Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Projeto Orientado à Computação

Recuperação de Documentos para o Formato Eletrônico

Orientadora: Mariza A. S. Bigonha
Aluna: Renata Castro Carvalho

Relatório Final do Projeto Orientado
Área: Linguagens de Programação

Av. Antônio Carlos, 6627
31270-010 - Belo Horizonte - MG
4 de Dezembro de 2000

Resumo

Este trabalho mostra o resultado de uma análise feita a partir de três linguagens, SGML, HTML e XML na busca da melhor linguagem, atualmente disponível, para efetuar de forma segura e eficiente a recuperação e publicação eletrônica de documentos. O SGML é uma linguagem estruturada e com semântica complexa; o HTML não possui esta estruturação e tem tags fixos e a XML, uma extensão do SGML, foi desenvolvida especialmente para a Web, possui formato estruturado e maior simplicidade. Após o levantamento e análise das principais características dessas linguagens chegou-se na linguagem XML. Para atingir o objetivo proposto neste trabalho, essa linguagem foi estudada em detalhes. A abordagem escolhida, via exemplos, mostra suas funcionalidades em vários tipos de documentos: textos e livros literários; artigos e livros técnico-científicos; listagens e dados estátisticos; textos, documentos com figuras; documentos com conexões diversas entre outros. Como sugestão de trabalhos futuros deveriam ser estudados os novos recursos em desenvolvimento da linguagem e também a viabilidade do uso de XML em aplicações comerciais.

Índice

1 Introdução **1**

2 Estado da Arte **3**
2.1 Introdução . . . 3
2.2 SGML . . . 3
2.2.1 Avaliação do SGML . . . 7
2.3 HTML . . . 8
2.3.1 Avaliação do HTML . . . 12
2.4 XML . . . 12
2.4.1 XML 1.0 . . . 14
2.4.2 XLink e XPointer . . . 16
2.4.3 XSL . . . 17
2.4.4 Avaliação do XML . . . 18
2.5 Outras Linguagens . . . 18
2.6 Conclusão . . . 19

3 A Linguagem XML × Publicação de Documentos **20**
3.1 Introdução . . . 20
3.2 Tipos de Documentos . . . 20
3.2.1 Textos e Livros Literários . . . 21
3.2.2 Textos, Artigos e Livros Técnicos-Científicos . . . 21
3.2.3 Listagens e Dados Estatísticos . . . 21
3.2.4 Documentos com Figuras . . . 22
3.2.5 Documentos com Conexões Diversas . . . 22
3.2.6 Documentos com Outros Recursos Interessantes . . . 22
3.3 Ferramentas para o Uso do XML . . . 23
3.4 XML × Textos Literários . . . 23
3.5 XML × Documentos Técnico-Científicos . . . 28
3.5.1 O Uso de Seções CDATA . . . 28
3.5.2 O Uso de DTDs . . . 29
3.6 XML × Listagens e Dados Estatísticos . . . 31
3.6.1 O CSS - Cascading Style Sheet . . . 31
3.6.2 O XSL - Extensible Style Sheet Language . . . 37
3.7 XML × Figuras . . . 45
3.8 XML × Conexões . . . 47
3.8.1 O XLink . . . 47
3.8.2 O XPointer . . . 49
3.9 Conclusão . . . 54

4 Conclusão Final **55**

5 Trabalhos Futuros **56**

1 Introdução

Quando falamos em um documento eletrônico [9] podemos estar nos referindo a inúmeras formas de documentos, por exemplo, uma imagem, uma seqüência de palavras, uma informação estruturada como: equações matemáticas, gráficos vetoriais, transações comerciais, etc. Imprimir, colocar o documento disponível na www, reescrever alguns trechos e arquivar de maneira segura são algumas das atividades comumente realizadas sobre estes documentos.

Existe na web referências [1] para uma grande quantidade de ferramentas e suporte para desenvolvimento de um documento. Além disso, existem três linguagens de anotação [1] para publicação eletrônica de documentos que podem, hoje em dia, ser consideradas como padrão. São elas HTML [7], SGML [6] e XML [8, 2, 3]. Passamos, então, a ter um problema: Qual ferramenta e suporte utilizar? Para nos auxiliar na escolha da ferramenta e suporte adequados, a solução do nosso problema se resume em uma palavra: padronização.

A idéia de conseguir imortalizar documentos importantes [4] em um formato onde sejam considerados portabilidade e compatibilidade é de grande interesse para a humanidade. Guardar um documento em um formato padronizado, conhecido internacionalmente, bem-estruturado, confiável e independente de plataforma é uma necessidade do mundo atual para poder proporcionar uma vida longa a uma informação importante a todos.

Neste sentido, a editoração eletrônica traz grandes vantagens, por exemplo, a economia de tempo na procura de papéis sobre um determinado assunto sem probabilidade de perda do documento. Com ela, informações podem ser rapidamente e facilmente distribuídas para múltiplos locais simultaneamente.

Outro ponto importante relacionado com a editoração de documentos é a realização desta atividade de forma que um documento seja visto como uma estrutura lógica. Informação estruturada contém ambos, conteúdo, por exemplo: palavras, figuras, etc. e alguma indicação do papel que este conteúdo desempenha, por exemplo, o conteúdo de um cabeçalho tem um significado diferente de um conteúdo de uma nota de rodapé. Quase todos os documentos possuem uma estrutura.

A necessidade de tornar um documento acessível a um maior número de pessoas de modo que elas possam trocar informações com garantia de recebimento sem alterações, incrementos ou perda de dados ou mesmo sem problema de compatibilidade foi um dos motivos que nos incentivaram no desenvolvimento deste trabalho cujo objetivo é procurar encontrar uma linguagem adequada para a recuperação de documentos, com alta qualidade e de forma duradoura.

Para tanto, nosso projeto foi dividido em três etapas:

1. Investigar o que o mercado oferece atualmente para a recuperação de documentos no formato eletrônico e, a partir daí, escolher a linguagem que melhor atende às necessidades apresentadas.

[1] Uma linguagem de anotação é um conjunto de etiquetas que se coloca em um texto para marcar sua estrutura.

2. Realizar um estudo mais aprofundado da linguagem escolhida, focando em sua sintaxe e forma de uso, assim como definindo as ferramentas existentes que proporcionam a utilização dessa linguagem.

3. Escolher documentos a serem recuperados. Estes documentos devem possuir características interessantes de modo a proporcionar uma visão do funcionamento da sintaxe da linguagem escolhida ressaltando seus recursos mais importantes durante a passagem para o novo formato.

Este relatório apresenta os resultados obtidos após a conclusão dos itens acima. Ele está organizado da seguinte forma:

- A Seção 2 apresenta as três linguagens que foram estudadas visando a recuperação de documentos para o formato eletrônico e a linguagem escolhida para tal intuito.

- A Seção 2.2 apresenta o SGML, *Standard Generalized Markup Language*, um padrão internacional que permite a publicação de documentos no formato original, não importando o recurso computacional utilizado.

- A Seção 2.3 apresenta o HTML, *Hypertext Markup Language*, uma aplicação do SGML, com restrições principalmente em relação à funcionalidades de *tags*.

- A Seção 2.4 apresenta o XML, *Extensible Markup Language*, uma extensão do SGML, cuja principal característica é adaptar o SGML para publicação de documentos especificamente para a www.

- A Seção 2.5 apresenta outras linguagens usadas para publicação eletrônica. Estas linguagens, contudo, não satisfazem o escopo deste trabalho e, por isso, não foram detalhadas.

- A Seção 2.6 apresenta os argumentos que nos fizeram escolher o XML como a linguagem adequada para o desenvolvimento da segunda etapa deste trabalho.

- A Seção 3 apresenta um estudo aprofundado do XML, bem como os tipos de documentos e as ferramentas para a utilização desta linguagem.

- A Seção 3.2 apresenta os tipos de documentos existentes que possuem características interessantes para que se utilize o XML.

- A Seção 3.3 apresenta as ferramentas e recursos necessários para que a publicação e recuperação de documentos usando XML seja possível.

- A Seção 3.4 apresenta algumas características da linguagem XML que possibilitam a criação de textos literários.

- A Seção 3.5 apresenta algumas características da linguagem XML que possibilitam a criação de textos técnicos-científicos.

- A Seção 3.6 apresenta algumas características da linguagem XML que possibilitam a criação de listagens e dados estatísticos.
- A Seção 3.7 apresenta algumas características da linguagem XML que possibilitam a criação de documentos com figuras.
- A Seção 3.8 apresenta algumas características da linguagem XML que possibilitam a criação de textos com conexões diversas.
- A Seção 3.9 apresenta os argumentos que nos fizeram escolher o XML como a linguagem adequada para a criação dos vários tipos de documentos existentes.
- A Seção 4 apresenta a conclusão deste trabalho, mostrando as vantagens do uso do XML para recuperação de documentos eletrônicos.
- A Seção 5 apresenta tópicos que podem ser desenvolvidos em um estudo posterior mais aprofundado da linguagem XML e que não fazem parte do escopo deste trabalho.

2 Estado da Arte

2.1 Introdução

Esta Seção apresenta o levantamento das principais características de três linguagens de editoração eletrônica visando a recuperação e publicação de documentos para o formato eletrônico. Foram analisadas: o SGML, que é uma linguagem estruturada e com semântica complexa; o HTML, que não possui esta estruturação e tem tags fixos e o XML, uma extensão do SGML que foi desenvolvido especialmente para a Web, possui formato estruturado e maior simplicidade. Após esta análise foi feita uma comparação buscando selecionar a linguagem que melhor atende ao objetivo proposto, mostrando o porquê da sua escolha.

2.2 SGML

Um dos grandes desafios da publicação eletrônica de documentos é conseguir encontrar uma forma de preservar as características literárias e linguísticas de um texto em diferentes plataformas de computação.

Este é um grande problema pois existe uma diversidade de recursos computacionais: sistemas operacionais: *Linux, Unix, Windows*, etc; processadores de texto; aplicativos; formas de armazenamento de informações: cartões, fitas, discos, etc, que diferem entre si e que, normalmente, são incompatíveis.

Tentando minimizar este problema foi criado um padrão internacional para garantir a visualização e leitura do documento no formato original, não importando o recurso computacional adotado. Este padrão denomina-se SGML [11], *Standard Generalized Markup Language*, cuja principal característica é a independência de plataforma.

O SGML é uma metalinguagem, ou seja, é uma linguagem que descreve outra linguagem, no caso, a linguagem de anotação, *Markup Language*.

Uma linguagem de anotação é formada por um conjunto de etiquetas, *tags*, que se coloca em um texto para marcar sua estrutura. Por exemplo:

- Quando se deseja imprimir certos trechos de um texto em negrito, são colocados marcadores antes e depois do texto que se deseja imprimir em negrito da seguinte forma:

```
<negrito> Parte que será impressa em negrito </negrito>
Parte que não será impressa em negrito.
```

- Quando se deseja garantir a completeza de um texto. Considere uma receita de bolo, ela precisa ter as seguintes partes:

```
<Nome do Bolo> seguido dos <Ingredientes>, que deve conter
<Nome dos Ingredientes> e  <Quantidade> deles, seguido do
<Modo de Fazer>.
```

  Sendo esquecida uma dessas partes, a receita fica incompleta.

O SGML, portanto, deve prover uma sintaxe que permita especificar uma forma de definir quais *tags* são permitidas, quais são obrigatórias, como distinguir estas *tags* do texto em si e o que cada uma das *tags* significa.

Especificamente, existem três pontos que caracterizam o SGML [10]:

- Sua independência de plataforma;
- Sua ênfase em marcadores descritivos e não procedurais;
- O conceito de Definição de Documento de Tipo.

A independência de plataforma foi o que motivou sua criação.

Por ênfase em marcadores descritivos entenda que o SGML simplesmente fornece nomes para categorizar partes de um documento. Assim, a *tag* <título> apenas indica que a parte seguindo a *tag* será o título do documento, mas não define a fonte, o tamanho, a formatação do título.

Outra característica importante do SGML é a Definição de Documento de Tipo ou DTD, *Definition Type Document*. Neste documento são feitas as definições de tipo e estrutura de um documento, por exemplo, em uma receita de bolo, primeiramente aparecem os ingredientes e depois o modo de fazer. O desrespeito a essas regras ou estruturas são identificadas durante o processamento verificando a correção dos elementos e da estrutura do texto. Além disso, os DTDs permitem que os programas encapsulem as estruturas. Este

fato favorece tanto a segurança das informações como o reaproveitamento das definições em vários outros documentos.

Em SGML, as instruções necessárias para processar o documento, por exemplo, para formatá-lo, são dadas normalmente em programas ou procedimentos separados, havendo grande distinção entre a parte descritiva e a parte procedural dos marcadores. Usando a definição descritiva, o mesmo documento pode ser processado de várias formas diferentes, dependendo do foco do programa utilizado. Por exemplo, um programa P1 pode querer coletar os títulos de cada capítulo para formar um índice, enquanto outro programa P2, apenas deseja imprimir os títulos em negrito, em uma fonte maior com alinhamento centralizado.

Para entender como funciona o SGML vamos definir alguns conceitos e dar um exemplo simples, explicando o seu formato. Este exemplo apresenta a estrutura de um poema [10] que possui a seguinte hierarquia:

- Um poema é constituído por um `título`, que pode estar ou não presente, seguido de uma ou mais `estrofes`
- Cada `estrofe` é constituída por uma ou mais `linhas`.
- A estrutura do documento deve, então, seguir a ordem ilustrada na Figura 1

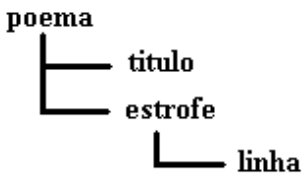

Figura 1: Hierarquia do Poema

De acordo com a hierarquia dada, uma `estrofe` não pode ser iniciada se antes não tiver sido indicado que o `poema` começou. Do mesmo modo, uma `linha` não pode ser inserida depois do `título` se não houver sido indicado que uma `estrofe` foi iniciada.

O primeiro passo para a especificação de uma estrutura de um documento SGML é estabelecer as regras que descrevem esta estrutura via DTD. Essas regras são importantes pois podem simplificar a tarefa de colocar *tags* em um texto sem causar perda na estrutura do texto. O DTD para a hierarquia do `poema` descrito poderia ser o seguinte:

```
<!ELEMENT poema>    --   (titulo?, estrofe+)
<!ELEMENT titulo>   -o   (#PCDATA)
<!ELEMENT estrofe>  -o   (linha+)
<!ELEMENT linha>    oo   (#PCDATA)
```

Neste DTD podemos identificar os seguintes componentes em ordem:

1. `ELEMENT`: identificam as partes do poema, seguido dos seus respectivos nomes. Um Elemento pode ser composto de outros elementos simples ou por # PCDATA que é um símbolo terminal ou *Token*.

