observacao> ... </Observacao>

  <Letra ... >  ....   </Letra>

  <Letra ... >  ....   </Letra>

</Glossario>
```

Figura V.108 RCML – Exemplo de utilização do elemento “Glossario”.

* Letra

a. Definição :

Um elemento do tipo “Letra” estrutura informações referentes a uma letra de glossário.

b. Estrutura

Este objeto contém o fragmento “ItemdeGlossario”. A Figura V.109 mostra sua estrutura hierárquica.

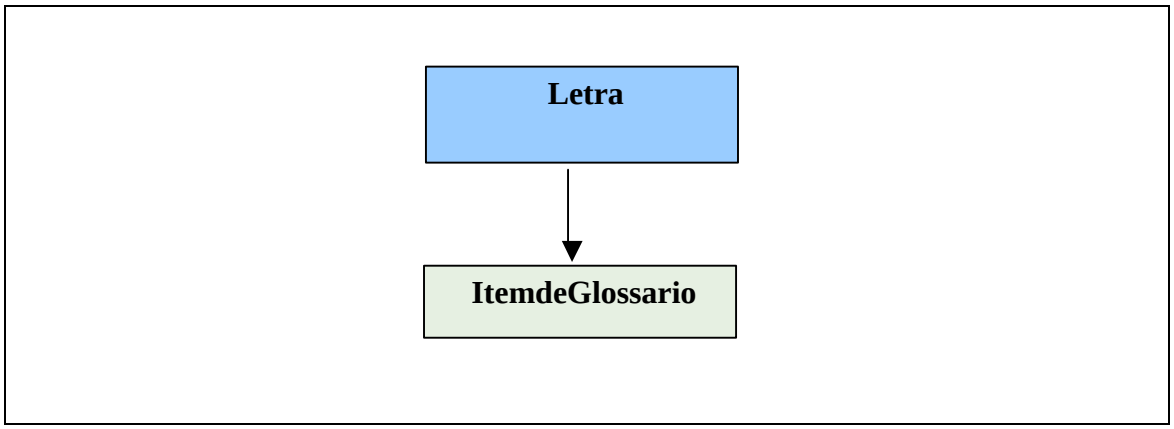


Figura V.109 RCML – Estrutura do elemento “Letra”.

c. Sintaxe :

```
<!ELEMENT Letra (ItemdeGlossario)*>

<!ATTLIST Letra %Nohname;>
```

Figura V.110 RCML– Sintaxe do elemento “Letra”.

d. Exemplo :

```
<Letra NohName = “A”>

<ItemdeGlossario> ... </ItemdeGlossario>

<ItemdeGlossario> ... </ItemdeGlossario>

<ItemdeGlossario> ... </ItemdeGlossario>

</Letra>
```

Figura V.111 RCML – Exemplo de utilização do elemento “Letra”.

* Artigos

a. Definição :

O elemento “Artigos” serve para agrupar vários elementos do tipo “ListadeArtigos”.

b. Estrutura

Este objeto contém o fragmento “ListadeArtigos”. A Figura V.112 mostra sua estrutura hierárquica.

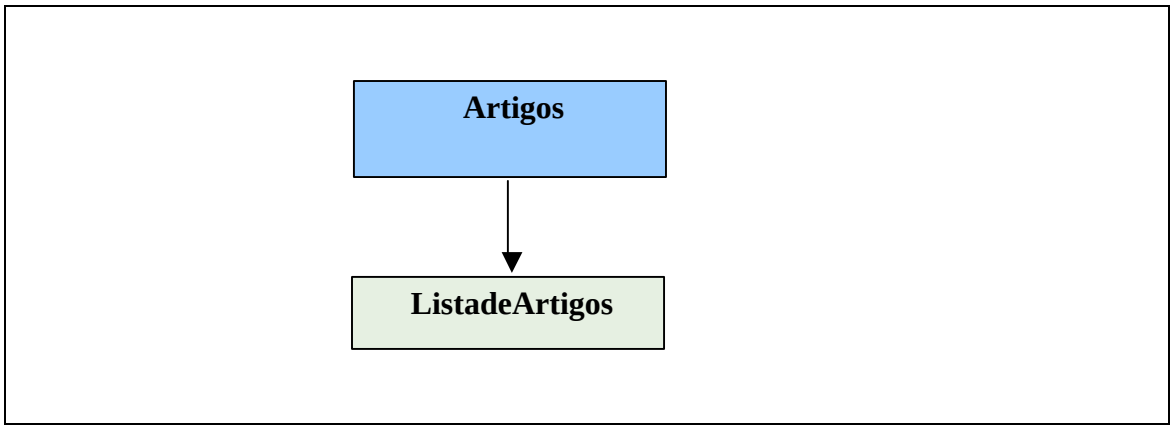


Figura V.112 RCML – Estrutura do elemento “Artigos”.

c. Sintaxe :

```
<!ELEMENT Artigos (ListadeArtigos)*>)

<!ATTLIST  Artigos %Nohname;>
```

Figura V.113 RCML – Sintaxe do elemento “Artigos”.

d. Exemplo :

```
<Artigos NohName = “Artigos”>

  <ListadeArtigos ...> .... </ListadeArtigos >

  <ListadeArtigos... > .... </ListadeArtigos >

  <ListadeArtigos ...> .... </ListadeArtigos >

</Artigos>
```

Figura V.114 RCML– Exemplo de utilização do elemento “Artigos”.

* ListadeArtigos

a. Definição :

O elemento “ListadeArtigos” serve para agrupar vários elementos do tipo “Artigo”. Este elemento armazena uma listagem de artigos. Pode também, agrupar elementos do mesmo tipo, ou seja, “ListadeArtigos”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “ListadeArtigos” e “Artigo”. A Figura V.115 mostra sua estrutura hierárquica.

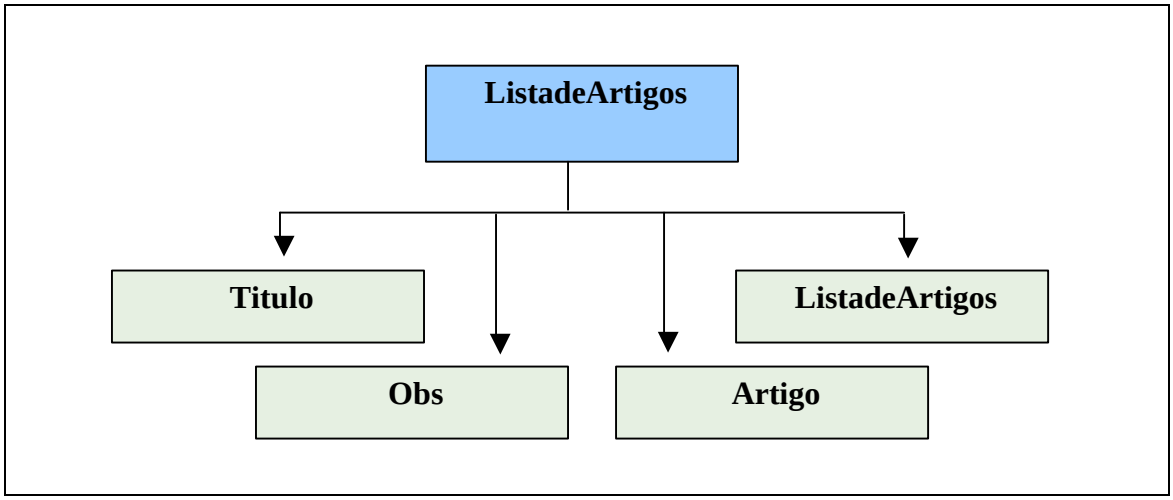


Figura V.115 RCML – Estrutura do elemento “ListadeArtigos”.

c. Sintaxe :

```
<!ELEMENT ListadeArtigos (Titulo, Observacao, (Artigo |
ListadeArtigos)*)>

<!ATTLIST  ListadeArtigos %Nohname;>
```

Figura V.116 RCML – Sintaxe do elemento “ListadeArtigos”.

d. Exemplo :

```
<ListadeArtigos NohName = “Artigos Gerais”>

  <Titulo> .... </Titulo>  <Obs> ... </Obs>

  < Artigo... >  ...  </Artigo>

  < Artigo... >  ...  </Artigo>

  <ListadeArtigos... >  ...  </ListadeArtigos></ListadeArtigos>
```

Figura V.117 RCML– Exemplo de utilização do elemento “ListadeArtigos”.

* Artigo

a. Definição :

Um elemento do tipo “Artigo” estrutura informações referentes a um artigo de um determinado assunto. Outros dados, como descrição e caminho do arquivo também são preservados.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “PalavrasChave”, “Descricao”, “Arquivo” e “Texto”. A Figura V.118 mostra sua estrutura hierárquica.

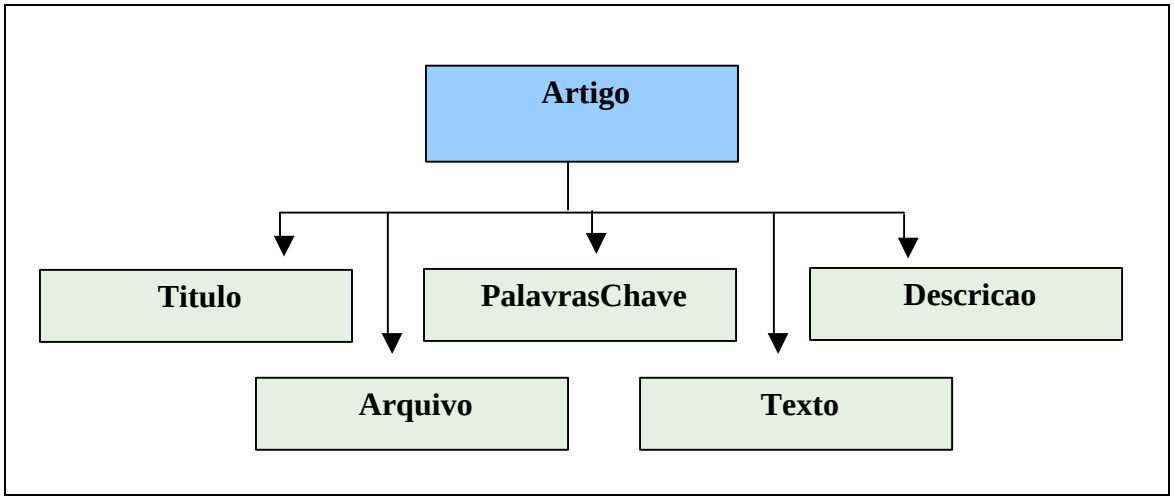


Figura V.118 RCML – Estrutura do elemento “Artigo”.

c. Sintaxe :

```
<!ELEMENT Artigo (Titulo, PalavrasChave, Descricao, Arquivo, Texto >

<!ATTLIST Artigo %Nohname;>
```

Figura V.119 RCML – Sintaxe do elemento “Artigo”.

d. Exemplo :

```
<Artigo  NohName = “Artigo 1” >

<Titulo> ... </Titulo>

<PalavrasChave> ... </PalavrasChave>

<Texto> ... </Texto>

</Artigo>
```

Figura V.120 RCML– Exemplo de utilização do elemento “Artigo”

* Arquivos

a. Definição :

O elemento “Arquivos” serve para agrupar vários elementos do tipo “Arquivos”.

b. Estrutura

Este objeto contém o fragmento “ListadeArquivos”. A Figura V.121 mostra sua estrutura hierárquica.

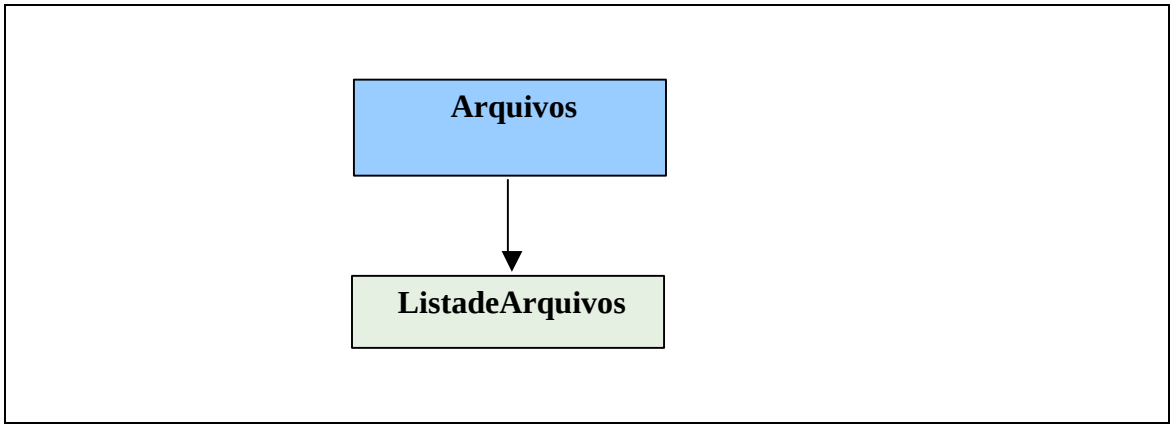


Figura V.121 RCML – Estrutura do elemento “Arquivos”.

c. Sintaxe :

```
<!ELEMENT Arquivos (ListadeArquivos)*>)

<!ATTLIST  Arquivos %Nohname;>
```

Figura V.122 RCML – Sintaxe do elemento “Arquivos”.

d. Exemplo :

```
<Arquivos NohName = “Arquivos”>

  <ListadeArquivos ...> .... </ListadeArquivos>

  <ListadeArquivos... > .... </ListadeArquivos>

  <ListadeArquivos ...> .... </ListadeArquivos>

</Arquivos>
```

Figura V.123 RCML – Exemplo de utilização do elemento “Arquivos”.

* ListadeArquivos

a. Definição :

O elemento “ListadeArquivos” serve para agrupar vários elementos do tipo “Arquivo”. Este elemento armazena uma listagem de arquivos. Pode também agrupar elementos do mesmo tipo, ou seja, “ListadeArquivos”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “ListadeArquivos” e “Arquivo”. A Figura V.124 mostra sua estrutura hierárquica.

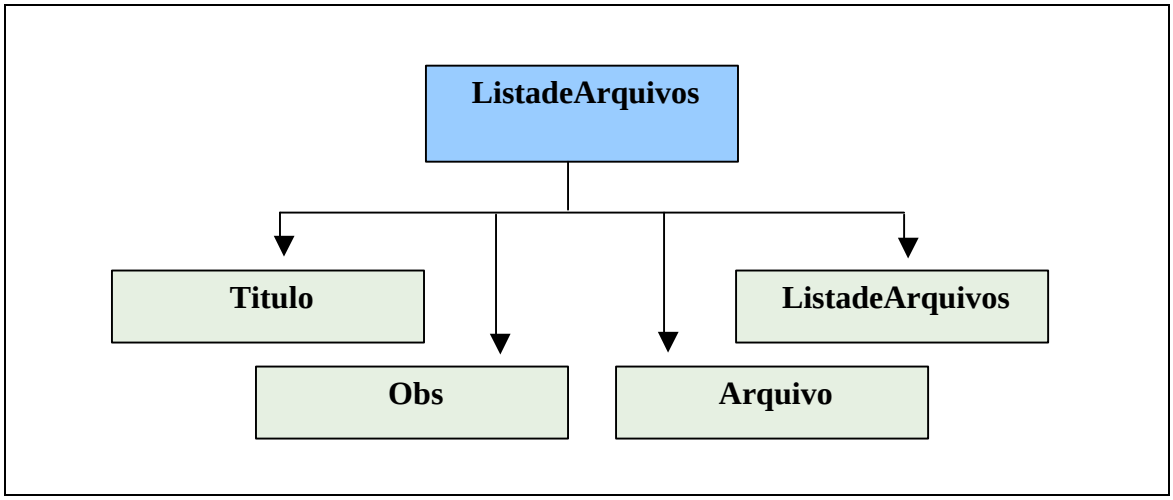


Figura V.124 RCML – Estrutura do elemento “ListadeArquivos”.

c. Sintaxe :

```
<!ELEMENT ListadeArquivos (Titulo, Observacao, (Arquivo |
ListadeArquivos)*)>

<!ATTLIST  ListadeArquivos %Nohname;>
```

Figura V.125 RCML – Exemplo de utilização do elemento “ListadeArquivos”.

d. Exemplo :

```
<ListadeArquivos NohName = “Arquivos Gerais”>

  <Titulo> .... </Titulo>  <Obs> ... </Obs>

  <Arquivo... >  ...  </Arquivo>

  <Arquivo... >  ...  </Arquivo>

  <ListadeArquivos... >  ...  </ListadeArquivos ></ListadeArquivos>
```

Figura V.126 RCML – Exemplo de utilização do elemento “ListadeArquivos”.

* Arquivo

a. Definição :

Um elemento do tipo “Arquivo” estrutura informações referentes a um determinado arquivo. Outros dados, como descrição e caminho do arquivo, também são preservados.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “PalavrasChave”, “Descricao”, “Arquivo” e “Texto”. A Figura V.127 mostra sua estrutura hierárquica.

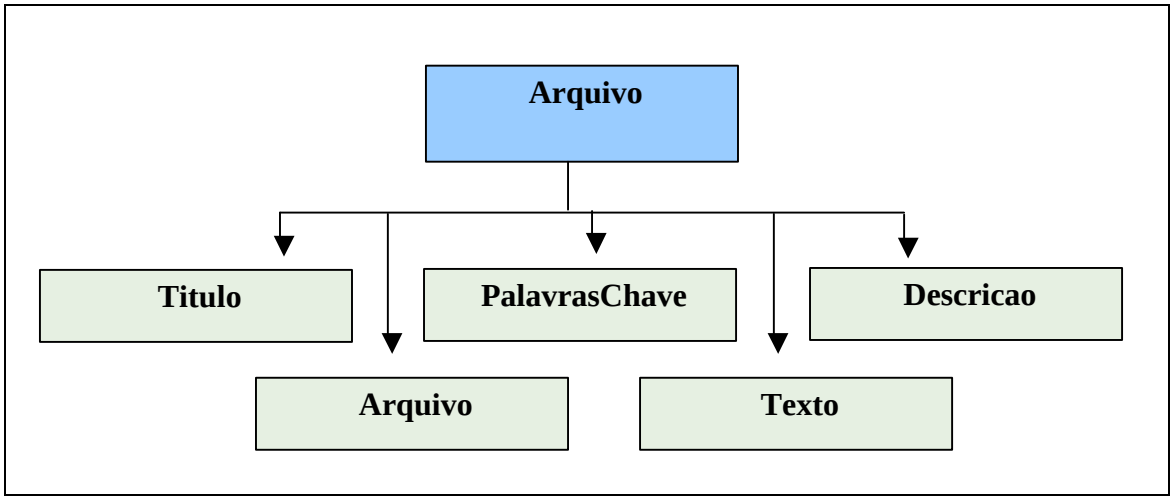


Figura V.127 RCML – Estrutura do elemento “Arquivo”.

c. Sintaxe :

```
<!ELEMENT Arquivo (Titulo, PalavrasChave, Descricao, Arquivo, Texto >

<!ATTLIST Arquivo %Nohname;>
```

Figura V.128 RCML – Sintaxe do elemento “Arquivo”.

d. Exemplo :

```
<Arquivo  NohName = “Arquivo 1” >

<Titulo> ... </Titulo>

<PalavrasChave> ... </PalavrasChave>

<Arquivo> ... </Arquivo>

<Texto> ... </Texto>

</Arquivo>
```

Figura V.129 RCML– Exemplo de utilização do elemento “Arquivo”

* Bibliografias

a. Definição :

O elemento “Bibliografias” serve para agrupar vários elementos do tipo “ListadeBibliografias”.

b. Estrutura

Este objeto contém o fragmento “ListadeBibliografias”. A Figura V.130 mostra sua estrutura hierárquica.

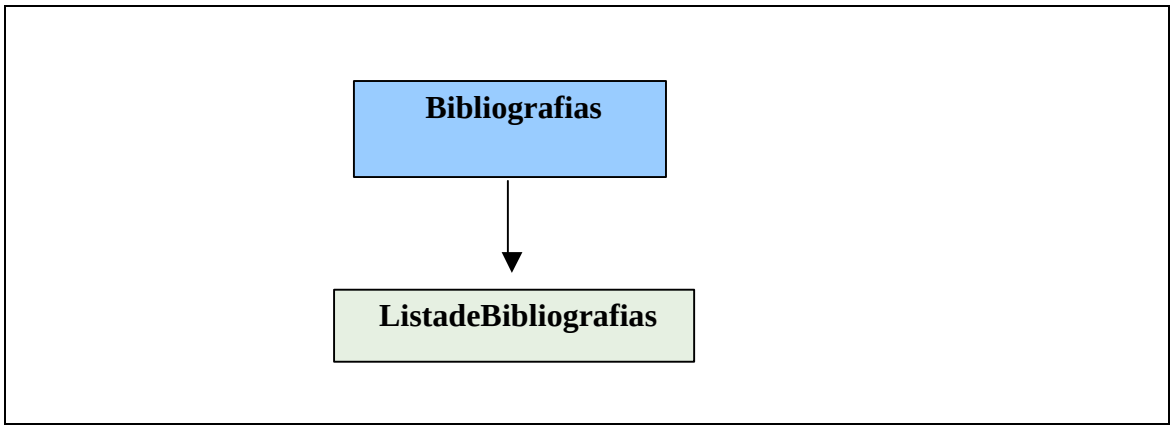


Figura V.130 RCML – Estrutura do elemento “Bibliografias”.

c. Sintaxe :

