

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

XHA: eXtensible Hyper-Automaton

por

CÉSAR COSTA MACHADO

Dissertação submetida à avaliação,
como requisito parcial para a obtenção do grau de Mestre
em Ciência da Computação

Prof. Paulo Blauth Menezes

Orientador

Porto Alegre, setembro de 2002.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Machado, César Costa

eXtensible Hyper-Automaton / por César Costa Machado. – Porto Alegre: PPGC da UFRGS, 2002.

148 f.: il.

Dissertação (mestrado) - Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR – RS, 2002. Orientador: Menezes, Paulo Fernando Blauth.

1. WWW. 2. Extensible Markup Language 3. Teoria dos autômatos 4. Gerenciamento de hiperdocumentos. 5. Cursos na web . I. Menezes, Paulo Fernando Blauth. II. Título

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitora: Profa. Wrana Panizzi

Pró-Reitor de ensino: Prof. José Carlos Ferraz Hennemann

Pró-Reitor Adjunto de Pós-Graduação: Jaime Evaldo Fensterseifer

Diretor do Instituto de Informática: Prof. Philippe Olivier Alexandre Navaux

Coordenador do PPGC: Prof. Carlos Alberto Heuser

Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

Agradecimentos

Acima de tudo está Deus, “O Criador”, e a ele vai o meu mais profundo agradecimento, pois Ele me deu vida e desta procuro extrair lições que engrandecem o meu espírito através do trabalho e do amor.

Agradeço também a minha esposa Patrícia e a meu filho Bruno, pois sem eles uma parte da minha vida não teria sentido. Neste período souberam compreender a ausência, entender as dificuldades, compartilhar as ansiedades e os momentos de alegria.

Ao meu filho:

“Tenho um filho de 4 anos, é meu único filho. E é uma xerox minha. Só que muito melhorado. Ele tem meu jeito, mas é doce e tranquilo.

Antes de ter um filho eu não tinha muita paciência com criança.

Meu filho mudou isso. Aliás, meu filho mudou tudo. Ele deu ao Universo um sentido maior. Eu tenho muitas razões para viver. Mas essa é uma razão que me beija, que me abraça. Uma razão de viver que precisa mais de mim do que as outras razões.

Hoje, definitivamente, sou uma pessoa melhor por causa dele.

O que ele me ensina é muito mais importante do que eu posso ensinar a ele. Eu o ensino a comer. Ele desenvolve o meu espírito, minha imaginação, me faz reaprender coisas esquecidas. Ele desenvolve a motricidade dos meus sentimentos. Porque num mundo tão duro, o coração da gente vai perdendo a coordenação motora.

Quando a noite ele se abriga em mim e faz um ninho na minha barriga, quem se sente na verdade protegido sou eu. Tenho certeza que meu "professor" ainda vai me ensinar muitas coisas. Seja a mexer no vídeo, no som do carro ou no meu coração, Antonio me ensinou que: Os monitores do século XXI são nossos filhos. Como diz ele: “eu sou seu professor, né pai?””

Nizan Guanaes

Por fim, aos meus Pais,

Pai e mãe, aqui cheguei, graças a vocês continuo agradecendo a Deus por sempre ter reconhecido o que por mim fizeram. Neste momento indescritível sinto-me mais perto, como em minha formatura. Desculpem pelos momentos em que falhei, pois como humano, foi difícil aprender todas as lições que me transmitiram e até hoje, estou encontrando na vida os fundamentos daquilo que me ensinaram.

Onde estiverem ouçam o meu muito obrigado.

Sumário

Lista de Abreviaturas	6
Lista de Figuras	7
Lista de Tabelas	8
Listagens de Programas	9
Resumo	10
Abstract	11
1 Introdução	12
1.1 Contexto do Trabalho	12
1.2 Objetivo	13
1.3 Descrição do Trabalho	14
1.4 Principais Contribuições	14
1.5 Apresentação da Dissertação	15
2 Características e Vantagens na Estruturação de Documentos em XML na WWW para EAD	17
2.1 Resumo	18
2.2 Introdução	18
2.3 O HTML e suas limitações	18
2.4 O XML e sua importância	19
2.5 Proposta de aplicação	22
2.6 Conclusões	24
3 Utilização do XML no Sistema Hyper-Automaton	25
3.1 Resumo	26
3.2 Introdução	27
3.3 Visão do Sistema	27
3.4 Gerência do Armazenamento	29
3.5 XML (eXtensible Markup Language)	29
3.6 A DTD (Data Type Definition)	29
3.7 Folhas de Estilo (Style Sheets)	30
3.8 Folhas de Estilo Avançadas	31
3.9 Comparação entre CSS e XSL	32
3.10 Conclusões	35
4 Definição e Aplicação de Regras para a Elaboração Adequada de Documentos XML para o Sistema Hyper-Automaton	36
4.1 Resumo	37
4.2 Introdução	37
4.3 Visão do Sistema	38
4.4 A DTD (Data Type Definition)	39
4.5 Declaração do tipo de documento – Document Type Declarations	40
4.6 Declaração de elementos	40
4.7 Entidades	41
4.8 Entidades gerais externas	43
4.9 Inclusão de dados não-XML	44
4.10 Reconhecendo arquivos não-XML	45
4.11 Conclusões	47
5 Flexibilização e Adequação do Formato de Saída ao Conteúdo Disponibilizado no Sistema de Hyper-Automaton	48
5.1 Resumo	49
5.2 Introdução	49

5.3 Visão do sistema adequada a estilos.....	50
5.4 Folhas de Estilos em cascata	51
5.5 A XSL padrão	52
5.6 Características da linguagem e exemplos práticos	55
5.7 Conclusões.....	64
6 Implementando o Novo Sistema-Hyper Automaton	65
6.1 Resumo	66
6.2 Introdução.....	66
6.3 XML e Java no H.A.....	67
6.3.1 Regras: DTD para XMLSchema, revisão de conceitos.....	67
6.3.2 A utilização de Java no H.A.	68
6.3.3 XML e Java juntos no H.A.....	69
6.4 Definição do objeto elementar.....	70
6.5 Armazenamento, recuperação e dinamicidade	73
6.5.1 XML e banco de dados relacional no H.A.	73
6.5.2 A utilização de Java Server Pages	74
6.6 Sistema de cursos H.A.....	75
6.7 Comparação com o sistema anterior.....	83
6.7.1 Organização de arquivos	83
6.7.2 Manutenção facilitada: uma máquina de estados que incorpora as duas funcionalidades	83
6.7.3 Acentuação e símbolos	83
6.7.4 Reuso inteligente	84
6.7.5 Expansibilidade	84
6.8 Conclusões e perspectivas futuras	84
6.8.1 Administrativo Web para a criação de cursos	84
6.8.2 Funções avançadas: índice remissivo, busca, etc	85
6.8.3 Software para a criação de objetos	85
6.8.4 Login e conteúdo sensível ao usuário.....	85
6.8.5 Log de utilização	85
6.8.6 Cursos sensíveis ao aplicativo	85
7 Conclusões e trabalhos futuros.....	86
7.1 Conclusão	86
7.2 Trabalhos futuros.....	88
Anexo 1 Artigo WIE'2000	90
Anexo 2 Artigo ISKM/DM'2000	100
Anexo 3 Artigo DEXA'2001.....	113
Anexo 4 Artigo IC'2001.....	122
Anexo 5 Artigo ISKM'2002.....	126
8 Bibliografia.....	139

Lista de Abreviaturas

ASCII	American Standard Code for Information Interchange
ASP	Active Server Pages
CDF	Channel Definition Format
CERN	Centre Européene pour la Reserche Nucléaire
CGI	Common Gateway Interface
CSS	Cascading Style Sheet
CSS1	Cascading Style Sheet level one
CSS2	Cascading Style Sheet level two
DER	Diagrama Entidade Relacionamento
DHTML	Dynamic Hypertext Markup Language
DOM	Document Object Model
DTD	Document Type Definition
EPS	Encapsulated PostScript
Fig.	Figura
FTP	File Transfer Protocol
GIF	Graphics Interchange Format
H.A.	Hyper-Automaton
HTML	Hypertext Markup Language
ie	Internet Explorer
JPEG	Joint Photographics Expert Group
JSP	Java Server Pages
MathML	Mathematical Markup Language
MIDI	Musical Instrument Digital Interface
MP3	MPEG-1 Audio Layer-3
OO	Orientado a Objeto
PDF	Portable Document Format
PS	PostScript
SAX	Simple API for XML
SGBD	Sistema de Gerência de Banco de Dados
SGML	Standard Generalized Markup Language
SMIL	Synchronized Multimedia Integration Language
SQL	Structured Query Language
TIFF	Tag Image File Format
UCS	Universal Character Set
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTF-8	UCS Transformation Format
VML	Vector Markup Language
VRML	Virtual Reality Model Language
W3	World Wide Web
W3C	World Wide Web Consortium
WAV	Microsoft Waveform
WWW	World Wide Web
XML	Extensible Markup Language
XPath	Extensible Path
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformations

Lista de Figuras

FIGURA 2.1 - Escopo SGML/HTML.....	19
FIGURA 2.2 - Estrutura de um documento XML.....	20
FIGURA 2.3 – Documento XML visualizado em um browser.....	21
FIGURA 2.4 - Aplicação de “ <i>stylesheet</i> ”	22
FIGURA 2.5 - Estruturação em blocos do sistema	24
FIGURA 3.1 – Estruturação em blocos do sistema	27
FIGURA 3.2 – Diagrama entidade-relacionamento do sistema	28
FIGURA 3.3 - Utilização do XML com CSS.....	33
FIGURA 3.4 - Resultado da visualização em navegador Web	35
FIGURA 4.1 - Esquema geral	39
FIGURA 4.2 - Entidades internas.....	42
FIGURA 4.3 - Entidades gerais externas	43
FIGURA 4.4 - Inserção de arquivos não-XML.....	45
FIGURA 4.5 - Reconhecendo arquivos externos	46
FIGURA 5.1 - Elementos gerais do sistema	51
FIGURA 5.2 – Aplicação de CSS	52
FIGURA 5.3 - Contexto dos padrões	54
FIGURA 5.4 - Processo de visualização	54
FIGURA 5.5 - Estrutura de um documento.....	55
FIGURA 5.6 - Saída para alunos.....	56
FIGURA 5.7 - Saída para exercícios	63
FIGURA 6.1 – Funcionamento de uma requisição JSP	74
FIGURA 6.2 - Saída no browser para uma determinada XSL	82
FIGURA 6.3 - Saída no browser para outra determinada XSL.....	83
FIGURA 7.1 - Sistema H.A na sua forma original	86
FIGURA 7.2 - Sistema H.A na forma atual	87
FIGURA 7.3 - Exemplo de saída do sistema H.A.....	88

Lista de Tabelas

TABELA 4.1 - Especificação de conteúdo	40
TABELA 7.1 – Comparação entre os sistemas HA e XHA.....	88

Listagem de Programas

LISTAGEM 2.1 – Documento XML básico	20
LISTAGEM 3.1 – CSS em XML	31
LISTAGEM 3.2 – Regras declarativas.....	31
LISTAGEM 3.3 – Exemplo de XSL	33
LISTAGEM 3.4 – Arquivo fonte XML para dados pessoais.....	34
LISTAGEM 3.5 – Arquivo fonte XSL para dados pessoais	34
LISTAGEM 4.1 – Especificação de conteúdo	41
LISTAGEM 4.2 - Entidades internas	42
LISTAGEM 4.3 - Inserção de arquivos não-XML.....	44
LISTAGEM 4.4 - Definição de entidades externas (nucleo.dtd)	45
LISTAGEM 4.5 - Reconhecendo arquivos externos.....	46
LISTAGEM 5.1 - Exemplo de XSLT	54
LISTAGEM 5.2 - Folha de estilo XSL para aluno.xml	57
LISTAGEM 5.3 - Arquivo de dados (exercicios.xml)	59
LISTAGEM 5.4 – Folha de estilo XSL para exercicios.xml	62
LISTAGEM 6.1 - XML Schema do objeto elementar	70
LISTAGEM 6.2 - Objeto de ensino da máquina de Mealy	72
LISTAGEM 6.3 - Criação do documento Page.....	76
LISTAGEM 6.4 - Consulta ao banco de dados	77
LISTAGEM 6.5 - Consulta ao banco de dados da palavra de saída	77
LISTAGEM 6.6 - Resultado da consulta	78
LISTAGEM 6.7 - Consulta ao banco de dados da palavra de entrada.....	79
LISTAGEM 6.8 - Transições de estado	79
LISTAGEM 6.9 - Templates padrão	80
LISTAGEM 6.10 – Transformação XSL para curso.....	81

Resumo

O objetivo geral desta dissertação é estudar as possibilidades de flexibilização da função de saída do Sistema *Hyper-Automaton* além das rígidas possibilidades utilizadas atualmente com a utilização direta do HTML, objetivando eliminar as limitações como execução de aplicações proprietárias, caracteres incompatíveis entre browsers, excesso de tráfego na rede, padronizar aplicações, incrementar recursos didáticos, melhorar o suporte a aplicações multimídia atuais e futuras, facilitar a manutenção, implementação e reuso, alterar o *layout* de saída no *browser* de maneira dinâmica, explorar outros recursos de *links*, estabelecer padrões de organização do material instrucional criado pelo professor e muitas outras. Tal sistema anteriormente desenvolvido e funcionando adequadamente, é baseado no formalismo de Autômatos Finitos com Saída como modelo estrutural para organização de hiperdocumentos instrucionais, em especial em cursos na *Web*, tornando o material hipermídia independente do controle da aplicação. O Sistema *Hyper-Automaton*, tornou-se portanto, um sistema semi-automatizado para suporte a cursos na *Web*.