2. Indicadores de obrigatoriedade para a *tag* de início e fim do elemento respectivamente:

   (a) '-' indica que a *tag* é obrigatória

   (b) 'o' indica que a *tag* é opcional

3. Indicadores de ocorrência: mostram a frequência que um elemento se repete na estrutura:

   (a) '*' indica que o elemento pode ocorrer nenhuma, uma ou várias vezes

   (b) '+' indica que o elemento pode ocorrer uma ou mais vezes

   (c) '?' indica que o elemento pode ocorrer uma ou nenhuma vez

4. Conectores: mostram a ordem em que um elemento pode aparecer:

   (a) '|' indica que apenas um dos elementos pode aparecer

   (b) ',' indica que os elementos devem aparecer na ordem em que foram definidos

   (c) '&' indica que os elementos podem aparecer em qualquer ordem, mas todos os elementos definidos devem aparecer.

Definido o DTD, a estrutura de um documento SGML para o exemplo em questão poderia ser o que se segue:

```
<!DOCTYPE Poema SYSTEM "c:\poc\dtd\poema.dtd">

<poema>
<titulo> TITULO
<estrofe>
<linha>
<linha>
</poema>
```

O exemplo que acabamos de mostrar ilustra um dos casos mais simples no uso de SGML, mas várias outras declarações e regras poderiam ser acrescentadas a esta estrutura para torná-la mais sofisticada. São elas [10]:

- A inclusão de elementos que podem aparecer em qualquer parte do documento e que, portanto, não está presente na hierarquia apresentada. Exemplo: nota de rodapé.
- A existência de múltiplas hierarquias paralelas dentro de um mesmo documento. Por exemplo: acrescentar uma hierarquia que fizesse a paginação dos poemas.
- A inclusão de atributos, informações adicionais aos elementos, para informar pontos que descrevem um elemento específico, mas não podem ser considerados uma parte constituinte do elemento. Por exemplo: *id* do poema; *status* do poema: publicado, original ou revisado.
- Uma aplicação do uso de atributos é a propriedade de ativar um *link*, conexão, entre elementos por meio dos seus respectivos identificadores que devem ser únicos.
- A definição de entidades, uma estrutura portável do documento que permite a economia na escrita ou a inclusão de trechos de outros documentos dentro de sua estrutura o que simplifica a tarefa de manutenção da consistência dos dados em um conjunto de documentos. Exemplos:

  ```
  <!ENTITY SGML "SGML - Standard Generalized Markup Language">
  ```

  Faz com que toda vez que aparecer SGML no texto, ele seja substituído pelo *string* relacionado acima.

  ```
  <!ENTITY trechoX SYSTEM "trechoX.txt">
  ```

  Faz com que o conteúdo do arquivo trechoX.txt seja incluído no texto toda vez que for escrito trechoX.

- A existência de "sessões marcadas" facilita bastante o reaproveitamento das informações. Sessões marcadas consiste na marcação de um certo trecho do texto que seria interessante aparecer em algumas aplicações e em outras não. Por exemplo: em um manual técnico de certos equipamentos, há informações que são instruções gerais para qualquer equipamento, mas outras características são individuais dependendo do aparelho.

2.2.1 Avaliação do SGML

Pode-se perceber pelo exposto, que o SGML possui um grande número de funcionalidades, podendo ser aplicado para descrever centenas de tipos de documentos, desde manuscritos antigos a documentos técnicos, desde registros médicos a notações musicais. Esta característica é devido ao caráter geral da linguagem que permite aplicações em vários ambientes sem necessidade de atualizações ou versões. Mas toda generalização traz consigo

uma bagagem de complexidade. Isto significa que problemas que poderiam ser resolvidos com simplicidade tornam-se complicados, sem necessidade, com o uso de SGML.

Justamente por ser tão geral, o SGML não foi muito difundido e não é aplicado para publicação de documentos. O SGML é usado, principalmente, para tratar excessões e casos especiais, especialmente quando a uniformidade da estrutura for o mais importante. Portanto não há necessidade de complicar quando existem soluções mais simples e compreensíveis.

Resumindo, o SGML foi desenvolvido para ser:

- Suficientemente formal a fim de permitir provas de validade do documento.
- Suficientemente estruturado para lidar com documentos complexos.
- Suficientemente extenso para dar suporte ao gerenciamento de grandes depósitos de informações.

Conseqüentemente, várias características do SGML não têm utilidade dentro do nosso propósito que é a recuperação de documentos para o formato eletrônico. Seu uso faria com que houvesse um desperdício de recursos, uma vez que o nosso objetivo poderia ser alcançado com mais simplicidade.

Outro ponto desfavorável ao uso do SGML é que ele é perigoso na rede, pois tem várias características opcionais, exigindo que o emissor e receptor concordem com algumas destas opções para permanecerem compatíveis e operáveis entre eles.

Visando buscar uma maior simplicidade, duas outras linguagens serão apresentadas nas Seções 2.3 e 2.4, respectivamente, o HTML e o XML. O HTML é uma aplicação do SGML com recursos restritos e o XML é uma versão do SGML simplificada omitindo as partes mais complexas e menos usadas das funcionalidades do SGML.

2.3 HTML

A www, a *World Wide Web*, é uma rede mundial de recursos de informação. Ela é baseada, principalmente, em três mecanismos que possibilitam que estes recursos se tornem acessíveis ao público. São eles [12]:

- Um esquema uniforme de nomenclatura que permite a localização dos recursos na rede, como por exemplo, as URIs, *Universal Resource Indentifier*.
- Os Protocolos, para acesso a recursos via web, por exemplo, o HTTP, *HyperText Transfer Protocol* .
- O HiperTexto [2], que permite uma navegação fácil entre os recursos disponíveis.

[2]Hipertexto são trechos de texto ou de qualquer outra informação que se encontram interconectados. Exemplo: nota de rodapé

Nesta pesquisa, estaremos interessados apenas no último dos mecanismos, o Hipertexto, relacionado com a linguagem de anotação a ser utilizada para permitir a criação e publicação de documentos eletrônicos na www.

Quando da criação da www, a Web não estava preparada para uma poderosa e genérica linguagem de anotação, como o SGML. O SGML apresentava uma estrutura complexa para o objetivo desejado na época. Nesse estágio inicial, a Web precisava apenas de um pequeno conjunto de marcadores, *tags*, que fosse adequado para a maioria dos documentos e que fosse simples o suficiente para ser entendido e proporcionar a proliferação da publicação de documentos eletrônicos.

Na tentativa de suprir esta necessidade da época e, ao mesmo tempo, aproveitar os recursos já disponíveis, como a linguagem SGML, a solução foi criar uma nova linguagem, o HTML, *Hypertext Markup Language*, contendo apenas as funcionalidades básicas e desejáveis no momento até que se tornasse possível a difusão de informações de um modo mais complexo.

O HTML [13] é, então, uma aplicação simplificada do SGML, sendo de caráter universal e também independente de plataforma, ou seja, documentos em HTML podem ser vistos por qualquer computador que esteja conectado à Internet e que possua um *browser* HTML. Ele foi criado para permitir a composição e publicação de documentos eletrônicos na www e obteve rápida ascensão e grande utilização que pode ser observada até hoje. O HTML foi um dos principais mecanismos que permitiram a explosão da www.

O HTML é composto por um conjunto de *tags* definidas e fixas que permitem a estruturação do texto em cabeçalhos, *headings*, parágrafos, listas, *links* de hipertexto, etc. Sintaticamente uma *tag* é definida por um "<" seguida de um nome e finalizada por um ">".

Para mostrar o funcionamento das *tags* vamos apresentar um pequeno exemplo de um texto em HTML e algumas das *tags* mais comuns. Um texto em HTML possui a seguinte estrutura básica:

```
<HTML>
<HEAD>
<TITLE> Aqui fica o Titulo </TITLE>
</HEAD>
<BODY>
<p>
Aqui fica o corpo do seu texto.
</BODY>
</HTML>
```

Observando este exemplo, pode-se perceber que cada vez que uma *tag* for inicializada <TAG> ela também deve ser finalizada, </TAG> com exceção para as *tags* de mudança de linha e parágrafo, ilustradas no exemplo pela *tag* <p> indicando o início de um parágrafo.

Todo documento em HTML começa pela *tag* <HTML> e termina com a sua respectiva *tag* de finalização.

Além dessa estrutura básica, há várias outras *tags* com outras funcionalidades presentes em HTML. Algumas delas são:

- *Tags* para formatação do texto:
  - Para centralizar T1

    ```
    <CENTER> T1 </CENTER>
    ```

  - Para colocar T2 com a fonte de tamanho 30

    ```
    <FONT SIZE =30> T2 </FONT>
    ```

- *Tags* para conexão de trechos de texto ou páginas entre si, por exemplo:

  ```
  <a HREF="#fim"> fim da p&aacute;gina</a>
  <a NAME="fim"></a>fim da p&aacute;gina
  ```

- *Tags* para inclusão de listas, tabelas, entre outras, por exemplo:
  - Para criação de itens como este.

    ```
    <ul>
    <li>
    Primeiro</li>
    <li>
    Segundo</li>
    <li>
    Terceiro</li>
    </ul>
    ```

Juntando todas estas *tags* mencionadas acima na estrutura básica, podemos produzir um arquivo em HTML, arquivo.html, e visualizar o documento gerado. Este arquivo gera o documento estruturado apresentado na Figura 2:

```
<HTML>
<HEAD>
<TITLE> Aqui fica o Titulo </TITLE>
</HEAD>
<BODY>
Aqui fica o corpo do seu texto.
<br>
LINK para o<a href="#fim"> fim da p&aacute;gina</a>
<br>
<CENTER> T1 </CENTER>
<FONT SIZE =30> T2 </FONT>
```

```
<br>
Exemplo de uma lista :
<ul>
<li>
Primeiro</li>
<li>
Segundo</li>
<li>
Terceiro</li>
</ul>
<br>
<p><a NAME="fim"></a>fim da p&aacute;gina
</BODY>
</HTML>
```

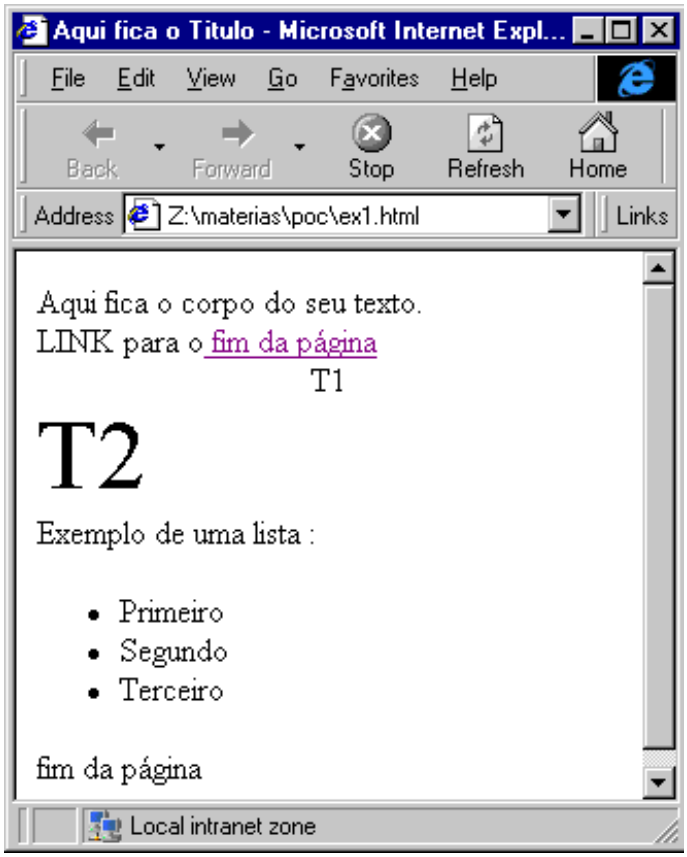

Figura 2: Exemplo em HTML

Com o desenvolvimento da www e a crescente popularização da publicação eletrônica, a demanda por um conjunto de *tags* com mais recursos e funcionalidades não previstas

na primeira versão do HTML foi inevitável. Para suprir estas necessidades, foram sendo criadas várias versões de HTML, cada uma delas acrescentando novas facilidades de acordo com a necessidade observada. São algumas delas [12]:

- Os estilo de página, *style sheets*: permitem ao autor e usuários maior controle sobre a apresentação dos documentos: fonte, alinhamentos, cores, etc;
- Os *frames*, quadros: responsável pela divisão da página em quadros independentes;
- A possibilidade de criação de tabelas mais ricas: permite maior controle sobre o laioute e estrutura das tabelas;
- O *scripting*: responsável pela criação de *home-pages* dinâmicas, que reagem de acordo com o modo de uso do usuário, permitindo a inserção de trechos de código em linguagem script e o desenvolvimento de aplicações via rede. O *scripting* é independente da linguagem de script utilizada.

2.3.1 Avaliação do HTML

Mesmo com o acréscimo de novas funcionalidades, o HTML tornou-se incapaz de suprir as necessidades cada vez mais exigentes e diversas dos usuários e criadores de documentos eletrônicos. Os *tags* fixos, representam, hoje, uma barreira no desenvolvimento de publicações de alto nível técnico, uma vez que a rede de computadores já amadureceu o suficiente para desejar documentos que possuem maiores recursos.

Os *tags* fixos restringem o uso das funcionalidades apenas às já definidas, não permitindo a criação de *tags* personalizadas. Além disso, a mudança de uma determinada característica da página acarreta grande trabalho já que as muitas definições, apesar da criação dos *stylesheets*, ainda continuam sendo feitas individualmente não ocorrendo reaproveitamento de código e economia de tempo na programação usando HTML.

Como pôde ser observado, a inflexibilidade dos *tags* em HTML, passou a ser, atualmente, uma grande restrição para as publicações que fazem uso de recursos mais sofisticados. Esta característica que, a princípio, foi essencial para o desenvolvimento da www, hoje é um grande empecilho para o seu contínuo crescimento.

Visando suprir as deficiências levantadas, uma outra linguagem de anotação não tão complexa como SGML e não tão simplificada como o HTML será apresentada na Seção 2.4. Esta linguagem, XML, foi desenvolvida para permitir o uso do SGML na www, uma vez que ela não apresenta as características opcionais que impedem o uso do SGML na rede, sendo este o principal motivo para descartarmos a escolha do SGML.

2.4 XML

O crescimento da www é recente. Nos dias de hoje, os computadores são utilizados para as mais diversas atividades, principalmente quando conectados entre si. Por exemplo para [14]:

Acesso a informações remotas: permitem acesso a instituições financeiras para pagamento de contas, gerenciamento das contas bancárias, gerenciamento de investimentos; permitem realização de compras via Internet; acesso a notícias em jornais e revistas e acesso a todo tipo de documentos com informações sobre arte, religião, negócios, culinária, governo, saúde, história, ciência, esportes, etc.

Comunicação entre pessoas: permite comunicação direta através de e-mail, videoconferência, ainda em evolução, *newsgroup*, ou seja, grupo de discussões eletrônicos sobre determinado assunto, etc.

Divertimento: permite os jogos eletrônicos à distância com simulação de jogos em tempo real, permite que se desenvolva a indústria do vídeo sob demanda permitindo a escolha da programação de TV ou cinema ou o desenvolvimento de filmes interativos.

Uso em empresas: permite divulgação e troca de informações dentro da empresa de maneira mais rápida de forma a proporcionar tanto economia de tempo como de dinheiro para as corporações, etc.