```
<!ELEMENT Bibliografias (ListadeBibliografias)*>)

<!ATTLIST  Bibliografias %Nohname;>
```

Figura V.131 RCML – Sintaxe do elemento “Bibliografias”.

d. Exemplo :

```
<Bibliografias NohName = “Bibliografias”>

  <ListadeBibliografias ...> .... </ListadeBibliografias>

  <ListadeBibliografias... > .... </ListadeBibliografias>

  <ListadeBibliografias ...> .... </ListadeBibliografias>

</Bibliografias>
```

Figura V.132 RCML– Exemplo de utilização do elemento “Bibliografias”.

* ListadeBibliografias

a. Definição :

O elemento “ListadeBibliografias” serve para agrupar vários elementos do tipo “Bibliografia”. Este elemento armazena uma listagem de artigos. Pode também, agrupar elementos do mesmo tipo, ou seja, “ListadeBibliografias”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “ListadeBibliografias” e “Bibliografia”. A Figura V.133 mostra sua estrutura hierárquica.

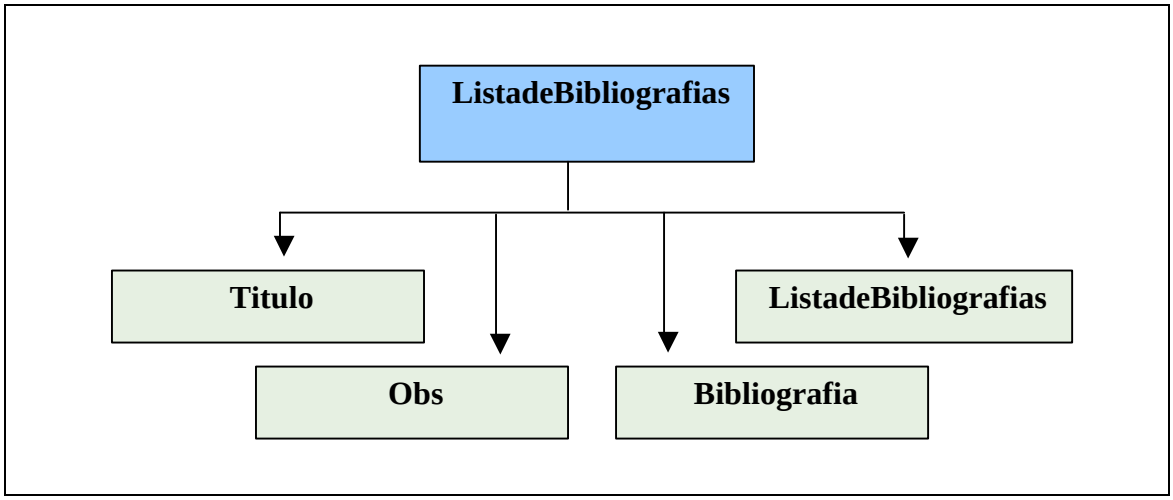


Figura V.133 RCML – Estrutura do elemento “ListadeBibliografias”.

c. Sintaxe :

```
<!ELEMENT ListadeBibliografias (Titulo, Observacao, (Bibliografia |
ListadeBibliografias)*)>

<!ATTLIST  ListadeBibliografias %Nohname;>
```

Figura V.134 RCML – Sintaxe do elemento “ListadeBibliografias”.

d. Exemplo :

```
<ListadeBibliografias NohName = “BibliografiasGerais”>

  <Titulo> .... </Titulo>  <Obs> ... </Obs>

  <Bibliografia...>  ...  </Bibliografia>  < Bibliografia...>  ...
</Bibliografia>

  <Listade Bibliografia...>  ...  </Listade Bibliografia>

</ListadeBibliografias>
```

Figura V.135 RCML– Exemplo de utilização do elemento “ListadeBibliografias”.

* Bibliografia

a. Definição :

Um elemento do tipo “Bibliografia” estrutura informações referentes a uma referência bibliográfica.

b. Estrutura

Este objeto contém os fragmentos “Abreviatura”, “Autores”, “Titulo”, “Endereco”, “Obs” e “PalavrasChave”. A Figura V.136 mostra sua estrutura hierárquica.

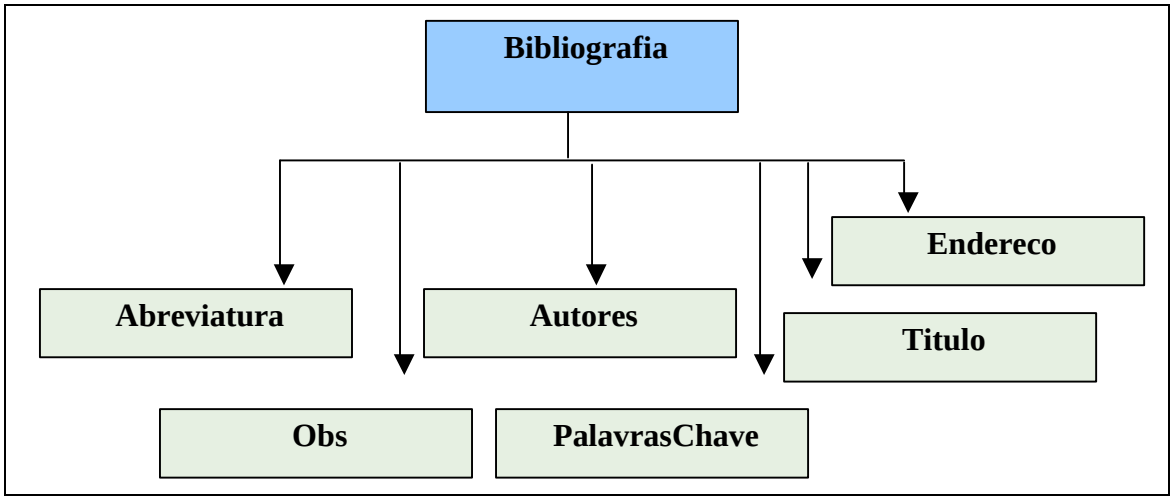


Figura V.136 RCML – Estrutura do elemento “Bibliografia”.

c. Sintaxe :

```
<!ELEMENT Bibliografia (Abreviatura,Autores,Endereco,Titulo,Obs,
PalavrasChave><!ATTLIST Bibliografia %Nohname;>
```

Figura V.137 RCML – Sintaxe do elemento “Bibliografia”.

d. Exemplo :

```
<Bibliografia NohName = “Bibliografia 1” >

   <Abreviatura> ... </Abreviatura>   <Autores> ... </Autores>

   <Endereco> ... </Endereco>

   <Titulo> ... </Titulo>   <Obs> ... </Obs>

   <PalavrasChave> ... </PalavrasChave>

</Bibliografia>
```

Figura V.138 RCML– Exemplo de utilização do elemento “Bibliografia”

* ObjetosMultimedia

a. Definição

O elemento “ObjetosMultimedia” armazena informações referentes a dados multimedia de um determinado repositório..

b. Estrutura

Este elemento possui a função de agrupar os fragmentos “Telas”, “Applets”, “Flashs” e “Videos”. A Figura V.139 mostra sua estrutura hierárquica.

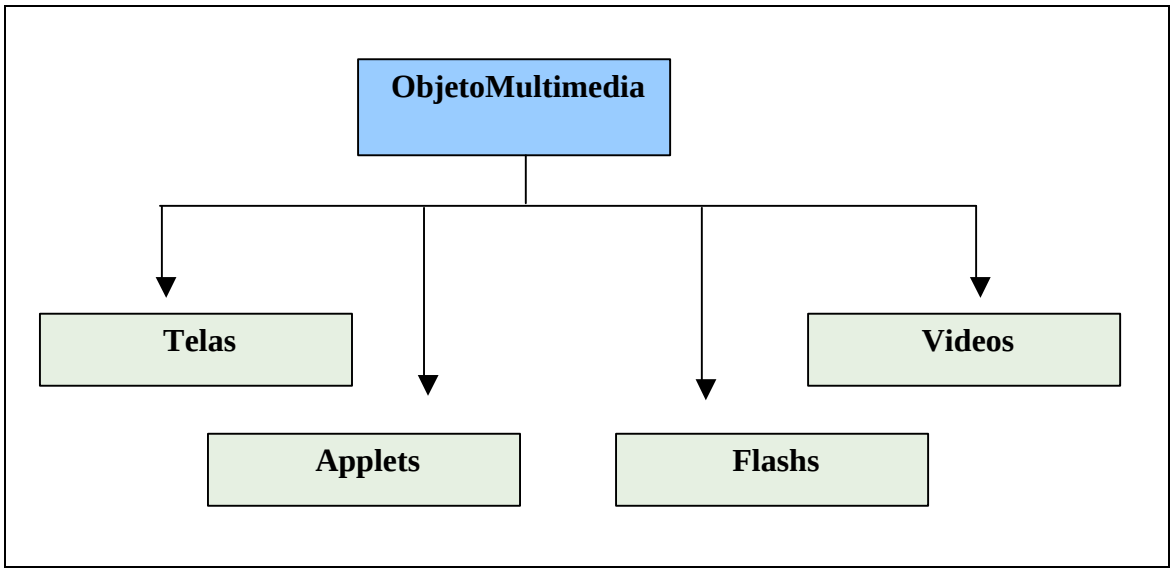


Figura V.139 RCML – Estrutura do elemento “ObjetosMultimedia”.

c. Sintaxe

```
<!ELEMENT ObjetosMultimedia(Telas,Applets,Flashs,Videos)>

<!ATTLIST  ObjetosMultimedia %Nohname;>
```

Figura V.140 RCML – Sintaxe do elemento “ObjetosMultimedia”.

d. Exemplo

```
<ObjetosMultimedia Nohname=”ObjetosMultimedia”>

   <Telas> ... </Telas>

   <Applets> ... </Applets>

   <Flashs> ... </Flashs>

   <Videos> ... </Videos>

</ObjetosdeMultimedia>
```

Figura V.141 RCML – Exemplo de utilização do elemento “ObjetosMultimedia”.

* Telas

a. Definição :

O elemento “Telas” serve para agrupar vários elementos do tipo “Telas”.

b. Estrutura

Este objeto contém o fragmento “ListadeTelas”. A Figura V.142 mostra sua estrutura hierárquica.

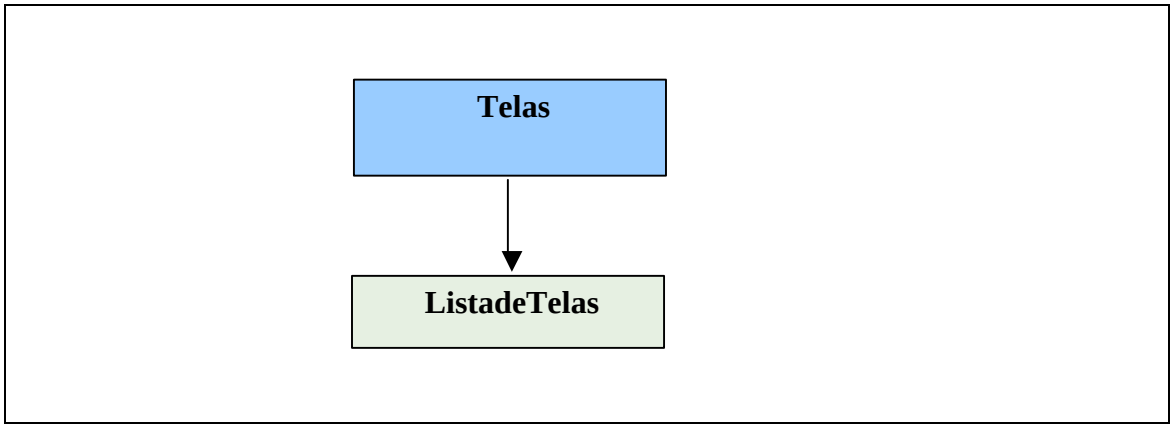


Figura V.142 RCML – Estrutura do elemento “Telas”.

c. Sintaxe :

```
<!ELEMENT Telas (ListadeTelas)*>)

<!ATTLIST  Telas %Nohname;>
```

Figura V.143 RCML – Sintaxe do elemento “Telas”.

d. Exemplo :

```
<Telas NohName = “Telas”>

  <ListadeTelas ...> .... </ListadeTelas>

  <ListadeTelas... > .... </ListadeTelas>

  <ListadeTelas ...> .... </ListadeTelas>

</Telas>
```

Figura V.144 RCML – Exemplo de utilização do elemento “Telas”.

* ListadeTelas

a. Definição :

O elemento “ListadeTelas” serve para agrupar vários elementos do tipo “Tela”. Este elemento armazena uma listagem de arquivos. Pode também agrupar elementos do mesmo tipo, ou seja, “ListadeTelas”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “ListadeTelas” e “Tela”. A Figura V.145 mostra sua estrutura hierárquica.

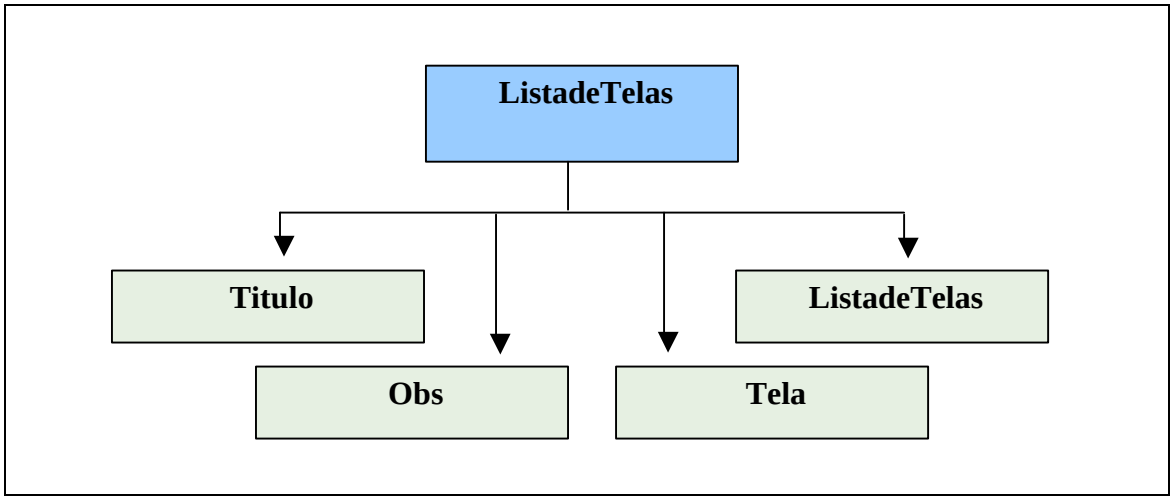


Figura V.145 RCML – Estrutura do elemento “ListadeTelas”.

c. Sintaxe :

```
<!ELEMENT ListadeTelas (Titulo, Observacao, (Tela | ListadeTelas)*)>

<!ATTLIST  ListadeTelas %Nohname;>
```

Figura V.146 RCML – Exemplo de utilização do elemento “ListadeTelas”.

d. Exemplo :

```
<ListadeTelas NohName = “Telas Gerais”>

  <Titulo> .... </Titulo>  <Obs> ... </Obs>

  <Tela... >  ...  </Tela>

  <Tela... >  ...  </Tela>

  <ListadeTela... >  ...  </ListadeTela>

</ListadeTelas>
```

Figura V.147 RCML – Exemplo de utilização do elemento “ListadeTelas”.

* Tela

a. Definição :

Um elemento do tipo “Tela” armazena informações referentes ao código HTML de uma determinada interface gráfica. Outros dados, como descrição e palavras-chave, também são preservados. Um objeto do tipo “Tela” pode agrupar outros objetos do mesmo tipo “Tela” para compor interfaces gráficas formadas por *frames*.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “PalavrasChave”, “Descricao”, “Texto” e “Tela”. A Figura V.148 mostra sua estrutura hierárquica.

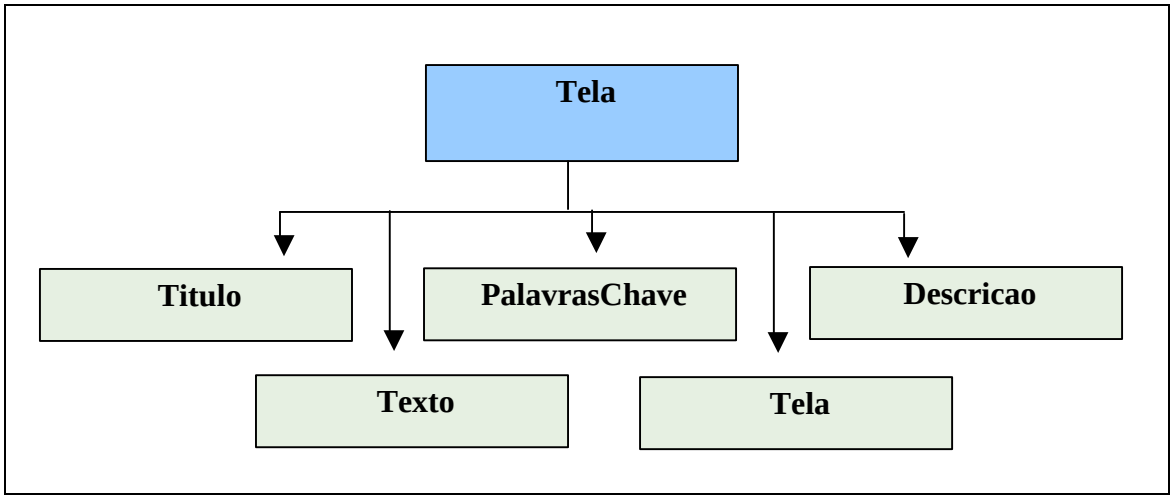


Figura V.148 RCML – Estrutura do elemento “Tela”.

c. Sintaxe :

```
<!ELEMENT Tela (Titulo, PalavrasChave, Descricao, Tela, Texto >

<!ATTLIST Tela %Nohname;>
```

Figura V.149 RCML – Sintaxe do elemento “Tela”.

d. Exemplo :

```
<Tela  NohName = “Tela 1” >

<Titulo> ... </Titulo><PalavrasChave> ... </PalavrasChave>

<Texto> ... </Textp>

<Tela> ... </Tela>

</Tela>
```

Figura V.150 RCML– Exemplo de utilização do elemento “Tela”

* Applets

a. Definição :

O elemento “Applets” serve para agrupar vários elementos do tipo “ListadeApplets”.

b. Estrutura

Este objeto contém o fragmento “ListadeApplets”. A Figura V.151 mostra sua estrutura hierárquica.

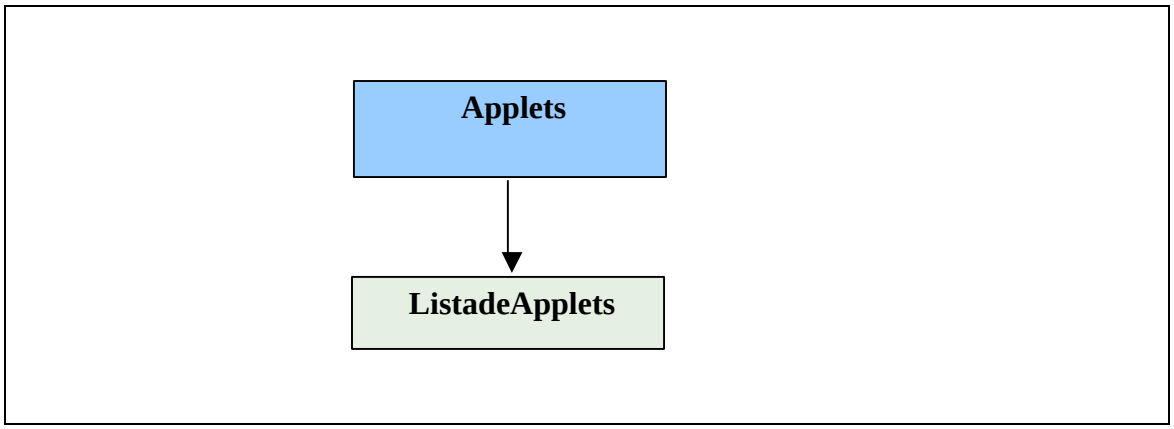


Figura V.151 RCML – Estrutura do elemento “Applets”.

c. Sintaxe :

```
<!ELEMENT Applets (ListadeApplets)*>)

<!ATTLIST  Applets %Nohname;>
```

Figura V.152 RCML – Sintaxe do elemento “Applets”.

d. Exemplo :