Com o desdobramento da pesquisa, esta procurou ir mais além e descreveu possibilidades de não só estudar os aspectos possíveis de formatação de saída do sistema, mas reestruturá-lo totalmente sobre uma linguagem de *markup* padrão, buscando atualizá-lo tecnologicamente definindo outras possibilidades para que significativos trabalhos futuros possam de maneira mais clara serem alcançados.

Dessa maneira, esta dissertação centra-se no estudo da aplicação de formas de flexibilização do Sistema *Hyper-Automaton*, tanto na parte da estruturação de documentos que são conteúdos instrucionais armazenados, bem como, na forma desse material tornar-se-á disponível através de navegadores WWW compatíveis com as tecnologias propostas, possibilitando o incremento substancial de funcionalidades necessárias para cursos onde a *Web* é o principal meio.

Esta pesquisa dá prosseguimento a dois trabalhos anteriormente concluídos no PPGC e do Curso de Bacharelado em Ciência da Computação da UFRGS no ano de 2000, na seqüência, *Hyper-Automaton: Hipertextos e Cursos na Web Utilizando Autômatos Finitos com Saída*, dissertação de mestrado de Júlio Henrique de A. P. Machado e *Hyper-Automaton: Implementação e Uso*, trabalho de diplomação de Leonardo Penczek.

Palavras-chave: WWW, XML, XSL, DTD, educação à distância, cursos remotos, autômatos finitos.

TITLE: “WEB COURSES USING OUTPUT AUTOMATA: FORMATTING & CONTENT VISUALIZATION ”

Abstract

The overall goal of this dissertation is to study the bending possibilities for the output function of the *Hyper-Automaton* system beyond the rigid possibilities nowadays used with the direct usage of HTML, aiming to eliminate limitations such as proprietary application executions, incompatible characters between browsers, network overload, standardizing applications, increasing teaching resources, improving nowadays and future multimedia applications support, facilitating maintenance, implementation, and reusability, altering the browser output layout dynamically, exploring other link resources, setting standards for the teaching material created by the educator, and many others. Such system, previously developed and working properly, is based on the Output Finite Automata formalism as structural model for the organization of instructional hyper documents, especially Web courses, making the hypermedia material independent from the application control. The *Hyper-Automaton* System has then become a semi-automated system for Web courses support.

As the research advanced, it went further away and described possibilities of not only studying the possible aspects of the system output formatting, but also totally restructuring it over a standard markup language, aiming technological update defining other possibilities so that future papers may, in a clearer way, be produced.

This way, the dissertation here presented is focused on the study of the application of bending forms for the *Hyper-Automaton* System, on the document structuring part, which are stored instructional contents, as well as, on the form of this paper will become available through WWW browsers compatible with the proposed technologies, making possible the substantial increase of the necessary functionalities for courses were the Web is the main ground.

This research proceeds two papers previously concluded on the PPGC and on the Computer Science Course of UFRGS during the year 2000: *Hyper-Automaton: Hypertext and Web Courses Using Output Finite Automata*, MSc dissertation of Júlio Henrique de A. P. Machado and *Hyper-Automaton: Implementation and Usage*, graduation paper of Leonardo Penczek.

Keywords: WWW, XML, XSL, DTD, distance learning, remote courses, finite automata.

1 Introdução

Este documento descreve a dissertação submetida à avaliação como requisito parcial para a obtenção do grau de mestre em Ciência da Computação junto ao programa de Pós-Graduação em Computação (PPGC) da Universidade Federal do Rio Grande do Sul. A dissertação aborda análise e implementação de aspectos de flexibilização estrutural sobre um sistema baseado em conceitos da Teoria de Autômatos para o controle e organização de hiperdocumentos na *World Wide Web*.

1.1 Contexto do Trabalho

O objetivo dos sistemas de ensino à distância é proporcionar material instrucional para um número maior de alunos potencialmente espalhados em uma grande área. Dessa forma, permite-se, por exemplo, que novos conhecimentos cheguem a usuários isolados dos grandes centros de ensino e que professores sejam compartilhados eficientemente por diversos alunos localizados em diferentes locais.

A partir dos trabalhos de [SCH95], [MAR96], [LOH96] e [MAJ00] pode-se apresentar as principais vantagens do uso da *WWW* no Ensino à Distância:

- permite apresentação de conteúdo multimídia, texto integrado com som, imagens e vídeo, fornecendo ao professor a possibilidade de enriquecer o material instrucional, tornando-o mais claro e motivador;
- consiste em um ambiente bastante amigável, o que permite sua manipulação por usuários com pouca prática no uso de computadores;
- é, potencialmente, um ambiente integrado, pois é possível visualizar diferentes mídias dentro do próprio navegador (*browser*). Através da execução de aplicações auxiliares externas (*helper applications*) e da instalação de programas que estendem as capacidades dos navegadores (*plug-ins*) é possível manipular diversas mídias de uma forma quase transparente ao usuário;
- permite a integração com outros serviços da *Internet*. Com a utilização da interface comum disponibilizada pelos navegadores é possível acessar serviços como o *ftp*, *telnet*, *newsgroup* e correio eletrônico;
- permite o uso de *hyperlinks*, possibilitando ao educador uma melhor estruturação do conteúdo;
- favorece uma educação ativa, já que é oferecido um ambiente no qual o aluno atua no processo de descoberta de novos conhecimentos, ao invés de ser apenas um passivo receptor de conhecimentos;
- possibilidade de acesso às informações através de diferentes plataformas de *hardware* e *software*;
- possui a flexibilidade de horário, permitindo que o aluno estude um material disponibilizado na *WWW* no momento que lhe for mais adequado.

Diferentemente do senso comum, o trabalho e a função do professor não desaparecem com a utilização da *Web*, mas mudam um pouco com este novo ambiente.

Nesse caso, o professor concentra as suas ações na coordenação das atividades, dirige o aprendizado e, principalmente, tira e/ou gera dúvidas. Também cabe ao professor gerar um conteúdo que seja estimulante para o aprendizado. Essa situação coloca o professor não mais apenas como um transmissor de conhecimentos, mas como um elemento coordenador da aquisição de conhecimento pelo aluno. Tampouco o trabalho do professor se torna mais leve. Ao contrário, a manutenção e o controle do conteúdo didático oferecido aos estudantes e o acompanhamento do aprendizado

exigem um considerável esforço em todas as etapas do processo. Também, a elaboração da nova formatação do conteúdo didático utilizado pode ser uma tarefa bastante árdua. O desafio fundamental para os pesquisadores na área do ensino de Ciência da Computação se torna, então, intensificar esforços no desenho, desenvolvimento e teste formal de recursos para o ensino de computação que façam uso mais efetivo do novo paradigma apresentado pela evolução da WWW e de suas tecnologias relacionadas [BOR99].

O presente trabalho está baseado no projeto TEIA – Técnicas de Ensino Interativas Assistidas por Computador [WEB98], em realização no Instituto de Informática da Universidade Federal do Rio Grande do Sul, cujos objetivos principais incluem:

- “Maior independência do aluno em relação ao professor e maior envolvimento do aluno no processo de aprendizagem, estabelecendo seu próprio ritmo de desenvolvimento e seu nível de aprofundamento ou abrangência de conteúdos”.
- “Maior disponibilidade de material didático a baixo custo. Hiperdocumentos podem ser copiados e posteriormente consultados mesmo em computadores não conectados à rede”.
- “Maior produtividade do professor. Liberado de repetir semestre a semestre os conteúdos das disciplinas, o professor pode dedicar-se ao acompanhamento individual dos alunos, ao incentivo da criatividade dos alunos e à promoção de debates, desenvolvimento e análise de experimentos e implementações dos conteúdos”.
- “Maior flexibilidade de formatação de saída dos objetos disponibilizados no *browser*, inclusive para caracteres matemáticos, pois possibilitará ao professor a alteração da disposição visual do conteúdo sem alterá-lo”.
- “Maior facilidade de elaboração do material instrucional, dessa maneira o professor apenas preencherá janelas de texto com conteúdo, não se preocupando com requintes da linguagem de *markup*”.
- “Maior facilidade de manipulação de documentos, dessa maneira o professor poderá distribuir outros formatos de documentos multiplataforma a partir de documentos XML”.
- “Maior organização na elaboração dos conteúdos, gerência e manutenção do sistema a partir de regras cuja flexibilização são ajustáveis”.

1.2 Objetivo

O objetivo geral desta dissertação centra-se na análise e aplicação no sistema baseado no formalismo dos Autômatos Finitos com Saída denominado por Hyper-Automaton, formas padronizadas e modernas de flexibilização na manipulação de hiperdocumentos em sistemas de hipertexto baseados na *World Wide Web*, em especial na sua função de saída.

Como um todo, a pesquisa científica foi balizada pelos seguintes objetivos mais específicos:

- Analisar as linguagens de *markup* existentes e disponíveis, propondo a mais adequada para a criação de hiperdocumentos instrucionais.
- Estender a função de saída dos autômatos.
- Analisar as formas atuais e futuras desta suportar aplicações multimídia.
- Desenvolver testes buscando avaliar as potencialidades disponíveis na linguagem.
- Desenvolver uma aplicação inicial baseada nos conteúdos estudados.
- Comparar as novas proposições com o sistema atual.

1.3 Descrição do Trabalho

Em projetos de software direcionados à *Internet*, a *WWW* aparece como uma base (composta de protocolos de transporte e comunicação, interfaces e armazenamento de dados) sobre a qual são construídas ferramentas para suprir deficiências como a necessidade de controlar a organização de informações dispersas em unidades coerentes e permitir a flexibilidade no desenvolvimento e distribuição de material hipermídia onde a base de hiperdocumentos é projetada de forma independente da estrutura de controle das aplicações hipermídia. Constata-se, portanto, a necessidade de ferramentas para complementar uma estrutura já existente, visando torná-la mais apta ao suporte às necessidades cada vez mais crescentes do ensino a distância.

A pesquisa enfoca especificamente maneiras de flexibilização de um sistema de hipertextos. A organização de hiperdocumentos como Autômatos Finitos com Saída está direcionada à criação de uma estrutura funcional para a apresentação de materiais na *Web* – uma solução que facilita a implementação, manutenção e reuso dos componentes instrucionais. Neste modelo, visões sobre uma base de hiperdocumentos são autômatos com saída (Máquina de Mealy/Moore) e *links* entre páginas são transições entre os estados do autômato.

Autômatos têm uma história de uso como uma técnica de diagramação formal e podem ser representados por uma estrutura de grafo, a qual pode ser usada para prover interfaces de programação para o controle de material hipermídia.

O que caracteriza um conjunto de hiperdocumentos organizados no conceito de autômato como um “curso” é a forma didática da apresentação da informação de acordo com premissas pedagógicas e motivacionais direcionadas a facilitar a aprendizagem.

Todo o prosseguimento do estudo do Sistema de Hyper-Automaton direcionado a formas de melhor manipular e disponibilizar o conteúdo armazenado foi desenvolvido como resultado desta pesquisa de dissertação.

Ao longo deste texto são tratadas e discutidas as principais vantagens obtidas com a utilização do XML no Sistema Hyper-Automaton, entre as quais: padronização, separação da formatação física, lógica e aspectos de visualização; facilidade do reuso das páginas, estilos e regras estruturais, com eliminação de redundância; independência dos hiperdocumentos da estrutura do autômato, cuja alteração continua não influenciando nas páginas e vice-versa; facilidade de implementação e manutenção; interface gráfica simples, direta e com maiores recursos de formatação e possibilidades de inclusão de outras aplicações, possibilidade de implementação de pesquisas sobre hiperdocumentos.

O capítulo seis descreve características de implementação de um protótipo em fase inicial de desenvolvimento, neste são incluídos potencialidades baseadas nos aspectos propostos nesta dissertação, bem como, uma comparação entre o sistema atual baseado nos novos conceitos abordados e o sistema anterior inflexível e com recursos limitados.

1.4 Principais Contribuições

Uma contribuição importante desta proposta está na facilitação para a utilização de recursos conhecidos na Teoria da Computação, Teoria das Categorias e Hipermídia, para a extensão de sistemas de hipermídia na *WWW* voltados para o ensino.

De uma forma geral, esta dissertação contribui para as áreas de Engenharia de Software, Hipermídia, Informática na Educação e Sistemas Formais, onde existe a preocupação em desenvolver soluções flexíveis e poderosas para o aperfeiçoamento de sistemas de hipertexto voltados para ambientes distribuídos de ensino.