Para se garantir que informações estruturadas em um documento possam ser acessíveis a um grande número de pessoas sem perda da estrutura e formatação original, é necessário que a linguagem em que se escreve um documento seja adaptada para a www com segurança.

A falta dessa adaptação para a rede presente no SGML que, até o momento, parecia a linguagem mais adequada para as novas necessidades na publicação eletrônica, fez com que a comunidade acadêmica, por meio do *WorldWide Web Consortium*, W3C, visse a necessidade de criação de uma nova linguagem, o XML, *Extensible Markup Language*, que na verdade é uma adaptação do SGML suprindo as características que impedem seu uso na rede. Nessa nova linguagem foram eliminadas certas propriedades para evitar excesso de transmissão de informações o que não é desejável em uma rede, pois pode gerar congestionamento. Foram também cortadas características que dificultavam a interpretação da linguagem para proprocionar maior rapidez na transmissão, além de outras características opcionais que impedem a compatibilidade na comunicação entre computadores.

Assim, o XML foi desenvolvido para facilitar e possibilitar o uso do SGML na www, tendo como principais características:

- Facilidade para definir tipos de documento;
- Facilidade para criação e gerenciamento dos documentos;
- Facilidade para transmissão e compartilhamento de documentos via www.

O objetivo de XML é possibilitar a publicação, o envio e o processamento de documentos na rede de forma tão simples como hoje é possível com o uso de HTML, mas utilizando recursos de estruturação mais sofisticados e eliminando as limitações do HTML.

O XML é, então, assim como o SGML, uma metalinguagem que permite a criação de *tags* próprias para a estruturação do texto.

O XML é definido a partir de três especificações básicas [8]:

1. A especificação própria da linguagem via *Extensible Markup Language 1.0*, onde foi definida a sintaxe da linguagem. Os detalhes desta especificação serão apresentados na Seção 2.4.1.

2. O *XML Pointer Language*, o XPointer e o *XML Linking Language*, o XLink, onde foi definido um modo padrão para representar conexão entre recursos. Maiores detalhes sobre as possibilidades de se realizar esta conexão em XML será apresentada na Seção 2.4.2.

3. O *Extensible Style Language*, o XSL, onde foi definido o padrão de formatação de documentos escritos em XML. A forma de se usar este recurso para fazer a formatação do documento será apresentada na Seção 2.4.3

2.4.1 XML 1.0

Nesta Seção serão apresentadas as principais características da linguagem relacionadas com a sua sintaxe e que foram definidas na primeira versão da especificação do XML.

As principais características de XML que a tornam diferente do SGML são [5]:

1. Em XML, foi definida uma sintaxe de forma a garantir que todas as pessoas que usem XML, façam uso da mesma sintaxe. Por exemplo: toda *tag* de início deve começar com "<"e terminar com ">".

2. Em XML, uma nova *tag* indicando elemento vazio é usada, indicando que a *tag* de fim não é esperada. Esta nova estrutura de *tag* é representada da seguinte forma: "<nometag/>"

3. Em XML, não é permitida a omissão de *tags*. Assim, todo elemento não vazio deve possuir tanto a *tag* de início, <nometag> como a *tag* de fim, </nometag>.

4. Em XML não é necessária a existência da Definição de Documento de Tipo ou DTD.

A estrutura de um documento XML com as características acima poderia ser [8]:

```
<?xml version="1.0"?>
<piada>
<joao> Diga <aspas> boa-noite </aspas>, Maria. </joao>
<jose><aspas>Boa-noite, Maria.</aspas></jose>
<aplausos/>
</piada>
```

Em relação ao exemplo mostrado, algumas observações precisam ser feitas:

- Um documento XML começa com uma instrução de processamento <?xml...?>. A presença dessa instrução identifica o documento como sendo escrito em XML, além de indicar a versão do XML a ser aplicada.

- Não há DTD para este documento, uma vez que a presença de DTD em XML é opcional. Os DTDs serão incorporados a um documento quando necessários para evitar ambigüidade ou interpretações diferentes de uma mesma estrutura.

- Há a presença de um elemento vazio o <aplauso/>, que apenas indica uma marcação onde algum evento, no caso, aplausos, irá ocorrer.

Um documento XML é composto de marcadores, *markups*, e conteúdo. Existem seis tipos de marcadores que podem ocorrer em um documento XML: elementos, referências a entidades, sessões marcadas, DTDs, instruções a serem processadas e comentários.

O conceito dos quatro primeiros marcadores foram suficientemente descritos na Seção 2.2 e são aplicados da mesma forma em XML. Já as duas últimas características necessitam de explicação.

Por instruções a serem processadas ou PIs entende-se "pedaços de informações"inseridas no documento para fornecer informações para uma aplicação. As PIs não são necessariamente parte de um documento XML, mas o processador XML deve passar esta informação adiante para a aplicação. Uma PI tem a seguinte forma:

```
<?nome dadosPI?>
```

O nome, também chamado de alvo da PI, identifica a PI para a aplicação. Aplicações devem processar apenas os alvos reconhecidos por ela e ignorar todos os outros que não lhe interessem.

Os comentários são informações inseridas ao longo do documento para fornecer dados extras que o construtor do documento achar necessário. Os comentários não precisam ser passados para nenhuma aplicação.

Para inserir um comentário, deve-se proceder da seguinte maneira:

```
Comentários começam com <!-- e terminam com -->
Comentários podem conter qualquer dado exceto o string literal --
Comentários podem ser inseridos entre qualquer marcador,
em qualquer lugar no seu documento.
```

Como os DTDs em XML não são obrigatórios, os documentos em XML foram divididos em duas categorias: Documentos bem-estruturados e Documentos Válidos.

Documentos bem-estruturados são aqueles que obedecem a todas as regras de sintaxe definidas para o XML. Assim, atributos não podem aparecer mais de uma vez em uma mesma *tag* de início, entidades precisam ser declaradas antes de serem usadas, etc.

Um documento bem-estruturado apenas é considerado Válido se ele contém um DTD e se o documento obedece às restrições impostas nesta declaração.

Em relação ao conteúdo dos documentos, temos a definição de um modelo, Esquema, que permite a inclusão de restrições ao conteúdo dos documentos. Um Esquema descreve um modelo que possibilita a inclusão de *tags* e textos de uma forma válida dentro de um documento XML. Um Esquema pode também ser visto como um acordo que estabelece um vocabulário comum para uma determinada aplicação que envolva compartilhamento de documentos.

Para tornar mais claro o funcionmento destas restrições vamos analisar o Esquema a seguir:

```
<endereco>
<nome>Renata Castro Carvalho</nome>
<rua>Rua 10 de Abril, número 56</rua>
<cidade>Belo Horizonte</cidade>
<estado>MG</estado>
<estado>PR</estado>
<cep>3098</cep>
</endereco>
```

Este exemplo viola duas restrições que podem ser observadas. Primeiramente é definido mais de um `estado` para um mesmo `endereço`. Depois o formato do `cep` é diferente do formato correto de um `cep`, que na realidade deveria possuir 8 dígitos.

A habilidade de testar a validade de documentos é um aspecto importante em aplicações *web* que estão recebendo e transmitindo informações de diferentes fontes. Não é desejado processar as informações que não estejam com seu Esquema válido.

2.4.2 XLink e XPointer

Um *link* ou conexão expressa o relacionamento entre recursos, como por exemplo, um elemento se relaciona com seu conteúdo ou parte do seu conteúdo e são conectados estando ambos em qualquer localização endereçada pelo *link*.

Apesar de toda publicidade em torno do HTML que é chamado de *Hypertext Markup Language*, esta linguagem implementa apenas uma forma bem simples de conexão: *links* unidirecionais para um local estabelecido.

Em um sistema verdadeiro de hipertexto, como é o caso do XML, busca-se uma padronização para permitir vários mecanismos de conexão [15]:

- Conexão independente de localização dos recursos, seja dentro ou fora do documento
- *Links* bidirecionais.
- *Links* multi-direcionais que podem ser especificados e gerenciados fora do documento,por exemplo, anéis, janelas múltiplas, etc.
- *Links* agregados, para acessar fontes múltiplas.

- Atributos em *links*, estabelecendo tipos para os *links*. Como o XML não possui um conjunto fixo de elementos, o nome do *link* não pode ser usado para fazer a conexão. Ao invés disso, os processadores identificam o *link* por meio do reconhecimento do atributo `xml:link`.

O XLink permite ao construtor ter controle sobre a semântica do *link*, podendo a conexão ser realizada entre dois ou mais recursos e os recursos podem ser desde arquivos de qualquer tipo até elementos de um arquivo ou documento.

O conceito de XPointer é bastante relacionado ao XLink, uma vez que o XPointer define os pontos onde os *links* serão aplicados. O XPointer oferece uma sintaxe que permite a localização do recurso percorrendo a árvore, ou seja, todo o caminho, do documento que contém o recurso.

Deve-se tomar cuidado com o uso deste poderoso recurso em XML, uma vez que a especificação do XPointer não impede que ele aponte para um trecho inválido do documento.

2.4.3 XSL

O uso de estilos de páginas é obrigatório em XML quando se deseja formatar um texto. Alguns *browser* possibilitam estilos simples para elementos populares como <paragrafo>, ou <listas> contendo <itens>, mas em geral, um estilo de página permite ao autor maior controle sobre o seu documento.

Em uma linguagem poderosa como o XML, existem várias características que permitem que a formatação do texto seja a mais completa possível [15]:

- O XML é livremente extensível, de modo a permitir que os projetistas possam definir um número ilimitado de tratamento na formatação para uma grande variedade de *tags*.
- O XML é baseado em uma sintaxe padrão, o que diminui a curva de aprendizagem.
- Em XML pode-se endereçar a estrutura inteira de um documento, assim o relacionamento entre elementos pode ser expresso em qualquer nível de complexidade.
- O XML é um padrão internacional, então *scripts* de formatação à direita, ou esquerda, ou para o topo, etc podem ser aplicadas mesmo se misturadas em um único e simples documento.
- O XML foi desenvolvido de forma a permitir o uso de modelos sofisticados que fazem uso de características avançadas para laioute de página como rotação de áreas de texto, trechos flutuantes, etc.

Para aproveitar estas características buscando uma formatação facilitada e completa, algumas soluções são apresentadas [8]:

- O uso de *Cascading Stylesheet Specification*, CSS, fornece uma sintaxe simples para associar estilos aos elementos. Esta solução é parcialmente implementada em alguns *browsers* HTML, mas é muito simplificada para uso em linguagens que possuem flexibilidade de *tags* como o XML.

- O uso do padrão *Document Style Semantics and Specification Language*, DSSSL, possui todas as características listadas como desejáveis para uso em documentos XML, mas apresenta uma especificação muito complicada para ser usada no momento.

- O uso do *Extensible Style Language*, XSL, está sendo desenvolvido especialmente para ser aplicado junto com o XML. Este utiliza a sintaxe do XML, mas combina características de formatação das outras propostas como o CSS e o DSSSL.

Como aplicações específicas normalmente são mais eficientes, o XSL parece ser a solução mais adequada para resolver a questão da formatação em documentos XML. Como este padrão ainda está em desenvolvimento, não há como verficar a sua verdadeira capacidade no momento.

2.4.4 Avaliação do XML

Após toda esta análise, podemos perceber que o XML elimina duas restrições que se mostravam um empecilho para o desenvolvimento da www. A primeira é a dependência de uma única e inflexível Definição de Documento de Tipo presente no HTML. A outra restrição diz respeito à complexidade de uso do SGML para a www, cuja sintaxe permite poderosas, mas difíceis de programar e nem sempre compatíveis, formas de estruturação de um documento.

O XML, ao redefinir algumas das características mais importantes do SGML e adaptá-las para o uso na www, tornou-se uma linguagem adequada para a publicação de documentos eletrônicos na *web*. O XML, ao mesmo tempo que permite que grupos e organizações criem suas próprias *tags* de forma a customizar e estruturar suas informações, também possibilita a transmissão e compartilhamento dos documentos construídos com garantia de compatibilidade.

Assim, o XML permite que os documentos tenham as principais características desejáveis para sua publicação, não restringindo a modelos ou estruturas pré-definidos, facilitando a reusabilidade dos recursos e sendo completamente independente de plataforma o que torna o documento acessível a qualquer pessoa.

2.5 Outras Linguagens

Existe na literatura uma grande quantidade de outras linguagens para publicação eletrônica, mas grande parte dessas linguagens não é interessante neste trabalho por serem de aplicação específica ou por se referirem a aplicações mais comerciais. São alguns exemplos:

- A Wap, *Wireless Aplication Protocol* [17]: apresenta um protocolo para criação de *sites* para fornecer conteúdo para acesso à *Internet* via celulares.

- A linguagem JavaScript ou Java [18]: junto com linguagens como o HTML ou mesmo XML proporcionam a definição de restrições e construção de aplicativos comerciais.

2.6 Conclusão

O objetivo desse desta etapa do projeto é procurar encontrar uma linguagem adequada para a recuperação de documentos para o formato eletrônico, com alta qualidade e de forma duradoura.

A linguagem escolhida após a análise de todas as características foi o XML. Procuraremos destacar os principais pontos que nos fizeram descartar o uso de SGML e HTML e escolher XML.

O SGML possui algumas características que o coloca em desvantagem em relação ao nosso objetivo final, apesar de ser semanticamente estruturado:

- O SGML possui certas características opcionais que não são necessárias e acabam por causar incompatibilidade entre computadores conectados em rede se estes não possuem a mesma configuração quanto a essas opcionalidades.
- O SGML possui funcionalidades específicas para as mais diversas aplicações que são raramente usadas. Esta complexidade não é interessante na *web*, pois torna-se inviável se forem analisados aspectos de custo-benefício.
- A interpretação do SGML torna-se quase impossível sem um DTD associado, o que ocasiona uma maior lentidão na rede o que não é interessante.

Em relação ao HTML, as desvantagens são outras [16]:

- Os padrões HTML mudam muito devagar. Isto porque sua sintaxe é baseada em um conjunto de *tags* fixas. Para acrescentar ou modificar algumas dessas *tags* é necessário esperar a liberação de uma nova versão ou sair da padronização.
- Buscando novas funcionalidades foram surgindo *tags* específicas a certos *browsers*, causando problema de acesso a certas *home-pages* que usam *tags* visíveis apenas a alguns *browsers*. Este problema acaba por causar incompatibilidade, contrariando uma das principais características necessárias em uma linguagem para a *web*: a independência de plataforma.
- O HTML não possui marcações que indicam o significado da estrutura do documento, preocupando-se apenas com a formatação e tornando o texto ininteligível em relação à sua semântica.
- A forma de conexão entre documentos é muito limitada havendo necessidade de recursos mais sofisticados neste sentido.

Podendo solucionar todas estas questões, o XML, mostrou-se adequado para ser aplicado neste projeto, tendo como características marcantes e essenciais para o nosso objetivo [15]:

- Extensibilidade que permite a definição de *tags* próprias.
- Permite a criação de documentos estruturados com significado possibilitando a sua aplicação para representar esquemas de banco de dados, hierarquias de aplicações orientadas a objeto entre outras.
- Permite a confecção de documentos válidos, o que garante a compatibilidade e independência de plataforma.
- Não necessita obrigatoriamente de um DTD para definir a estrutura do documento, garantindo maior simplicidade à linguagem.
- Permite conexões múltiplas e multi-direcionais fornecendo maiores recursos e funcionalidades necessários para a recuperação de documentos eletrônicos de forma mais satisfatória.