```
<Applets NohName = “Applets”>

  <ListadeApplets ...> .... </ListadeApplets>

  <ListadeApplets... > .... </ListadeApplets>

  <ListadeApplets ...> .... </ListadeApplets>

</Applets>
```

Figura V.153 RCML – Exemplo de utilização do elemento “Applets”.

* ListadeApplets

a. Definição :

O elemento “ListadeTelas” serve para agrupar vários elementos do tipo “Applet”. Este elemento armazena uma listagem de *applets*. Pode também agrupar elementos do mesmo tipo, ou seja, “ListadeApplets”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “ListadeApplets” e “Applet”. A Figura V.154 mostra sua estrutura hierárquica.

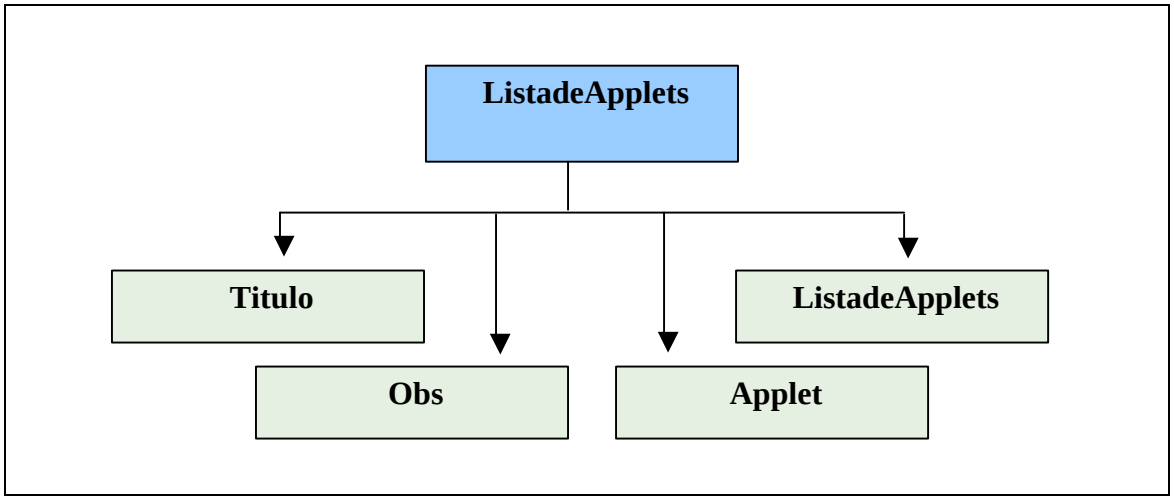


Figura V.154 RCML – Estrutura do elemento “ListadeApplets”.

c. Sintaxe :

```
<!ELEMENT ListadeApplets (Titulo, Observacao, (Applet |
ListadeApplets)*)>

<!ATTLIST  ListadeApplets %Nohname;>
```

Figura V.155 RCML – Exemplo de utilização do elemento “ListadeApplets”.

d. Exemplo :

```
<ListadeApplets NohName = “Applets Gerais”>

  <Titulo> .... </Titulo>  <Obs> ... </Obs>

  <Applet... >  ...  </Applets>  <Applet... >  ...  </Applets>

  <ListadeApplets... >  ...  </ListadeApplets>

</ListadeApplets>
```

Figura V.156 RCML – Exemplo de utilização do elemento “ListadeApplets”.

* Applet

a. Definição :

Um elemento do tipo “Applet” armazena informações referentes a *applets java*.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “PalavrasChave”, “Descricao”, “Codigo”, “Base”, “Largura”, “Altura”, “Alinhamento”, “Parametros” e “Texto”. A Figura V.157 mostra sua estrutura hierárquica.

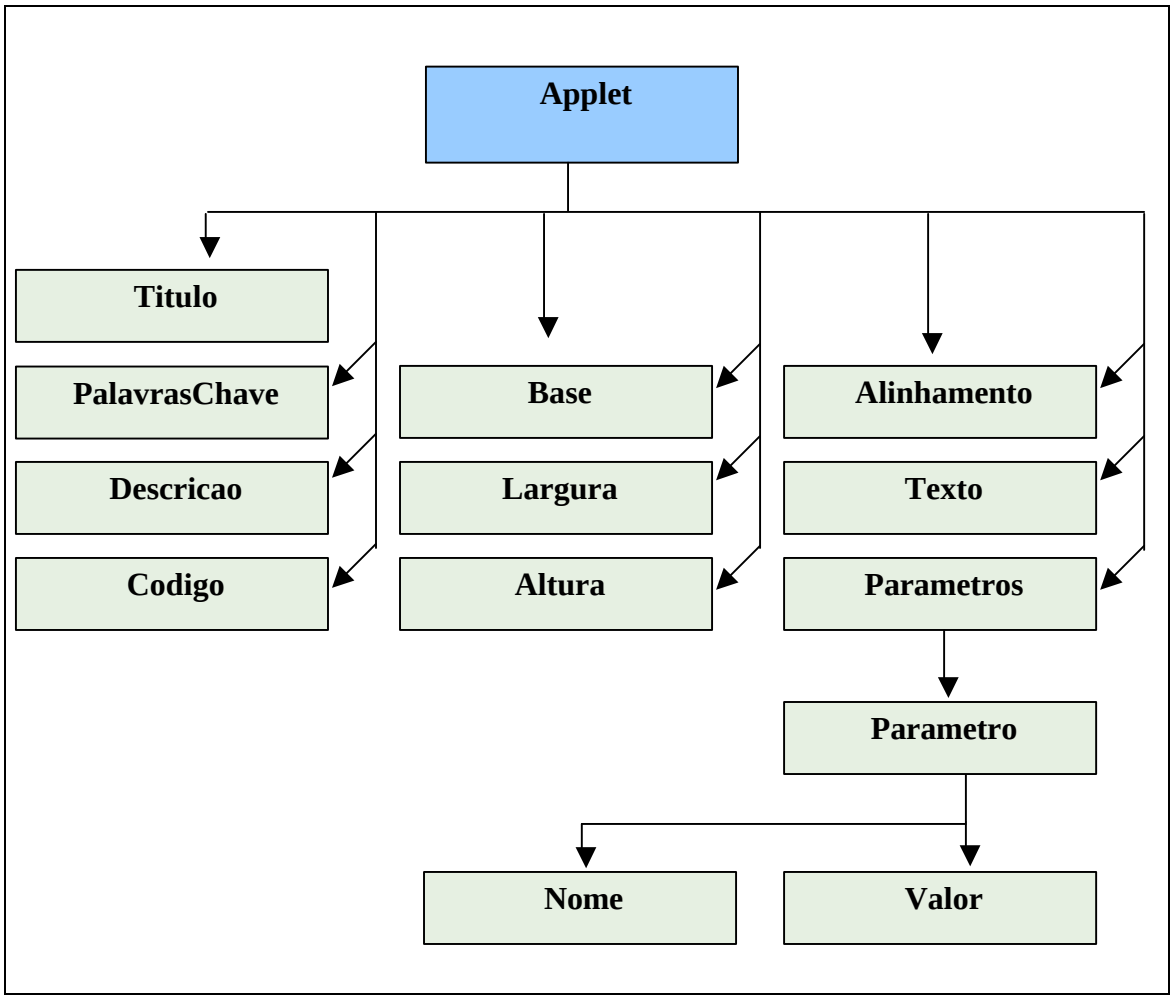


Figura V.157 RCML – Estrutura do elemento “Applet”.

c. Sintaxe :

```
<!ELEMENT Applet (Titulo, PalavrasChave, Descricao,Codigo,Base,
Largura,Altura,Alinhamento,Texto,Parametros)>

<!ATTLIST Applet %Nohname;>

<!ELEMENT Parametros (Parametro)+>

<!ATTLIST Parametros %Nohname;>

<!ELEMENT Parametro (Nome,Valor)

<!ATTLIST Parametro %Nohname;>
```

Figura V.158 RCML – Sintaxe do elemento “Applet”.

d. Exemplo :

```
<Applet NohName = “Applet 1” >

   <Titulo> ... </Titulo>

   <PalavrasChave> ... </PalavrasChave>

   <Descricao> … </Descricao>

   <Codigo> … </Codigo>

   <Base> ... </Base>

   <Largura> ... </Largura>

   <Altura> ... </Altura>

   <Alinhamento> ... </Alinhamento>

   <Texto> ... </Texto>

   <Parametros>  ... </Parametros>

</Applet>
```

Figura V.159 RCML– Exemplo de utilização do elemento “Applet”

* Flashs

a. Definição :

O elemento “Flahs” serve para agrupar vários elementos do tipo “ListadeFlashs”.

b. Estrutura

Este objeto contém o fragmento “ListadeFlashs”. A Figura V.160 mostra sua estrutura hierárquica.

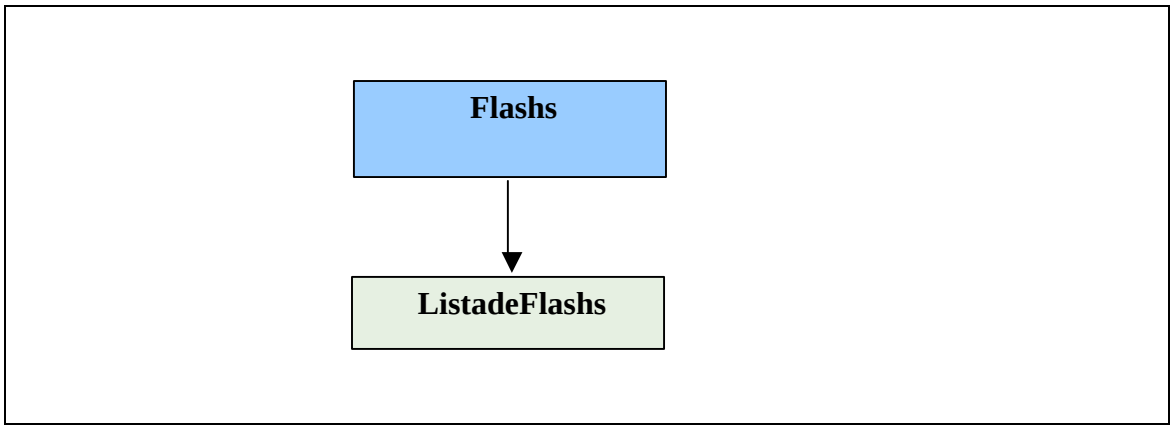


Figura V.160 RCML – Estrutura do elemento “Flashs”.

c. Sintaxe :

```
<!ELEMENT Flashs (ListadeFlashs)*>)

<!ATTLIST  Flashs %Nohname;>
```

Figura V.161 RCML – Sintaxe do elemento “Flashs”.

d. Exemplo :

```
<Flashs NohName = “Flashs”>

  <ListadeFlashs ...> .... </ListadeFlashs>

  <ListadeFlashs... > .... </ListadeFlashs>

  <ListadeFlashs ...> .... </ListadeFlashs>

</Flashs>
```

Figura V.162 RCML – Exemplo de utilização do elemento “Flashs”.

* ListadeFlashs

a. Definição :

O elemento “ListadeFlashs” serve para agrupar vários elementos do tipo “Flash”. Este elemento armazena uma listagem de animações em flash. Pode também agrupar elementos do mesmo tipo, ou seja, “ListadeFlashs”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “ListadeFlashs” e “Flash”. A Figura V.163 mostra sua estrutura hierárquica.

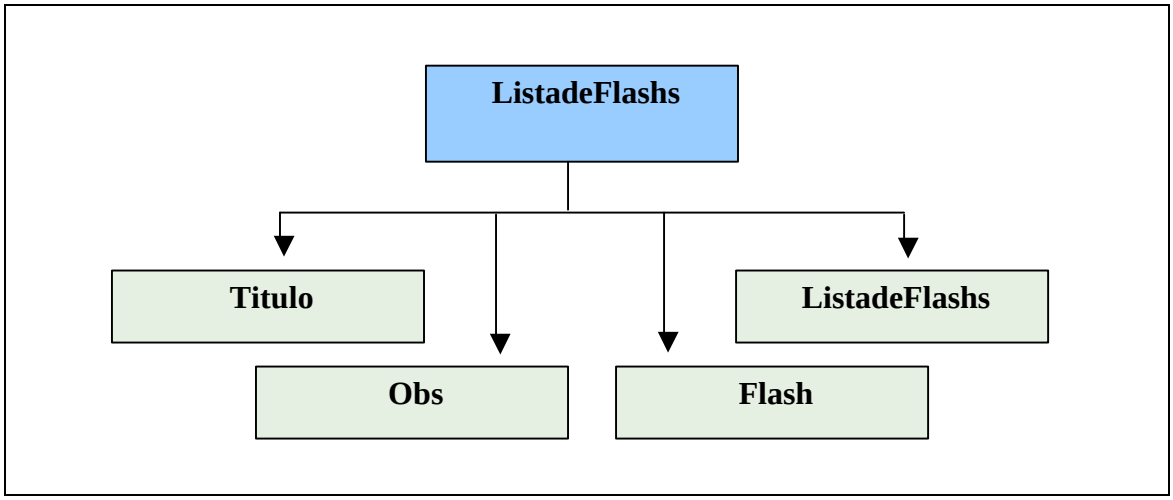


Figura V.163 RCML – Estrutura do elemento “ListadeFlashs”.

c. Sintaxe :

```
<!ELEMENT ListadeFlashs (Titulo, Observacao, (Flash |
ListadeFlashs)*)>

<!ATTLIST  ListadeFlashs %Nohname;>
```

Figura V.164 RCML – Exemplo de utilização do elemento “ListadeFlashs”.

d. Exemplo :

```
<ListadeFlashs NohName = “Arquivos Gerais”>

  <Titulo> .... </Titulo>  <Obs> ... </Obse>

  <Flash... >  ...  </Flash>    <Flash... >  ...  </Flash>

  <ListadeFlashs... >  ...  </ListadeFlashs>

</ListadeFlashs>
```

Figura V.165 RCML – Exemplo de utilização do elemento “ListadeFlashs”.

* Flash

a. Definição :

Um elemento do tipo “Flash” estrutura informações referentes à uma animação em flash. Outros dados, como descrição e caminho do arquivo, também são preservados.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “PalavrasChave”, “Descricao”, “Arquivo”, “Largura”, “Altura”, “Alinhamento”, “Escala”, “Cor”, “Loop”, “AutoPlay”, “Qualidade”, “Parametros” e “Texto”. A Figura V.166 mostra sua estrutura hierárquica.

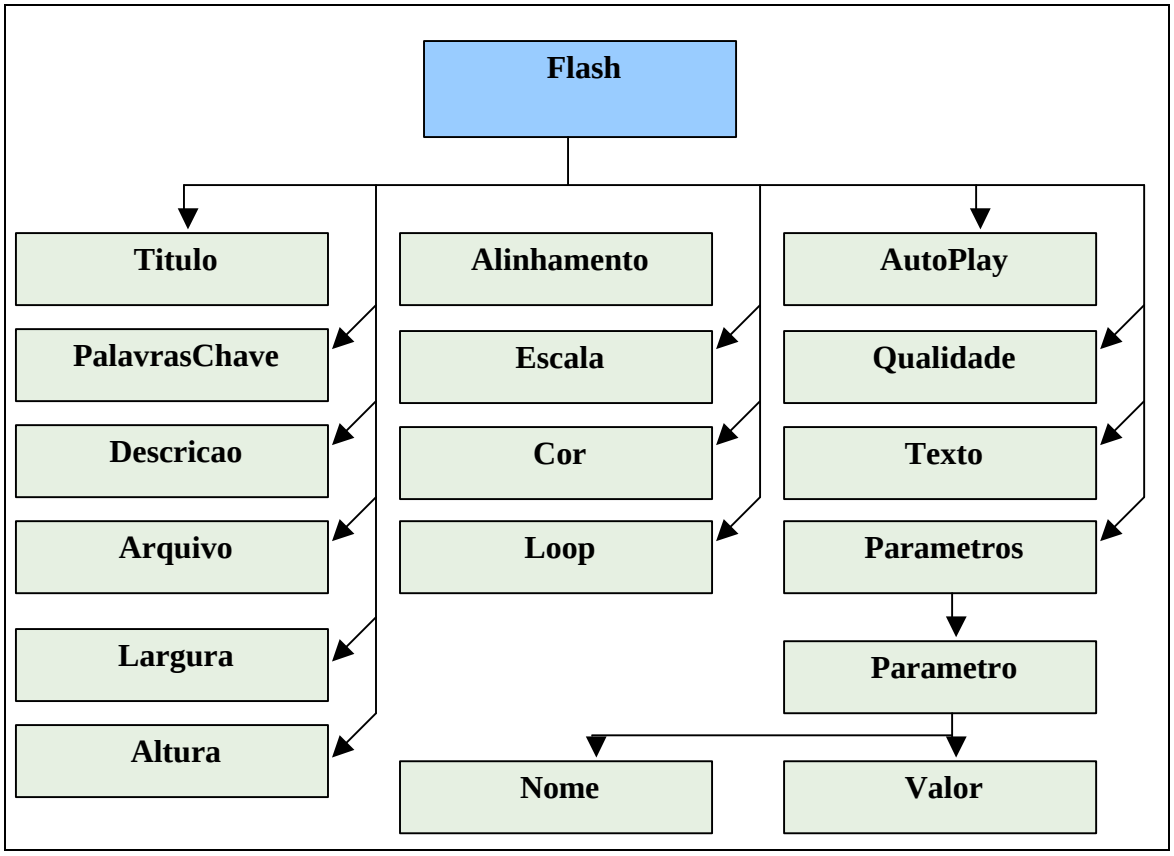


Figura V.166 RCML – Estrutura do elemento “Flash”.

c. Sintaxe :

```
<!ELEMENT Flash (Titulo, PalavrasChave, Descricao, Arquivo,Largura,
Altura, Alinhamento,Escala,Cor,Loop,AutoPlay, Qualidade, Texto,
Parametros)>

<!ATTLIST Flash %Nohname;>
```

Figura V.167 RCML – Sintaxe do elemento “Flash”.

d. Exemplo :

```
<Flash  NohName = “Flash 1” >

   <Titulo> ... </Titulo>

   <PalavrasChave> ... </PalavrasChave>

   <Descricao> … </Descricao>

   <Arquivo> … </Arquivo>

   <Largura> … </Largura>

   <Altura> … </Altura>

   <Alinhamento> … </Alinhamento>

   <Escala> ... </Escala>

   <Cor> ... </Cor>

   <Loop> ... </Loop>

   <AutoPlay> ... </AutoPlay>

   <Qualidade> ... </Qualidade>

   <Texto> ... </Texto>

   <Parametros> ... </Parametros>

</Flash>
```

Figura V.168 RCML– Exemplo de utilização do elemento “Flash”

* Videos

a. Definição :

O elemento “Videos” serve para agrupar vários elementos do tipo “Videos”.

b. Estrutura

Este objeto contém o fragmento “ListadeVideos”. A Figura V.169 mostra sua estrutura hierárquica.

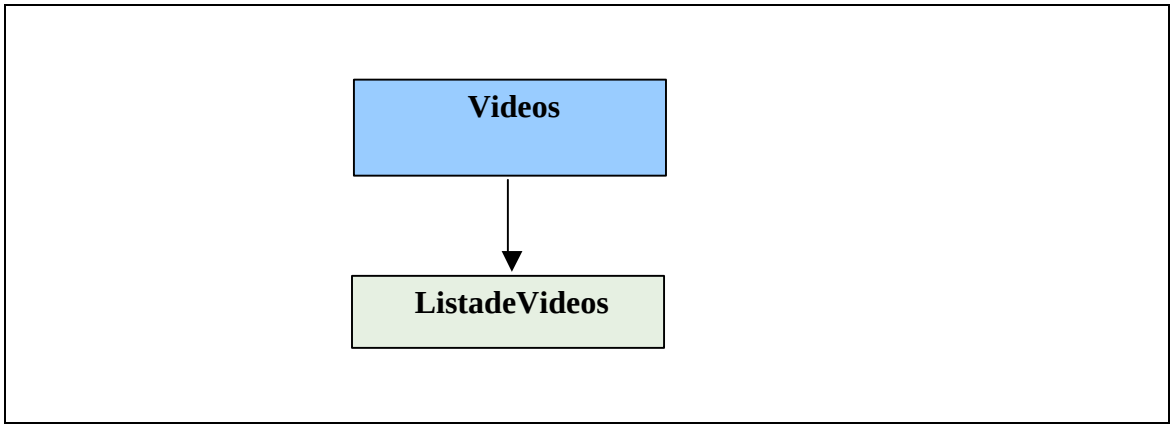


Figura V.169 RCML – Estrutura do elemento “Videos”.

c. Sintaxe :

```
<!ELEMENT Videos (ListadeVideos)*>)

<!ATTLIST Videos %Nohname;>
```

Figura V.170 RCML – Sintaxe do elemento “Videos”.

d. Exemplo :