- Desenvolvimento de um modelo simples e eficiente para a estruturação de hiperdocumentos na *Web*.
- Desenvolvimento de implementações padronizadas.
- Manipulação mais eficiente de documentos.
- Facilidade de criação de documentos instrucionais por parte do professor.

1.5 Apresentação da Dissertação

Esta dissertação foi elaborada de acordo com o novo formato recomendado pelo PPGC da UFRGS. A dissertação está organizada como uma coletânea de artigos e baseada no trabalho individual produzidos durante o curso. Ao todo foram produzidos 5 artigos, tanto em nível de autoria quanto co-autoria, para eventos relacionados aos temas da dissertação (Informática na Educação, Computação na Internet, Banco de dados), listados abaixo:

- **Estudo Comparativo Sobre as Ferramentas de Autoria Existentes para a Criação de Documentos para a World Wide Web.** Porto Alegre: PGCC da UFRGS, 1999. Trabalho Individual. [MAC99] traz um estudo comparativo sobre as principais linguagens baseadas em *tags*, descrevendo suas principais características visando futuras possibilidades de implementação.
- **Características e Vantagens na Estruturação de Documentos em XML na WWW para EAD.** I Workshop Informática e Educação: uma Nova Abordagem Educacional, 2000, Passo Fundo. [MAC00] traz uma análise preliminar das características peculiares do XML para sua utilização no sistema de Hyper-Automaton para ensino à distância, descrevendo algumas maneiras de construção de documentos XML, regras de estrutura e maneiras de disponibilização do conteúdo armazenado.
- **Utilização de XML no Sistema Hyper-Automaton.** *International Symposium on Knowledge Management / Document Management*, 2000, Curitiba. [MAC00a] discute de forma mais ampla a implementação do sistema de uma maneira que objetive maior flexibilidade, introduzindo as primeiras idéias e aplicações.
- **Definition and Application of Rules for the Adequate Designing of XML Documents for the Hyper-Automaton System.** *Fifth International Query Processing and Multimedia Issues in Distributed Systems Workshop* em conjunto com a 12th *International Conference on Data and Expert Systems Application* e *IEEE Computer Society*, 2001, Munich. [MAC01a] este artigo descreve de maneira prática e com exemplos previamente testados com forte ênfase na construção de regras corretas e rígidas que determinarão o nível de flexibilidade sobre estruturas XML armazenadas pertencentes a cursos na *Web* com conteúdo multimídia.
- **Flexibility and Adequacy of the Output Layout in the Hyper-Automaton System.** *International Conference on Internet Computing*, 2001, Las Vegas. [MAC01] este trabalho trata das formas de disponibilizar conteúdo visualmente, descrevendo, comparando e exemplificando as mais importantes características, de acordo com uma visão aplicada e amparada bibliograficamente.

Os artigos publicados estão organizados nos capítulos subseqüentes. Cada um desses capítulos se inicia por uma subseção de introdução, na qual estão descritas as contribuições de cada artigo constante do capítulo, juntamente com informações pertinentes ao evento no qual o artigo foi publicado e um resumo dos principais assuntos tratados no texto.

Todos os artigos incluídos nesta dissertação estão na língua portuguesa, mesmo sendo submetidos originalmente em inglês, na parte reservada aos anexos encontramos tais artigos presentes no seu formato original.

O volume desta dissertação está organizado da seguinte forma:

- **Capítulo 1** – apresenta a introdução da dissertação, com a definição dos objetivos da pesquisa e as principais contribuições do trabalho realizado.
- **Capítulo 2** – é composto pelo artigo “*Características e Vantagens na Estruturação de Documentos em XML na WWW para EAD*” e tem a intenção de localizar a área de uso do XML na Educação a Distância.
- **Capítulo 3** – é composto pelo artigo “*Utilização de XML no Sistema Hyper-Automaton*” e tem a intenção de fazer as primeiras proposições do que será o sistema quando da incorporação de uma moderna e padronizada estruturação de documentos anteriormente proposta. São realizados as primeiras comparações, testes e outros estudos relacionados ao seu armazenamento.
- **Capítulo 4** – é composto pelo artigo “*Definition and Application of Rules for the Adequate Designing of XML Documents for the Hyper-Automaton System*” e descreve de maneira mais detalhada a estruturação geral para que tenhamos documentos válidos e bem formados, localização e acesso a outros tipos de documentos não-XML.
- **Capítulo 5** – é composto pelo artigo “*Flexibility and Adequacy of the Output Layout in the Hyper-Automaton System*” e tem por objetivos, de acordo com uma visão prática, a partir de testes, mostrar as possibilidades iniciais de formatação para os hiperdocumentos característicos em nosso sistema.
- **Capítulo 6** – “*Implementando o Novo Sistema-Hyper Automaton*” estuda os principais aspectos da implementação do XML no sistema, descrevendo a sua estrutura de dados, linguagens de programação utilizadas, características diversas do sistema, dificuldades encontradas as quais não foram descritas anteriormente, bem como, futuros desenvolvimentos relacionados.
- **Capítulo 7** – Conclusões e trabalhos futuros descreve os objetivos propostos e alcançados, as facilidades que passaram a fazer parte do sistema e a ampla gama de trabalhos futuros que advirão das novas propostas
- **Anexos** – são encontrados os artigos que fazem parte da dissertação, presentes na folha de rosto e introdução.

A seqüência de numeração dos capítulos indicam a ordem recomendada de leitura da dissertação, pois tais capítulos trazem uma idéia completa da evolução da pesquisa.

Após a leitura do item 3.2, dispensa-se a leitura dos itens 4.2 e 5.2, por serem contextualizações de artigos.

2 Características e vantagens na estruturação de documentos em XML na WWW para EAD

Este artigo [MAC00] foi submetido e publicado na categoria de artigo completo no I Workshop Informática e Educação: uma Nova Abordagem Educacional, promovido pela Faculdade de Educação – FAED conjuntamente com o Grupo de Estudos e Pesquisa em Software Educacional – GEPESE nos dias 9 e 11 de Novembro de 2000 na Universidade de Passo Fundo (UPF).

Esse evento tem por objetivo divulgar, refletir e discutir questões, consideradas prioritárias no desenvolvimento e na pesquisa de softwares educacionais e seus reflexos quanto ao uso da informática na educação e teve por clientela envolvida: professores, alunos e profissionais da área de informática e educação.

O artigo faz uma análise preliminar das características peculiares do XML para sua utilização no sistema de Hyper-Automaton para ensino a distância, descrevendo algumas maneiras de construção de documentos XML, regras de estrutura e maneiras de disponibilização do conteúdos armazenados.

Neste capítulo são descritos as limitações do HTML e os motivos pelos quais o XML passa a ser introduzido em nossas aplicações, logo após são abordados os conceitos iniciais do XML e os problemas enfrentados com sua utilização.

2 Características e vantagens na estruturação de documentos em XML na WWW para EAD

2.1 Resumo

Este trabalho apresenta uma forma de estruturação e visualização do conteúdo do material utilizado em cursos disponibilizados na WWW utilizando-se construções formais conhecidas como Autômatos Finitos Determinísticos com Saída. O estudo baseado nas evoluções tecnológicas da Internet sugere que o sistema atual incorpore os benefícios da estruturação moderna de documentos, consoantes com as necessidades e benefícios tecnológicos, bem como, faça uso de importantes recursos derivados das linguagens de marcação referentes às possibilidades de formatação do conteúdo disponibilizado visualmente. O objetivo deste trabalho é reestruturar o curso, que embora funcionando perfeitamente à base de fragmentos HTML, possa ser reconstruído perante estas novas tecnologias, mais adequadamente para o formato XML e folhas de estilo XSL.

2.2 Introdução

O ensino à distância é uma das áreas de grande desenvolvimento e que gera inúmeras fontes de pesquisa nas universidades brasileiras. Na Universidade Federal do Rio Grande do Sul (UFRGS), mais especificamente, desenvolvemos muitas aplicações voltadas a esta área.

Este trabalho tratará das principais características da linguagem de marcação de documentos estruturados na Web, a *eXtensible Markup Language (XML)*, que oferece significativos avanços em termos de disponibilização e visualização de documentos dos mais diferentes formatos, principalmente orientadas ao ensino auxiliado por computador [MAC99].

2.3 O HTML e suas limitações

Um dos importantes serviços que a Internet oferece é a conhecida como *World Wide Web – WWW*, onde vários serviços disponíveis na Internet podem ser acessados.

A *WWW* que suporta inúmeros recursos multimídia é o local destinado para a publicação dos mais variados tipos de informações já imaginados, com inúmeras possibilidades de disposição de textos, imagens dos mais diferentes formatos, sons e outros recursos também associados a hiperlinks, que permitem inicializar outras funções como serviços, aplicativos ou redirecionar a navegação para outro endereço eletrônico [LYN99], [MAC99].

Atualmente a ferramenta mais comum utilizada pelos desenvolvedores a fim de disponibilizar conteúdo na *Web* que suporta satisfatoriamente bem os recursos de hiperlinks, é o conhecido HTML (*HyperText Markup Language*), bem como, outras linguagens de hipertexto derivadas desta, que encapsulam scripts oferecendo recursos adicionais poderosos de grande valia.

Embora o HTML tenha sido definido a partir do SGML (*Standard Generalized Markup Language*), fig. 2.1, através da apropriação de *tags* adequados com a finalidade de suporte entre plataformas para a apresentação de páginas *Web*, esta começa a dar sinais de fraqueza, vítima do seu próprio sucesso, pois sua facilidade de compreensão, manipulação e alta flexibilidade, torna-se evidente [GOL90], [LIG99].

Problemas decorrentes destas facilidades começam a surgir e, como se não bastasse, grandes empresas desenvolvedoras de *browsers* passaram a criar implementações conhecidas como proprietárias, destinadas a navegadores *Web* específicos, portanto não reguladas pela *W3C* (*World Wide Web Consortium*), órgão este que busca a tão sonhada padronização na *WWW*.

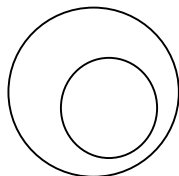


FIGURA 2.1 - Escopo SGML/HTML

Onde a HTML falha [LIG99]?

- Exibir páginas extremamente longas.
- Fornecer melhor controle sobre aparência das páginas.
- Suportar vários tipos de hiperlinks.
- Distribuir uma variedade cada vez maior de informações em Intranets bem como na Internet.

O XML (*eXtensible Markup Language*), torna-se adequado a tipos de documentos que requerem formatações que não se encaixam no molde do HTML. Eis alguns exemplos:

- Livros;
- Transações financeiras;
- Manuais técnicos;
- Fórmulas químicas;
- Registros médicos;
- Catálogos de registros para museus;

2.4 O XML e sua importância

XML é um texto baseado em linguagem de markup (*tags*) que está se tornando um rápido padrão de troca de dados na *Web*. Em parte é similar a HTML, pois pode-se identificar dados usando *tags* (identificadores inclusos, como estes: <...>).

Os tags XML identificam (explicitam) o que o dado significa, melhor do que o HTML, pois preocupa-se somente com a maneira como os dados serão dispostos visualmente quando vistos pelo usuário, como por exemplo: este dados será exibido em negrito, itálico, na cor vermelha, etc., (...); um *tag* XML age como um campo de nome, ele permite identificar com um rótulo em um dado, tornando o documento muito mais claro ao desenvolvedor (por exemplo: <capitulo>...</capitulo>).

Da mesma maneira que são definidos os nomes dos rótulos dos dados, é também definida a estrutura do documento, existe uma liberdade para que seja usado qualquer *tag* XML que faça sentido para uma certa aplicação, obviamente, dessa maneira para múltiplas aplicações podem-se utilizar os mesmos dados XML. Aqui está um exemplo de um mesmo dado XML que poderia ser utilizado para utilização de uma aplicação de mensagens: [LIG99], [BRA98]

```

<message>
  <to>cmu@info.ucel.com</to>
  <from>frng@spock.ucpel.com</from>
  <subject>XML - estrutura</subject>
  <text>Um documento XML, torna-se claro a partir do momento que
  seus tags são auto descritivos</text>
</message>

```

LISTAGEM 2.1 – Documento XML básico

Os *tags* (em negrito) nesse exemplo identificam inteiramente a mensagem, o destino e o endereço de quem está enviando, o resumo e o texto da mensagem. Como em HTML, o *tag* <to> possui um *tag* final: </to>. O dado entre o tag inicial e final define o elemento do dado XML. Nota-se também, que o conteúdo do *tag* <to> está totalmente contido no escopo da *markup* mensagem <message>...</message>. É esta característica de um *tag* conter outros *tags* que dá ao XML a habilidade de representar hierarquicamente a estrutura de dados de qualquer documento, o que torna o acesso mais fácil e a estruturação do documento (fig.2.2).

Em comparação com o HTML, espaços em branco são essencialmente irrelevantes, desta maneira, pode-se formatar o dado com legibilidade, processá-lo facilmente, e ainda, procurar e localizar conjuntos de dados, pois como já explicado, XML identifica o dado contido, tornando fácil a localização destes pelo usuário ou qualquer implementação de pesquisa de dados.