3 A Linguagem XML × Publicação de Documentos

3.1 Introdução

Após feita a análise e escolhida a linguagem para a publicação de documentos na *Web*, uma outra questão torna-se importante. Quais são os tipos de documentos que existem e quais são os recursos do XML que podem proporcionar características diferenciais nestes documentos?

Para responder a estas questões, primeiramente vamos analisar alguns tipos de documentos disponíveis e suas principais características.

Em seguida mostraremos recursos do XML para publicá-los na *Web*, paralelamente a um estudo mais aprofundado da linguagem. Esta abordagem via exemplos não só facilita o entendimento dos recursos da linguagem como facilita o uso da linguagem uma vez que mostra aplicações práticas de sua utilização.

3.2 Tipos de Documentos

São vários os tipos de documentos disponíveis hoje na www. Estes documentos vão desde simples listas de nomes de pessoas a complexas e extensas páginas dinâmicas.

O escopo deste projeto não abrange o uso de XML para aplicações comerciais e uso de recursos mais sofisticados como vídeos, sons e similares. Os principais tipos de documentos que vamos analisar são: textos e livros literários; textos, artigos e livros técnicos-científicos; listagens e dados estatísticos; documentos com figuras; documentos com conexões diversas e alguns outros documentos por apresentarem outras características interessantes.

3.2.1 Textos e Livros Literários

A publicação de textos e artigos literários na *Web* é comum. Vários são os motivos que levam um autor a querer seu texto divulgado no formato eletrônico. O autor pode querer publicidade para o seu nome ou livro ou ele pode apenas estar querendo colocar seu texto disponível para um maior número de pessoas.

Já existem vários artigos e livros publicados na *Web* e que não estão no formato XML. Passar estes documentos para a linguagem XML é interessante no sentido que estes artigos ou livros, estando estruturados, o entendimento do assunto abordado torna-se mais fácil, além de torna-los inteligíveis para um leigo em XML. Neste sentido, várias conversões já foram feitas e já existem versões disponíveis na rede, por exemplo, da Bíblia e do Alcorão em XML.

3.2.2 Textos, Artigos e Livros Técnicos-Científicos

Neste tipo de documento talvez esteja a grande vantagem da estruturação. Um documento técnico ou científico estruturado permite que sua leitura seja inteligível para qualquer usuário que seja da área, possibilitando buscas de assuntos específicos com mais facilidade, correções de erros mais localizados, além de proporcionar ao documento uma vida mais longa que não depende de sua formatação.

O grande avanço do XML encontra-se justamente neste ponto e até existem várias variações do XML sendo desenvolvidas para facilitar sua utilização em aplicações para áreas específicas. São exemplos dessas variações: o CML, *Chemical Markup Language*, o MathML, *Mathematical Markup Language*, o MusicML, *Music Markup Language*, dentre outros.

Não faz parte deste projeto analisar cada uma destas aplicações, mas apenas mostrar sua importância na área técnica-científica.

3.2.3 Listagens e Dados Estatísticos

Este tipo de documento é publicado facilmente na *Web* como fonte de informação para o usuário. Compreende desde listas dos mais diversos assuntos a tabelas contendo dados estatísticos coletados para alguma pesquisa.

Uma das principais vantagens e o grande diferencial da publicação de documentos como estes usando a linguagem XML está relacionado com o fato de que se estes documentos estiverem bem estruturados, podem servir também como entrada de dados para outros programas que irão analisar ou agrupar características específicas destes dados.

Outra vantagem do uso de XML para publicação destes documentos é que o XML possibilita diferentes formatações dos dados, dependendo da necessidade do usuário, através dos estilos de página. Essa versatilidade é interessante, pois torna-se desnecessário reescrever os dados se, por exemplo, em um momento desejamos vê-los em formato de tabela e, em outro instante, em forma de gráfico.

3.2.4 Documentos com Figuras

A grande maioria das páginas da Internet hoje possuem figuras. Figuras são interessantes não apenas para enfeitar a página, mas também servem como fonte de informação, ilustração, ou divulgação de logomarcas de empresas.

Apesar do uso de figuras ser intenso, os recursos para o uso delas é bastante restrito e limitado. O XML pode expandir e facilitar o uso delas, tornando possível a utilização dessas figuras de maneira mais interessante e funcional.

Um dos grandes problemas hoje é que para se mudar de figura em uma página é preciso ir em todos os lugares onde esta figura é referenciada para trocar sua referência. Assim, imagine uma grande empresa que decida modificar sua logomarca e imagine o trabalho que teria para procurar em todas as suas páginas onde estão as logomarcas antigas para mudá-las para a nova versão. Para solucionar este problema, o XML oferece as facilidades conhecidas como entidades que permite a inclusão de arquivos ou trechos de texto via referência. Com a utilização deste conceito este trabalho praticamente desaparece, pois simplifica a tarefa de manutenção da consistência dos dados.

Além disso o XML pode proporcionar conexões mais interessantes e abrangentes com a utilização ou não de figuras, como é explicitado na Seção 3.2.5.

3.2.5 Documentos com Conexões Diversas

O grande diferencial do XML com os recursos existentes hoje na rede para a conexão de textos ou trechos de textos está nos mecanismos proporcionados pelo *XLink* e *XPointer*.

Por ser extremamente interessante e novo, a análise deste tipo de documento merece uma atenção especial, uma vez que o XML oferece alternativas de conexões hoje não possíveis de se fazer na *Web* sem a utilização dos mecanismos citados.

Buscando estas novas possibilidades, serão testados *links* biderecionais e multidirecionais; *links* entre trechos de um mesmo documento e entre documentos localizados em diferentes locais; *links* agregados que permitem acesso a fontes múltiplas de documentos ou trechos deles.

Estas novas formas de conexão proporcionam páginas mais interessantes de manusear, mais dinâmicas e didáticas, além de facilitar o uso pelo usuário, tornando a página mais atrativa.

3.2.6 Documentos com Outros Recursos Interessantes

Outros recursos interessantes ou casos especiais do uso do XML podem exigir atenção especial depois que for feito um estudo mais aprofundado da linguagem XML.

Assim, deixamos esta Seção para apresentar estes casos que forem surgindo e se tornarem relevantes para o nosso projeto.

3.3 Ferramentas para o Uso do XML

Para usar o XML são necessários poucos recursos computacionais: um editor de texto e um *browser*.

O editor de texto, por exemplo *WordPad*, *Microsoft Word*, é onde o documento XML será escrito e salvo com a extensão .xml. O mesmo serve para os arquivos adicionais como: os arquivos que fazem a formatação do documento XML que tem extensão .css ou .xsl e os arquivos que contêm os DTD's com extensão .dtd.

O *browser* é onde o documento XML é visto. Para que as instruções escritas em um documento XML seja interpretada corretamente, é necessário que a versão do *browser* em questão possa possibilitar essa interpretação. Existem hoje dois *browsers* que realizam essa conversão corretamente. São eles: o *Internet Explorer* Versões 5.0 ou superior e o Mozilla.

3.4 XML × Textos Literários

A grande vantagem de se escrever um texto literário usando XML está na possibilidade de estruturação do texto. Dessa forma, para que um texto de literatura seja publicado usando XML, primeiramente precisamos saber o que caracteriza um documento XML como bem-estruturado.

Para que um documento XML seja bem-estruturado ele deve obedecer às oito regras descritas a seguir [22]:

1. A declaração XML deve iniciar o documento.

   Por declaração XML entende-se uma linha que deve iniciar o documento para que ele seja corretamente interpretado. Esta linha é:

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

   onde *version* indica a versão do XML e *standalone* indica a necessidade de se fazer referências a outros arquivos quando do uso de entidades. Se *yes*, o documento XML em questão não necessita de outros arquivos para ser interpretado, caso contrário, estes arquivos são necessários e devem ser incluídos no documento.

2. Elementos que contêm dados, devem ter *tags* de início e fim obrigatoriamente.

   Quando uma *tag* for iniciada via <Documento>, por exemplo, ela deve ser obrigatoriamente fechada com </Documento>. Se isso não ocorrer será emitida uma mensagem de erro.

3. Elementos que não contêm dados e que usam apenas uma única *tag* devem terminar com />.

   Elementos vazios servem apenas para indicar alguma ação que deve ser realizada naquele momento do texto, não tendo nenhum conteúdo. Assim, desejando escrever um elemento como este, ele deve ser <vazio/>, por exemplo.

Um exemplo do uso de elementos vazios para textos literários é o elemento <RISADAS/> na piada abaixo [21]:

```
<?xml version="1.0" standalone="yes"?>
<PIADA>
<PERGUNTA>
- Qual a diferença entre um namorado e um marido?
</PERGUNTA>
<RESPOSTA>
- 45 minutos.
</RESPOSTA>
<RISADAS/>
</PIADA>
```

4. O documento deve conter um elemento que contém todos os outros elementos.

   Um documento XML deve possuir um elemento raiz que contém todos os outros elementos do documento de forma a tornar mais clara a definição da estrutura. Exemplo:

```
Elemento raiz em um poema:
<POEMA>
</POEMA>
Elemento raiz em uma piada:
<PIADA>
</PIADA>
Elemento raiz em um conto:
<CONTO>
</CONTO>
Elemento raiz em um documento:
<DOCUMENTO>
</DOCUMENTO>
```

5. Elementos podem estar aninhados mas não sobrepostos Elementos podem conter outros elementos desde que estes não estejam sobrepostos. Isto significa que se houver o início de uma *tag* dentro de um elemento, esta *tag* deve ser fechada ainda dentro deste elemento. Exemplo:

```
Errado:
<POEMA>
<ESTROFE>
</POEMA>
</ESTROFE>
Certo:
<POEMA>
```

```
<ESTROFE>
</ESTROFE>
</POEMA>
```

6. Os valores dos atributos devem vir entre aspas

   Um atributo é uma informação adicional que pode ser acrescentada a um elemento para informar pontos que descrevem o elemento, mas não são podem ser considerados uma parte constituinte do elemento.

   Uma das grandes polêmicas em relação aos atributos, é quando considerar uma característica como elemento ou atributo, por exemplo:

   Considere um conto.

   O título deste conto deve ser um atributo do elemento conto:

```
<CONTO TITULO="A Velha Contrabandista">
</CONTO>
```

   ou deve ser um outro elemento:

```
<CONTO>
<TITULO>
A Velha Contrabandista
</TITULO>
</CONTO>
```

   Para solucionar este problema, existem umas regras simples que devem ser verficadas antes da especificação da melhor estrutura para o documento, não esquecendo nunca de, quando usar atributos, colocá-los entre aspas. São elas [23]:

   - A informação é estrutural? Um objeto por natureza deve estar obrigatoriamente presente na estrutura, por exemplo, o remetente ou o destinatário de uma carta, é um bom candidato a Elemento.
   - A informação qualifica o conteúdo ou faz parte de um padrão repetitivo? Uma carta pode ter a ela associada um indicaador que identifica o seu tipo como "privada"ou "negócios", não estando relacionado estruturalmente com o elemento carta. Sempre que a informação puder ser descrita por um valor pertencente a uma lista enumerada, tem-se um bom candidato a atributo.
   - A informação está relacionada com o aspecto visual? Os atributos não devem ser usados para descrever o aspecto visual da informação. Por exemplo, se se pretende realçar certas partes do texto, não se deve fazê-lo recorrendo a atributos, mas sim recorrendo a um novo elemento.

Para tornar mais claro essa distinção, seguem alguns exemplos de quando o uso de atributos é recomendado.

```
<IMAGEM HEIGHT="10cm" WIDTH="10cm" >
</IMAGEM>

<CONTO>
<AUTOR ID="A1">Graciliano Ramos></AUTOR>
<AUTOR ID="A2">Lima Barreto></AUTOR>
</CONTO>
```

7. As únicas entidades referenciadas que devem aparecer são: &*amp*;, &*lt*;, &*gt*;, &*apos*; e &*quote*;.

   Entidades referenciadas são *strings* que são substituídas por caracteres reservados da linguagem XML quando o documento é traduzido. Existem cinco entidades referenciadas pré-definidas em XML. São elas:

   &*amp*; que representa o caracter &

   &*lt*; que representa o caracter <

   &*gt*; que representa o caracter >

   &*apos*; que representa o caracter “

   &*quote*; que representa o caracter ‘

8. Os caracteres < e & devem ser utilizados somente para iniciar *tags* e entidades referenciadas respectivamente.

   Por serem caracteres reservados o seu uso fica restrito a essas funcionalidades.

Para mostrar como todas estas características são importantes para a publicação de um texto literário, segue um exemplo de um documento bem-estruturado [20]. Após a declaração foi inserida uma linha de comentário que são informações inseridas ao longo de um documento para fornecer dados extras que o autor queira informar ao leitor. Estes comentários não são exibidos no *browser*.

```
<?xml version="1.0" standalone="yes"?>
<CONTO>
<!-- Este conto foi extraído do livro Para Gostar de Ler Volume 8 Contos -->
<TITULO>
A VELHA CONTRABANDISTA
</TITULO>
<NARRADOR>
<P>Diz que era uma velhinha que sabia andar de lambreta. Todo dia ela
passava pela fronteira montada numa lambreta, com um bruto saco atrás
da lambreta. O pessoal da alfândega  tudo malandro velho  começou
```

a desconfiar da velhinha.</P>
<P>Um dia, quando ela vinha na lambreta com um saco atrás, o fiscal da alfândega mandou ela parar. A velhinha parou e então o fiscal perguntou assim para ela:</P>
</NARRADOR>
<FISCAL>
<P>- Escuta aqui, vovozinha, a senhora passa por aqui todo dia, com este saco aí atrás. Que diabo a senhora leva neste saco?</P>
</FISCAL>
<NARRADOR>
<P>A velhinha sorriu com os poucos dentes que lhe restavam e mais os outros, que ela adquirira no odontólogo, e respondeu:</P>
</NARRADOR>
<VELHINHA>
<P>- É areia!</P>
</VELHINHA>
<NARRADOR>
<P>Aí quem sorriu foi o fiscal. Achou que não era areia nenhuma e mandou a velhinha saltar da lambreta para examinar o saco. A velhinha saltou, o fiscal esvaziou o saco e dentro só tinha areia. Muito encabulado, ordenou à velhinha que fosse em frente. Ela montou na lambreta e foi embora, com o saco de areia atrás.</P>
<P>Mas o fiscal ficou desconfiado ainda. Talvez a velhinha passasse um dia com areia e no outro com muamba, dentro daquele maldito saco.</P>
<P>No dia seguinte, quando ela passou com a lambreta com o saco atrás, o fiscal mandou parar outra vez. Perguntou o que é que ela levava no saco e ela respondeu que era areia, uai! O fiscal examinou e era mesmo. Durante um mês seguido o fiscal interceptou a velhinha e, todas as vezes, o que ela levava no saco era areia.</P>
<P>Diz que foi aí que o fiscal se chateou:</P>
</NARRADOR>
<FISCAL>
<P>- Olha, vovozinha, eu sou fiscal da alfândega com 40 anos de serviço. Manjo essa coisa de contrabando para burro. Ninguém me tira da cabeça que a senhora é contrabandista.</P>
</FISCAL>
<VELHINHA>
<P>- Mas no saco só tem areia!
<NARRADOR>
insistiu a velhinha.
</NARRADOR> </P>
</VELHINHA>
<NARRADOR>
<P>E já ia tocar a lambreta, quando o fiscal propôs:</P>
</NARRADOR>

```
<FISCAL>
<P>- Eu prometo à senhora que deixo a senhora passar. Não dou parte, não
apreendo, não conto nada a ninguém, mas a senhora vai me dizer: Qual é o
contrabando que a senhora está passando por aqui todos os dias?</P>
</FISCAL>
<VELHINHA>
<P>- O senhor promete que não "espáia"?
<NARRADOR>
quis saber a velhinha.
</NARRADOR> </P>
</VELHINHA>
<FISCAL>
<P>- Juro. respondeu o fiscal.</P>
</FISCAL>
<SUSPENSE/>
<VELHINHA>
<P>- É lambreta.</P>
</VELHINHA>
<RISADAS/>
</CONTO>
```

3.5 XML × Documentos Técnico-Científicos

A grande vantagem de se utilizar XML para publicar documentos técnicos-científicos também está na estruturação. Assim, além dos oito pontos levantados na Seção 3.4, existem outras características do XML que tornam a estruturação mais bem-feita, especialmente para documentos técnicos-científicos. São elas: o uso de Seções CDATA e o uso de DTDs [22].