```
<Videos NohName = “Videos”>

  <ListadeVideos ...> .... </ListadeVideos>

  <ListadeVideos... > .... </ListadeVideos>

  <ListadeVideos ...> .... </ListadeVideos>

</Videos>
```

Figura V.171 RCML – Exemplo de utilização do elemento “Videos”.

* ListadeVideos

a. Definição :

O elemento “ListadeVideos” serve para agrupar vários elementos do tipo “Video”. Este elemento armazena uma listagem de arquivos. Pode também agrupar elementos do mesmo tipo, ou seja, “ListadeVideos”.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “Obs”, “ListadeVideos” e “Video”. A Figura V.172 mostra sua estrutura hierárquica.

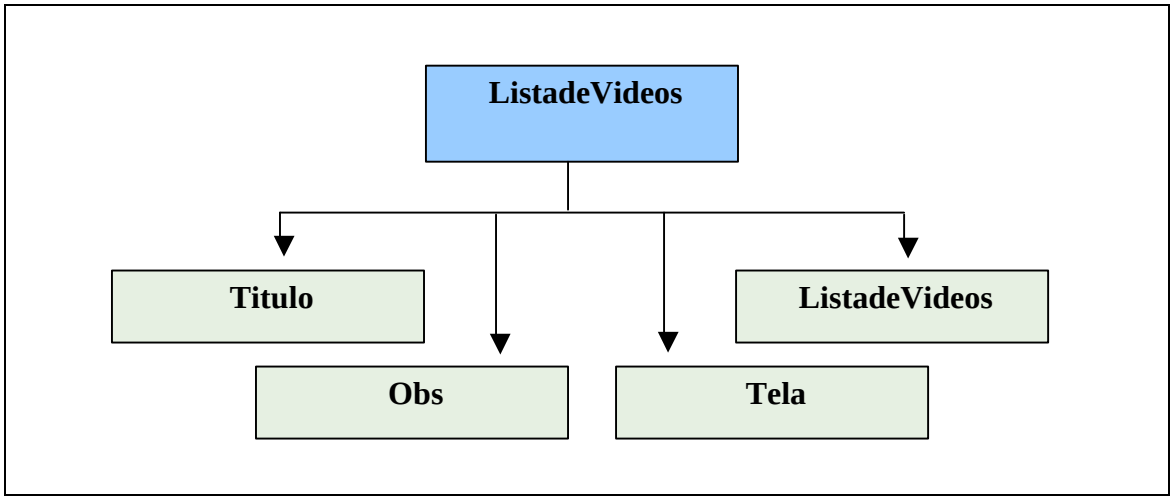


Figura V.172 RCML – Estrutura do elemento “ListadeVideos”.

c. Sintaxe :

```
<!ELEMENT ListadeVideos (Titulo, Observacao, (Video| ListadeVideos)*)>

<!ATTLIST  ListadeVideos %Nohname;>
```

Figura V.173 RCML – Exemplo de utilização do elemento “ListadeVideos”.

d. Exemplo :

```
<ListadeVideos NohName = “Videos Gerais”>

  <Titulo> .... </Titulo>  <Obs> ... </Obs>

  <Video... >  ...  </Video>

  <Video... >  ...  </Video>

  <ListadeVideos... >  ...  </ListadeVideos>

</ListadeVideos>
```

Figura V.174 RCML – Exemplo de utilização do elemento “ListadeVideos”.

* Video

a. Definição :

Um elemento do tipo “Video” armazena informações referentes à um video em algum formato como mpeg.. Outros dados, como descrição e palavras-chave também são preservados.

b. Estrutura

Este objeto contém os fragmentos “Titulo”, “PalavrasChave”, “Descricao”, “Arquivo”, “Largura”, “Altura” e “Alinhamento”. A Figura V.175 mostra sua estrutura hierárquica.

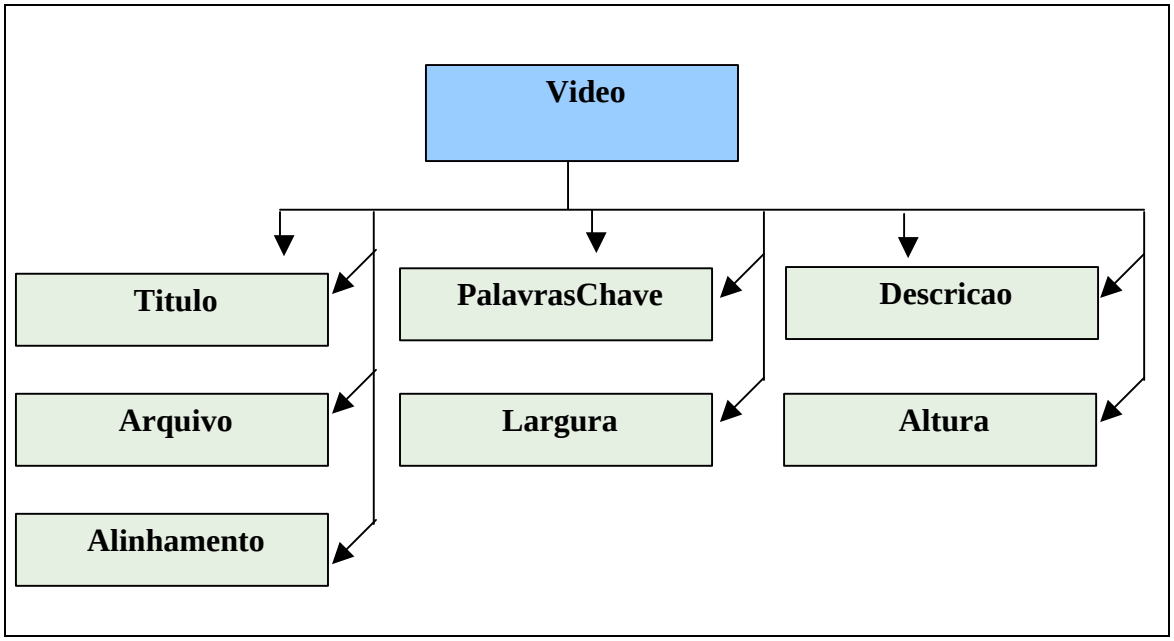


Figura V.175 RCML – Estrutura do elemento “Video”.

c. Sintaxe :

```
<!ELEMENT Video (Titulo, PalavrasChave, Descricao, Arquivo,Largura,
Altura,Alinhamento)>

<!ATTLIST Video %Nohname;>
```

Figura V.176 RCML – Sintaxe do elemento “Video”.

d. Exemplo :

```
<Video  NohName = “Video 1” >

   <Titulo> ... </Titulo>

   <PalavrasChave> ... </PalavrasChave>

   <Descricao> … </Descricao>

   <Arquivo> … </Arquivo>

   <Largura> … </Largura>

   <Altura> … </Altura>

   <Alinhamento> … </Alinhamento>

</Video>
```

Figura V.177 RCML– Exemplo de utilização do elemento “Video”

Anexo VI

Linguagem RCM Script

A linguagem RCM Script foi desenvolvida para prover a manipulação de dados estruturados. Enquanto que XSL traz uma sintaxe muito complexa e genérica, esta linguagem busca facilitar a implementação através de uma sintaxe muito simples.

VI.1 A Gramática RCM Script

Conforme [COM 03] uma gramática é definida pela quádrupla G = (N,T,P,S), onde N indica os símbolos não terminais, T, os símbolos terminais, P o conjunto de regras de produção e S o símbolo inicial, temos o exposto na Figura VI.1.

```
N = (REF,IF,CONDTION,THEN,ELSE,FOR,BREAK,PARAMETRO,COMMAND);

T = ();

P = (<,>,=,”,REF,TIPO,ID,GET,CONDITION,THEN,ELSE,OP,>=,<=,



S -> COMMAND

COMMAND -> REF | IF | FOR | BREAK | PARAMAETRO | TEXTO

REF -> <REF TIPO = “TIPO” ID = “TEXTO”/> | <REF TIPO = “TIPO” ID =
“TEXTO” GET = “TEXTO”/>

IF -> <IF> CONDIION THEN ELSE </IF>

CONDITION -> <CONDITION> (REF | TEXTO) OP (REF | TEXTO) </CONDITION>

THEN -> <THEN>COMMAND</THEN>

ELSE -> <ELSE>COMMAND</ELSE>

OP -> <OP ID=”OPLIST”/>

OPLIST ->   =,>.<.>=.<=

FOR -> <FOR TIPO = “TIPO” ID = “TEXTO”>COMMAND </FOR> |

<FOR TIPO = “TIPO” ID = “TEXTO” GET = “TEXTO”>COMMAND</FOR>

TEXTO -> string

)
```

Figura VI.1 RCM Script - Gramática.

VI.2 Principais comandos :

* Referências

a. Descrição

Este comando faz referência a um nó específico. Os parâmetros “Tipo” e “ID” identificam o nó a ser referenciado. Assim, como resultado, o conteúdo do nó é copiado diretamente para o documento que está utilizando o comando “REF”. O atributo “GET” serve para especificar qual fragmento será referenciado.

b. Sintaxe

```
<REF TIPO =”Tipo de Dado” ID = “Nome de um nó” GET”Texto” />
```

Figura VI.2 RCM Script – Comando “REF”.

c. Exemplo

Supomos a existência de um nó para um determinado teorema cujo nome identificador é “Teorema 1”. Para incluirmos esta teorema dentro de um texto, será necessário o exposto na Figura VI.3.

```
Esse é o texto esse é o  texto esse é o texto
A tabela abaixo mostra o exemplo de integrar conteúdos :
O texto do teorema é
<REF TIPO = “Teorema” ID =”Teorema 1” Get=”Texto” />
Já o título do teorema é
<REF TIPO = “Teorema” ID =”Teorema 1” Get=”Titulo” />
Segue o texto segue o texto segue o texto.
```

Figura VI.3 RCM Script – Utilizando Referências.

No caso acima, dois fragmentos foram copiados diretamente para o texto : título e texto. O comando GET deve ser utilizado para buscar apenas o fragmento desejado. O exemplo da Figura VI..4 mostra como buscar alguns dados específicos de um “gráfico”.

```
            Esse é um exemplo de integração através de referencias

            O gráfico possui altura <REF TIPO =”Gráfico” ID =”Gráfico
1” GET=”Altura” />

      e largura  <REF TIPO =”Gráfico” ID =”Gráfico 1” GET=”Largura”
/>.

      Este gráfico está em formato SVG. Segue abaixo o código SVG
deste gráfico :

            <REF TIPO =”Gráfico” ID =”Gráfico 1” GET=”SVG” />

            Veja abaixo a visualização do gráfico :

            <REF TIPO =”Gráfico” ID =”Gráfico 1” />
```

Figura VI.4 RCM Script – Utilizando Referências com o método GET.

A Tabela VI.1 mostra que informações podem ser buscadas através do atributo “GET” com cada tipo de conteúdo :

Elemento	Informações referenciadas com “Get”
Lista de Recursos	Título, Observação,Nome,Tipo,Depth
Recurso	Título, Palavras Chave, Texto,Nome,Tipo,Depth
Lista de Fórmulas	Título, Observação,Nome,Tipo,Depth
Fórmula	Título, Palavras Chave, Descrição MathTeX, MathML,Nome,Tipo,Depth
Lista de Gráficos	Título, Observação,Nome,Tipo,Depth
Gráfico	Título, Palavras Chave, DescriçãoSVG, Nome,Tipo,Depth
Lista de Tabelas	Título, Observação,Nome,Tipo,Depth
Tabela	Título, Palavras Chave, Descrição,Linhas, Colunas, Texto,Nome,Tipo,Depth
Lista de Figuras	Título, Observação,Nome,Tipo,Depth
Figura	Título, Descrição, Palavras Chave, Arquivo,Nome,Tipo,Depth
Lista de Exemplos	Título, Observação,Nome,Tipo,Depth
Exemplo	Título, Palavras Chave, Texto,Nome,Tipo,Depth
Lista de Exercícios	Título, Observação,Nome,Tipo,Depth
Exercício	Título, Palavras Chave, Descrição, Resolução,

	Resposta,Nome,Tipo,Depth
Lista de Teoremas	Título, Observação,Nome,Tipo,Depth
Teorema	Título, Palavras Chave, Descrição, Resolução, Resposta,Nome,Tipo,Depth
Lista de Definições	Título, Observação,Nome,Tipo,Depth
Definição	Título, Palavras Chave, Texto,Nome,Tipo,Depth
Glossário	Título,Observação,Nome,Tipo,Depth
Letra	Nome,Tipo,Depth
Lista de Bibliografias	Título, Observação,Nome,Tipo,Depth
Bibliografia	Abreviatura, Autores, Título, Web, Obs, Palavras Chave, Nome,Tipo,Depth
Lista de Applets	Título, Observação,Nome,Tipo,Depth
Applet	Título, Palavras Chave, Descrição, Path, Base, Largura, Altura, Alinhamento, Texto,Nome,Tipo,Depth
Lista de Flashs	Título, Observação,Nome,Tipo,Depth
Flash	Título, Palavras Chave, Descrição, Arquivo, Largura, Altura, Alinhamento, Escala, Cor, Loop, AutoPlay, Qualidade,Nome,Tipo,Depth
Lista de Videos	Título, Observação,Nome,Tipo,Depth
Vídeo	Título, Palavras Chave, Descrição, Arquivo, Largura, Altura, Alinhamento,Nome,Tipo,Depth

Tabela VI.1 RCM Script– Utilização do método “GET”.

* Desvio Condicional

a. Descrição

O objetivo deste comando é realizar um desvio condicional baseado numa determinada condição. Qualquer outro comando pode ser aninhado dentro das *tags* do desvio condicional..

b. Sintaxe

```
<IF>
<CONDITION>operando1 <OP ID="=,>=,<=,>,>"/> operando2</CONDITION>
<THEN></THEN>
<ELSE></ELSE>
</IF>
```

Figura VI.5 RCM Script – Desvio Condicional

c. Exemplo

Listar todas as fórmulas cuja altura é maior ou igual a 100 dentro de uma tabela HTML. O código correspondente é apresentado na Figura VI.6.

```
      <table border = "1">
      <FOR TIPO="Lista de Fórmulas" ID="Lista 01">
      <tr>
       <IF>
       <CONDITION>< REF GET="Altura"/> <OP ID=">="/> 100 </CONDITION>
              <THEN><td><REF /></td></THEN>
      </IF>
      </tr>
      </FOR>
      </table>
```

Figura VI.6 RCM Script – Desvio Condicional.

* Laços Interativos

a. Descrição

Através do comando "FOR" é possível percorrer todos os nós descendentes do nó identificado pelos atributos "TIPO" e "ID". Pode-se restringir este percurso através do atributo GET direcionando o laço apenas para os elemento do tipo "lista de items" ou do tipo "item". O comando "FOR" pode ser utilizado para a geração dinâmica de documentos.

b. Sintaxe

```
<FOR TIPO =”Tipo de Dado” ID = “Nome de um nó” GET” Tipo de Dado”>

... comandos ...

</FOR>
```

Figura VI.7 RCM Script – Laços Interativos.

c. Exemplo

Supõe-se a necessidade de uma tabela com a listagem de todos os nomes e respectivos arquivos das figuras que estão localizadas na lista de figuras nomeada com “Lista de Figuras 001”. Assim, efetua-se um laço para localizar o nó e percorrer cada descendente e, em cada interação, adicionar as colunas correspondentes da tabela cujos valores (nomes e arquivos) são obtidos através do comando “REF”. No exemplo abaixo será mostrado que a linguagem de scripts é orientada ao contexto, ou seja, qualquer comando utilizado dentro do contexto de um laço “FOR” não é necessário especificar atributos de localização do nó (Tipo e ID), pois pressupõe-se que o nó já está localizado, sendo o nó da interação atual do laço. A Figura VI.8 ilustra o código correspondente.

```
      <table border=”1”>

      <FOR TIPO=”Lista de Figuras” ID=”Lista de Figuras 001” GET=”Figura”>

       <tr>

       <td><REF GET=”Nome”/></td><td><REF GET=”Arquivo”/></td>

      </tr>

      </FOR>

      </table>
```

Figura VI.8 RCM Script – Laços Interativos

Caso o atributo “GET” do comando “FOR” não fosse utilizado, seria necessária a utilização de um desvio condicional para garantir que o laço percorra somente os tipos de nós necessários. Segue na Figura VI.9 o código equivalente sem a utilização do atributo “GET”.

```
<table border=”1”>
<FOR TIPO=”Lista de Figuras” ID=”Lista de Figuras 001” >
<IF>
<CONDITION><REF GET=”Tipo”/><OP ID=”=”/>Figura</CONDITION>
<THEN>
 <tr>
 <td><REF GET=”Nome”/></td><td><REF GET=”Arquivo”/></td>
</tr>
</THEN>
</FOR>
</table>
```

Figura VI.9 RCM Script – Laços Interativos.

* Links

a. Descrição

O gerenciador de conteúdo fornece um mecanismo de gerenciamento orientado ao conteúdo (nó), e não ao arquivo, tornando a existência de arquivos transparente para o usuário. Isso traz uma maior facilidade na manutenção dos dados. É possível criar links que apontam para determinados nós do conteúdo através dos atributos “TIPO” e “ID” do comando “LINK”.

b. Sintaxe

```
<LINK TIPO = “Tipo de Dado” ID=”Texto”> Texto </LINK>
```

Figura VI.10 RCM Script – Links.

c. Exemplo

O exemplo na Figura VI.11 cria uma lista em uma tabela de links de todas as fórmulas da lista nomeada “Lista de Fórmulas 005”. É possível observar que o comando “REF”utilizado dentro do contexto do “FOR”, não utiliza os atributos “TIPO” e “ID” para identificar um determinado nó.

```
<table border = “1”>
<FOR TIPO=”Lista de Fórmulas” ID=”Lista de Fórmulas 005” GET=”Fórmula”>
<tr>
<td><LINK><REF GET=”Nome”/></LINK></td>
</tr>
</FOR>
</table>
```

Figura VI.11 RCM Script – Links.

Caso o atributo “GET” do comando “FOR” não fosse utilizado, seria necessária a utilização de um desvio conditional para garantir que o laço percorra somente os tipos de nós necessários. Segue, na Figura VI.12, o código equivalente sem a utilização do atributo “GET”.

```
<table border = “1”>
<FOR TIPO=”Lista de Fórmulas” ID=”Lista de Fórmulas 005” GET=”Fórmula”>
<IF>
<CONDITION><REF GET=”Tipo”/><OP ID=”=”/>Fórmula</CONDITION>
<THEN><tr>
<td><LINK><REF GET=”Nome”/></LINK></td>
</tr>
</THEN>
</IF>
</FOR>
</table>
```

Figura VI.12 RCM Script – Links.

* Gerando arquivos

a. Definição :

Para viabilizar um mecanismo de exportação de dados, desenvolveu-se um comando para gerar arquivos capaz de integrar-se com toda a sintaxe da linguagem RCM Script. Assim, qualquer modelo de exportação poderá ser definido pelo usuário através da utilização desta linguagem.

b. Sintaxe :

```
<ToFILE>

<NAME> NomedoArquivo.html </NAME>

<TEXT>

…

</TEXT>

</TOFILE>
```

Figura VI.13 RCM Script – Gerando Arquivos.

c. Exemplo :

O exemplo da Figura VI.14 irá gerar uma seqüência de arquivos HTML baseados nos elementos do tipo “Recurso” definidos através da linguagem.