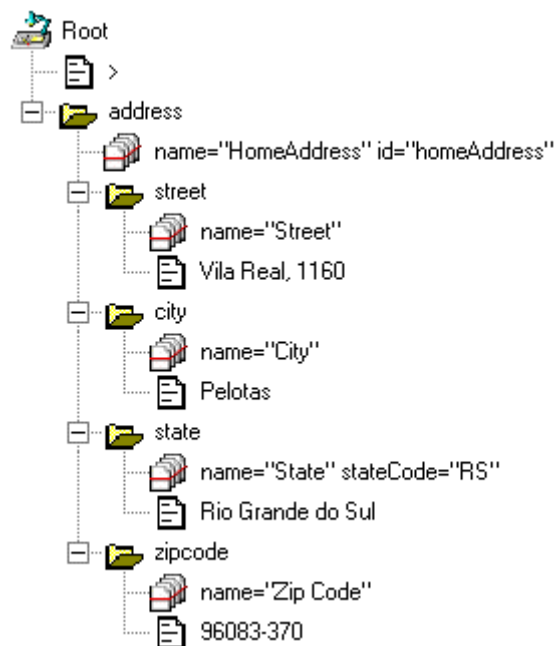


FIGURA 2.2 – Estrutura de um documento XML

Há um número de razões para a larga aceitação do padrão XML. Abaixo estão listadas as mais importantes:.

- Texto de fácil clareza
Como o XML não é um formato binário, pode-se criar e editar arquivos a partir de editores de textos, bem como ferramentas de desenvolvimento em ambientes visuais. Este é de fácil depuração e torna-se apropriado para o armazenamento de pouca

quantidade de dados. Nesta mesma visão, o XML “*front end*” de um banco de dados torna possível e eficiente o armazenamento de grande volume de dados. Então o XML permite escalabilidade para muitas aplicações a partir de pequenas adaptações nos arquivos, permite o gerenciamento de grandes volumes de dados [HAR99].

- Identificação de dados

XML identifica com clareza o tipo de dados que este condiciona, e não como este é disposto na tela de um monitor de vídeo ou em outro dispositivo que permita visualização, pois sua *markup* identifica a informação e a separa em partes, tornando mais fácil o processamento por um programa de *e-mail*, por exemplo: um *address book* teria a possibilidade de procurar mais facilmente a pessoa em particular e extrair a informação de endereço do resto do corpo da mensagem. Resumindo: pelo fato de poder-se identificar pequenas partes de informação, estas podem ser utilizados por diferentes aplicações.(fig.2.3) [HAR99], [TID99] .

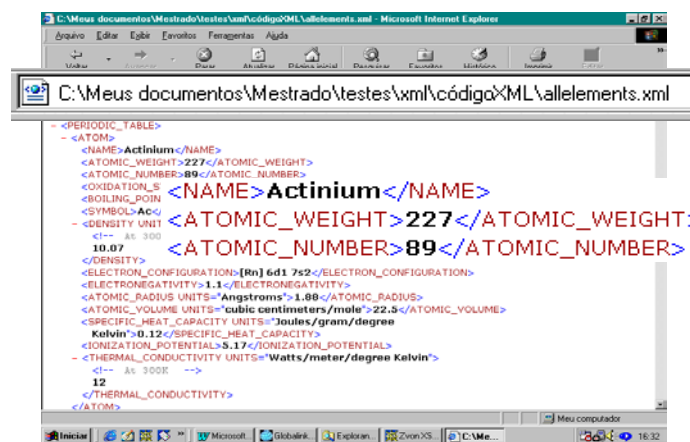


FIGURA 2.3 – Documento XML visualizado em um *browser*

- Estilização

Quando a visualização é um fator importante, a folha de estilo padrão, XSL, permite que se defina como o dado deverá ser apresentado [HAR99], [ARM99], [ADL00], [MAC01]. Por exemplo, a folha de estilo (*stylesheet*) para:

<to>you@yourAddress.com</to>, pode informar:

1. Começar em nova linha.
2. Mostrar “To:” em negrito, seguido por um espaço.
3. Mostra o destino.

O que produz: **To:** you@yourAddress.com

Naturalmente que mesma tarefa poderia ser feita em HTML, mas neste formato, não haveria a possibilidade destes dados serem processados por programas de busca ou programas para a extração de endereços eletrônicos ou semelhantes. O mais importante é que este padrão é independente do estilo, o que permite que para cada documento sejam aplicados os mais variados formatos de visualização, podendo aplicar diferentes folhas de estilos para produzir como saída documentos em *postscript*, TEX, PDF, ou algum novo formato que ainda não tenha sido inventado (fig.2.4) [ADL00].

- Reusabilidade

Um dos melhores aspectos de documentos em XML é que estes podem ser compostos por vários módulos. Até então igual ao HTML, mas com a grande diferença, com XML pode-se ligá-lo a outros documentos. É permitido assim que se possa acessar uma parte de um documento sem que seja necessário o cliente carregá-lo totalmente. Pode-se construir um documento em módulos, onde estes podem estar em qualquer local no mundo, como em um trabalho cooperativo, onde resultados devem ser trocados e atualizados via *Web*, assim sendo, esse documento será constituído como os pedaços de uma peça, facilitando sua atualização [LIG99], [MOU99].



FIGURA 2.4 – Aplicação de “stylesheet” fonte: www.ibm.com

- *Linkability*

Graças ao HTML, a habilidade de definir *links* entre documentos é considerada como uma necessidade. O XML possui a habilidade de manipulá-los das mais diferentes formas, isto é, permite que seja definido *links* duplex (dois caminhos), multi-alvo, *links* expandidos e *links* entre dois documentos existentes que estão definidos em um terceiro [ARM99].

- Fácil Processamento

Com uma regular e consistente notação torna-se fácil construir programas para processar dados XML. Por exemplo, em HTML por sua exagerada flexibilidade [HAR99], as regras de construção de documentos não são rígidas, permitindo que algumas vezes *tags* sejam suprimidos ou subtendidos. Isso torna difícil o processamento das informações e leva a resultados inesperados. Mas em XML, o *tag* deve obrigatoriamente possuir o *tag* terminador. Essas restrições obedecidas, tornam o documento elaborado nesta linguagem de *markup* bem formado (*document well-formed*). De outro modo, o processador XML não estaria apto a ler os dados contidos em tal documento. Para concluir existem softwares adequados para analisar a estrutura de um documento XML, esses programas são denominados *parsers*.

2.5 Proposta de aplicação

A proposta de aplicação que será alvo de estudos, visa dar prosseguimento ao trabalho Modelagem de Cursos na Web Utilizando Sistemas Formais [MAJ00], [MEN99] e tem como objetivo propor alternativas com respeito à formatação da interface com o usuário através da utilização do XML e de suas aplicações (fig. 2.4) descritas nos itens anteriores.

Cursos na Web Utilizando Sistemas Formais, utilizam construções formais conhecidas como Autômatos Finitos Determinísticos [MEN98]. Nós introduzimos o conceito de "cursos são autômatos" como uma estrutura que permite fácil implementação, criação de material hipermídia independente do autômato, ao mesmo tempo que encoraja o reuso de páginas *Web* em vários cursos, os quais podem ser construídos com enfoques diferentes, diminuindo a redundância na criação de páginas [MAJ98], [MAJ00], [MEN99].

Este projeto já é realidade e cria um sistema semi-automatizado para o suporte do ensino de Informática Teórica através da aplicação de conceitos inerentes à Ciência da Computação, em especial da Teoria de Autômatos e Teoria das Categorias, juntamente à tecnologia de hiperdocumentos, reunindo os benefícios de ambas [MAJ00].

O objetivo geral do modelo Hyper-Automaton centra-se no estudo da aplicação do formalismo de Autômatos Finitos com Saída (Máquina de *Mealy* e Máquina de *Moore*) como um modelo estrutural para a organização de hiperdocumentos instrucionais, em especial de Cursos na *Web*. O modelo é inspirado em pesquisas clássicas na área de hiperdocumentos e recentes iniciativas na *WWW*, com especial enfoque no desenvolvimento de sistemas de hipertexto, onde a base de hiperdocumentos é projetada de forma independente da estrutura de controle da aplicação hipermídia e suporta algumas facilidades descritas no Modelo Dexter como a composição de estruturas hierárquicas, especificação de vários conjuntos de links sobre um mesmo corpo de hiperdocumentos e objetos separados da estrutura de navegação [HAL94].

Cada autômato define um curso e consiste de um conjunto de hiperdocumentos independentes, os quais podem pertencer a outros cursos. A função de transição funciona como ligação lógica entre os hiperdocumentos e a função de saída compõe as páginas. O resultado final é a estrutura básica de páginas e links de hipertexto em um *site* na *Web*. O modelo leva a um alto grau de modularização do material instrucional, apresentando as seguintes vantagens: facilidade de reuso de páginas em diversos cursos, com eliminação da redundância; independência dos hiperdocumentos da estrutura do autômato, cuja alteração não influi nas páginas e vice-versa; permite que qualquer usuário crie links de e para qualquer documento; facilidade de implementação e manutenção; interface gráfica simples e direta; elaboração de seqüências instrucionais com enfoques específicos e capaz de oferecer estudo individualizado; operações categoriais fornecem um esquema de composição de cursos que permite a construção de novos cursos sobre cursos já existentes através de procedimentos de alto nível [MAJ00], [MEN99].

O sistema Hyper-Automaton constitui-se de um sistema semi-automatizado para o suporte a cursos na *Web* (com base em uma arquitetura cliente/servidor e interface desenvolvida em HTML) através da aplicação de conceitos inerentes à Ciência da Computação, em especial da Teoria de Autômatos, tecnologia de Hipermídia e Teoria das Categorias, reunindo os benefícios destas. Embora a ênfase utilizada para a validação do modelo seja a implementação de sistemas para o ensino a distância, os resultados obtidos se aplicam para sistemas de hipertexto como um todo.

Nosso trabalho na fase atual, está centrado nas várias possibilidades de formatação do conteúdo disponibilizado, culminando em uma adequada interface com o usuário, onde, a flexibilização da formatação do conteúdo obtido na função de saída do autômato será uma aplicação que remonte o documento nas partes desejadas e com as características mais adequadas relacionadas às necessidades do usuário e do curso em questão. O estudo proposto trará uma maior flexibilidade ao já existe, pois no estado atual, o que se apresenta é uma única e rígida possibilidade composta de fragmentos em HTML.

O diagrama em blocos que ilustra as partes do sistema (fig. 2.5), inclui o que estamos desenvolvendo atualmente. A novidade está em estruturar em XML todas as informações do curso e utilizar templates XSL a fim de visualização e *layout* do referido curso *on-line*.

Essas alterações objetivarão principalmente a atualização tecnológica do sistema de acordo com os benefícios já descritos do XML.

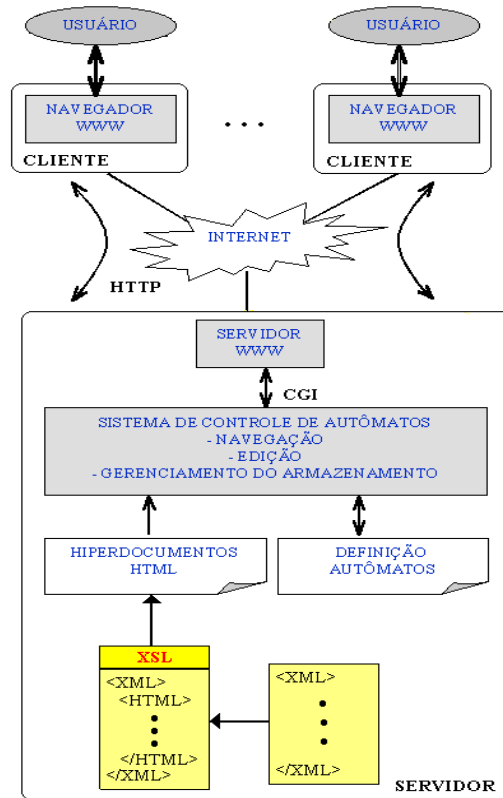


FIGURA 2.5 – Estruturação em blocos do sistema

2.6 Conclusões

As principais conclusões encontradas até este ponto da pesquisa, giram em torno da incompatibilidade entre diversos *browsers*, pois somente a partir da versão 4.0 do browser Internet Explorer da Microsoft, este começa a suportar a utilização de *Extensible Style Sheets (XSL)*.

Outro aspecto importante é escassa bibliografia à respeito, justificada pelo recém desenvolvimento e utilização destas tecnologias.

3 Utilização do XML no sistema Hyper-Automaton

Este artigo [MAC00a] foi submetido e publicado na categoria de artigo completo no *International Symposium on Knowledge Management / Document Management* – Simpósio Internacional da Gestão do Conhecimento e Gestão Documentos, promovido pela Pontifícia Universidade Católica de Curitiba – PUC PR e Centro Internacional de Tecnologia de Software – CITS, ocorrido em 26 a 29 de Novembro de 2000 na cidade de Curitiba.