3.5.1 O Uso de Seções CDATA

Seções CDATA são partes do texto onde os caracteres escritos nestas seções são interpretados como texto puro. Qualquer caractere, seja ele < ou &, é interpretado apenas como texto e não como palavras reservadas se estiverem em uma seção CDATA. Seções CDATA são usadas quando se quer que todo o texto seja interpretado como caracteres puros e não como palavras reservadas. Estas seções são extremamente úteis quando se quer escrever um trecho onde há presença constante das entidades referenciadas, como por exemplo em trechos de código em C[26] ou Java[18]. Um exemplo de como se usa este recurso está no trecho da aplicação Java a seguir. Ele inicia com <!CDATA e termina com]]>, e todo o texto escrito entre eles está na seção CDATA.

```
<!CDATA[/**
 * O HelloWorldApp apenas implementa uma aplicação Java que
 * imprime "Hello World!" como saída.
 */
```

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}]]>
```

3.5.2 O Uso de DTDs

Document Type Definition, DTD, fornece uma lista dos elementos, atributos, notações e entidades presentes em um documento, assim como o relacionamento entre eles. Os DTDs especificam um conjunto de regras para a montagem da estrutura de um documento. Por exemplo, um DTD pode estabelecer que um elemento LIVRO tenha exatamente um TITULO e um ou mais AUTORES.

DTDs podem ser incluídos no arquivo que possui o documento que ele descreve ou ser associado por meio de uma URL externa. Tais DTDs externos podem ser compartilhados por diferentes documentos e *sites* da *Web*. DTDs permitem que organizações ou grupos com interesses em comum concordem em estabelecer padrões para a estrutura de um documento técnico da área que eles atuam.

DTDs também possibilitam que pessoas e programas diferentes possam ler e entender os arquivos uns dos outros. Por exemplo, se os químicos concordarem em adotar um DTD específico como base para a notação química, então fica assegurado que todo o grupo acadêmico de químicos poderá ler e entender os artigos uns dos outros. O DTD define exatamente o que pode ou não pode aparecer dentro de um documento. O DTD estabelece um padrão para comunicação entre grupos de áreas afins.

Para que o DTD cumpra o seu papel de possibilitar ao documento uma estrutura mais sofisticada, o documento tem que obedecer às regras impostas pelo DTD. Um documento cuja estrutura obedece às restrições impostas no DTD é dito válido.

Para verificar a validade de um documento existem analisadores sintáticos que ao detectarem erros de sintaxe os reportam ao autor do documento para que a validação seja feita. Alguns desses analisadores sintáticos disponíveis para validação são:

- XML for Java: http://www.alphaworks.ibm.com/tech/xml
- XJParser: http://www.datachannel.com/xmlresources/
- SXP: http://www.loria.fr/projets/XSilfide/EN/sxp

Para se criar um DTD, primeiramente deve-se conhecer a estrutura do documento para então aplicá-las às regras sintáticas do DTD. A gramática usada em um DTD foi apresentada na Seção 2.2, aqui será mostrado um exemplo para ilustrar o seu funcionamento.

O documento a seguir apresenta os elementos de um composto químico. Uma MOLECULA é formada por um ou mais ATOMOS ou por uma ou mais MOLECULAS. O elemento QUIMICA é formado por qualquer tipo de dados representado pelo comando ANY, enquanto o elemento ATOMO é formado por um texto puro representado pelo comando #PCDATA.

```
<?xml version="1.0" standalone="yes">
<!DOCTYPE  QUIMICA[
<!ELEMENT QUIMICA ANY>
<!ELEMENT MOLECULA (ATOMO+ | MOLECULA+)>
<!ELEMENT ATOMO (#PCDATA)>
]>
<QUIMICA>
<MOLECULA TITLE = "Água">
<ATOMO>
H
</ATOMO>
<ATOMO>
H
</ATOMO>
<ATOMO>
O
</ATOMO>
</MOLECULA>

<MOLECULA TITLE = "Sal de Cozinha ou Cloreto de Sódio">

A fórmula do sal de cozinha é NaCl formada pela reação química
dos seguintes elementos:

<MOLECULA TITLE = "Ácido Clorídrico">
<ATOMO>
H
</ATOMO>
<ATOMO>
Cl
</ATOMO>
</MOLECULA>

<MOLECULA TITLE = "Soda Cáustica ou Hidróxido de Sódio">
<ATOMO>
Na
</ATOMO>
<ATOMO>
O
</ATOMO>
<ATOMO>
H
</ATOMO>
</MOLECULA>
</MOLECULA>
```

```
</QUIMICA>
```

3.6 XML × Listagens e Dados Estatísticos

A grande vantagem de se utilizar XML para publicação de Listagens e Dados Estatísticos está na versatilidade da formatação dos documentos que permite que um mesmo documento possa ser visualizado de várias formas diferentes sem a necessidade de reescrevê-lo.

Como em XML *tags* arbritárias podem ser incluídas em um documento, o *browser* não tem controle de como elas serão mostradas. Portanto, quando se publica um documento é necessário que o mesmo tenha um estilo de página, *style sheets*, para que o *browser* possa saber como formatar os elementos do documento. Existem dois tipos de estilos de páginas que podem ser usados em documentos XML. São eles , o CSS, *Cascading Style Sheet* e o XSL, *Extensible Style Sheet Language*, respectivamente nas Seções 3.6.1 e 3.6.2[22].

3.6.1 O CSS - Cascading Style Sheet

Inicialmente desenvolvido para HTML, o CSS define as propriedades de formatação de um elemento do documento como: tamanho da fonte, tipo da fonte, identação de parágrafo e outros estilos que possam ser aplicados a um elemento em particular.

O estilo de página CSS é uma lista de regras. Cada regra estabelece o nome do elemento e o estilo que se deseja aplicar àquele elemento. O CSS possui duas versões: o CSS1 e o CSS2.

Na primeira, o CSS1, apenas algumas formatações simples são permitidas como: modificação do tipo, estilo, cor e tamanho de fonte; inserção de bordas; inserção de fundo de página; espaçamento e alinhamento de caracteres.

Para suprir várias carências do CSS1, foi criado o CSS2. Ele permite formatações mais avançadas como: inserção de tabelas; possibilidade de inserção de textos não presentes no documento XML; capacidade de ordenar textos; possibilidade de inserção de arquivos de áudio e vídeo, entre outras. O grande problema é que todas essas funcionalidades acrescentadas pelo CSS2 ainda não são suportadas por nenhum *browser* existente hoje no mercado, não havendo, por exemplo, uma forma de construir um documento com tabela.

Para ilustrar como funciona o estilo de página CSS será apresentado um exemplo. Neste exemplo será aplicado um estilo de página CSS1 num documento XML que possui dados estatísticos de um campeonato de futebol [27]. Não faz parte deste trabalho apresentar todas as formas de formatação possíveis de serem feitas com o CSS1 ou CSS2. Para tanto basta consultar a especificação oficial [19] da linguagem.

Segue o documento XML com seu respectivo DTD e a chamada para o estilo de página CSS1.

```
<?xml version="1.0" standalone="yes"?>
<?xml-stylesheet type="text/css" href="futebol.css"?>
```

```
<!DOCTYPE  FUTEBOL[
<!ELEMENT FUTEBOL (CAMPEONATO+)>
<!ELEMENT CAMPEONATO (ANO, NOME, DIVISAO, TIME+)>
<!ELEMENT ANO (#PCDATA)>
<!ELEMENT NOME_CAMPEONATO (#PCDATA)>
<!ELEMENT DIVISAO (#PCDATA)>
<!ELEMENT TIME (ESTADO_TIME, TECNICO, PONTOS_GANHOS, JOGOS+,  JOGADOR+)>
<!ELEMENT ESTADO_TIME (#PCDATA)>
<!ELEMENT TECNICO (#PCDATA)>
<!ELEMENT PONTOS_GANHOS (#PCDATA)>
<!ELEMENT JOGOS (NUM_JOGOS, VITORIAS, EMPATES, DERROTAS,
 GOLS_PROS, GOLS_CONTRA, SALDO_GOLS, MEDIA_PUBLICO,
 MEDIA_RENDA )>
<!ELEMENT NUM_JOGOS (#PCDATA)>
<!ELEMENT VITORIAS (#PCDATA)>
<!ELEMENT EMPATES (#PCDATA)>
<!ELEMENT DERROTAS (#PCDATA)>
<!ELEMENT GOLS_PROS (#PCDATA)>
<!ELEMENT GOLS_CONTRA (#PCDATA)>
<!ELEMENT SALDO_GOLS (#PCDATA)>
<!ELEMENT MEDIA_PUBLICO (#PCDATA)>
<!ELEMENT MEDIA_RENDA (#PCDATA)>
<!ELEMENT JOGADOR (APELIDO, POSICAO, NUM_GOLS, NUM_EXPULSOES )>
<!ELEMENT APELIDO (#PCDATA)>
<!ELEMENT POSICAO (#PCDATA)>
<!ELEMENT NUM_GOLS (#PCDATA)>
<!ELEMENT NUM_EXPULSOES (#PCDATA)>
]>


<FUTEBOL>
<CAMPEONATO>
<ANO>1998</ANO>
<NOME_CAMPEONATO>Campeonato Brasileiro</NOME_CAMPEONATO>
<DIVISAO>Serie A</DIVISAO>
<TIME>
<ESTADO_TIME>Minas Gerais</ESTADO_TIME>
<NOME_TIME>Cruzeiro</NOME_TIME>
<TECNICO>Levir Culpi</TECNICO>
<PONTOS_GANHOS>51</PONTOS_GANHOS>
<JOGOS>
<NUM_JOGOS>32</NUM_JOGOS>
<VITORIAS>14</VITORIAS>
<EMPATES>9</EMPATES>
<DERROTAS>9</DERROTAS>
```

<GOLS_PROS>56</GOLS_PROS>
<GOLS_CONTRA>41</GOLS_CONTRA>
<SALDO_GOLS>15</SALDO_GOLS>
<MEDIA_PUBLICO>22923</MEDIA_PUBLICO>
<MEDIA_RENDA>196864</MEDIA_RENDA>
</JOGOS>
<JOGADOR>
<APELIDO>Dida</APELIDO>
<POSICAO>Goleiro</POSICAO>
<NUM_GOLS></NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Ronaldo</APELIDO>
<POSICAO>Lateral Direito</POSICAO>
<NUM_GOLS></NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Marcelo Djian</APELIDO>
<POSICAO>Zagueiro</POSICAO>
<NUM_GOLS>3</NUM_GOLS>
<NUM_EXPULSOES>1</NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Wilson Gottardo</APELIDO>
<POSICAO>Zagueiro</POSICAO>
<NUM_GOLS>1</NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Gustavo</APELIDO>
<POSICAO>Lateral Direito</POSICAO>
<NUM_GOLS></NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Marcos Paulo</APELIDO>
<POSICAO>Volante</POSICAO>
<NUM_GOLS>1</NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Caio</APELIDO>
<POSICAO>Meia-atacante</POSICAO>

```
<NUM_GOLS></NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Djair</APELIDO>
<POSICAO>Volante</POSICAO>
<NUM_GOLS>3</NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>

</JOGADOR>
<JOGADOR>
<APELIDO>Valdo</APELIDO>
<POSICAO>Meia-atacante</POSICAO>
<NUM_GOLS>6</NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Muller</APELIDO>
<POSICAO>Atacante</POSICAO>
<NUM_GOLS>5</NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Fabio Junior</APELIDO>
<POSICAO>Atacante</POSICAO>
<NUM_GOLS>18</NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Marcelo Ramos</APELIDO>
<POSICAO>Atacante</POSICAO>
<NUM_GOLS>8</NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Ricardinho</APELIDO>
<POSICAO>Volante</POSICAO>
<NUM_GOLS>1</NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Alex Alves</APELIDO>
<POSICAO>Atacante</POSICAO>
<NUM_GOLS>6</NUM_GOLS>
<NUM_EXPULSOES>1</NUM_EXPULSOES>
```

```
</JOGADOR>
<JOGADOR>
<APELIDO>Gilberto</APELIDO>
<POSICAO>Lateral Esquerdo</POSICAO>
<NUM_GOLS>2</NUM_GOLS>
<NUM_EXPULSOES>1</NUM_EXPULSOES>
</JOGADOR>

</TIME>
<TIME>
<ESTADO_TIME>Minas Gerais</ESTADO_TIME>
<NOME_TIME>Atletico Mineiro</NOME_TIME>
<TECNICO>Carlos Alberto Torres</TECNICO>
<PONTOS_GANHOS>36</PONTOS_GANHOS>
<JOGOS>
<NUM_JOGOS>23</NUM_JOGOS>
<VITORIAS>9</VITORIAS>
<EMPATES>9</EMPATES>
<DERROTAS>5</DERROTAS>
<GOLS_PROS>38</GOLS_PROS>
<GOLS_CONTRA>33</GOLS_CONTRA>
<SALDO_GOLS>5</SALDO_GOLS>
<MEDIA_PUBLICO>14052</MEDIA_PUBLICO>
<MEDIA_RENDA>125176</MEDIA_RENDA>
</JOGOS>
<JOGADOR>
<APELIDO>Emerson</APELIDO>
<POSICAO>Goleiro</POSICAO>
<NUM_GOLS></NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Paulo Cesar</APELIDO>
<POSICAO>Lateral Direito</POSICAO>
<NUM_GOLS>1</NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Galvan</APELIDO>
<POSICAO>Zagueiro</POSICAO>
<NUM_GOLS>1</NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Claudio Cacapa</APELIDO>
```

<POSICAO>Zagueiro</POSICAO>
<NUM_GOLS></NUM_GOLS>
<NUM_EXPULSOES>1</NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Vitor</APELIDO>
<POSICAO>Lateral Esquerdo</POSICAO>
<NUM_GOLS>1</NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Edgar</APELIDO>
<POSICAO>Volante</POSICAO>
<NUM_GOLS></NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Bruno</APELIDO>
<POSICAO>Volante</POSICAO>
<NUM_GOLS></NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Hernani</APELIDO>
<POSICAO>Meia-atacante</POSICAO>
<NUM_GOLS></NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Juninho Petrolina</APELIDO>
<POSICAO>Meia-atacante</POSICAO>
<NUM_GOLS></NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Lincoln</APELIDO>
<POSICAO>Meia-atacante</POSICAO>
<NUM_GOLS>2</NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Marco Antonio Boiadeiro</APELIDO>
<POSICAO>Volante</POSICAO>
<NUM_GOLS></NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>