```
      <TOFILE>
        <NAME>Recursos.HTML</NAME>
        <TEXT>
          <table      border      =      "2">                        <tr>
<td><b>Título</b></td><td><b>Descrição</b></td></tr>
          <FOR>
            <FOR >
                   <IF>
                   <CONDITION>
                     <REF GET="Tipo"/> <OP ID="="/> Tópico
                   </CONDITION>
                   <THEN>
                     <TOFILE>
                     <NAME><REF GET="Título"/>.html </NAME>
                     <TEXT><REF /></TEXT>
                     </TOFILE>
                     <tr>
                     <td> <REF GET="Título"/> </td>
                     <td> <REF GET="Palavras Chave"/> </td>
                     </tr>
                    </THEN>
                    </IF>
            </FOR>
          </FOR>
          </table>
        </TEXT>
      </TOFILE>
```

Figura VI.14 RCM Script – Gerando Arquivos.

5.5.1.2 Funções :

As funções definidas nessa linguagem são utilizadas através do parâmetro “GET”. Essas funções têm por objetivo buscar valores internos dos conteúdos.

As funções propostas inicialmente são as que seguem:

* **Depth** : Retorna o nível de profundidade na hierarquia XML de um nó específico.
* **Nome** : Retorna o nome de nó um nó específico.
* **Cont** : Retorna o valor do contador dentro do contexto de um laço “FOR”.

Anexo VII

Linguagem GraphML

VII.1 Introcução

A linguagem GraphML (*Graph Marckup Language*) foi desenvolvida com o propósito de descrever conteúdos de gráficos estatísticos preservando informações referentes aos seus dados e à sua renderização.

VII.2 Elementos GraphML

* Elemento Grafico

a. Definição :

O elemento gráfico é o elemento de documento, [W3C 03d], da linguagem GraphML. Dentro deste elemento estarão todos os outros elementos da linguagem. Possui um atributo tipo, que descreve o tipo de gráfico (barras ou pizza).

b. Estrutura

Este objeto possui o fragmento “Item” A Figura VII.1 mostra sua estrutura hierárquica.

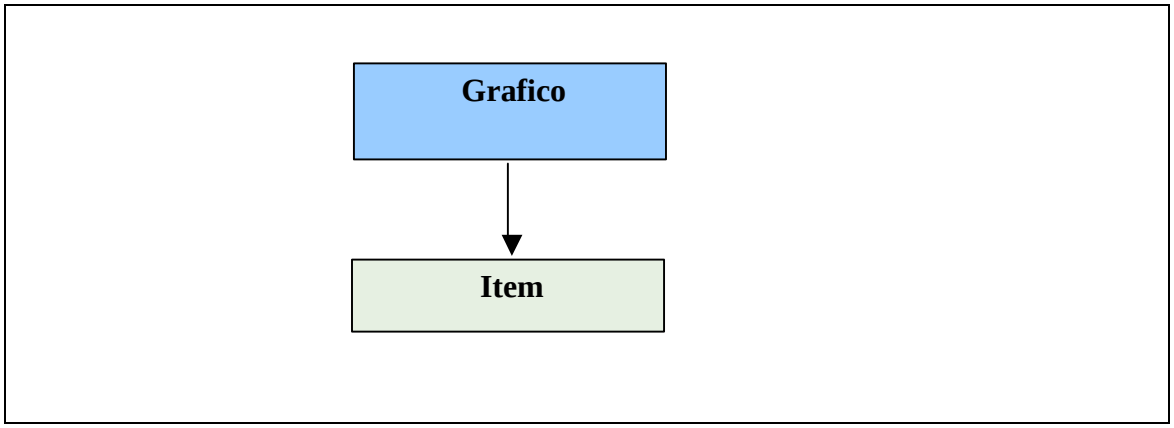


Figura VII.1 GraphML – Estrutura do elemento “Grafico”.

c. Sintaxe :

```
<!ELEMENT Grafico (Item)*>

<!ATTLIST Grafico Tipo #PCDATA>
```

Figura VII.2 GraphML – Sintaxe do elemento “Grafico”.

d. Exemplo :

```
<Grafico Tipo = “Pizza”>

......

</Grafico>
```

Figura VII.3 GraphML – Exemplo de utilização do elemento “Grafico”.

* Elemento Item

a. Definição :

O elemento *Item* descreve cada item do gráfico, ou seja, cada barra, no caso do gráfico de barras, ou cada “fatia”, no caso do gráfico de pizza. Possui um atributo *descricao*, que fornece a descrição de cada item. Possui, também, outros atributos que não serão citados.

b. Estrutura

Este elemento possui o fragmento “Valor”. A Figura VII.4 mostra sua estrutura hierárquica.

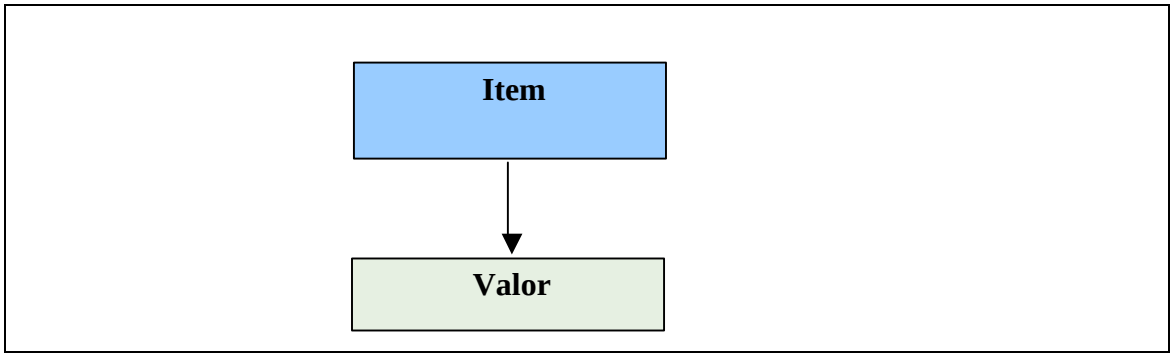


Figura VII.4 GraphML – Estrutura do elemento “Item”.

c. Sintaxe :

```
<!ELEMENT Item (Valor)>

<!ATTLIST Item Descricao #PCDATA>
```

Figura VII.5 Linguagem GraphML – Sintaxe do elemento “Item”.

d. Exemplo:

```
<Item Descricao = “Brasil”>

.............. </Item>
```

Figura VII.6 GraphML – Exemplo de utilização do elemento “Item”.

* Elemento Valor

a. Definição :

O elemento *Valor* descreve o valor que cada item do gráfico irá armazenar.. Não possui atributos e seu conteúdo é somente texto contendo o valor.

b. Estrutura

Este objeto possui texto ou os fragmentos"EX" e "EY". Sua estrutura pode ser observada na Figura VII.7.

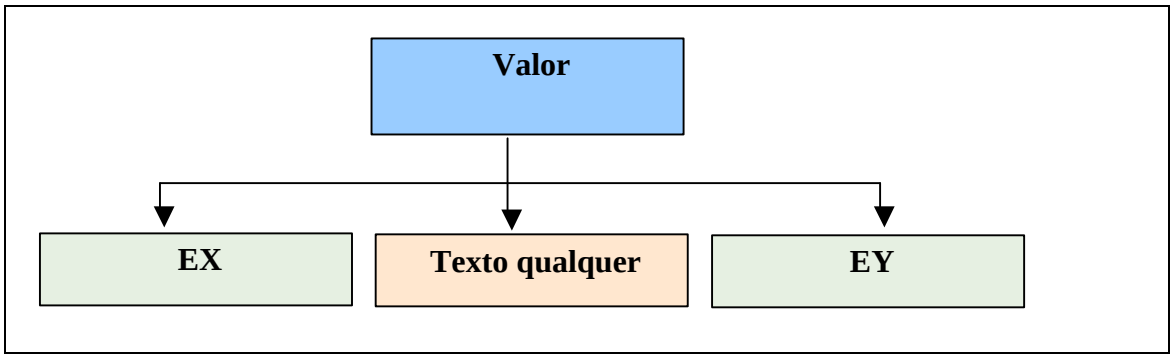


Figura VII.7 GraphML – Estrutura do elemento "Valor".

c. Sintaxe :

```
<!ELEMENT Valor (#PCDATA |(EX,EY) )>
```

Figura VII.8 GraphML – Sintaxe do elemento "Valor".

d. Exemplo:

```
<Valor>

85

</Valor> OU

<Valor>

<EX> ... </EX>

<EY> ... </EY> </Valor>
```

Figura VII.9 GraphML – Exemplo de utilização do elemento "Valor".

* Elemento EX, e EY

a. Definição :

Para o caso dos gráficos de pontos e linhas, é necessário que se mantenham os valores para as coordenadas de um ponto, ou seja, um valor para "x" e um valor para "y". Assim, dois elementos fazem esta distinção: EX e EY. Estes elementos não contêm atributos, e seus conteúdos são somente texto contendo os valores para "x" e para "y", respectivamente.

b. Estrutura

Estes objetos não possuem fragmentos aninhados, apenas texto. Suas estruturas hierárquicas podem ser vistas na Figura VII.10.

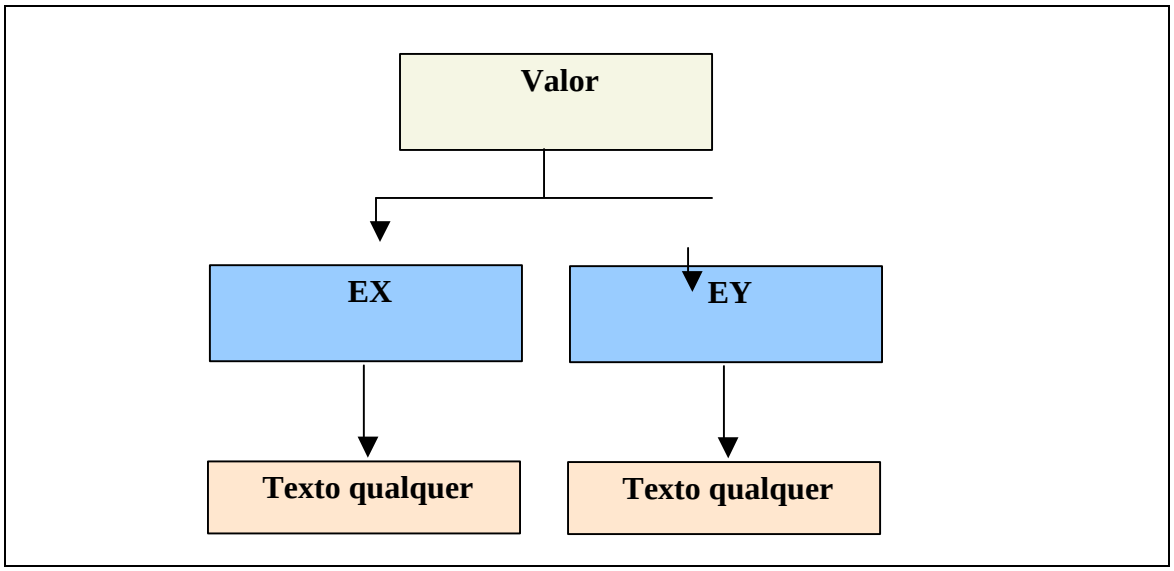


Figura VII.10 GraphML – Estrutura dos elementos "EX" e "EY".

c. Sintaxe :

```
<!ELEMENT EX (#PCDDATA)>

<!ELEMENT EY (#PCDDATA)>
```

Figura VII.11 GraphML – Sintaxe do elementos "EX" e "EY".

d. Exemplo

```
<Valor>

   <EX>  0  </EX>

   <EY> 10  </EY> </Valor>
```

Figura VII.12 GraphML – Exemplo de utilização dos elementos "EX" e "EY".

Segue, na Figura VII.13, um exemplo de um documento completo para representar um gráfico de pizza.

```
<Grafico Tipo="Pizza">
    <Item Titulo="Brasil" Cor="blue" Degradee="85">
        <Valor>
        85
        </Valor>
    </Item>
    <Item Titulo="Argentina" Cor="red" Degradee="10">
        <Valor>
        10
        </Valor>
    </Item>
    <Item Titulo="Uruguai" Cor="yellow" Degradee="5">
        <Valor>
        5
        </Valor>
   </Item>
</Grafico>
```

Figura VII.13 GraphML – Exemplo de utilização.

Anexo VIII

Utilizando o Gerenciador de Hipermídias

Está seção mostrará um exemplo de utilização do Gerenciador de Hipermídias.

VIII.1 Criando um novo projeto

Para criar um novo projeto de conteúdos, é necessário utilizar o botão “Novo”. O gerenciador de conteúdos suporta vários tipos de conteúdos, conforme mostrado na Figura VIII.1.

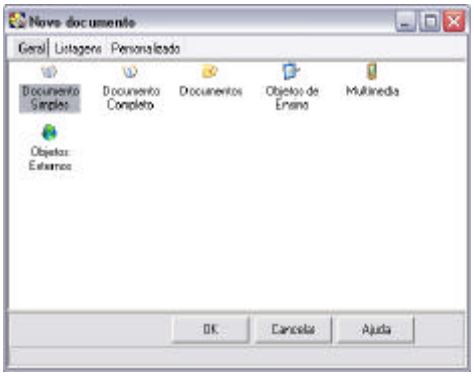


Figura VIII.1 Gerenciador de Hipermídias – Adicionando um novo documento.

VIII.2 Abrindo um projeto existente

Para abrir um projeto existente, é necessário clicar no botão “Abrir”. Uma janela será aberta, onde deve ser escolhido o arquivo XML, conforme Figura VIII.2.

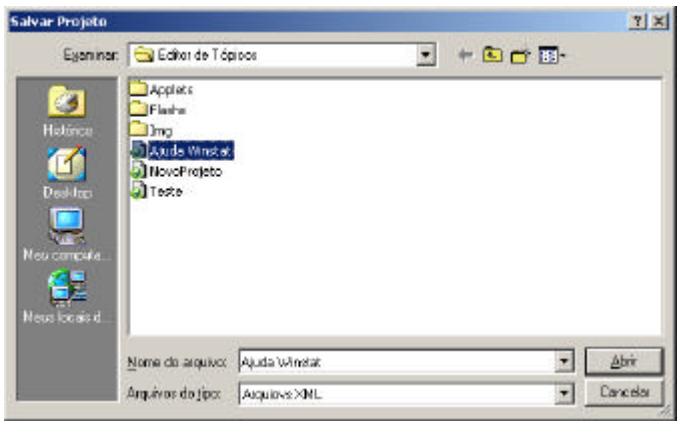


Figura VIII.2 Gerenciador de Hipermídias – Abrindo um projeto existente.

VIII.3 Salvando um projeto

Os botões “Salvar” e “Salvar como” servem, respectivamente, para salvar qualquer alteração em um documento, e salvar os dados em um novo arquivo. Qualquer modificação feita nos conteúdos deve ser salvada imediatamente antes de qualquer navegação pelo restante do conteúdo. A Figura VIII.3 ilustra este processo.

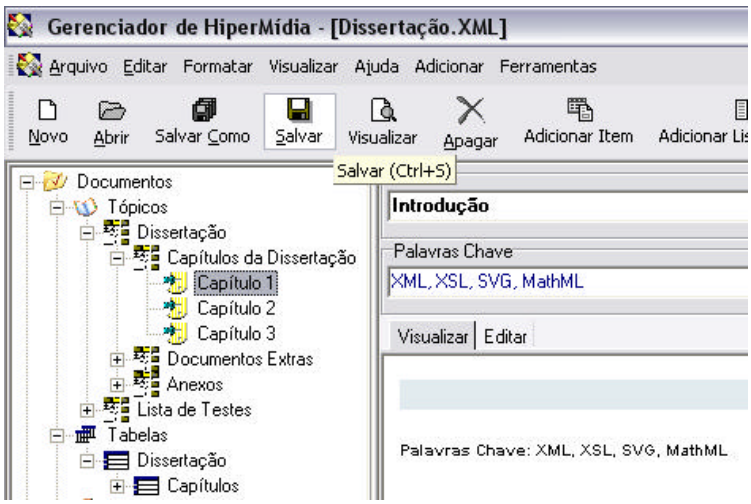


Figura VIII.3 Gerenciador de Hipermídias – Salvando um projeto.

VIII.4 Visualizando um documento

É possível visualizar os conteúdos editados e processados em HTML através do botão “Visualizar”.

Suponha o seguinte documento ilustrado na Figura VIII.4.