Em sua quarta edição, o ISKM/DM consolida-se como o evento de referência destas áreas, atraindo contribuições de alto nível dos mais diversos setores. Esta edição do ISKM/DM, contou com 150 participantes de 50 diferentes organizações. Foram apresentados 22 trabalhos técnicos e 8 relatos de experiências, além de 5 palestras convidadas.

O artigo faz uma análise mais aprofundada do XML orientado ao sistema de Hyper-Automaton. Outras abordagens importantes da linguagem são colocadas, conjuntamente com testes e seus respectivos resultados, bem como, início do estudo da aplicação correta de folhas de estilos e pesquisa na árvore estrutural do documento, objetivando a disponibilização do conteúdo armazenado em XML.

Neste capítulo são descritos os conceitos iniciais do gerenciamento do armazenamento de documentos, regras estruturais para a elaboração padrão de documentos e introdução ao uso de folhas de estilos, ao final, listamos os desafios que tivemos que enfrentar com as novas alterações no sistema, visando adequá-lo às novas necessidades.

3 Utilização do XML no sistema Hyper-Automaton

3.1 Resumo

Este trabalho dá prosseguimento a estudos anteriores buscando oferecer uma forma de estruturação e visualização do conteúdo do material utilizado em cursos disponibilizados na *WWW* utilizando-se construções formais conhecidas como Autômatos Finitos com Saída. O estudo baseado nas evoluções tecnológicas da Internet sugere que o sistema atual incorpore os benefícios da estruturação moderna de documentos, consoantes com as necessidades e benefícios tecnológicos, bem como, incorpore importantes recursos derivados das linguagens de marcação referentes às possibilidades de formatação do conteúdo disponibilizado visualmente. O objetivo deste trabalho é esquematizar e descrever, de acordo com uma visão prática, o que é necessário dentro das inúmeras possibilidades do XML a fim de apontar caminhos para a reestruturação e inclusão do XML e XSL no sistema Hyper-Automaton.

3.2 Introdução

Este artigo dá prosseguimento ao trabalho Modelagem de Cursos na Web Utilizando Sistemas Formais [MAJ00a], [MAJ00], [MEN99] e tem como objetivo propor alternativas com respeito à modelagem de hiperdocumentos e formatação da interface com o usuário através da utilização do XML e de suas aplicações.

Cursos na *Web* baseados no modelo Hyper-Automaton [MAJ00], [MEN99], [MAJ00a] utilizam construções formais conhecidas como Autômatos Finitos Determinísticos [MEN00]. Introduzimos o conceito de "cursos são autômatos" como uma estrutura que permite fácil implementação, criação de material hipermídia independente do autômato, ao mesmo tempo que encoraja o reuso de páginas *Web* em vários cursos, os quais podem ser construídos com enfoques diferentes, diminuindo a redundância na criação de páginas.

Este projeto já é realidade e constitui-se de um ambiente semi-automatizado para o suporte do ensino de Informática Teórica (<http://teia.inf.ufrgs.br>) através da aplicação de conceitos inerentes à Ciência da Computação, em especial da Teoria de Autômatos e Teoria das Categorias, juntamente à tecnologia de Hiperdocumentos, reunindo os benefícios de ambas [MAJ00].

O objetivo geral do modelo Hyper-Automaton centra-se no estudo da aplicação do formalismo de Autômatos Finitos com Saída (Máquina de *Mealy* e Máquina de *Moore*) [MEN00] como um modelo estrutural para a organização de hiperdocumentos instrucionais, em especial de Cursos na Web. O modelo é inspirado em pesquisas clássicas na área de hiperdocumentos e recentes iniciativas na *WWW*, com especial enfoque no desenvolvimento de sistemas de hipertexto, onde a base de hiperdocumentos é projetada de forma independente da estrutura de controle da aplicação hipermídia e suporta algumas facilidades descritas no Modelo Dexter [HAL94] como a composição de estruturas hierárquicas, especificação de vários conjuntos de links sobre um mesmo corpo de hiperdocumentos e objetos separados da estrutura de navegação.

Cada autômato define um curso e consiste em um conjunto de hiperdocumentos independentes, os quais podem pertencer a outros cursos. A função de transição funciona como ligação lógica entre os hiperdocumentos e a função de saída compõe as páginas. O resultado final é a estrutura básica de páginas e links de hipertexto em um site na *Web*. O modelo leva a um alto grau de modularização do material instrucional, apresentando as seguintes vantagens: facilidade de reuso de páginas em diversos cursos,

com eliminação da redundância; independência dos hiperdocumentos da estrutura do autômato, cuja alteração não influi nas páginas e vice-versa; permite que qualquer usuário crie links de e para qualquer documento; facilidade de implementação e manutenção; interface gráfica simples e direta; elaboração de seqüências instrucionais com enfoques específicos e capaz de oferecer estudo individualizado; operações categoriais fornecem um esquema de composição de cursos que permite a construção de novos cursos sobre cursos já existentes através de procedimentos de alto nível [MAJ00], [MEN99], [MAJ00a].

Este trabalho está centrado nas várias possibilidades de formatação do conteúdo disponibilizado, culminando em uma adequada interface com o usuário, onde, a flexibilização da formatação do conteúdo obtido na função de saída do autômato será uma aplicação que remonte o documento nas partes desejadas, com características mais adequadas às necessidades do usuário e do curso em questão. O estudo proposto trará uma maior flexibilidade ao sistema já existente pois, no estado atual, o que se apresenta é uma única possibilidade composta de fragmentos em HTML (concatenação de páginas).

3.3 Visão do Sistema

O diagrama em blocos, que ilustra as partes do sistema (fig.3.1), apresenta em destaque a extensão ao sistema Hyper-Automaton discutida neste trabalho. A novidade está em estruturar os conteúdos (hiperdocumentos) em XML obedecendo às definições preestabelecidas para os documentos (DTD) [LIG99], [HAR99], [MAC01a] e utilização de *templates* XSL e/ou folhas de estilos em cascata (CSS) [LIE99], [BOS98], [MAC01] a fim de visualização e *layout* do referido curso *on-line*. Estas alterações objetivarão principalmente a atualização tecnológica do sistema de acordo com os benefícios relacionados ao XML e descritos nas seções seguintes.

A partir deste momento, este trabalho descreverá outras importantes características do XML [BOS98] intimamente ligadas aos nossos objetivos, explicitando sua utilização conforme descrito na fig.3.1. Características estas que serão amplamente utilizadas na nova proposta de implementação sobre o sistema Hyper-Automaton.

Observando a fig.3.1 (direita), procura-se detalhar de forma gráfica o que deverá ser implementado. As características de cada bloco serão detalhadas na seqüência.

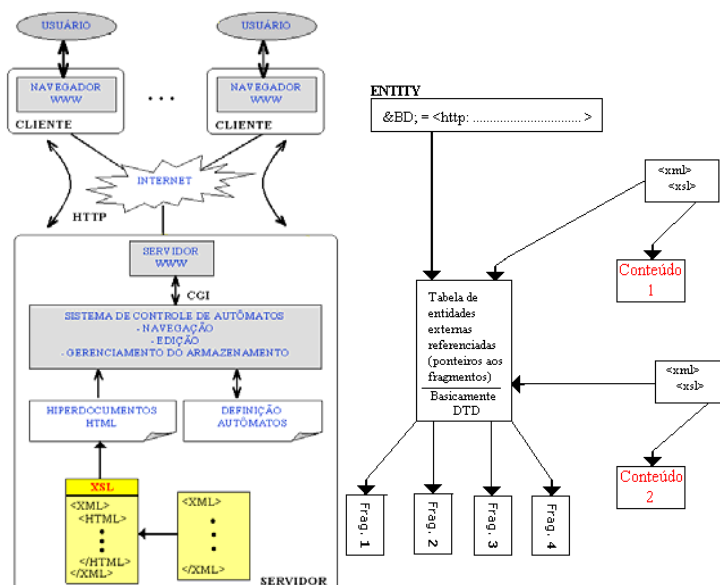


FIGURA 3.1 – Estrutura em blocos do sistema

Conforme o modelo de autômatos com saída utilizado, no nível mais baixo do sistema têm-se fragmentos de informação. Estes são considerados unidades atômicas se levarmos em consideração o conceito de alfabeto de saída. Um fragmento pode vir a ser um parágrafo de um texto, uma imagem, um vídeo, etc. A unidade de apresentação é uma página *Web* (documento HTML) chamada de palavra de saída no universo dos autômatos (na terminologia de hipermídia o conceito é equivalente ao termo nodo). Uma página é, então, construída sobre esses fragmentos. Na versão atual do Hyper-Automaton, cada página é uma seqüência linear de fragmentos estáticos. Novos construtores podem ser definidos a partir da extensão da função de saída dos autômatos. Um benefício imediato da utilização de hiperdocumentos marcados com XML, é a criação de funções de saída que constroem automaticamente um novo hiperdocumento (como um índice, ou resumo de um livro) a partir de consultas XSL. Como a função de saída também é responsável pela finalização das páginas *Web* a serem visualizadas por um navegador, um uso imediato do XSL é permitir que a função determine aspectos de *layout* das páginas para os conteúdos (fontes, alinhamento, cores, etc.) utilizando-se de folhas de estilo. Além disso, através de um modelo de usuário mantido em banco de dados, é possível permitir aos usuários armazenarem opções pessoais para a visualização dos conteúdos.

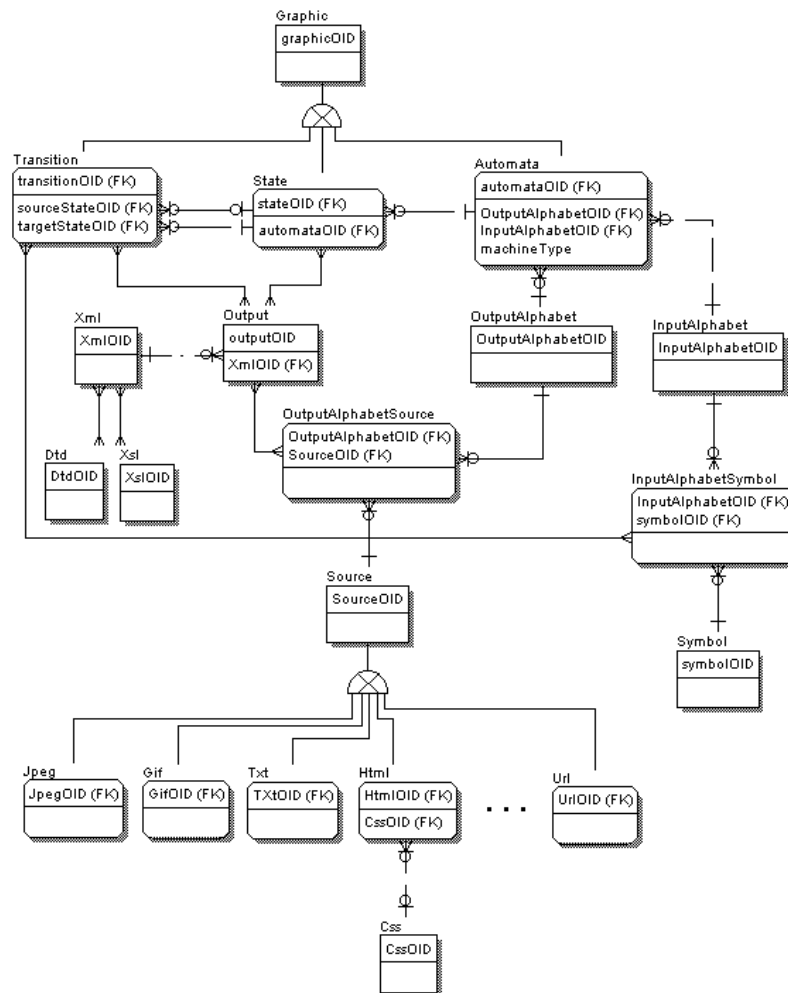


FIGURA 3.2 – Diagrama entidade-relacionamento do sistema

3.4 Gerenciamento do Armazenamento

Uma última abordagem a considerar sobre os formatos para os cursos realizados através do Hyper-Automaton é seu armazenamento persistente em banco de dados. Foi elaborado um diagrama entidades-relacionamentos (DER) a partir do diagrama de classes do Hyper-Automaton, que foi gerado através de mapeamento do paradigma da orientação a objetos para o paradigma relacional [AMB00], [LER99], [SIL99]. Como podemos observar (fig.3.2), existem várias tabelas relacionadas à formatação dos documentos apresentados via *Web*. Em relação aos conceitos do Hyper-Automaton, os documentos disponibilizados aos usuários são funções de saída sobre um alfabeto.

As saídas, (tabela *Output*), especializadas em documentos GIF, JPEG, etc., associam-se, portanto, a alfabetos de saída (tabela *OutputAlphabetSource*), que as vinculam aos hyper-automata, e um formato que as define, conforme os conceitos de XML (tabela *Xml*). Cada documento XML armazenado pode se relacionar, por sua vez, a declarações de tipo de documento (tabela *Did*) e a códigos XSL (tabela *Xsl*). Ainda relacionado ao tema formatação, podemos relacionar estilos diretamente aos documentos HTML armazenados através da tabela CSS.