```
</JOGADOR>
<JOGADOR>
<APELIDO>Marques</APELIDO>
<POSICAO>Atacante</POSICAO>
<NUM_GOLS>8</NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Valdir</APELIDO>
<POSICAO>Atacante</POSICAO>
<NUM_GOLS>18</NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
<JOGADOR>
<APELIDO>Edmilson</APELIDO>
<POSICAO>Atacante</POSICAO>
<NUM_GOLS>2</NUM_GOLS>
<NUM_EXPULSOES></NUM_EXPULSOES>
</JOGADOR>
</TIME>
</CAMPEONATO>
</FUTEBOL>
```

O estilo de página CSS1 a ser aplicado ao documento é o futebol.css. Este arquivo encontra-se vazio, mostrando portanto todos os dados do documento de maneira não formatada. A Figura 3 mostra como é visualizado o futebol.xml no Interntet Explorer 5.0.

3.6.2 O XSL - Extensible Style Sheet Language

O XSL ainda está em desenvolvimento. A linguagem XSL mudou radicalmente no passado e certamente continuará mudando no futuro próximo. Assim, alguns dos recursos utilizados nesta Seção podem não funcionar mais em breve. O XSL é a linguagem mais adequada para fazer a formatação de documentos XML, pois permite todas as formatações do CSS1 e CSS2 e é suportado pelos *browsers* existentes hoje.

O XSL compreende uma linguagem de tranformação e uma linguagem de formatação. Estas duas linguagens podem funcionar independentemente uma da outra. Por exemplo, a linguagem de transformação pode transformar um documento XML em um documento HTML bem-estruturado e ignorar completamente os objetos de formatação do XSL.

A linguagem de transformação fornece elementos que definem as regras de como transformar um documento XML em um outro documento XML. O documento XML transformado usa as *tags* e o DTD do documento original ou usa um conjunto de *tags* completamente diferente. A transformação pode ocorrer de três maneiras diferentes [22]:

1. O documento XML é associado a um documento XSL e ambos são enviados ao cliente, *Web browser*, que irá transformar o documento XML de acordo com o que

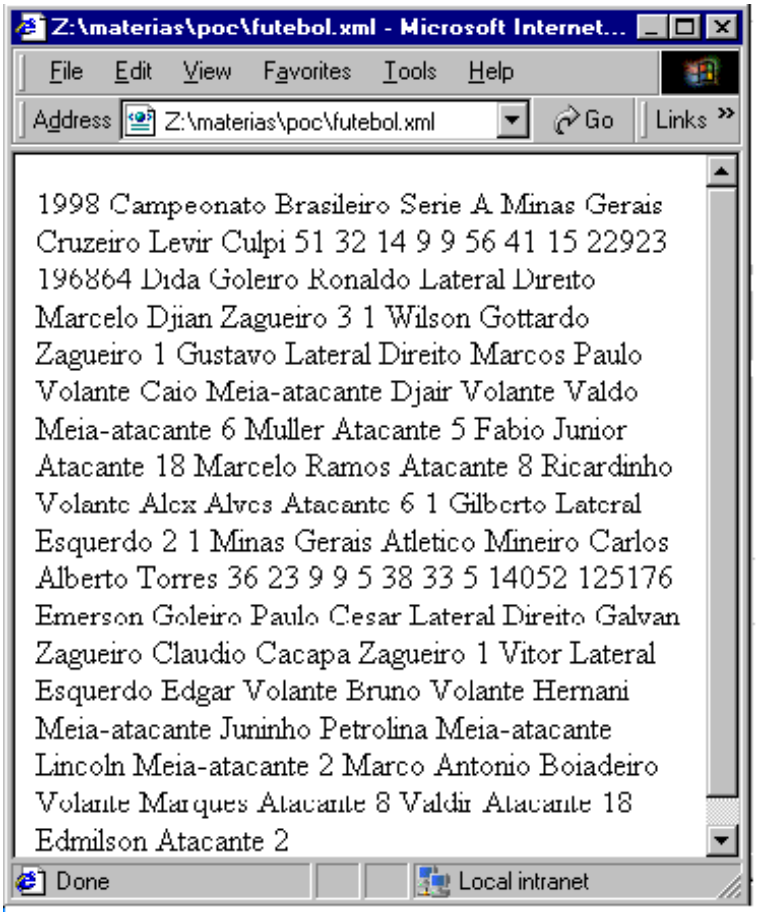

Figura 3: Futebol.xml no Internet Explorer 5.0 com documento de estilo de página vazio.

foi especificado no estilo de página. O documento é, então, apresentado ao usuário já formatado. Um exemplo é a visualização de um documento XML no Internet Explorer 5.0. Esta abordagem ainda não é possível de ser feita com perfeição, pois os *browsers* existentes no mercado, inclusive o Internet Explorer 5.0, ainda não conseguem fazer uma tradução bem-feita.

2. O servidor aplica um estilo de página XSL em um documento XML para tranformá-lo em outro formato, geralmente HTML, e retorna o documento transformado para ser visualizado no cliente, *Web browser*. Essa abordagem requer softwares especiais para servidores que hoje não são satisfatórios.

3. Um programa de terceiros transforma o documento XML original em outro formato, geralmente HTML, antes do documento ser colocado no servidor. Assim, tanto o servidor quanto o cliente lidam com o documento já transformado. Esta abordagem é a que se encontra mais desenvolvida e aplicável no momento via programas como o o XT [24] ou o LotusXSL [25].

Nesta Seção são apresentados exemplos de formatação do documento XML sobre futebol da Seção 3.6.1 usando a terceira maneira descrita acima. Não faz parte deste trabalho apresentar todas as formas de formatação possíveis de serem feitas com o XSL. Para tanto basta consultar a especificação oficial [19] da linguagem.

O primeiro estilo de página XLS a ser aplicada no documento apenas faz com que o texto do documento XML seja exibido. Assim, o estilo de página que se segue gera a mesma Figura 3 da Seção 3.6.1.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

<xsl:template match="/">
<html>
<xsl:apply-templates/>
</html>
</xsl:template>

<xsl:template match="FUTEBOL">
<body>
<xsl:apply-templates/>
</body>
</xsl:template>

<xsl:template match="CAMPEONATO">
<xsl:value-of select="ANO"/>
<xsl:value-of select="NOME_CAMPEONATO"/>
<xsl:value-of select="DIVISAO"/>
<xsl:value-of select="TIME"/>
```

```
<xsl:value-of select="TIME"/>
</xsl:template>

</xsl:stylesheet>
```

O segundo estilo de página aplicado sobre o documento XML em questão é o que se segue e o resultado está apresentado nas Figuras 4 e ??.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

<xsl:template match="/">
<html>
<xsl:apply-templates/>
</html>
</xsl:template>

<xsl:template match="FUTEBOL">
<body>
<xsl:apply-templates/>
</body>
</xsl:template>

<xsl:template match="CAMPEONATO">
<title>
<xsl:value-of select="ANO"/>
<xsl:value-of select="NOME_CAMPEONATO"/>
<xsl:value-of select="DIVISAO"/>
</title>
<h1>
<center>
<xsl:value-of select="ANO"/>
</center>
</h1>
<h2>
<center>
<xsl:value-of select="NOME_CAMPEONATO"/>
-
<xsl:value-of select="DIVISAO"/>
</center>
</h2>
<xsl:apply-templates select="TIME"/>
</xsl:template>

<xsl:template match="TIME">
<h3>
```

```
<xsl:value-of select="NOME_TIME"/>
-
<xsl:value-of select="ESTADO_TIME"/>
</h3>
<b>
Tecnico:
<xsl:value-of select="TECNICO"/>
<br/>
Pontos Ganhos:
<xsl:value-of select="PONTOS_GANHOS"/>
</b>
<br></br>
<table border="1" >
<th>Num.Jogos</th>
<th>Vitorias</th>
<th>Empates</th>
<th>Derrotas</th>
<th>Gols Pros</th>
<th>Gols Contra</th>
<th>Saldo Gols</th>
<th>Media Publico</th>
<th>Media Renda(R$)</th>
<xsl:apply-templates select="JOGOS"/>
</table>
<br></br>
<table border="1">
<th>Nome Jogador</th>
<th>Posicao</th>
<th>Num.Gols</th>
<th>Num.Expulsoes</th>
<xsl:apply-templates select="JOGADOR"/>
</table>
</xsl:template>

<xsl:template match="JOGOS">
<tr>
<td><xsl:value-of select="NUM_JOGOS"/></td>
<td><xsl:value-of select="VITORIAS"/></td>
<td><xsl:value-of select="EMPATES"/></td>
<td><xsl:value-of select="DERROTAS"/></td>
<td><xsl:value-of select="GOLS_PROS"/></td>
<td><xsl:value-of select="GOLS_CONTRA"/></td>
<td><xsl:value-of select="SALDO_GOLS"/></td>
<td><xsl:value-of select="MEDIA_PUBLICO"/></td>
<td><xsl:value-of select="MEDIA_RENDA"/></td>
```

```
</tr>
</xsl:template>

<xsl:template match="JOGADOR">
<tr>
<td><xsl:value-of select="APELIDO"/></td>
<td><xsl:value-of select="POSICAO"/></td>
<td><xsl:value-of select="NUM_GOLS"/></td>
<td><xsl:value-of select="NUM_EXPULSOES"/></td>
</tr>
</xsl:template>

</xsl:stylesheet>
```

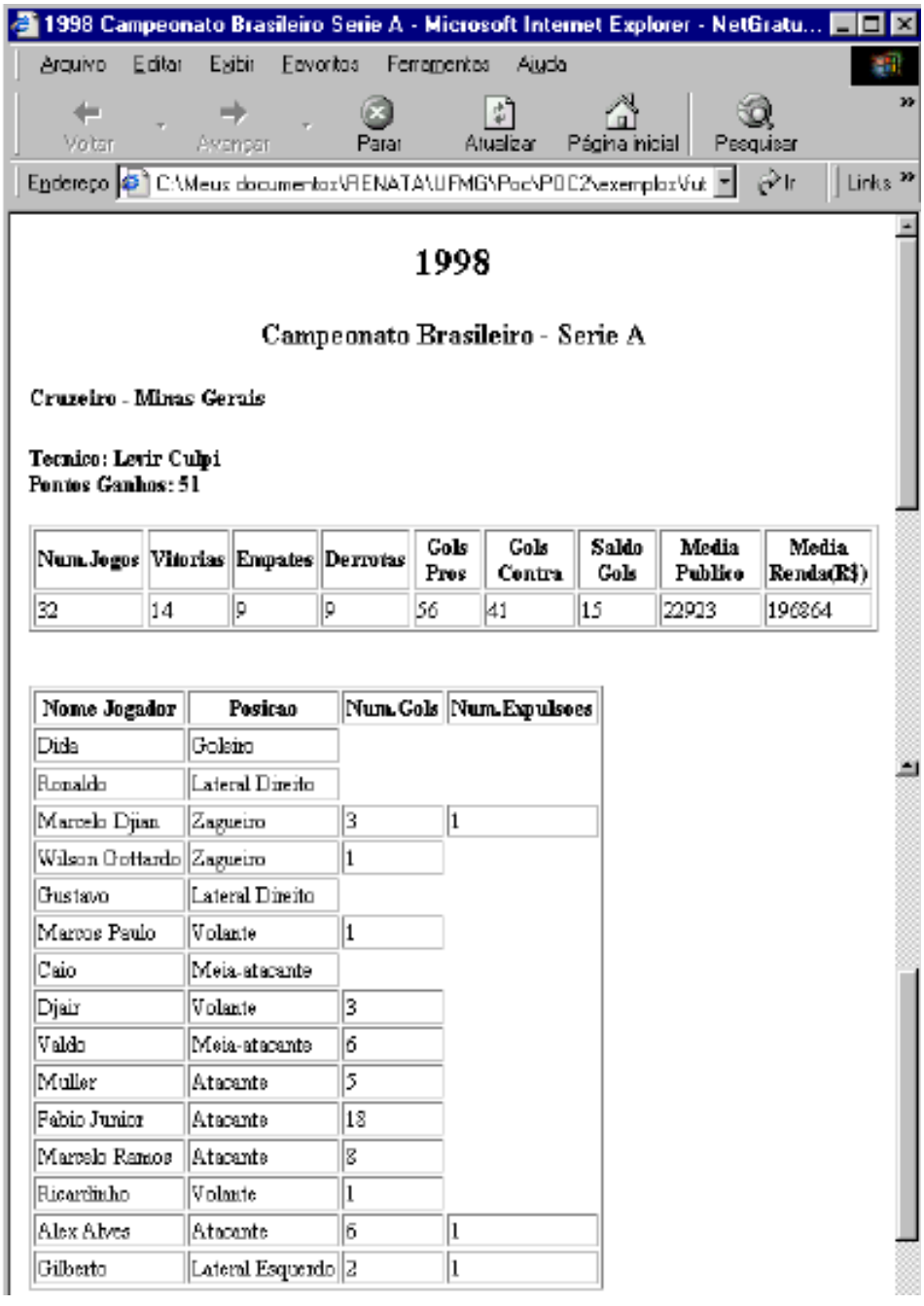

Figura 4: futebolxsl.xml no Internet Explorer 5.0

x

A linguagem de formatação descreve como o conteúdo deve ser apresentado ao usuário. Geralmente, um estilo de página utiliza a linguagem de transformação do XSL para transformar um documento XML em um novo documento que utiliza o vocabulário dos objetos de formatação do XSL.

Embora muitos pesquisadores acreditem que os *Web browsers* sejam capazes de fazer essa formatação sem a necessidade de um objeto de formatação, atualmente ainda é necessário que o documento de saída seja transformado em outro formato, por exemplo, para o PDF.

3.7 XML × Figuras

A grande vantagem de se utilizar XML para publicação de documentos com figuras está capacidade de possibilitar que a sua troca, quando necessária, seja feita da forma menos trabalhosa possível. Para explicar como isso é possível e como o XML facilita essa tarefa, primeiramente temos que explicar o conceito de entidades.

Entidades são unidades de armazenamento de partes de um documento XML, por exemplo, todos os arquivos XML e os estilos de páginas são entidades. As entidades geralmente possuem nomes por meio dos quais elas são referenciadas. Existem dois tipos de entidades: internas e externas.

Entidades internas são definidas no DTD do documento XML e possui uma funcionalidade bastante interessante: permite a substituição de textos dentro de um documento, facilitando não só na escrita do documento como contribuindo para o reaproveitamento de informações.

O documento a seguir apresenta uma aplicação de entidades internas.

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE DOCUMENTO [
<!ENTITY RCC "Renata Castro Carvalho">
<!ELEMENT DOCUMENTO (TITULO, ASSINATURA)>
<!ELEMENT TITULO (#PCDATA)>
<!ELEMENT ASSINATURA (EMAIL, ULTIMA_MODIFICACAO, ASSINA)>
<!ELEMENT EMAIL (#PCDATA)>
<!ELEMENT ULTIMA_MODIFICACAO(#PCDATA)>
<!ELEMENT ASSINA (#PCDATA)>
]>
<DOCUMENTO>
<TITULO> A assinatura de &RCC;</TITULO>
<ASSINATURA>
<EMAIL>renata@dcc.ufmg.br</EMAIL>
<ULTIMA_MODIFICACAO> Dezembro de 2000</ULTIMA_MODIFICACAO>
<ASSINA >&RCC;</ASSINA>
</ASSINATURA>
</DOCUMENTO>
```

A Figura 5 mostra como este documento é exibido no Internet Explorer 5.0.