```
      O WinStat é um sistema de análise estatística que, dentre outras características,
      possibilita:
      <ul>
      <li>O gerenciamento de conjuntos de dados, utilizados tanto para entrada de dados como para
      armazenar saídas; </li>
      <li>Uma série de metodologias de análise de dados, incluindo análise de frequências, modelos
      lineares, modelos não lineares, correlação, análise gráfica, etc.;
      Cálculo de probabilidade, construção de tabelas de distribuições, gráficos de distribuições
       e geração de variáveis aleatórias; </li>
      <li>Álgebra de matrizes e vetores;</li>
      <li>Armazenamento de objetos de vários tipos: conjuntos de dados, matrizes,
      vetores e escalares;</li>
      <li>Ferramentas para gerenciamento de tabelas de bancos de dados, arquivos texto e saídas;
      Saídas em vários formatos: texto, html e planilha; </li>
      <li>Sistema de ajuda em html, possibilitando intercâmbio entre saídas, sistema
      de ajuda e material didático; </li>
      <li>Navegação totalmente no estilo Windows, aproveitando integralmente os recursos
      desse sistema operacional. </li>
      </ul>
      <BREAK/>
      O acesso a todas as possibilidades do sistema é feito a partir da Janela Principal:
      <REF Tipo = "Figura" ID = "Janela principal do WinStat"/>
```

Figura VIII.4 Gerenciador de Hipermídias – Visualizando um documento.

No documento acima existem tags HTML (*ul*, *li*) e também um script da linguagem acima proposta para fazer referência a uma figura. O gerenciador de Conteúdos possui um mecanismo de visualizar todo o documento acima de forma integrada (tags html, figuras, etc) em formato HTML. A visualização do documento acima, seria mostrada no aplicativo conforme a Figura VIII.5.

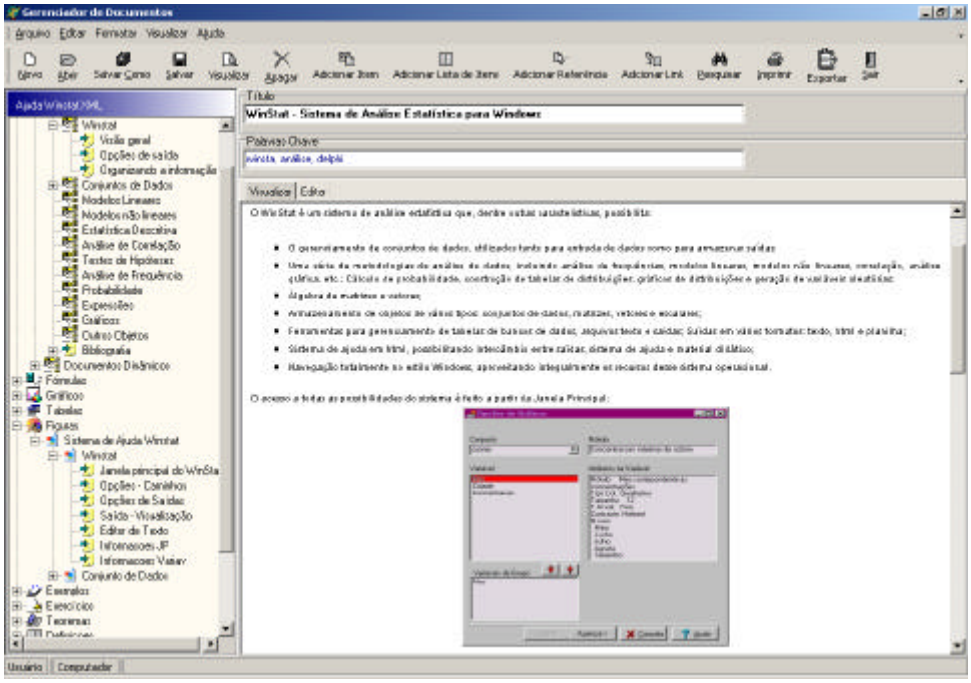


Figura VIII.5 Gerenciador de Hipermídias – Visualizando um documento.

VIII.5 Apagando um elemento

Para apagar um elemento específico basta selecionar o nó desejado e clicar no botão “Apagar”. Se o tipo do nó selecionado for *Item*, o elemento será apagado normalmente. Caso o tipo do nó selecionado for *Lista de Item*, o elemento, juntamente com todos os elementos descendentes, serão apagados. A Figura VIII.6 apresenta este processo.

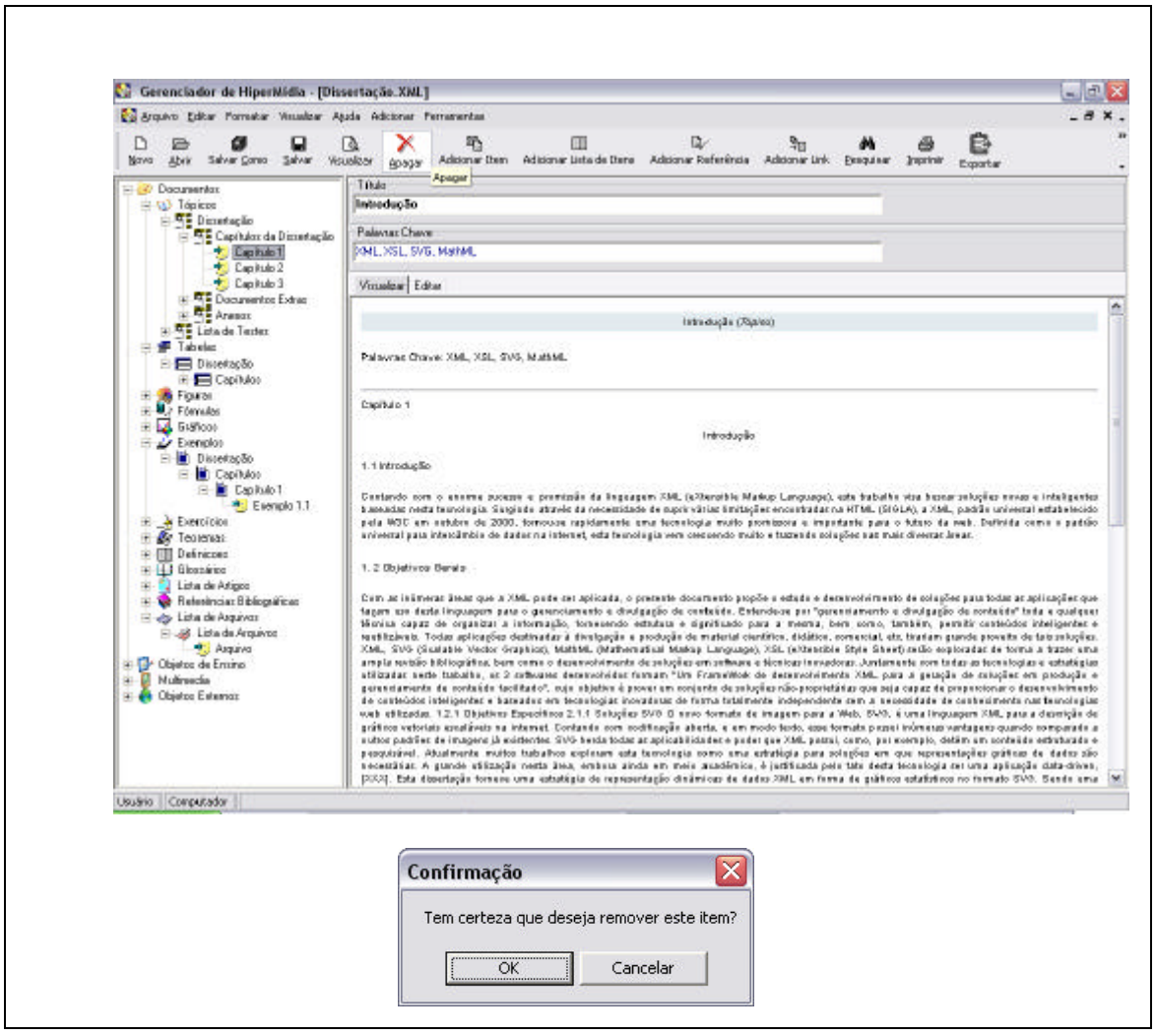


Figura VIII.6 Gerenciador de Hipermídias – Apagando um elemento.

VIII.6 Adicionando novos itens ao projeto

A cada novo item adicionado no documento, um novo nó XML é gerado fisicamente na base de dados. Supõe-se a geração de um novo documento do tipo “Lista de Recursos”. A Figura VIII.7 ilusta este processo.

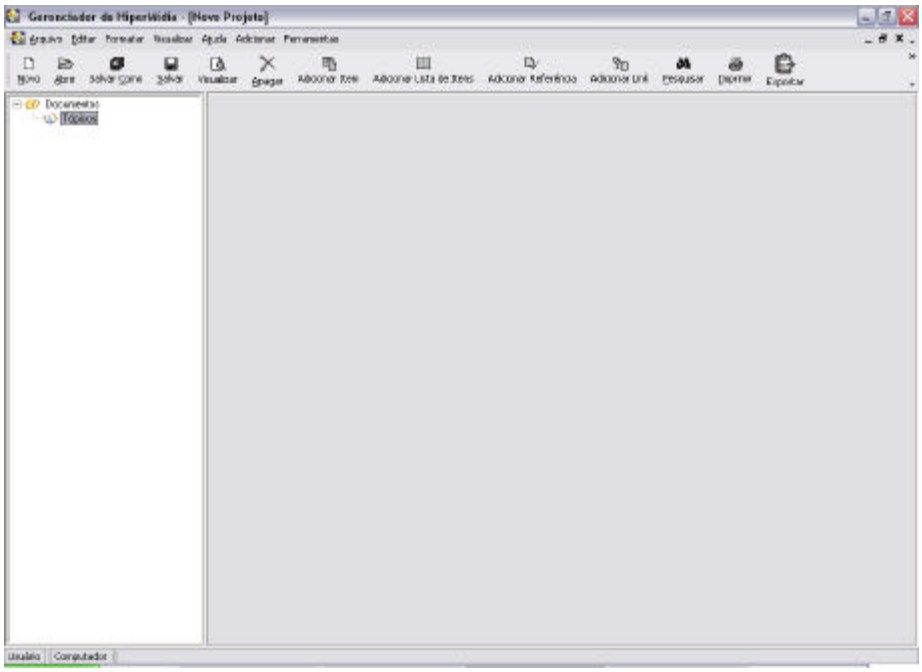


Figura VIII.7 Gerenciador de Hipermídias – Adicionando novos itens ao projeto 1.

Supõe-se, agora, a necessidade de um documento com a seguinte hierarquia exposta na Figura VIII.8.

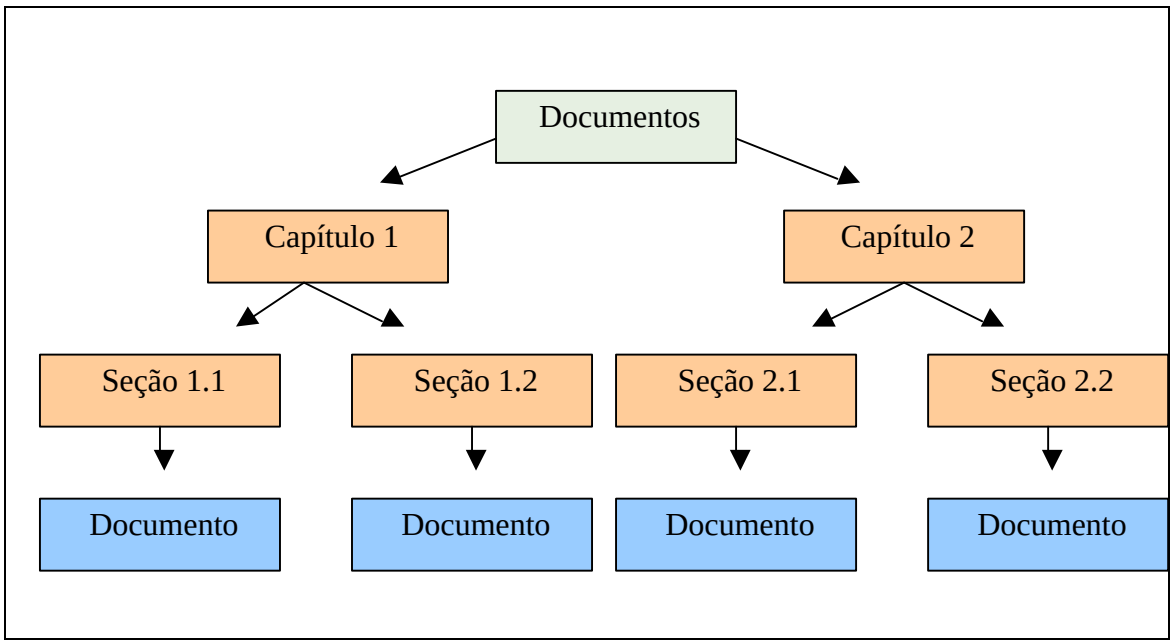


Figura VIII.8 Gerenciador de Hipermídias – Adicionando novos itens ao projeto 2.

Para gerar a estrutura apresentada na Figura VIII.8, no Gerenciador de Conteúdos a seguinte seqüência de ações será necessária :

1. Alterar o nome do único elemento (*Lista de Recursos*) para “Capítulo 1”;
2.
    a. Selecionar o elemento cujo nome é “Objetos de Dados”;
    b. Criar um novo elemento “Lista de Recursos” clicando em “Adicionar Lista de Itens”;
    c. Trocar o nome do novo elemento para “Capítulo 2”;
3.
    a. Selecionar o elemento cujo nome é “Capítulo 1”;
    b. Criar um novo elemento “Lista de Recursos” clicando em “Adicionar Lista de Itens”;
    c. Trocar o nome do novo elemento para “Seção 1.1”;
4.
    a. Selecionar o elemento cujo nome é “Seção 1.1”;
    b. Criar um novo elemento “Recurso” clicando em “Adicionar Item”;
    c. Trocar o nome do novo elemento para “Documento”;

5.
    a. Selecionar o elemento cujo nome é “Capítulo 1”;
    b. Criar um novo elemento “Lista de Recursos” clicando em “Adicionar Lista de Itens”;
    c. Trocar o nome do novo elemento para “Seção 1.2”;
6.
    a. Selecionar o elemento cujo nome é “Seção 1.2”;
    b. Criar um novo elemento “Recurso” clicando em “Adicionar Item”;
    c. Trocar o nome do novo elemento para “Documento”;
7.
    a. Selecionar o elemento cujo nome é “Capítulo 2”;
    b. Criar um novo elemento “Item” clicando em “Adicionar Item”;
    c. Trocar o nome do novo elemento para “Seção 2.1”;
8.
    a. Selecionar o elemento cujo nome é “Seção 2.1”;
    b. Criar um novo elemento “Recurso” clicando em “Adicionar Item”;
    c. Trocar o nome do novo elemento para “Documento”;
9.
    a. Selecionar o elemento cujo nome é “Capítulo 2”;
    b. Criar um novo elemento “Item” clicando em “Adicionar Item”;
    c. Trocar o nome do novo elemento para “Seção 2.2”;
10.
    a. Selecionar o elemento cujo nome é “Seção 2.2”;
    b. Criar um novo elemento “Recurso” clicando em “Adicionar Item”;
    c. Trocar o nome do novo elemento para “Documento”.

O resultado obtido através da lista de ações especificadas acima é o que segue na Figura VIII.9.

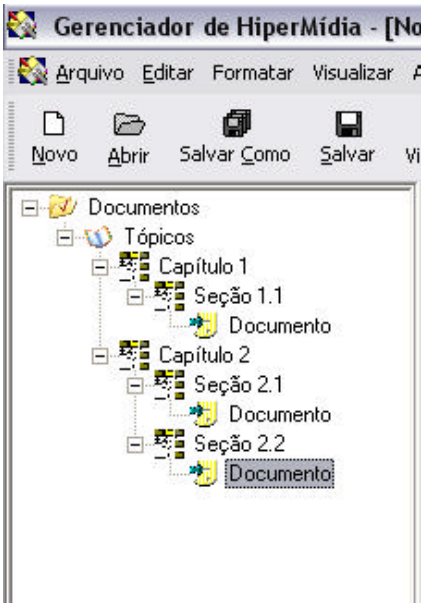


Figura VIII.9 Gerenciador de Hipermídias – Adicionando novos itens ao projeto 3.

VIII.7 Pesquisando elementos

O Gerenciador de Conteúdos possui um poderoso mecanismo para pesquisar qualquer tipo de conteúdo (recurso, fórmula, tabela, etc) . Todas as pesquisas podem ser feitas por palavras – chave ou por nome. Pode-se definir o escopo da pesquisa para todos os documentos ou para algum elemento específico. Os resultados são mostrados numa tabela que direciona para o elemento desejado, ou seja, um duplo-clique no elemento que está na tabela de resultados desloca o usuário para a posição específica na árvore de elementos em que o elemento se encontra. A Figura VIII.10 mostra a tela de pesquisas.

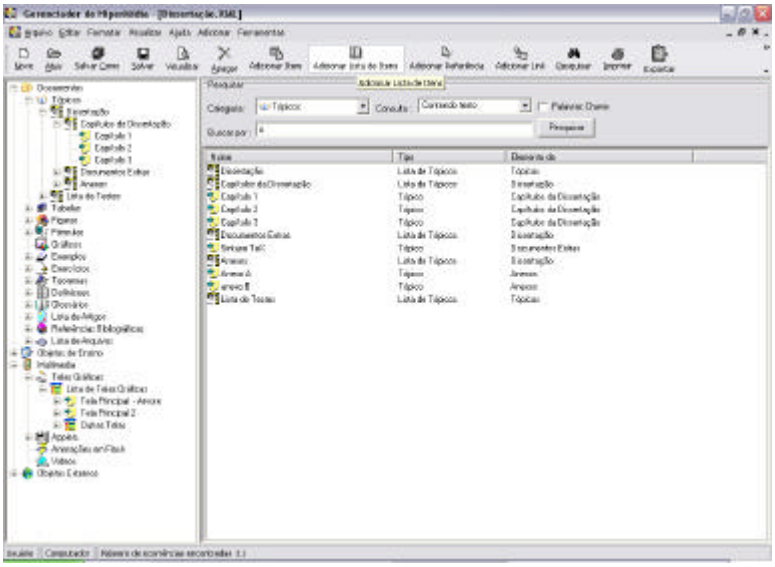


Figura VIII.10 Gerenciador de Hipermídias – Pesquisando elementos 1.

É possível restringir a busca através das opções : “Iniciando com”,”Exatamente igual” e “Contendo Texto” , conforme Figura VIII.11.

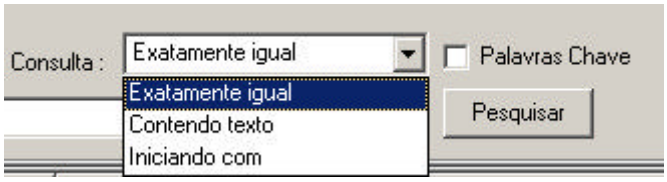


Figura VIII.11 Gerenciador de Hipermídias – Pesquisando elementos 2.

Pode-se filtrar a pesquisa cujo escopo define um determinado tipo de elemento, conforme Figura VIII.12.

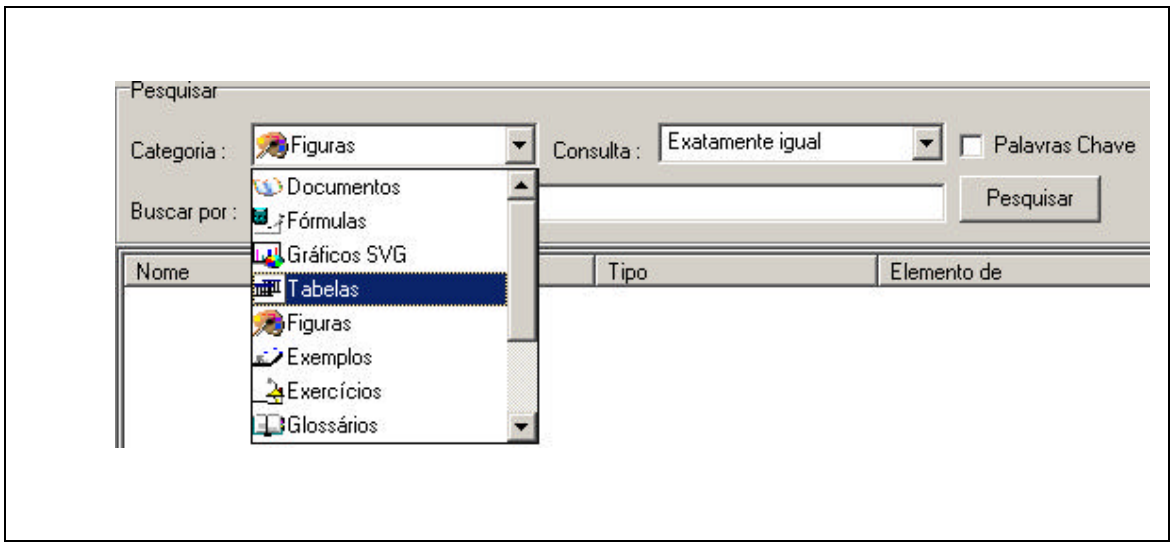


Figura VIII.12 Gerenciador de Hipermídias – Pesquisando elementos 3.

Após a escolha da categoria e do tipo de consulta, basta preencher o campo “Buscar por” e clicar no botão “Pesquisar”. Todos os elementos que satisfazem a configuração de pesquisa escolhida serão listados em uma tabela. Segue, na Figura VIII.13, um exemplo de configuração de pesquisa.