3.5 XML (*eXtensible Markup Language*)

Conforme abordado em [MAC 00], XML é um texto baseado em linguagem de marcação que está se tornando rapidamente um padrão de troca de dados na Web. Em parte é similar à HTML, pois pode-se identificar dados utilizando marcações (identificadores inclusos).

As marcações XML identificam (explicitam) o que o dado significa de forma melhor que o HTML, o qual preocupa-se somente como os dados serão dispostos visualmente quando acessados pelos usuários. Uma marcação XML age como um campo de nome, permitindo identificar com um rótulo um determinado dado (por exemplo, "<capítulo>...</capítulo>"), tornando o documento muito mais claro ao desenvolvedor e aos programas de manipulação (denominados de parsers).

Um outro aspecto fundamental dos documentos XML é que estes podem ser compostos por vários módulos. É permitido assim que se possa acessar uma parte de um documento sem que seja necessário ao cliente carregá-lo totalmente. Assim sendo, documentos podem ser constituídos por peças separadas, as quais podem estar localizadas em qualquer repositório na rede, facilitando a atualização e reutilização de hiperdocumentos [LIG99].

3.6 A DTD (*Data Type Definition*)

A DTD significa, em português, declaração de tipo de documento. Esta tem por função indicar as regras que o documento XML está seguindo. Como se pode observar, é o centro do esquema da fig.3.1 (direita).

As regras contidas na DTD podem estar inclusas em um documento XML, chamado de subgrupo interno, ou em outra unidade, referenciadas pelo URF, chamado de subgrupo externo. É oportuno ressaltar que nada impede que tenhamos em um mesmo documento XML os dois tipos de DTD's.

A importância da utilização de documentos XML em acordo com uma DTD é fundamental quando é necessário construir documentos obedecendo a padrões corretos de elaboração física e/ou lógica [MAC01a]. Dessa maneira os documentos elaborados podem ser:

- Documentos inválidos: é quando um documento XML não é validado contra uma DTD associada ao documento, não existindo elementos e atributos declarados. Assim sendo, nenhuma verificação é feita para assegurar que cada elemento contenha os subelementos que a DTD informa que deveriam conter.
- Documentos válidos: o documento XML válido deve obedecer a todas as regras. Este só não deve ter uma DTD, mas cada elemento deve estar em conformidade com as regras que a DTD contém. Este é o modelo no qual os documentos XML geralmente serão criados e atualizados. As regras de validade do XML são uma camada adicional às regras para serem bem formados. Todo documento XML válido deve ser também bem formado.

Os documentos XML podem fazer referência a recursos não-XML que estão fora do documento, como por exemplo: arquivos de imagem, clipes de vídeo, clipes de áudio, arquivos de processador de texto e *applets* Java [LIG99]. Para que se possa vincular estes arquivos externos em um documento é necessário que estejam vinculados a uma entidade externa, convenientemente declarada na DTD, por exemplo:

```
<!ENTITY fig.estrela SYSTEM "/images/estrela.gif" NDATA GIF>
```

Desta maneira, entidades terão função fundamental em nosso sistema, pois é nesta parte da DTD que todos os arquivos pertencentes a uma página do curso *on-line* serão "apontados". Para enfatizar melhor o uso de entidades, veja a lista de possibilidades [LIG99], [HAR99]:

- Representar caracteres que não são padrão no seu documento XML;
- Funcionar como abreviação para frases freqüentemente utilizadas;
- Manter partes de marcação que podem aparecer em mais de um documento XML;
- Manter seções ou capítulos de um amplo documento XML;
- Representar recursos que não são XML.

3.7 Folhas de Estilo (Style Sheets)

A noção de folhas de estilo é complementar a estrutura de documentos. Documentos contêm conteúdos e estrutura, e as folhas de estilo descrevem como os documentos devem ser apresentados. Esta apresentação é uma necessidade para tornar independente do dispositivo o conteúdo do documento, isto é, todas as características necessárias para o conteúdo ser apresentado de maneira correta em um dispositivo específico é informada na folha de estilo, pois isto simplifica o gerenciamento do documento, uma vez que um único documento pode estar associado a muitas folhas de estilo [HAR99], [BRA00], [MAC01].

Por exemplo, se um documento XML utiliza elementos como: autor, nome e *e-mail* (list.3.1), não haveria modo de dizer como este conteúdo deveria ser apresentado.

Markup:

```
<autor>
  <nome>Irwing Cosow</nome>
  <email>cosow@w3.org</email>
</autor>
```

Style sheet:

```
autor {font: 12pt Times}  
nome {font-weight: bold}  
email {font-style: italic}
```

LISTAGEM 3.1 - CSS em XML

CSS foi desenvolvido em 1994 pela CERN com a meta de criar uma linguagem de folha de estilo para a Web que fornecesse ao autor um maior controle de estilos sobre documentos HTML. Em 1996, CSS1 (o primeiro nível de CSS) tornou-se uma recomendação da W3C. Em 1997, CSS1 passou a ser suportada pelos principais navegadores, incluindo o Netscape Navigator e o Microsoft Internet Explorer, bem como várias ferramentas de autoria [LIE99a].

CSS utiliza regras declarativas para anexar estilos aos documentos. No exemplo abaixo, uma simples regra informa que todos os elementos "P", da classe "definição" são exibidos com o texto em vermelho com um fundo branco:

```
P.Definição {  
    color: red;  
    background: white;  
}
```

LISTAGEM 3.2 – Regras declarativas

CSS1 suporta *screen-based formatting*, incluindo fontes, cores, e *layout* [LIE99], [LIE99a]. Antes das folhas de estilo existirem, autores de conteúdo para a *Web* criavam imagens de texto para construir fontes (no caso de notação matemática) e cores convenientes. Isto resultou que a maior parte da largura de banda é usado não para texto e sim para elementos gráficos. Além do mais, folhas de estilo tem o potencial de aumentar significativamente a performance da rede de comunicação de dados, como concluído por um recente estudo de como a tecnologia afeta a performance da rede [LIE99a].

Utilizando-se folhas de estilo ao invés de imagens, permite-se acessibilidade, característica de grande valia em sistemas de ensino a distância. Como exemplos: um sintetizador de voz pode ler um texto codificado em HTML para um usuário cego; o texto pode também ser apresentado em algum dispositivo que permita a leitura em braille; notação matemática com fontes não-padrão pode ser visualizada em diversos navegadores.

O próximo nível da CSS, a CSS2, surgiu em 1998 [BOS98] e, de acordo com a W3C, fortalece a acessibilidade a *Web* através da adição de conceitos de multimídia em folhas de estilo. Por exemplo, uma folha de estilo pode aplicar no documento, caso o dispositivo suportar, recursos sonoros.

3.8 Folhas de Estilo Avançadas

A *Extensible Stylesheet Language* (XSL), que está sendo definida pela W3C, oferece um passo adiante neste conceito, pelo fato desta estar apta a transformar a estrutura do documento. Por exemplo, uma folha de estilo XSL pode automaticamente gerar uma tabela de conteúdos (ou índice automático) através da extração dos títulos dos capítulos de um documento. O uso de XSL para transformar os dados estruturados em XML, traz grandes benefícios para a área de publicação de documentos nos mais variados formatos.

Muitas páginas *Web* misturam dados declarativos (como HTML, XML e CSS) com programas executáveis (como *scripts* e *applets*), pois desenvolvedores de conteúdo são motivados a usá-los para permitir efeitos especiais de apresentação (como menus especiais *pop-up*). O mais importante é o fato de que até agora não foi levado em consideração pelos desenvolvedores os custos e benefícios antes confiados aos *scripts* e *applets* para mostrar a informação. Os custos incluem [LIE99a] :

- Acessibilidade: conteúdo misturado com programa, torna este escondido aos *sites* de busca (*search engines*), e torna difícil se não impossível, converter este conteúdo em outro formato.
- Manteneabilidade: dados declarativos são mais fáceis de manter sua longevidade comparados aos programas.
- Independência do equipamento: muitos *scripts* são incompatíveis entre navegadores.

Com o desenvolvimento das folhas de estilo, espera-se que os efeitos mais comuns de apresentação estejam fazendo parte de regras declaradas no estilo. Por exemplo, a CSS2 inclui a funcionalidade de alteração na cor de um elemento ao passar do cursor do *mouse*, tal efeito somente era possível através da utilização de *scripts*.

O mais importante é que o padrão XML é independente do estilo, o que permite que para cada documento seja aplicado os mais variados formatos de visualização, podendo ser aplicado diferentes folhas de estilo para produzir documentos de saída diversos [CAG99].

3.9 Comparação entre CSS e XSL

XSL é frequentemente comparada a CSS como uma maneira de aplicar diferentes formatos para as marcações XML. Entretanto esta comparação é um pequeno engano. CSS lê cada elemento XML como se este fosse "escaneado" (fig.3.3) no documento e aplica estilos em ordem. Em outras palavras, CSS não altera a estrutura do documento XML, ela somente muda a aparência visual para cada elemento. Se for colocado um nome no topo de um documento XML, CSS irá colocar este nome no mesmo local, ao menos que seja explicitado outra posição absoluta. Além do mais, CSS irá tratar cada marcação de um dado tipo exatamente da mesma maneira.

Do ponto de vista dos desenvolvedores, pode-se alcançar uma maior flexibilidade usando uma combinação das três tecnologias: XML contém os dados, CSS (na forma interna ou externa) provê o estilo ao documento (manipulação da apresentação), enquanto o XSL é usado para modificar a estrutura do documento. Através da separação destes pedaços, o controle sobre alteração dos dados começa a ser total e os benefícios são significativos [CAG99].

Para concluir, CSS não pode filtrar, reordenar dados, adicionar texto ou subordinar a estruturas HTML. Alguns destes problemas, podem ser resolvidos através da utilização de ambientes DHTML (*Dynamic HTML*), mas esta tecnologia é caracterizada por implementações proprietárias.

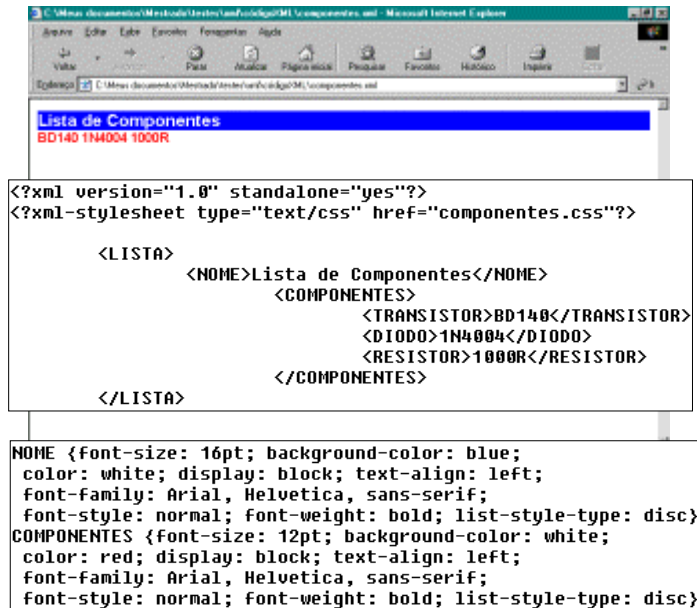


FIGURA 3.3 – Utilização do XML com CSS

XSL permite solução complementar para a formatação de XML. Diferente de CSS, o qual aplica informação de estilo para cada nodo XML que for encontrado na seqüência, XSL efetivamente repõe uma informação com a outra, independentemente da ordem que a informação encontra-se estruturada. Outras características de grande importância é a transformação que o XSL pode fazer em um documento XML tornando-o em outro documento de estrutura diferente como XML, HTML, texto [CAG99], [MOU99]. Diferentemente de outra linguagem de transformação, XSL possui o benefício de ser escrita em XML, o que significa que o mesmo *parser* que pode manipular dados XML pode também manipular XSL.

XSL consiste em uma série de *templates* que podem ser usados para encontrar informações em um documento. Estas *templates* aplicam padrões ao fluxo de informações de entrada que as transformam em um outro fluxo de saída, o qual pode conter código HTML.

XSL possui um número de ferramentas para fazer comparações condicionais, ordenamento e executar operações em grupo. Assim sendo, fica justificado que a saída do processamento XSL, está de longe, desvinculada da seqüência em que as marcações aparecem no documento XML. Por exemplo, consideraremos o seguinte código XSL que manipula a criação de um "nome":

```
<xsl: template match="nome">
  <h1><xsl:value-of/></h1>
</xsl:template>
```

LISTAGEM 3.3 – Exemplo de XSL

Esta simples *template* XSL irá chamar em qualquer momento o processador XSL a encontrar a marcação "<nome>" (neste caso somente será um). Quando houver a ocorrência, o *parser* XSL irá, de acordo com o texto da marcação "<xsl:value-of/>", colocá-lo entre duas marcações "<h1>" na saída da informação.