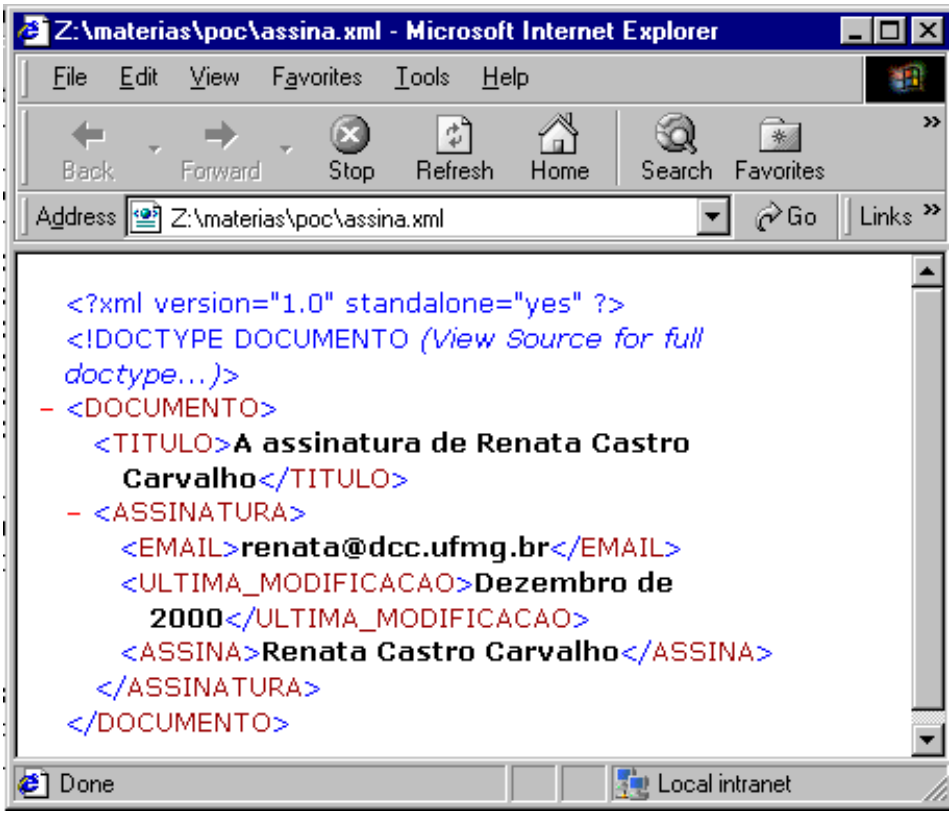

Figura 5: assina.xml no Internet Explorer 5.0

Entidades externas buscam seu conteúdo de outra fonte externa através de uma URL. O documento principal apenas contém a referência para a URL onde o documento original reside.

O documento a seguir apresenta exemplos de entidades externas. Observe que se por algum motivo o endereço de e-mail de Renata mudar, ou a figura inserida precisar ser modificada, apenas a declaração no DTD precisa ser modificada e não o conteúdo do documento XML.

```
<?xml version="1.0" standalone="no"?>
<?xml-stylesheet type="text/css" href="projeto.css"?>
<!DOCTYPE PROJETO [
<!ENTITY ASS SYSTEM "assina2.xml">
<!ENTITY FIG SYSTEM "Files.gif" NDATA GIF>
<!NOTATION GIF SYSTEM "image/gif">
<!ELEMENT IMAGEM EMPTY>
<!ATTLIST IMAGEM SOURCE ENTITY #REQUIRED>
<!ELEMENT PROJETO ANY>
<!ELEMENT INTRODUCAO (#PCDATA)>
<!ELEMENT CONCLUSAO (#PCDATA)>
```

```
<!ELEMENT ASSINATURA (#PCDATA)>
]>
<PROJETO>
<INTRODUCAO>
Este projeto e apenas um exemplo do uso de entidades externas
e contem apenas uma figura de um arquivo.
</INTRODUCAO>
<IMAGEM SOURCE="FIG"/>
<CONCLUSAO> Bem-sucedido</CONCLUSAO>
<ASSINATURA>&ASS;</ASSINATURA>
</PROJETO>
```

3.8 XML × Conexões

A grande vantagem da utilização do XML para os documentos com conexões é a possibilidade se serem realizadas formas de conexão entre documentos que hoje não são possíveis e facilitar a construção de conexões já em uso constante. Esse avanço só é possível devido a duas linguagens: o XLink e o XPointer.

3.8.1 O XLink

O XLink define como um documento será conectado com outro documento. Para isso, ela aponta para uma URL que especifica o recurso alvo. O XLink suporta *links* multidirecionais onde o *link* pode ser direcionado para mais de uma direção. Além disso, qualquer elemento pode se tornar um *link* e os *links* não precisam estar armazenados no mesmo arquivo do documento que os utiliza.

O primeiro tipo de conexão que pode ser utilizada com XLink são as Conexões Simples. Essa conexão apenas direciona o documento original para o alvo especificado. Para fazer essa conexão em XML é necessário que seja criado um Elemento de conexão identificado pelo atributo `xlink:form` com um valor `simples`. Além disso, cada elemento de conexão apresenta um atributo `href` cujo valor é a URL alvo. Um elemento de conexão também possui alguns outros atributos que facilitam a conexão. São eles:

content-title: Este atributo dá um título ao propósito do *link* estar naquela página. Geralmente é utilizado para que seja mostrada uma informação extra ao leitor do documento XML a respeito do *link*. Esta informação pode ser exibida na barra de status do *browser* quando o usuário passa o cursor do *mouse* sobre o *link*.

content-role: Este atributo descreve o propósito daquele *link* estar inserido no documento XML. Pode ser ou não exibido de alguma forma ao usuário depende do desejado.

title: Este atributo descreve o recurso remoto dando um título ao documento alvo.

role: Este atributo descreve o papel do documento alvo, podendo ser usado para distinguir entre citações, notas de rodapé, etc.

show: Este atributo descreve como o conteúdo deve ser exibido quando o *link* é ativado. Ele pode ter três valores: `replace, new` e `embed`. Quando o valor `replace` é indicado, o documento alvo é aberto na mesma janela do documento original. Quando o valor `new` é indicado, o documento alvo é aberto em uma outra janela que é ativada automaticamente. Quando o valor `embed` é indicado o documento alvo é inserido no documento original.

actuate: Este atributo está relacionado com a ativação ou não do *link*. Ele pode ter dois valores: `user` ou `auto`. Quando o valor `user` é indicado, o *link* é ativado apenas quando o usuário clica nele. Quando o valor `auto` é indicado, o *link* é ativado sempre que se abrir o documento XML.

behavior: Este atributo é usado para transmitir dados em formatos diversos para a aplicação que está lendo os dados. A aplicação usa este dado para realizar tarefas adicionais quando o *link* é ativado. Por exemplo, quando se deseja tocar uma música quando se clicar em determinado *link*.

Um exemplo desse tipo de conexão é a que se segue:

```
<!ENTITY % links-attributes>
"xlink:form CDATA   #FIXED 'simple'
 href CDATA #REQUIRED
 behavior CDATA #IMPLIED
 content-hole CDATA #IMPLIED
 content-title CDATA #IMPLIED
 role CDATA #IMPLIED
 title CDATA #IMPLIED
 show (new | replace | embed)
 actuate (user | auto)"
>
<!ELEMENT AUTOR (#PCDATA)>
<!ATTLIST AUTOR %link-attributes;>

<AUTOR href="http://www.dcc.ufmg.br/~renata"
content-title="autor da página"
content-role="quem contactar em caso de dúvida"
title="Página da Renata"
role="Mais informações sobre o autor do projeto"
show="new"
actuate="user"
behavior="musica.au">
Renata Castro Carvalho
Olhe esta página, sem deixar este site completamente.
</AUTOR>
```

Outro tipo de conexão que pode ser feita utilizando XLink são as Conexões Extendidas. Esse tipo de *link* permite a inclusão de *links* multidirecionais, ou seja, direciona o documento XML para diferentes páginas de acordo com certa premissa. Este tipo de conexão possui o elemento `locator` que possui dois atributos básicos além dos já descritos no item anterior: `href` indica os diferentes documentos alvos e `role` indica a premissa que deve ser observada. Exemplo [22]:

```
<WEBSITE xlink:form="extended"
   content-title="Cafe au Lait"
   content-role="Novidades sobre Java">

<locator href="http://sunsite.univie.ac.at/jcca/mirrors/javafaq/
   role=".at"/>

<locator href="http://metalab.unc.edu/javafaq/
   role=".us"/>

<locator href="http://sunsite.icm.edu.pl/java-corner/faq/
   role=".pl"/>

<locator href="http://sunsite.uakom.sk/javafaq/
   role=".sk"/>

<locator href="http://sunsite.cnlab-switch.ch/javafaq/
   role=".ch"/>

</WEBSITE>
```

Além desses dois tipos de *links*, existe outra forma de se fazer a conexão via *links out-of-line*. O *Links Out-of-Line* permite que o alvo não esteja presente no documento, ao invés disso, os *links* são armazenados em um documento próprio que estabelece as conexões. Esta conexão é útil, por exemplo, para manter a sequência na apresentação de transparências, onde cada uma aponta para a próxima e para a anterior.

3.8.2 O XPointer

O XPointer especifica mais detalhadamente o alvo dentro do documento que foi conectado, ou seja, aponta para uma parte específica do documento alvo. Esta propriedade é interessante pois o usuário pode desejar ir para um determinado local do documento. Por exemplo, no documento estatístico sobre futebol, um leitor poderia querer ir apenas para a parte do documento que fala do seu time e não ser posicionado no início do documento para ter que procurar a parte desejada.

Além dessa função, o XPointer pode ser utilizado para mais duas atividades: fornecer um protocolo para que o servidor *Web* envie para o *browser* apenas uma parte do documento

ao invés do documento inteiro e assegurar que o documento solicitado faça sentido no contexto do documento original.

No entanto, como o XPointer ainda está em desenvolvimento pelo W3C [19], apenas a primeira função pode ser utilizada. Nesta função o XPointer é usado para carregar todo o documento solicitado, mas posiciona o cursor na parte indicada por ele. Assim, iremos mostrar os recursos do XPointer que possibilitam esse posicionamento. Como exemplo considere o documento XML abaixo:

```
<?xml version="1.0"?>
<!DOCTYPE ARVOREGENEALOGICA[
<!ELEMENT ARVOREGENEALOGICA (PESSOA| FAMILIA)+>
<!ELEMENT PESSOA (NOME*, NASCIMENTO*, MORTE*, COMPANHEIRO*)>
<!ATTLIST PESSOA
ID  ID  #REQUIRED
PAI  CDATA  #IMPLIED
MAE  CDATA  #IMPLIED>
<!ELEMENT NOME (#PCDATA)>
<!ELEMENT NASCIMENTO (#PCDATA)>
<!ELEMENT MORTE (#PCDATA)>
<!ELEMENT COMPANHEIRO EMPTY>
<!ATTLIST COMPANHEIRO IDREF IDREF #REQUIRED>
<!ELEMENT FAMILIA (MARIDO?, ESPOSA?, FILHO+)>
<!ATTLIST FAMILIA ID  ID  #REQUIRED>
<!ELEMENT MARIDO EMPTY>
<!ATTLIST MARIDO IDREF IDREF #REQUIRED>
<!ELEMENT ESPOSA EMPTY>
<!ATTLIST ESPOSA IDREF IDREF #REQUIRED>
<!ELEMENT FILHO EMPTY>
<!ATTLIST FILHO IDREF IDREF #REQUIRED>
]>

<ARVOREGENEALOGICA>
<PESSOA ID="p1">
<NOME> Maria Aparecida da Silva </NOME>
<NASCIMENTO>11/02/1858</NASCIMENTO>
<MORTE>12/05/1898</MORTE>
<COMPANHEIRO IDREF="p2"/>
</PESSOA>

<PESSOA ID="p2">
<NOME> Joao Carlos da Silva </NOME>
<COMPANHEIRO IDREF="p1"/>
</PESSOA>
```

```
<PESSOA ID="p3" PAI="p2" MAE="p1">
<NOME> Joana da Silva Moreira </NOME>
<NASCIMENTO>11/02/1878</NASCIMENTO>
<MORTE>12/05/1908</MORTE>
<COMPANHEIRO IDREF="p4"/>
</PESSOA>

<PESSOA ID="p4">
<NOME> Roberto Moreira </NOME>
<COMPANHEIRO IDREF="p3"/>
</PESSOA>

<PESSOA ID="p7">
<NOME>Juca Santos </NOME>
<COMPANHEIRO IDREF="p6"/>
</PESSOA>

<PESSOA ID="p6" PAI="p2" MAE="p1">
<NOME> Maria da Silva </NOME>
<COMPANHEIRO IDREF="p7"/>
</PESSOA>

<PESSOA ID="p5" PAI="p2" MAE="p1">
<NOME> Eugenia da Silva </NOME>
</PESSOA>

<PESSOA ID="p8" PAI="p2" MAE="p1">
<NOME> Luisa da Silva </NOME>
<NASCIMENTO>29/10/1861</NASCIMENTO>
<MORTE>11/05/1921</MORTE>
<COMPANHEIRO IDREF="p9"/>
</PESSOA>

<PESSOA ID="p9">
<NOME> Charles Castro </NOME>
<NASCIMENTO>1861</NASCIMENTO>
<MORTE>1938</MORTE>
<COMPANHEIRO IDREF="p8"/>
</PESSOA>

<PESSOA ID="p10" PAI="p2" MAE="p1">
<NOME> Vitor da Silva </NOME>
<COMPANHEIRO IDREF="p11"/>
</PESSOA>
```

```
<PESSOA ID="p11">
<NOME> Helena Ribeiro </NOME>
<COMPANHEIRO IDREF="p10"/>
</PESSOA>

<PESSOA ID="p12" PAI="p2" MAE="p1">
<NOME> Honorio da Silva </NOME>
</PESSOA>

<FAMILIA>
<MARIDO IDREF="p2"/>
<ESPOSA IDREF="p1"/>
<FILHO IDREF="p3"/>
<FILHO IDREF="p5"/>
<FILHO IDREF="p6"/>
<FILHO IDREF="p8"/>
<FILHO IDREF="p10"/>
<FILHO IDREF="p12"/>
</FAMILIA>

<FAMILIA>
<MARIDO IDREF="p7"/>
<ESPOSA IDREF="p6"/>
</FAMILIA>
</ARVOREGENEALOGICA>
```