```
Categoria : Todos;
Consulta : Contendo Texto;
      Palavras Chave : Não
Buscar por : a
```

Figura VIII.13 Gerenciador de Hipermídias – Pesquisando elementos.

Assim, todos os elementos na base XML, cujo nome contém a letra “a”, serão adicionados na listagem de resultados. A Figura VIII.14 mostra o resultado da pesquisa da Figura VIII.13.

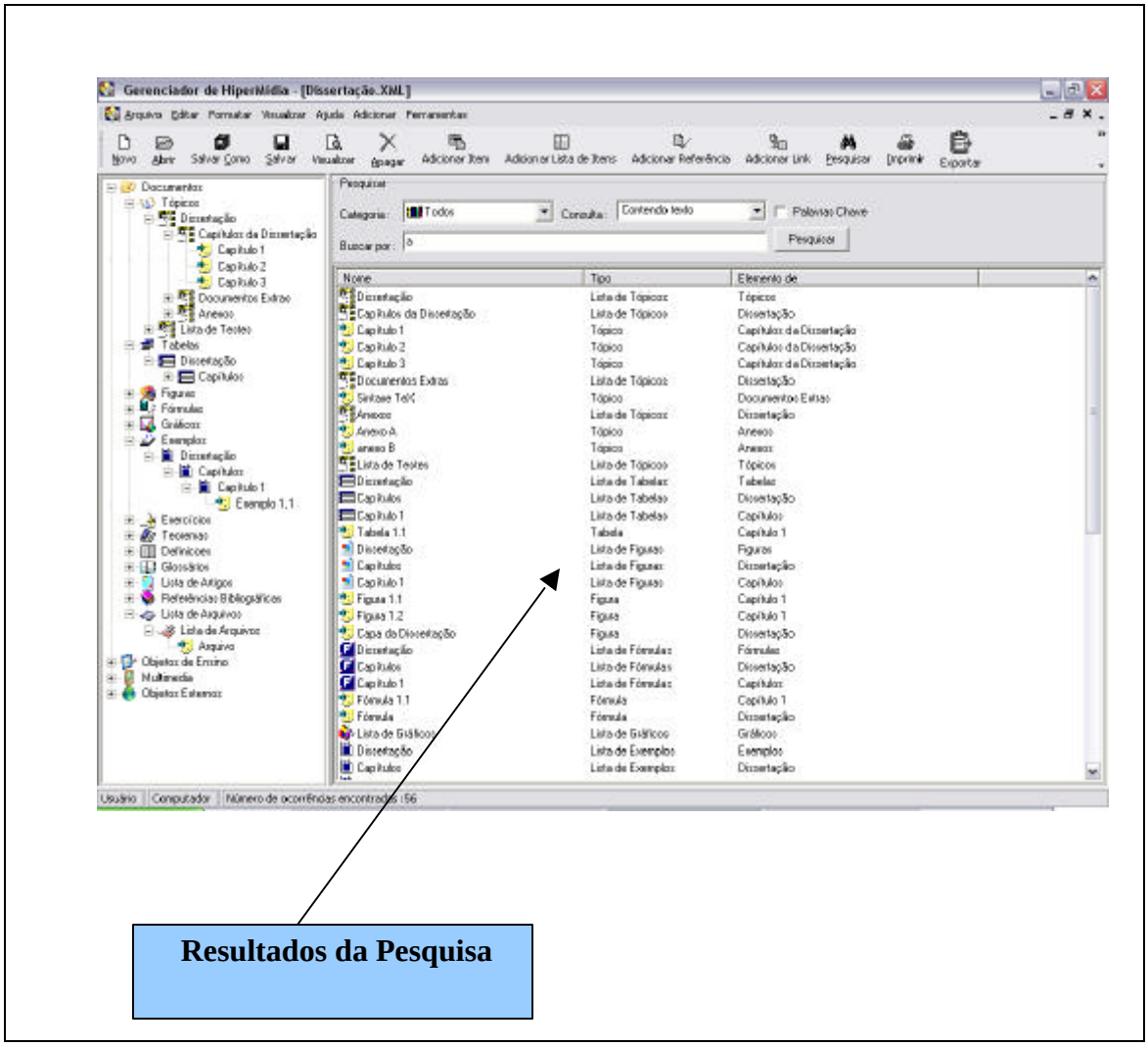


Figura VIII.14 Gerenciador de Hipermídias – Pesquisando elementos 4.

Anexo IX

Utilizando o Editor de Fórmulas

IX.1 Introdução

Nesta seção, será mostrado um exemplo de utilização da ferramenta. Supõe-se a necessidade de obtenção de código MathML relativo à seguinte fórmula : $\sum_{b}^{a} x ? y$

IX.2 Produção do código MathTeX/LaTeX

* Na paleta de funções, preencha os campos necessários para a função somatório, conforme Figura IX.1.

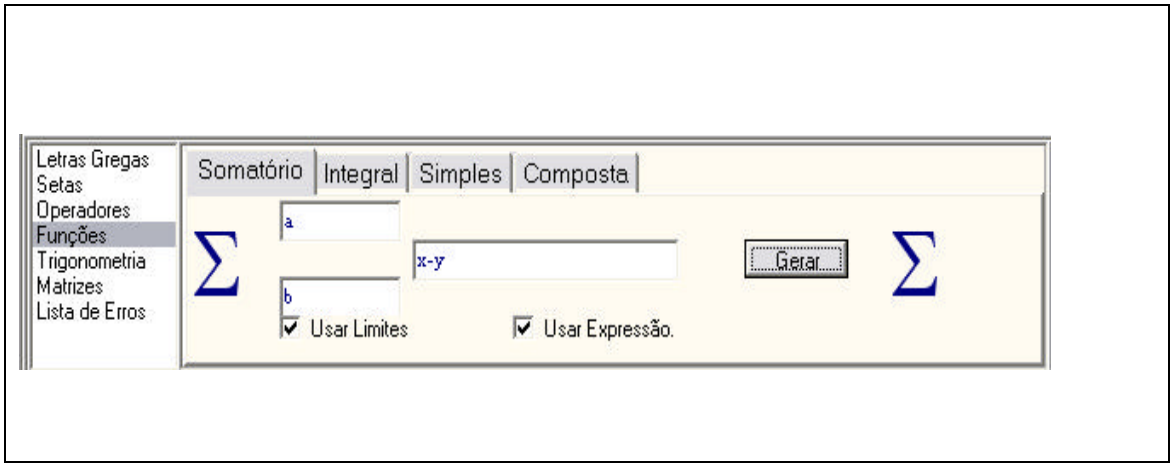


Figura IX.1 Editor de Fórmulas – Utilizando paletas de símbolos.

Após o preenchimento, clique no botão gerar. O resultado obtido será o exposto na Figura IX.2.

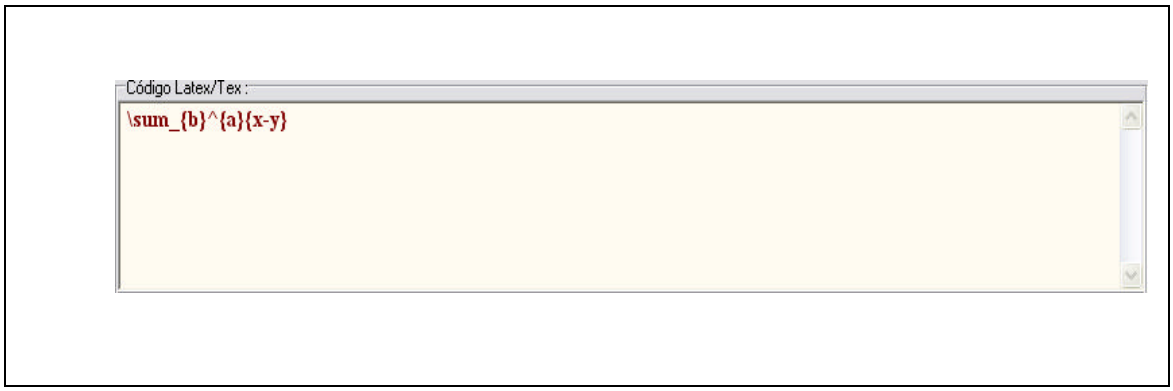


Figura IX.2 Editor de Fórmulas – Editando Código MathTeX.

* Após a produção de código, é necessário iniciar o processamento de interpretação e conversão. Para isso, clica-se no botão gerar, conforme Figura IX.3.

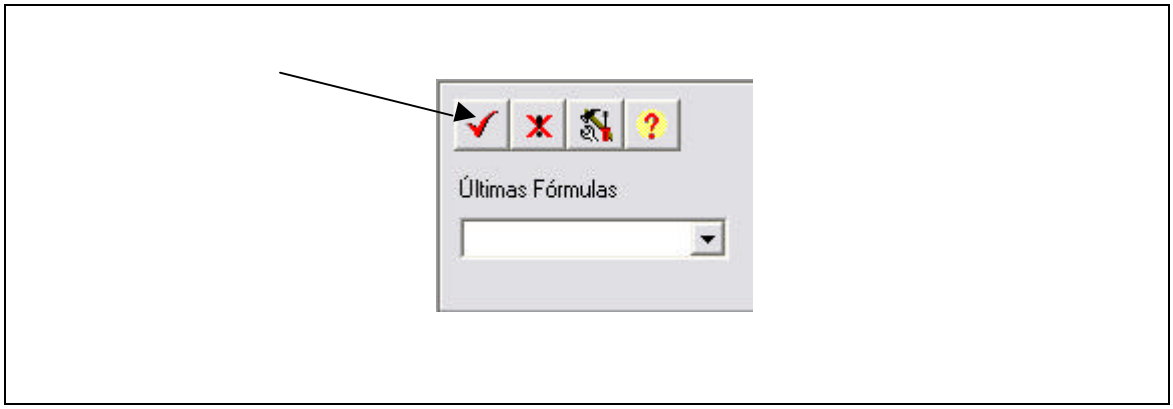


Figura IX.3 Editor de Fórmulas – Convertendo código MathTeX em MathML.

Após isso, caso o código esteja sintaticamente correto, conforme a sintaxe da linguagem LaTeX, os seguintes passos ocorrerão :

a. Surgirá a mensagem apresentada na Figura IX.4.

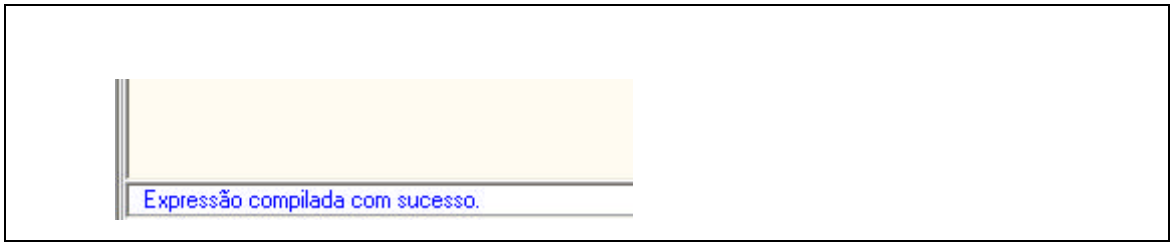


Figura IX.4 Editor de Fórmulas – Resultado da conversão.

b.O código MathML correspondente surgirá no editor de código mathml, conforme Figura IX.5.

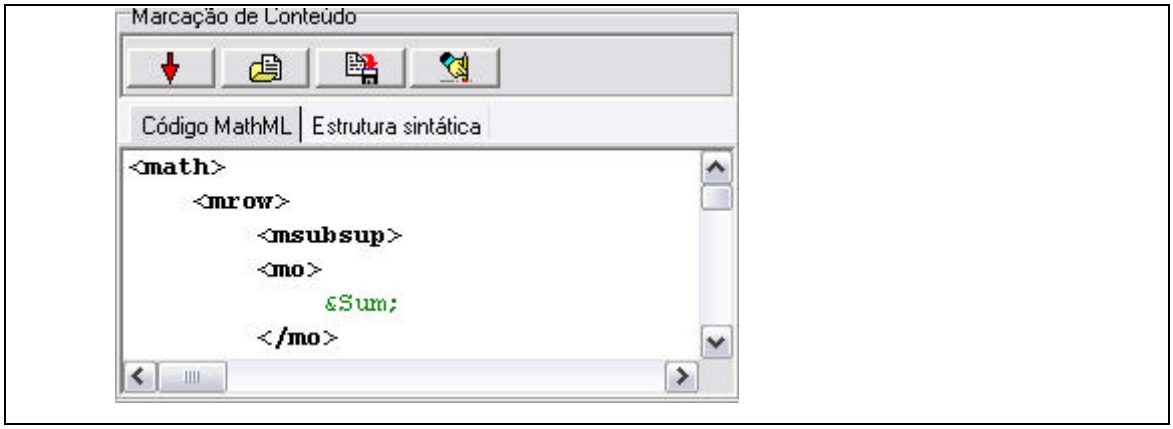


Figura IX.5 Editor de Fórmulas – Código MathML gerado.

c.Surgirá a visualização correspondente ao código MathML gerado (Figura IX.6).

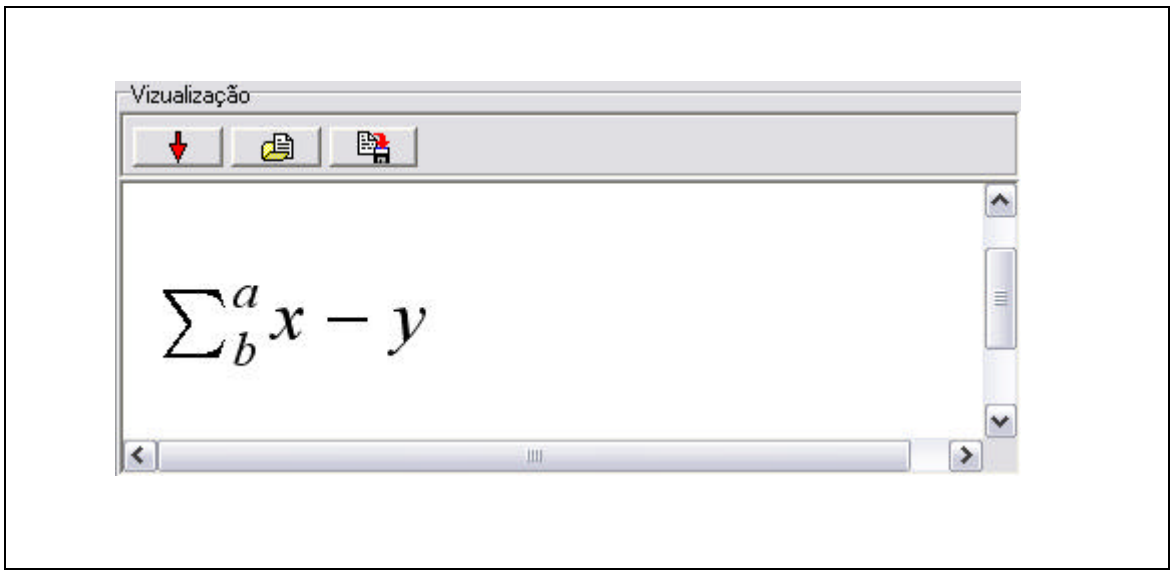


Figura IX.6 Editor de Fórmulas – Visualização do código MathML gerado.

Anexo X

Folhas de Estilos para GraphML

X.1 Gráfico de Barras

```
<?xml version="1.0" ?>

- <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msxml="urn:schemas-microsoft-com:xslt"
xmlns="http://www.w3.org/2000/svg-20000303-stylable"
xmlns:math="urn:schemas-cagle-com:math" exclude-result-prefixes="math
msxml" version="1.0">

  <xsl:output method="xml" media-type="image/svg" omit-xml-
declaration="yes" indent="yes" />

  <msxml:script language="JavaScript" implements-prefix="math">function
max(numSet){ var maxVal=0; for (var index=0;index !=
numSet.length;index++){ node=numSet[index];
nodeVal=parseFloat(node.text); if (nodeVal > maxVal){ maxVal=nodeVal;
} } return maxVal; } function min(numSet,valMax){ var minVal=valMax;
for (var index=0;index != numSet.length;index++){ node=numSet[index];
nodeVal=parseFloat(node.text); if (nodeVal < minVal){ minVal=nodeVal; }
} return minVal; }</msxml:script>

  <xsl:param name="maxValue" select="math:max(//ITEM/PORCENTAGEM)" />

  <xsl:param name="minValue"
select="math:min(//ITEM/PORCENTAGEM,$maxValue)" />

  <xsl:param name="Valor" select="//ITEM/PORCENTAGEM/@VALOR)" />

- <xsl:template match="/">

- <svg width="500" height="500" viewBox="0 0 1200 1300" left="0" top="0">

- <defs>

- <linearGradient id="boxGradient" gradientTransform="rotate(90)
translate(0 -1000) scale({1000 div 1300})">

  <stop offset="0%" style="stop-color:green" />

  <stop offset=".5" style="stop-color:lightGreen" />

  <stop offset="100%" style="stop-color:white" />

  </linearGradient>
```

```
- <linearGradient id="hotGradient" gradientTransform="rotate(90)
translate(0 -1000) scale({1000 div 1300})">

  <stop offset="0%" style="stop-color:red" />

  <stop offset=".5" style="stop-color:yellow" />

  <stop offset="100%" style="stop-color:white" />

  </linearGradient>

- <linearGradient id="coldGradient" gradientTransform="rotate(90)
translate(0 -1000) scale({1000 div 1300})">

  <stop offset="0%" style="stop-color:blue" />

  <stop offset=".5" style="stop-color:lightBlue" />

  <stop offset="100%" style="stop-color:white" />

  </linearGradient>

  </defs>

- <g transform="translate(20 100)">

  <xsl:apply-templates select="GRAFICO" />

  </g>

  </svg>

  </xsl:template>

- <xsl:template match="GRAFICO">

  <xsl:apply-templates select="ITEM" />

  </xsl:template>

- <xsl:template match="ITEM">

  <xsl:variable name="cx" select="floor(1000 div count(../ITEM))" />

  <xsl:variable name="cy" select="floor( $valor* 1000 div $maxValue )" />

  <xsl:variable name="valoritem" select="PORCENTAGEM" />

- <g transform="translate({position()*$cx - $cx} {1000-$cy})">

- <xsl:choose>
```

```
- <xsl:when test="$valoritem = $maxValue">

  <rect width="{$cx}" height="{$cy}"
style="fill:url(#hotGradient);stroke:black;stroke-width:2;" />

  </xsl:when>

- <xsl:when test="$valoritem = $minValue">

  <rect width="{$cx}" height="{$cy}"
style="fill:url(#coldGradient);stroke:black;stroke-width:2;" />

  </xsl:when>

- <xsl:otherwise>

  <rect width="{$cx}" height="{$cy}"
style="fill:url(#boxGradient);stroke:black;stroke-width:2;" />

  </xsl:otherwise>

  </xsl:choose>

- <text style="color:black;font-size:40;" y="-10">

  <xsl:value-of select="PORCENTAGEM" />

  </text>

- <g transform="translate({$cx div 3} {$cy + 24})">

- <g transform="rotate(30)">

- <text style="color:black;font-size:30;">

  <xsl:value-of select="@TITULO" />

  </text>

  </g>

  </g>

  </g>

  </xsl:template>

  </xsl:stylesheet>
```

Figura X.1 Folhas de Estilos – Gráfico de Barras.

X.2 Gráfico de Pizza

```
<?xml version="1.0" ?>

- <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msxml="urn:schemas-microsoft-com:xslt"
xmlns="http://www.w3.org/2000/svg-20000303-stylable"
xmlns:math="urn:schemas-cagle-com:math" exclude-result-prefixes="math
msxml" version="1.0">

  <xsl:output method="xml" media-type="image/svg" omit-xml-
declaration="yes" indent="yes" />

- <xsl:template match="/">

- <svg width="500" height="500" viewBox="0 0 1200 1300" left="0" top="0">

- <g transform="translate(20 100)">

  <xsl:apply-templates select="GRAFICO" />

  </g>

  </svg>

  </xsl:template>

- <xsl:template match="GRAFICO">

- <g transform="translate(0 0)">

- <text style="font-size:50;color:black;font-family:Verdana">

  <xsl:value-of select="@TITULO" />

  </text>

  </g>

  <xsl:apply-templates select="ITEM" />

  </xsl:template>

- <xsl:template match="ITEM">

  <xsl:variable name="raio" select="150" />

  <xsl:variable name="x1" select="EX" />

  <xsl:variable name="x2" select="EX1" />
```

```
  <xsl:variable name="y1" select="EY" />
  <xsl:variable name="y2" select="EY1" />
  <xsl:variable name="valor" select="VALOR" />
  <xsl:variable name="cor" select="@COR" />
- <!--
        <xsl:variable name = "movex"/>
        <xsl:variable name = "movey"/>




    <xsl:if test="$valor < 100">
     <xsl:if test="$x1 < 0">
           <xsl:if test="$x2 < 0">
                 <xsl:if test="$y1 < 0">
                       <xsl:if test="$y2 < 0">



                       <xsl:if>


                  <xsl:if>


           <xsl:if>


        <xsl:if>


    <xsl:if>
```

```
  -->

- <xsl:choose>

- <xsl:when test="$valor = 100">

  <path id="fatia{position()}" d="M200,200 l{$x1},{-$y1}
a{$raio},{$raio} 0 0,0 -300,{-$y2} a{$raio},{$raio} 0 0,0 300,{-$y2} z"
style="fill:{$cor}; stroke:black; stroke-width:3" />

  </xsl:when>

- <xsl:when test="$valor>=50">

  <path id="fatia{position()}" d="M200,200 l{$x1},{-$y1}
a{$raio},{$raio} 0 1,0 {$x2},{-$y2} z" style="fill:{$cor}; stroke:black;
stroke-width:3" />

  </xsl:when>

- <xsl:otherwise>

  <path id="fatia{position()}" d="M200,200 l{$x1},{-$y1}
a{$raio},{$raio} 0 0,0 {$x2},{-$y2} z" style="fill:{$cor}; stroke:black;
stroke-width:3" />

  </xsl:otherwise>

  </xsl:choose>

- <g transform="translate(500 {position()*40 - 60})">

- <text dx="-80" dy="20" style="font-size:30;color:black;font-
family:Arial">

  <xsl:value-of select="VALOR" />

  %

  </text>

  <rect width="20" height="20" style="fill:{$cor}; stroke:black; stroke-
width:3" />

- <text dx="30" dy="20" style="font-size:30;color:black;font-
family:Arial">

  <xsl:value-of select="@TITULO" />

  </text>
```

```
  </g>

- <text x="200" y="800" style="font-size:40;color:black;font-
family:Arial" visibility="hidden">

  <set attributeName="visibility" to="visible"
begin="fatia{position()}.mouseover" />

  <set attributeName="visibility" to="hidden"
begin="fatia{position()}.mouseout" />

  <xsl:value-of select="@TITULO" />

  <xsl:value-of select="VALOR" />

  %

  </text>

  </xsl:template>

  </xsl:stylesheet>
```

Figura X.2 Folhas de Estilos – Gráfico de Pizza.

X.3 Gráfico de Pontos

```
<?xml version="1.0" ?>

- <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msxml="urn:schemas-microsoft-com:xslt"
xmlns="http://www.w3.org/2000/svg-20000303-stylable"
xmlns:math="urn:schemas-cagle-com:math" exclude-result-prefixes="math
msxml" version="1.0">

  <xsl:output method="xml" media-type="image/svg" omit-xml-
declaration="yes" indent="yes" />

- <xsl:template match="/">

- <svg width="500" height="500" viewBox="0 0 1200 1300" left="0" top="0">

  <xsl:apply-templates select="GRAFICO" />

  </svg>

  </xsl:template>

- <xsl:template match="GRAFICO">

  <xsl:apply-templates select="ITEM" />

  </xsl:template>

- <xsl:template match="ITEM">

  <xsl:variable name="cx" select="EX" />

  <xsl:variable name="cy" select="EY" />

  <xsl:variable name="abscx" select="ABSEX" />

  <xsl:variable name="abscy" select="ABSEY" />

- <g style="color:green">

  <line x1="100" y1="100" x2="100" y2="1000" />

  <line x1="100" y1="1000" x2="1000" y2="1000" />

  <line x1="100" y1="100" x2="100" y2="150" transform="rotate(45 100 100)"
/>

  <line x1="100" y1="100" x2="100" y2="150" transform="rotate(-45 100
100)" />
```

```
  <line x1="950" y1="1000" x2="1000" y2="1000" transform="rotate(45 1000
1000)" />

  <line x1="950" y1="1000" x2="1000" y2="1000" transform="rotate(-45 1000
1000)" />

  </g>

- <g transform="translate({$abscx+100} {1000-$abscy})">

  <circle r="0.2cm" style="fill:red; stroke:blue; stroke-width:0.1cm" />

- <text style="color:black;font-size:20;" x="{-($abscx+50)}">

  <xsl:value-of select="EY" />

  </text>

- <text style="color:black;font-size:20;" y="{$abscy+50}">

  <xsl:value-of select="EX" />

  </text>

  </g>

  </xsl:template>

  </xsl:stylesheet>
```

Figura X.3 Folhas de Estilos – Gráfico de Pontos.

X.4 Gráfico de Linhas

```
<?xml version="1.0" ?>

- <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msxml="urn:schemas-microsoft-com:xslt"
xmlns="http://www.w3.org/2000/svg-20000303-stylable"
xmlns:math="urn:schemas-cagle-com:math" exclude-result-prefixes="math
msxml" version="1.0">

  <xsl:output method="xml" media-type="image/svg" omit-xml-
declaration="yes" indent="yes" />

  <msxml:script language="JavaScript" implements-prefix="math">function
max(numSet){ var maxVal=0; for (var index=0;index !=
numSet.length;index++){ node=numSet[index];
nodeVal=parseFloat(node.text); if (nodeVal > maxVal){ maxVal=nodeVal;
} } return maxVal; } function min(numSet,valMax){ var minVal=valMax;
for (var index=0;index != numSet.length;index++){ node=numSet[index];
nodeVal=parseFloat(node.text); if (nodeVal < minVal){ minVal=nodeVal; }
} return minVal; }</msxml:script>