O exemplo final abaixo, mais completo, ilustra com listagens dos códigos XML/XSL, exemplificando como ficaria a saída (fig. 3.4) para o documento, através da

aplicação da folha de estilo em questão. Pode-se ainda combinar XML/XSL com JavaScript para criar complexas aplicações [CAG99], [MOU99].

```
<?xml version="1.0"?>
<?xml:stylesheet type="text/xsl" href="dados.xsl"?>
<ender ano="2000">
  <nome>Cesar C. Machado</nome>
  <residencial>
    <rua>Vila Real, 1160</rua>
    <cep>96083-370</cep>
    <tel>278.8704</tel>
    <estado>RS</estado>
  </residencial>
  <email>
    <email1>machado@atlas.ucpel.tche.br</email1>
    <email2>machado@etfpel.tche.br</email2>
  </email>
  <url>atlas.ucpel.tche.br/~cmachado</url>
</ender>
```

LISTAGEM 3.4 – Arquivo fonte XML para dados pessoais

```
<xsl:template match="/">
  <HTML xmlns:xsl="http://www.w3.org/TR/WD-xsl">
    <HEAD>
      <TITLE>
        <xsl:for-each select="ender">
          <xsl:value-of select="@ano"/>
        </xsl:for-each>
        - Dados pessoais atualizados
      </TITLE>
    </HEAD>
    <BODY>
      <xsl:for-each select="ender">
        <H1>
          <xsl:value-of select="@ano"/>
          Dados pessoais atualizados
        </H1>
        <H2>
          <xsl:value-of select="nome"/>
        </H2>
        <xsl:for-each select="residencial">
          <H2>
            telefone - <xsl:value-of select="tel"/>
          </H2>
        </xsl:for-each>
      </xsl:for-each>
      <HR/>
      <A HREF="http://atlas.ucpel.tche.br/~cmachado">Cesar C. Machado
      </A><BR/>
      <A HREF="mailto:cmachado@atlas.ucpel.tche.br">e-mail</A>
    </BODY>
  </HTML>
</xsl:template>
</xsl:stylesheet>
```

LISTAGEM 3.5 – Arquivo fonte XSL para dados pessoais



FIGURA 3.4 – Resultado da visualização em navegador Web

3.10 Conclusões

Este artigo apresentou as principais características da linguagem de marcação de documentos estruturados na *Web*, a *eXtensible Markup Language* (XML), a qual oferece significativos avanços em termos de disponibilização e visualização de documentos e a sua utilização como extensão ao sistema *Hyper-Automaton*.

Todas as características tratadas neste trabalho, fazem parte da implementação para a inclusão da tecnologia XML, pois como se pode observar há o envolvimento de vários módulos com diferentes implementações, isto é, estrutura dos documentos em marcações apropriadas, elaboração correta da DTD e folhas de estilos XSL.

O núcleo central deste trabalho compreende uma DTD que funcionará basicamente com ponteiros a fragmentos de documentos existentes na hiperbase do curso na *Web*, os quais são alvo de folhas de estilo XSL, que de posse de uma DTD interna referenciada a este núcleo, terá então a capacidade de formatação visual dos aspectos necessários para a visualização dos conteúdos no navegador do usuário. O projeto neste estágio, encontra-se em fase de estudos para a construção adequada desta DTD.

As características básicas do XML e XSL identificadas neste artigo, como a estrutura clara e de fácil depuração e processamento, a capacidade de identificação de dados, o poder de estilização de conteúdos, a reusabilidade de definições e a manipulação de *links*, trazem benefícios adicionais ao sistema *Hyper-Automaton*, entre os quais: possibilidade de implementação de ferramentas de busca sobre hiperdocumentos; maior estruturação das informações disponibilizadas em hiperdocumentos e facilidades para a manipulação e gerenciamento dos dados; flexibilização da função de saída; múltiplas formatações de hiperdocumentos para um mesmo conteúdo; capacidade do usuário em alterar a formatação de saída dos hiperdocumentos de acordo com preferências pessoais; flexibilização do processamento de hiperdocumentos em operações de composição; capacidade de formatação de caracteres matemáticos.

4 Definição e aplicação de regras para a elaboração adequada de documentos XML para o sistema Hyper-Automaton

Este artigo [MAC01a] foi submetido e publicado na categoria de artigo completo no *Fifth International Query Processing and Multimedia Issues in Distributed Systems Workshop QPMIDS'2001* em conjunto com a *12th International Conference on Data and Expert Systems Application DEXA'01* e *IEEE Computer Society*, ocorrido de 3 a 4 de Setembro de 2001 na cidade de Munich na Alemanha.

Este workshop visa reunir pesquisadores, desenvolvedores, usuário e demais interessados na área de Processamento de Consultas e Gerenciamento de Multimídia em Sistemas Distribuídos. Avanços recentes na área de gerenciamento de multimídia tornam o uso de processamento de consultas e recuperação do conteúdo mais crítica em sistemas distribuídos. Efetivos e eficientes métodos de gerenciamento de processamento de consultas sobre dados multimídia (imagem, vídeo, som e texto) são necessários para habilitar de maneira rápida e com alta qualidade o acesso a estes tipos de dados para bancos de dados distribuídos.

Nessa direção, este artigo descreve de maneira prática e com exemplos previamente testados com forte ênfase na construção de regras corretas e rígidas que determinarão o nível de flexibilidade sobre estruturas XML armazenadas pertencentes a cursos na *Web* com conteúdo multimídia. Tal desenvolvimento traz uma análise de como estas regras poderão ser aplicadas no Sistema *Hyper-Automaton* para que tenhamos um formalismo correto na estruturação de documentos XML sempre obedecendo o conceito de documentos válidos e bem formados, pois a padronização e não ocorrência de erros de processamento são fundamentais.

4 Definição e aplicação de regras para a elaboração adequada de documentos XML para o sistema Hyper-Automaton

4.1 Resumo

Este trabalho dá prosseguimento a estudos anteriores buscando oferecer uma forma de estruturação e visualização do conteúdo de cursos disponibilizados na *WWW* através da utilização de construções formais conhecidas como Autômatos Finitos Determinísticos com Saída. O estudo baseado nas evoluções tecnológicas da Internet sugere que o sistema atual incorpore os benefícios da estruturação moderna de documentos, consoantes com as necessidades e benefícios tecnológicos, bem como, faça uso de importantes recursos derivados das linguagens de marcação referentes às possibilidades de formatação do conteúdo disponibilizado visualmente. O objetivo deste trabalho é esquematizar e descrever as regras de acordo com uma visão aplicada e amparada bibliograficamente, regras estas, que em um sentido geral, opcionalmente fazem parte de um documento XML, mas no sistema *Hyper-Automaton* o qual desenvolvemos, serão indubitavelmente um elemento permanentemente integrante e imprescindível.

4.2 Introdução

Este trabalho continua o estudo da Modelagem de Cursos na *Web* Utilizando Sistemas Formais [MAJ00], [MEN99] e tem como objetivo propor alternativas com respeito à formatação da interface com o usuário através da utilização do XML e de suas aplicações.

Cursos na *Web* Utilizando Sistemas Formais, utilizam construções formais conhecidas como Autômatos Finitos Determinísticos [MEN00]. Foi introduzido pelo grupo o conceito de "cursos são autômatos" como uma estrutura que permite fácil implementação, criação de material hipermídia independente do autômato, ao mesmo tempo que encoraja o reuso de páginas *Web* em vários cursos, os quais podem ser construídos com enfoques diferentes, diminuindo a redundância na criação de páginas [MAJ00], [MEN99], [MAJ98].

O objetivo geral do modelo *Hyper-Automaton* centra-se no estudo da aplicação do formalismo de Autômatos Finitos com Saída (Máquina de *Mealy* e Máquina de *Moore*) como um modelo estrutural para a organização de hiperdocumentos instrucionais, em especial de Cursos na *Web*. O modelo é inspirado por pesquisas clássicas na área de hiperdocumentos e recentes iniciativas na *WWW*, com especial enfoque no desenvolvimento de sistemas de hipertexto onde a base de hiperdocumentos é projetada de forma independente da estrutura de controle da aplicação hipermídia, e suporta algumas facilidades como a composição de estruturas hierárquicas, especificação de vários conjuntos de *links* sobre um mesmo corpo de hiperdocumentos e objetos separados da estrutura de navegação.

Cada autômato define um curso e consiste de um conjunto de hiperdocumentos independentes, os quais podem pertencer a outros cursos. A função de transição funciona como ligação lógica entre os hiperdocumentos e a função de saída compõe as páginas. O resultado final é a estrutura básica de páginas e *links* de hipertexto em um site na *Web*. O modelo leva a um alto grau de modularização do material instrucional, apresentando as seguintes vantagens: facilidade de reuso de páginas em diversos cursos,

com eliminação da redundância; independência dos hiperdocumentos da estrutura do autômato, cuja alteração não influi nas páginas e vice-versa; permite que qualquer usuário crie *links* de e para qualquer documento; facilidade de implementação e manutenção; interface gráfica simples e direta; elaboração de seqüências instrucionais com enfoques específicos e capaz de oferecer estudo individualizado; operações categoriais fornecem um esquema de composição de cursos que permite a construção de novos cursos sobre cursos já existentes através de procedimentos de alto nível [MAJ00], [MEN99].

O sistema *Hyper-Automaton* constitui-se de um sistema semi-automatizado para o suporte a cursos na Web (com base em uma arquitetura cliente/servidor e interface desenvolvida em HTML) através da aplicação de conceitos inerentes à Ciência da Computação, em especial da Teoria de Autômatos, tecnologia de Hiper-mídia e Teoria das Categorias, reunindo os benefícios de ambas. Embora a ênfase utilizada para a validação do modelo seja a implementação de sistemas para o ensino a distância, os resultados obtidos se aplicam para sistemas de hipertexto como um todo.

Nosso trabalho na fase atual, está centrado nas várias possibilidades de formatação do conteúdo disponibilizado, culminando em uma adequada interface com o usuário, onde, a flexibilização da formatação do conteúdo obtido na função de saída do autômato será uma aplicação que remonte o documento nas partes desejadas e com as características mais adequadas relacionadas as necessidades do usuário e do curso em questão. O estudo proposto trará uma maior flexibilidade ao já existe, pois no estado atual, o que se apresenta é uma única e rígida possibilidade composta de fragmentos em HTML.

Por considerações que excluem explicitamente qualquer conotação comercial, os trabalhos aqui desenvolvidos foram avaliados utilizando o Internet Explorer 5.5, sendo este, o único browser que a partir da versão 4, suporta os recursos XML padrão.

4.3 Visão do Sistema

O diagrama em blocos que ilustra as partes do sistema, figura 4.1, inclui o que estamos atualmente desenvolvendo. A novidade encontra-se em estruturar os conteúdos em XML obedecendo as definições preestabelecidas para os documentos (DTD) [LIG99], [HAR99] e utilização de *templates* XSL e/ou folhas de estilos em cascata (CSS) [LIE99], [BOS98] afim de visualização e *layout* do referido curso on-line.

Estas alterações objetivarão principalmente a atualização tecnológica do sistema de acordo com os benefícios já descritos em trabalhos anteriores relacionados ao XML [MAC00], [MAC00a].

A partir deste momento este trabalho descreverá uma das mais importantes características do XML no tocante ao estabelecimento de regras de construção. Conhecidas como *Data Type Definition* ou Definição do Tipo de Documento (DTD), estas regras estão intimamente ligadas aos nossos objetivos, pois estabelecem critérios de elaboração e ligação com arquivos não-XML ou outros tipos de arquivos que normalmente compõe documentos normalmente exibidos na *Web*, estes arquivos em nosso sistema, serão denominados por fragmentos.

Pormenorizadas na figura abaixo, estas características indubitavelmente farão parte e serão amplamente utilizadas na nova proposta de implementação sobre o sistema de *Hyper-Automaton*.

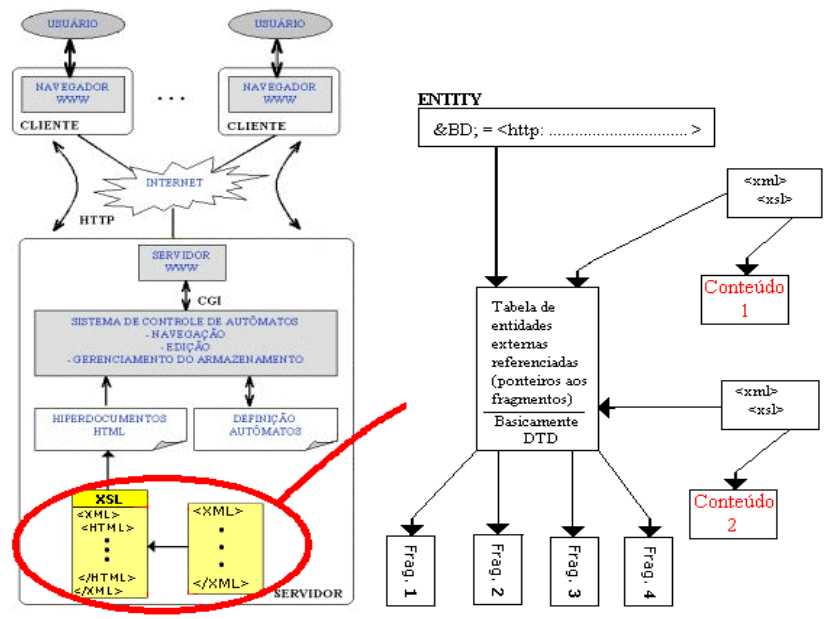


FIGURA 4.1 – Esquema geral

4.4 A DTD (Data Type Definition)

Segundo [HAR99], XML é descrito como uma *meta-markup language*, isto é, uma linguagem elaborada para descrever linguagens de markup, como por exemplo: MathML, VML, entre outras. Desta maneira a DTD é o local de estabelecimento das regras que a aplicação seguirá.