XPointers são construídos via termos de localização. Cada um desses termos especifica um ponto no documento alvo, geralmente relacionado a algum outro ponto já conhecido do documento como o seu início ou outro termo de localização. O tipo do termo de localização é dado por palavras chaves como:

id: Este termo de localização é um dos mais simples e o mais usado. Ele seleciona o elemento no documento que tem um atributo do tipo ID com o valor especificado. Exemplo: a URL http://www.familia.com/familia.xml#id(p12) seleciona o seguinte elemento do documento:

```
<PESSOA ID="p12" PAI="p2" MAE="p1">
<NOME> Honorio da Silva </NOME>
</PESSOA>
```

root: Este termo de localização aponta para o elemento raiz do documento. Ele é mais usado em conjunto com outros termos de localização. Se nenhum termo for informado, o `root` é considerado o *default*.

html: Este termo de localização seleciona um elemento HTML que esteja nomeado. Ele foi criado para permitir a compatibilidade entre documentos HTML e XML.

child: Este termo de localização seleciona do elemento filho imediato ao elemento fonte. Por exemplo, a seguinte URL http://www.famila.com/familia.xml#root().child(1,PESSOA).child(1,NOME) seleciona:

```
<NOME> Maria Aparecida da Silva </NOME>
```

descendant: Este termo de localização procura entre os descendentes do elemento fonte e não apenas os elementos filhos imediatos. Assim, a URL http://www.famila.com/familia.xml#root().descendant(3,NASCIMENTO) seleciona o seguinte elemento de Luisa da Silva:

```
<NASCIMENTO>29/10/1861</NASCIMENTO>
```

ancestor: Este termo de localização procura entre os ancestores do elemento fonte. Assim, a URL http://www.famila.com/familia.xml#root().descendant(2,NASCIMENTO).ancestor(1) seleciona o elemento <PESSOA> de Joana da Silva Moreira.

preceding: Este termo de localização procura entre todos os elementos que antecedem o elemento fonte. Assim, a URL http://www.famila.com/familia.xml#root().descendant(3,NASCIMENTO).preceding(5) seleciona o seguinte elemento de Luisa da Silva:

```
<NASCIMENTO>29/10/1861</NASCIMENTO>
```

following: Este termo de localização procura entre todos os elementos que sucedem o elemento fonte. Assim, a URL http://www.famila.com/familia.xml#root().descendant(2,NASCIMENTO).following(5) seleciona o seguinte elemento:

```
<NOME> Roberto Moreira </NOME>
```

psibling: Este termo de localização seleciona o elemento que precede o elemento fonte dentro de um mesmo elemento filho. Assim, a URL http://www.famila.com/familia.xml#root().descendant(2,NASCIMENTO).psibling(1) seleciona o seguinte elemento:

```
<NOME> Joana da Silva Moreira </NOME>
```

fsibling: Este termo de localização seleciona o elemento que sucede o elemento fonte dentro de um mesmo elemento filho. Assim, a URL http://www.famila.com/familia.xml#root().descendant(2,NASCIMENTO).fsibling(1) seleciona o seguinte elemento de Joana da Silva Moreira:

```
<MORTE>12/05/1908</MORTE>
```

Além desses termos de localização, existem outros recursos que facilitam a localização e que podem ser aplicada sobre esses termos: os argumentos do termos de localização. Estes argumentos podem ou não estar presentes nos termos de localização. São eles:

número: Este argumento é o primeiro elemento do termo de localização e indica o número de elementos que devem ser contados. Números positivos contam para frente e números negativos para trás. Exemplo: `root(),descendant(1)` onde 1 é o argumento.

tipo: Este argumento é o segundo elemento do termo de localização e indica o tipo do elemento a ser selecionado. O tipo pode ser: `#element`, para elementos; `#pi`, para instruções de processamento; `#comment`, para comentários; `#text` para qualquer texto que não seja marcador; `#cdata`, para seções cdata; `#all`, para todos os anteriores. Exemplo:`root().child(6,FILHO)` onde FILHO é o argumento.

atributo: Este argumento pode ser o terceiro e quarto elementos do termo de localização. O terceiro argumento é o nome do atributo. O quarto elemento é o valor do atributo. Exemplo: `root().child(1,PESSOA,PAI,p2` onde PAI e p2 são o nome e o valor do atributo respectivamente.

3.9 Conclusão

O objetivo dessa Seção do projeto foi mostrar a funcionalidade das principais características do XML, via exemplos, em diferentes tipos de documentos: textos e livros literários; artigos e livros técnico-científicos; listagens e dados estatísticos; textos, documentos com figuras; documentos com conexões diversas entre outros.

Em textos e livros literários foi observado que o XML é de grande importância pois garante a construção de um texto bem-estruturado. Essa estrutura mostrou-se necessária por possibilitar o entendimento do assunto do texto; possibilitar a caracterização dos personagens do texto; possibilitar a indentificação das ações a serem realizadas no caso de uma peça de teatro entre outras características interessantes. Dessa forma, um documento bem-estruturado pode tornar este tipo de documento mais inteligível mesmo para um leigo em XML.

Em artigos e livros técnico-científicos foi observado que o XML além de proporcionar uma estrutura bem elaborada para o documento também permite a validação desta estrutura via DTDs e permite a inclusão de trechos de códigos específicos de cada área de estudo. A estruturação neste tipo de documento garante a padronização de *tags* para áreas técnicas-científicas o que facilita o entendimento e a divulgação de documentos entre os estudiosos de cada assunto. Para que todas essas características sejam mais abrangentes estão sendo criada variações do XML para como: o CML, *Chemical Markup Language*, o MathML, *Mathematical Markup Language*, o MusicML, *Music Markup Language*, dentre outros.

Em documentos com listagens e dados estatísticos foi observado que o XML além de proporcionar as vantagens de uma boa estruturação, também permite que formatações diferentes sejam aplicadas aos documentos via CSS ou XSL. Apesar do XSL estar em desenvolvimento e não ter sido usado neste momento, o CSS apresentou-se satisfatório no sentido de proporcionar diferentes formatos de documentos sem necessidade de reescrevê-los. Em relação à estruturação, o XML permite que os dados estátisticos sirvam de entrada para outros programas, que não foi devidamente abordado neste projeto, mas merece ser citado pois é um grande diferencial em relação às outras linguagens de marcação.

Em documentos com figuras foi observado que o XML pode expandir e facilitar o uso de figuras na Internet, tornando possível a utilização dessas figuras de maneira mais interessante e funcional. O principal avanço está na capacidade de que, via Entidades, as modificações de figuras tornam-se mais fácil e menos trabalhosas. As entidades mostraram-se úteis não só para figuras, mas também para garantir o reaproveitamento e eliminar o trabalho desnecessário em relação a vários trechos de texto que são repetitivos ou sujeitos a mudanças constantes.

Em documentos com conexões diversas foi observado que o XML traz grandes progressos além de permitir tipos de conexões diferentes do que se tem com o HTML, ele também facilita a construção e manutenção dessas conexões por meio do XLink e do XPointer. Essas duas linguagens ampliam os recursos para a criação de *links* mais dinâmicos e mais próximos do que o usuário deseja.

Foi observado que várias características que são de grande interesse para um tipo de documento também são importantes para os outros tipos de documentos. Assim, pode-se perceber que as características explicadas em cada item podem ser adaptadas e usadas em todos os tipos de documentos XML.

4 Conclusão Final

O objetivo deste trabalho é encontrar a melhor linguagem existente hoje para a recuperação de documentos eletrônicos e mostrar os recursos desta linguagem que podem ser aplicados para a publicação de diversos tipos de documentos.

Buscando alcançar este objetivo, este projeto compreendem um estudo comparativo das três linguagens de marcação atualmente disponíveis no mercado: o SGML, o HTML e o XML.

O SGML é uma linguagem estruturada e com semântica complexa possuindo certas características opcionais que não são necessárias e acabam por causar incompatibilidade entre computadores conectados em rede se estes não possuem a mesma configuração quanto a essas opcionalidades. Além disso, o SGML possui funcionalidades específicas para as mais diversas aplicações que são raramente usadas.

O HTML não possui esta estruturação e tem tags fixos o que faz com que os padrões mudem muito devagar. Além disso, o HTML não possui marcações que indicam o significado da estrutura do documento, preocupando-se apenas com a formatação e tornando o texto ininteligível em relação à sua semântica.

Podendo solucionar todas estas questões, o XML, mostrou-se adequado para ser aplicado neste projeto, pois possui formato estruturado e maior simplicidade. As principais características que nos fizeram decidir a sua escolha foram: extensibilidade que permite a definição de *tags* próprias; estruturação que possibilita a sua aplicação para representar esquemas de banco de dados, hierarquias de aplicações orientadas a objeto entre outras; multi-conexões que permitem conexões múltiplas e multi-direcionais fornecendo maiores recursos e funcionalidades para a recuperação de documentos eletrônicos.

Escolhida a linguagem XML, procuramos mostrar as principais características da linguagem que são interessantes para a recuperação de documentos eletrônicos, via exemplos de acordo com os vários tipos de documentos mais publicados na *Web*. São eles: textos e livros literários; artigos e livros técnico-científicos; listagens e dados estátisticos; textos, documentos com figuras; documentos com conexões diversas entre outros.

Em textos e livros literários foi observado que o XML é de grande importância pois garante a construção de um texto bem-estruturado e o torna mais inteligível a qualquer usuário.

Em artigos e livros técnico-científicos foi observado que o XML além de proporcionar uma estrutura bem elaborada para o documento também permite a validação desta estrutura via DTDs e permite a inclusão de trechos de códigos específicos de cada área de estudo via seções CDATA

Em documentos com listagens e dados estatísticos foi observado que o XML além de proporcionar as vantagens de uma boa estruturação, também permite que formatações diferentes sejam aplicadas aos documentos via CSS ou XSL.

Em documentos com figuras foi observado que o XML pode expandir e facilitar o uso de figuras na Internet via Entidades que possibilitam que possíveis modificações de figuras tornam-se mais fácil e menos trabalhosas de serem feitas.

Em documentos com conexões diversas foi observado que o XML traz grandes progressos além de permitir tipos de conexões diferentes do que se tem com o HTML, ele também facilita a construção e manutenção dessas conexões através do XLink e do XPointer.

Foi observado que as características estudadas se aplicam a todos os tipos de documentos e não apenas àquele a que está relacionada, mostrando a importância da aplicação de um certo recurso para mais de um tipo de documento.

5 Trabalhos Futuros

Uma complementação deste trabalho pode ser feita estudando algumas características do XML que não encontram-se totalmente desenvolvidas pelo W3C [19], mas que podem vir a ser de grande interesse para a publicação dos tipos de documentos propostos neste trabalho.

Uma área bastante interessante que encontra-se em desenvolvimento é o estudo do CSS2. As formas de uso e sintaxe desta linguagem parece ser bem interessante para se aplicar em documentos XML e como não foram explicados neste projeto por nenhum *browser* ainda suportá-lo este seria um tópico interessante.

Uma outra abordagem do XML que também pode ser explorada diz respeito ao estudo das características da linguagem que mais se encaixam para as aplicações comerciais, o que não é o tema deste projeto.

Referências

[1] *Learn more about XML - some XML resources to get you started. (http://www.xml.clearlyonline/xml/learnmore.asp)*

[2] *Why XML - because it enables real information value. (http://www.xml.clearlyonline/xml/whyxml.asp)*

[3] XML Conversion Services. *(http://www.xml.clearlyonline/xml/convert.asp)*

[4] Ramalho, J. C. *Recovering Old Paper Historical Documents Through SGML/XML Modelling. in proc Digital Resources for the Humanities, 1999*

[5] Frequently Asked Questions about the Extensible Markup Language. *(http://www.ucc.ie/xml/), June, 1999*

[6] Sperberg,C.M., Burnard, L. *Guidelines for Eletronic Text Encoding and Interchange, 1993*

[7] Musciano,C., Kennedy, B., Loukides, M. *HTML: The Definitive Guide, 1998*

[8] XML.Com. *(http://www.xml.com)*

[9] Ramalho, J.C. *(http://beta.di.uminho.pt/ jcr/projectos/david/COMS/proc-doc/proc-doc-acet.html)*

[10] Chapter 2 - A Gentle Introduction to SGML. *(http://www-tei.uic.edu/orgs/tei/sgml/teip3sg/index.html)*

[11] Cover, R. *SGML - A Textual Representation for Information Structure. in Academic Computing.*

[12] 2-Introduction to HTML4. *(http://www.w3.org/TR/html4/intro/intro.html)*

[13] 13-HTML Origins, Owners, Good Practices. *(http://ei.cs.vt.edu/ wwwbtb/book/chap13/index.htm*

[14] Tanenbaum,A.S. *Computer Networks, Prentice Hall PTR, 3th Edition, 1996*

[15] Bosak,J. *XML, Java, and the future of the Web. (http://www.xml.com/pub/w3j/s3.bosak.html)*

[16] Sall,K. *XML: Structuring Data for the Web: An Introduction. (http://wdvl.com/Authoring/Languages/XML/Intro/)*

[17] Grego,M. *Nasce a Internet sem fio. Revista Info, Abril 2000, pags 30 a 37*

[18] The Source for Java Technology. *(http://java.sun.com/)*

[19] W3C - World Wide Web Consortium*(http://www.w3.org)*

[20] Ramos,G., Brandão,I.G., Veiga,J.J., Barreto,L., Vilela,L., Rey,M., Preta,S.P., *Para Gostar de Ler Volume 8 - Contos, Editora Ática, 1983, A velha Contrabandista, página 17*

[21] Casseta&Planeta *as melhores piadas do planeta... e da casseta também!, Objetiva, 13a edição, 1997, página 33*

[22] Harold,E.R. *XML Bible, IDG Books WorldWide, 1998*

[23] Ramalho,J.C.L. *Anotação Estrutural de Documentos e sua Semântica, Tese de Doutorado, Escola de Engenharia - Departamento de Informática - Universidade do Minho, 2000.*

[24] Clark,J.*XT (http://www.jclark.com/xml/xt.html)*

[25] LotusXSL *(http://www.alphaworks.ibm.com/tech/lotusxsl)*

[26] Kernighan,B.W., Ritchie,D.M. *The C Programming Language, Prentice Hall PTR, 2nd edition, 1998*

[27] *Tabelão Placar 1998. Revista Placar, Janeiro de 1999, encarte*

Renata Castro Carvalho
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Tel.: (031)499-5860 / (031)499-5842
Fax.: (031) 499-5858

Prof. Mariza Andrade da Silva Bigonha
Professor Adjunto
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Tel.: (031)499-5860 / (031)499-5891
Fax.: (031) 499-5858