  <xsl:param name="maxValueX" select="math:max(//ITEM/EX)" />

  <xsl:param name="maxValueY" select="math:max(//ITEM/EY)" />

- <xsl:template match="/">

- <svg width="500" height="500" viewBox="0 0 1200 1300" left="0" top="0">

- <defs>

- <linearGradient id="boxGradient" gradientTransform="rotate(90)
translate(0 -1000) scale({1000 div 1300})">

  <stop offset="0%" style="stop-color:green" />

  <stop offset=".5" style="stop-color:lightGreen" />

  <stop offset="100%" style="stop-color:white" />

  </linearGradient>

- <linearGradient id="hotGradient" gradientTransform="rotate(90)
translate(0 -1000) scale({1000 div 1300})">

  <stop offset="0%" style="stop-color:red" />

  <stop offset=".5" style="stop-color:yellow" />

  <stop offset="100%" style="stop-color:white" />
```

```
  </linearGradient>
- <linearGradient id="coldGradient" gradientTransform="rotate(90)
translate(0 -1000) scale({1000 div 1300})">
  <stop offset="0%" style="stop-color:blue" />
  <stop offset=".5" style="stop-color:lightBlue" />
  <stop offset="100%" style="stop-color:white" />
  </linearGradient>
  </defs>
- <g transform="translate(20 100)">
  <xsl:apply-templates select="GRAFICO" />
  </g>
  </svg>
  </xsl:template>
- <xsl:template match="GRAFICO">
- <g style="color:green">
  <line x1="100" y1="100" x2="100" y2="1000" />
  <line x1="100" y1="1000" x2="1000" y2="1000" />
  </g>
  <xsl:apply-templates select="ITEM" />
  </xsl:template>
- <xsl:template match="ITEM">
  <xsl:variable name="x1" select="EX" />
- <!--
Ponto atual calculado
  -->
  <xsl:variable name="y1" select="EY" />
```

```
- <!--
Ponto atual calculado
  -->
  <xsl:variable name="x2" select="EX1" />
- <!--
Ponto anterior calculado
  -->
  <xsl:variable name="y2" select="EY1" />
- <!--
Ponto anterior calculado
  -->
  <xsl:variable name="absx" select="ABSEX" />
- <!--
Ponto relativo anterior
  -->
  <xsl:variable name="absy" select="ABSEY" />
- <!--
Ponto relativo anterior
  -->
- <g style="color:green">
  <line x1="{$x1+100}" y1="{1000-$y1}" x2="{$x2+100}" y2="{1000-$y2}" />
  </g>
- <g transform="translate({$x1+100} {1000-$y1})">
  <circle r="0.2cm" style="fill:black" />
- <text style="color:black;font-size:20;" x="{-($x1+50)}">
  <xsl:value-of select="ABSEY" />
```

```
  </text>
- <text style="color:black;font-size:20;" y="{$y1+50}">
  <xsl:value-of select="ABSEX" />
  </text>
  </g>
  </xsl:template>
  </xsl:stylesheet>
```

Figura X.4 Folhas de Estilos – Gráfico de Linhas.

REFERÊNCIAS BIBLIOGRÁFICAS

[ADO 03] ADOBE Systems Incorporated , 2003,
Disponível em : <http://www.adobe.com>.
Acesso em : 12-04-2002

[ADO 03b] ADOBE Systems Incorporated, **SVG Zone**, Adobe, 2003,
Disponível em :<http://www.adobe.com/svg>.
Acesso em :18-04-2002

[BER 03] BERNERS-Lee ,Tim, Dan Connolly, and Sandro Hawke, **Semantic Web Tutorial Using N3,** 2003,
Disponível em :
<http://www.w3.org/2000/10/swap/doc/>.
Acesso em : 07-01-2003

[BER 01] BERNERS-Lee, Tim; Hendler, James; Lassila, Ora. **The Semantic Web**. **A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities**. Scientific American, Issue 01/05/2001.
Disponível em :
<http://www.sciam.com/2001/0501issue/0501berners-lee.html.>.
Acesso em : 23-06-2003

[BRI 99] BRICKLEY, Dan, **Semantic Web History: Nodes and Arcs 1989-1999**, Initial version: 1999-11-12,
Disponível em :
<http://www.w3.org/1999/11/11-WWWProposal/ >.
Acesso em : 17-11-2002

[BUL 00] BULLWER , Lez; ***XML and SVG as an aid to Distance Learning***, MSc Information Technology, De Montfort University

2000,

Disponível em :

<http://www.bullwer.co.uk/lez/project.html>.

Acesso em : 02-02-2003

[CAG 01] CAGLE, Kurt, **Professional XSL**, Wrox, 2001.

[CHA 03] CHAVES ,Marcirio Silveira, **Uso de ontologias para gerenciamento e acesso a documentos na web**, Universidade do Vale do Rio dos Sinos – UNISINOS, 2003.

[CLA 03] CLARK, J, **Document Style Semantics and Specification Language (DSSSL)**, 2003,

Disponível em : <http://www.jclark.com/dsssl/>., ISO/IEC 10179

Acesso em : 15-07-2002

[COM 03] **COMPILERS.NET**, 2003,

Disponível em :

<http://www.compilers.net/>.

Acesso em : 10-05-2002

[COV 98] COVER, Robin, **XML and Semantic Transparency,**

November 24, 1998. Represents reworked text from an article ´ submitted in draft to Sun/OASIS.

Disponível em :

<http://www.oasisopen.org/cover/xmlAndSemantics.html>.

Acesso em : 28-10-2002

[CRE 01] CRESPO, Pedro; **HSBC Brasil - Integrando soluções de Internet e sistemas legado**, HSBC, 2001, XML Forum, 26 de Setembro, São Paulo e Rio de Janeiro.

Disponível em :< http://www.ibpinet.net/xmlforum/index.html >;

Acesso em : 03-03-2002

[DEA 02] DEACH, Stephen; **What Is XSL-FO and When Should I Use It?**, 2002 by Seybold Publications,

Disponível em :

<http://www.seyboldreports.com/TSR/free/0217/techwatch.html>

Acesso em : 21-11-2002

[DES 03] DESIGN Science, **Math on the Web**, Copyright © 1996-2003 Design Science,

Disponível em :

<http://www.mathtype.com/en/reference/webmath/>.

Acesso em : 02-08-2002

[DES 03b] DESIGN Science, **Strategies for Math on the Web**, Copyright 1996-2003 Design Science,

Disponível em :

<http://www.mathtype.com/en/reference/webmath/strategies.htm

Acesso em : 07-08-2002

[FEN 00] FENSEL, Dieter. **Ontologies: Silver Bullet for knowledge Management and Eletronic Commerce,** Copyright © 1997-2002 NEC Research Institute , Berlin.

[FER 02] FERGNANI, Alain , **Ontology dynamics for Semantic Web:the MOMIS approach**, Tese de Doutorado, Universita' Degli Studi di Modena e Reggio Emilia, Facoltà di Ingegneria – Sede di Modena, Corso di Laurea in Ingegneria Informatica.

[GEE 03] GEERTS, Guido L. , **XML Support for an Operational Enterprise Ontology**, University of Delaware,USA, European Conference Accounting Information Systems Website **ECAIS 2002**

[GEO 00] GEORGE, Randy ; **SVG - Scalable Vector Graphics**, GeoTechnologies, Inc, Disponível em <http://www.web-maps.com/svg-gis.html >. Acesso em : 02-07-2002

[GOL 94] Goldstein, Robert, F. ***A Manifesto for Adding SGML Intelligence to the World-Wide Web*** , WWW'94 conference, 1994, C.M. Sperberg-McQueen Robert F. Goldstein 15 Sep 1994, rev. 4 July 1995. Disponível em : <http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Autools/sperberg-mcqueen/sperberg.html >. Acesso em : 08-06-2002

[GRU 93] GRUBER, T. R.. **A Translation Approach to Portable Ontology Specifications**. Knowledge Acquisition, 5 199-200, 1993. KNOWLEDGE SYSTEMS LABORATORY Computer Science Department, Stanford University, Stanford, California.

[GUA 97] GUARINO, Nicola. **Understanding, Building and Using Ontologies.** A commentary to "*Using Explicit Ontologies in KBS Development*", by van Heijst, Schreiber, and Wielinga. International Journal of Human and Computer Studies , 46, 293-310, 1997. Disponível em

<http://ksi.cpsc.ucalgary.ca/KAW/KAW96/guarino/guarino.html

Acesso em : 20-04-2003

[GUS 01] GUSMAO, Cristine M. G.; **Proposta de um Modelo de Comunicação entre Sistemas de Informação em Saúde: INTERACTMed;** Grupo de Tecnologias da Informação em Saúde, , Tese de Doutorado, Universidade Federal de Pernambuco, desembro de 2001.

[HAR 01] HARDY ,Vincent , **Scalable Vector Graphics (SVG): An Executive Summary**, 2001, distribuído pela Sun Microsystems

Disponível em :

<http://wwws.sun.com/software/xml/developers/svg/>.

Acesso em : 04-01-2003

[HAM 02] HARMELEN, Frank Van; **Ontology-based Information** Visualisation, Fifth International Conference on Information Visualisation (IV'01). July 25 - 27, 2001 London, England

[HEF 01] HEFLIN, Jeffrey Douglas Heflin, **Towards The Semantic Web Knowledge Representation in a Dynamic, Distributed Environment**, Dissertation submitted to the Faculty of the Graduate School of the University of Maryland, College Park in partial fulfillment of the requirements for the degree of Doctor of Philosophy 2001,

Disponível em :

<http://www.cs.umd.edu/projects/plus/SHOE/pubs/heflin-thesis-orig.pdf>.

Acesso em : 12-03-2003

[HEN 03] HENDLER , James; Berners-Lee , Tim ; Miller , Eric ; **Integrating Applications on the Semantic Web,** Journal of

the Institute of Electrical Engineers of Japan, Vol 122(10), October, 2002, p. 676- 680,

Disponível em <http://www.w3.org/2002/07/swint>.

Acesso em : 05-12-2003

[IEE 03] IEEE Learning Technology Standards Committee (LTSC). **Draft Standard for Learning Object Metadata**, 2003.

Disponível em :

<http://ltsc.ieee.org/doc/wg12/LOM_1484_12_1_v1_

Final_Draft.pdf>.

Acesso em : 02-04-2003

[JAS 99] JASPER, Robert and USCHOLD, Mike. **A Framework for understanding and classifying ontology applications.** in KAW'99 Twelfth Workshop on Knowledge Acquisition, Modeling and Management. 1999. Voyager Inn, Banff, Alberta, Canada.

Disponível em :

<http://sern.ucalgary.ca/KSI/KAW/KAW99/papers/

Uschold2/final-ont-apn-fmk.pdf >.

Acesso em : 23-03-2003

[KOH 01] KOHLHASE, Michael ; **Presenting and Capturing Mathematics for the Web**, School of Computer Science Carnegie Mellon University, 2001, Tutorial publicado pela W3C.

Disponível em :

<http://www.w3.org/Math/Documents/mathml-tutorial.pdf>.

Acesso em : 13-10-2002

[LEN 02] LENZ, Evan; **What's New in XSLT 2.0**, April 10, 2002, artigo publicado na "*XML.com*" (www.xml.com).

Disponível em :

<http://www.xml.com/pub/a/2002/04/10/xslt2.html>.

Acesso em : 14-05-2002

[LEN 02b] LENZ, Evan; **What's New n XPath 2.0**, March 20, 2002, artigo publicado na xml.com

Disponível em :

<http://www.xml.com/pub/a/2002/03/20/xpath2.html>.

Acesso em : 22-06-2003

[LUC 02] LUCENA ,Carlos J. P. publicado em SEMISH 2002 XXIX Seminário Integrado de Software e Hardware Evento Integrante do XXII Congresso da SBC - SBC2002 ,

Disponível em :

<http://www.inf.ufsc.br/sbc2002/palestras/lucena.ppt>.

Acesso em : 27-03-2003

[MAR 01] MARTINS ,Werley Ribeiro ; **Servidor de Documentos XML Usando Java**, Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação, da Universidade de São Paulo USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Área de Ciências de Computação e Matemática Computacional. São Carlos Agosto de 2001.

[MOU 00] MOULTIS, Natanya Pitts; Kirk, Chery; **XML Black Book : Solução e Poder**, Makron Books, São Paulo, 2000.

[MUN 02] MUNOZ, Lydia, Oliveira Palazzo Jozé, **Uma Ontologia para Apoio na Administração de Conteúdo Educativo na Web**, Dissertação de mestrado,

Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre - RS – Brazil,

[NIE 97] NIELSEN , Jakob; **Effective Use of Style Sheets**, July 1, 1997, Disponível em : <http://www.useit.com/alertbox/9707a.html>. Acesso em : 11-06-2002

[SAN 03] SANTANCHE , André , **Explorando Linguagens de Markup Extensíveis na Construção de Sistemas de Educação Baseados na Web**, Anais do XIX Congresso Nacional da Sociedade Brasileira de Computação – Volume 1. Rio de Janeiro. pp. 39-55, 1999.

[SAX 03] SAXPROJECT.ORG, **SAX** , ***Simple API for XML***, 2003, Disponível em : <http://www.saxproject.org/>. Acesso em : 24-11-2002

[SEA 01] Sean B. Palmer, Semantic Web Agreement Group, 2001-06. Disponível em : <http://www.purl.org/net/isbp> Acesso em : 21-12-2002

[SEM 03] SEMANTICWEB.ORG, **Markup Languages and Ontologies**, 2003, Disponível em : <http://www.semanticweb.org/knowmarkup.html>. Acesso em : 02-06-2003

[SEM 03b] SEMANTICWEB.ORG, **The Semantic Web Community Portal**, Disponível em <http://www.semanticweb.org>. Acesso em : 13-05-2003

[SIL 01] SILVEIRA, Ricardo Azambuja, **Modelagem Orientada a Agentes Aplicada a Ambientes Inteligentes Distribuídos de Ensino: JADE *Java Agent framework for Distance learning***

Environments, Tese de Doutorado, Porto Alegre, 2001.

[TAR 03] TAROUCO, Liane Margarida Rockenbach; **Reusabilidade de objetos educacionais**, publicado no I Ciclo de Palestras Novas Tecnologias na Educação fev/2003, CINTED-UFRGS, fevereiro, 2003.

[THE 02] *THE OpenMath Society,* **Open Math**, *2002,*

Disponível em :

<http://www.openmath.org/cocoon/openmath//index.html>.

Acesso em : 07-04-2002

[TRA 01] TRAVERSA ,Eddie, **Scalable Vector Graphics: The Art is in the Code**, 2001, artigo publicado em webreference.com.

Disponível em :

<http://www.webreference.com/authoring/languages/svg/intro/ >.

Acesso em : 18-10-2002

[VAZ 02] VAZI, Maria Fernanda Rodrigues, **Padrões emergentes em ensino à distância**, The VII International Conference on Engineering and Technology Education was held in Santos, Brazil on March 17-20, 2002.

[W3C 95] W3C – *World Wide Wibe Consortium,* **Report on the W3C style sheet workshop**, Paris '95,

Disponível em :

<http://www.w3.org/Style/951106_Workshop/report1.html#clark

Acesso em : 12-07-2002

[W3C 98a] W3C – *World Wide Wibe Consortium,* **Using XSL and CSS together**,W3C Note, 11 September 1998,

Disponível em :

<http://www.w3.org/TR/NOTE-XSL-and-CSS>.

Acesso em : 18-07-2002

[W3C 98b] W3C – *World Wide Web Consortium,* **XSL Requirements Summary**, W3C, 1998,

Disponível em :

<http://www.w3.org/TR/WD-XSLReq >.

Acesso em : 02-02-2003

[W3C 99a] W3C – *World Wide Wibe Consortium,* **Historical Style Sheet proposals**, 1999,

Disponível em :

<http://www.w3.org/Style/History/ >.

Acesso em : 03-08-2002

[W3C 99b] W3C – *World Wide Wibe Consortium,* **Resource Description Framework (RDF) Model and Syntax Specification**, W3C Recommendation 22 February 1999,

Disponível em :

<http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>.

Acesso em : 11-04-2003

[W3C 99c] W3C – *World Wide Wibe Consortium,* **XML Path Language (XPath) Version 1.0**, W3C Recommendation, 16 November 1999,

Disponível em < http://www.w3.org/TR/xpath>.

Acesso em : 16-09-2002

[W3C 99d] W3C – *World Wide Wibe Consortium,* **XSL Transformations (XSLT) Version 1.0**, W3C Recommendation 16 November

1999,

Disponível em < http://www.w3.org/TR/xslt>.

Acesso em : 01-09-2002

[W3C 01a] W3C – *World Wide Wibe Consortium,* **DAML+OIL (March 2001) Reference Description**, W3C Note 18 December 2001,

Disponível em <http://www.w3.org/TR/daml+oil-reference>.

Acesso em : 23-01-2003

[W3C 01b] W3C – *World Wide Wibe Consortium,* **XQuery 1.0 and XPath 2.0 Data Model,** W3C Working Draft 20 December 2001,

Disponível em <http://www.w3.org/TR/2001/WD-query-datamodel-20011220/>.

Acesso em : 13-09-2002

[W3C 03a] W3C – *World Wide Wibe Consortium,* **Cascade Style Sheets**, W3C, 2003,

Disponível em : <http://www.w3.org/Style/CSS/ >.

Acesso em : 25-03-2002

[W3C 03b] W3C – *World Wide Wibe Consortium,***Comparison of SGML and XML**, W3C, 2003,

Disponível em :

<http://www.w3.org/TR/NOTE-sgml-xml-971215 >.

Acesso em : 01-04-2002

[W3C 03c] W3C – *World Wide Wibe Consortium,***Document Object Model (DOM)**, 1997-2003,

Disponível em : <http://www.w3.org/DOM/>.

Acesso em : 10-09-2002

[W3C 03d] W3C – *World Wide Wibe Consortium,* **Extensible Markup**

Language (XML), 1996-2003 ,

Disponível em : <http://www.w3.org/XML/>.

Acesso em : 27-06-2002

[W3C 03e] W3C – *World Wide Wibe Consortium,* **HyperText Markup Language (HTML),** 1995-2003,

Disponível em :

<http://www.w3.org/MarkUp/>.

Acesso em : 22-03-2002

[W3C 03f] W3C – *World Wide Wibe Consortium,* **MathML – *MatheMatical Marckup Language***, W3C, 2003,

Disponível em :

< http://www.w3.org/Math/ >.

Acesso em : 21-05-2002

[W3C 03g] W3C – *World Wide Wibe Consortium,* **OWL Web Ontology Language Overview** , W3C Working Draft 31 March 2003,

Disponível em : <http://www.w3.org/TR/owl-features/>.

Acesso em : 17-02-2003

[W3C 03h] W3C – *World Wide Wibe Consortium,* **Scalable Vector Grafics**, 2003,

Disponível em :

<http://www.w3.org/Graphics/SVG/Overview.htm8>.

Acesso em : 10-08-2002

[W3C 03i] W3C – *World Wide Wibe Consortium,* **Semantic Web**,

Disponível em <http://www.w3.org/2001/sw/> 1994-2003.

Acesso em : 13-06-2003

[W3C 03j] W3C – *World Wide Wibe Consortium,* **The Extensible**

Stylesheet Language Family (XSL), 2003,

Disponível em <http://www.w3.org/Style/XSL/>.

Acesso em : 04-09-202

[W3C 03k] W3C – *World Wide Wibe Consortium,* **XSL Transformations (XSLT) Version 2.0**, W3C Working Draft 2 May 2003,

Disponível em : <http://www.w3.org/TR/xslt20/>.

Acesso em : 24-09-2002

[W3C 03l] W3C – *World Wide Wibe Consortium* **, Web-Ontology**, 2003,

Disponível em :

<http://www.w3.org/2001/sw/WebOnt/>.

Acesso em : 15-12-2002

[W3C 03m] W3C – *World Wide Wibe Consortium,* **World Wide Web Consortium**, W3C, 2003,

Disponível em <http://www.w3c.org>.

Acesso em : 10-07-2002

[W3S 03] W3SCHOOLS, **XSL Tutorial**, 2003, tutorial publicado pela w3schools.com

Disponível em : <http://www.w3schools.com/xsl/>.

Acesso em : 22-07-2002

[WIL 95] WILKINS ,David R. ; **Getting Started with LaTeX** ,2nd Edition, Copyright David R. Wilkins 1995,

Disponível em :

<http://www.maths.tcd.ie/~dwilkins/LaTeXPrimer/ >.

Acesso em : 03-04-2002

[WIN 93] WINBLAD, Ann L., **Software Orientado ao Objeto**, Makron Books, São Paulo, 1993.

[WTR 00] WILSON , Tracey; **XSLT & XPath Tutorial,** 2003, Disponível em : <http://www.vbxml.com/xsl/tutorials/intro/default.asp>. Acesso em : 11-09-2002

[WWW 03] WWW2003.ORG, **The Twelfth International World Wide Web Conference ,** 20-24 May 2003, Budapest, HUNGARY,