A DTD, significa em português, declaração de tipo de documento e como se pode observar é o centro do esquema da figura 4.1. Esta tem por função indicar a regra que o documento XML está seguindo. De maneira mais específica, a DTD relaciona uma lista de elementos, atributos, notações, e entidades contidas em um documento, bem como as relações entre estas. A DTD especifica um conjunto de regras estruturais de um documento. Por exemplo, uma DTD determina que um documento *página* (de um curso, por exemplo) possua exatamente um *título* filho, um ou mais *autores* filhos e não deve obrigatoriamente conter um *subtítulo*.

DTD's podem ser incluídas no arquivo que contém a descrição do documento ou ligadas a uma URL externa.

DTD externas podem ser compartilhadas por diferentes documentos e *Web sites*. Estas determinam que documentos elaborados em qualquer tempo e lugar, obedeçam a regras as quais objetivam estabelecer padrões de estrutura, clareza e legibilidade.

Um exemplo prático que se pode ilustrar, é o fato de que várias pessoas estarem juntas desenvolvendo documentos relacionados a um mesmo curso na Web, estas podem estar separadas geograficamente trabalhando individualmente em capítulos distintos, mas pelo fato destas estarem inserindo conteúdo obedecendo à regras estruturais especificadas em uma DTD, no momento em que houver a integralização dos documentos como um todo, haverá, desta maneira, um padrão estrutural do referido curso.

Assim sendo, a DTD descreve como os diferentes elementos de uma página são arranjados sem manipular diretamente estes dados. A DTD permite que seja visualizada a estrutura do documento separado dos dados. Isto significa que independente das mais

exageradas formas de estilos aplicado ao documento relacionado este estará sempre imune, separado das alterações que este possa vir a sofrer, em outras palavras, pintar uma casa sem alterar sua arquitetura básica. A DTD está em um nível em que o usuário não o percebe, mas onde os aplicativos como *JavaScripts*, CGI, banco de dados e outros programas podem usá-la [HAR99].

Os documentos XML podem fazer referência a recursos não-XML que estão fora do documento, como por exemplo: arquivos de imagem, clipes de vídeo, clipes de áudio, arquivos de processador de texto e applets Java [LIG99]. Para que se possa vincular estes arquivos externos em um documento é necessário estejam vinculados a uma entidade externa, convenientemente declarada na DTD.

Desta maneira entidades terão função fundamental em nosso sistema, pois é nesta parte da DTD que todos os arquivos pertencentes serão “apontados”. Para enfatizar melhor o uso de entidades, veja a lista de possibilidades [LIG99]:

- Representar caracteres que não são padrão no seu documento XML;
- Funcionar como abreviação para frases freqüentemente usadas;
- Manter partes de marcação que podem aparecer em mais de um documento XML;
- Manter seções ou capítulos de um amplo documento XML;
- Organizar a sua DTD em unidades lógicas;
- Representar recursos que não são XML.

4.5 Declaração do tipo de documento – *Document Type Declarations*

A declaração do tipo de documento especifica a DTD que é utilizada pelo documento e esta aparece no prólogo de um documento, após a declaração XML mas antes da raiz do elemento. Este deve conter a definição do tipo de documento ou a URL que identifica o arquivo onde a definição do tipo de documento está localizada. Em nosso caso utilizaremos duas DTD’s, comumente chamadas de DTD interna e externa. A interna estabelecerá regras estruturais para a construção do documento, sendo esta não obrigatória, e a externa servirá como ponteiro aos fragmentos que fazem parte da composição das páginas do curso armazenado.

4.6 Declaração de elementos

Cada *tag* utilizado em um documento XML válido deve ser declarado como um elemento na DTD. A declaração de um elemento define o nome e o possível conteúdo do elemento. A lista de conteúdos é algumas vezes chamada de especificação de conteúdo [HAR99]. Esta especificação utiliza uma gramática simples para precisamente especificar o que é permitido e o que não é permitido em um documento. A figura abaixo, exemplifica a quantidade de ocorrências de um elemento em um documento:

TABELA 4.1 – Especificação de conteúdo

Símbolo	Nome	Significado
?	Interrogação	Opcional (zero ou um)
*	Asterisco	Zero ou mais
+	Mais	Um ou mais

Observe o exemplo abaixo, mais algumas regras utilizadas em nossas especificações:

```
<?xml version="1.0" standalone="yes">
<?xml-stylesheet type="text/css" href="standard.css"?>
<!DOCTYPE fragmentoum
[
  <!ELEMENT FRAGMENTO (INDENTFRAG, TITULO, CONTEUDO, IMAGEM*, SOM*)>
  <!ELEMENT INDENTFRAG (#PCDATA)>
  <!ELEMENT TITULO (#PCDATA)>
  <!ELEMENT CONTEUDO (#PCDATA)>
  <!ENTITY IMAGEM SYSTEM "logo.gif" NDATA GIF>
  <!ENTITY SOM SYSTEM "regra.mp3" NDATA MP3>
]>
```

LISTAGEM 4.1 – Especificação de conteúdo

Nota-se na listagem anterior que denota um fragmento, pela sua simplicidade, foi associada uma folha de estilo em cascata [LIE99a], [NIE99]. Nossa idéia é aplicar a folha de estilo na saída da página do documento para o usuário, com a finalidade de flexibilizar ao máximo a formatação dos aspectos relevantes ao desejo de quem utiliza o referido curso.

Na quarta linha impõe-se uma restrição importante que relaciona a seqüência em que os itens que compõe um fragmento deverão ser compostos, isto é: INDENTFRAG, TITULO e CONTEUDO; deverão existir em uma só vez, IMAGEM* e SOM*, zero ou mais vezes em um documento.

Os elementos INDENTFRAG, TITULO e CONTEUDO, são compostos de caracteres convencionais menos *markup*, por isso a definição #PCDATA. Finalmente os elementos IMAGEM e SOM, entidades binárias, que requerem tratamento especial na especificação e cuja ocorrência não é freqüente, serão arquivos totalmente desvencilhados e apontados pelos documentos que as incluem. O fato de serem vinculados e não incluídos no documento XML reforça a idéia de máxima flexibilidade. Esta uma pequena desvantagem do XML em relação ao HTML, pois nesta segunda linguagem, a inserção de imagens e sons é um pouco mais direta.

4.7 Entidades

Um documento XML deve conter dados e declarações de muitas diferentes fontes e arquivos. Deste modo os dados devem estar direcionados para um banco de dados, CGI scripts, ou ainda outra fonte de dados. Os itens aonde os pedaços de um XML são armazenados são chamados de entidades. A referência a estas entidades tem por objetivo carregá-las para o documento principal XML .

A unidade de armazenamento que contém as declarações XML, a DTD, e a raiz do elemento é chamada de *documento entidade*. Entretanto, o elemento raiz e seus descendentes podem também conter entidades referenciadas apontando para dados adicionais que devem ser incluídos no documento [HAR99], [MOU99].

Existem dois tipos de entidades: internas e externas. As entidades internas são definidas completamente dentro do documento entidade. As entidades externas relacionam seu conteúdo para uma outra fonte localizada por uma URL. O documento principal somente inclui a referência para a URL onde o conteúdo reside.

Uma utilização prática das entidades internas é quando existe repetição excessiva

de um mesmo texto em várias partes do documento. Outra grande vantagem é possibilidade de fácil atualização dos conteúdos referenciados. Observe o seguinte exemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DOCUMENT [
  <!ENTITY omega "&#937;">
  <!ENTITY aacute "&#225;">
  <!ENTITY Eacute "&#201;">
  <!ENTITY Ecirc "&#202;">
  <!ENTITY UFRGS "UFRGS - Universidade Federal do Rio Grande do Sul">
  <!ENTITY IN "Conceitos b&aacute;sicos de eletricidade">

  <!ELEMENT DOCUMENT (TITLE, CONTEUDO)>
  <!ELEMENT CONTEUDO (TITULO, ITEM, LAST_MODIFIED)>
  <!ELEMENT TITLE (#PCDATA)>
  <!ELEMENT CAPITULO (#PCDATA)>
  <!ELEMENT ITEM (#PCDATA)>
  <!ELEMENT LAST_MODIFIED (#PCDATA)>
]>
<LICAO>
  <TITLE>&IN;</TITLE>
  <CONTEUDO>
    <CAPITULO>2000 &UFRGS;</CAPITULO>
    <ITEM>RESIST&Ecirc;NCIA      EL&Eacute;TRICA      -      UNIDADE:
OHM(&omega;)</ITEM>
    <LAST_MODIFIED>Outubro 16, 2000</LAST_MODIFIED>
  </CONTEUDO>
</LICAO>
```

LISTAGEM 4.2 – Entidades internas

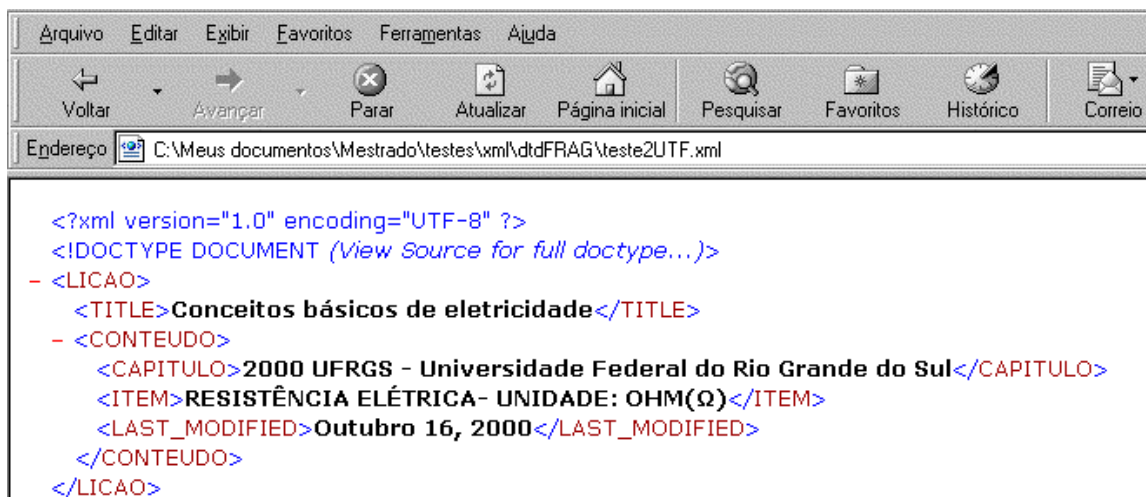


FIGURA 4.2 – Entidades internas

Observando o exemplo acima, nota-se que ainda não houve preocupação com a formatação visual, ou seja, não foi aplicada para esta listagem nenhuma folha de estilo (fig. 4.2). Outro importante aspecto é a definição do padrão de codificação de caracteres, para isto foi escolhido, neste exemplo, UTF-8, suportando além do padrão ASCII uma outra gama de caracteres, possibilitando acentuação correta para a observância da ortografia.

4.8 Entidades gerais externas

Entidades externas são dados localizados no exterior do arquivo principal que contém o elemento raiz / documento entidade. Entidades externas permitem que sejam incluídos estes arquivos (entidades) externos fazendo com que um documento XML seja constituído de vários outros arquivos independentes [LIG99], [HAR99], [MOU99].

Documentos que fazem uso de somente entidades internas, remontam claramente o modelo HTML. O documento texto completo é disponibilizado em somente um arquivo. Imagens, *applets*, sons e outros dados não-HTML devem ser ligados, mas todo o texto está presente. Naturalmente o modelo HTML apresenta alguns problemas, tais como a dificuldade de incluir informações dinâmicas no arquivo, sendo possível somente através de CGI, *Java applets*, programa de banco de dados, etc. O modelo HTML somente permite um documento estático. Uma solução para este problema, seria a utilização de *frames*, mas este se comporta de maneira irregular e confusa para uma grande parte dos usuários [HAR99].

Parte deste problema é o fato de que um documento HTML não encaixar dentro de um outro de maneira natural. Todo documento HTML deve possuir somente um corpo (BODY), e não mais. Os servidores somente habilitam fragmentos de HTML e nunca um documento inteiro válido dentro de outro documento.

XML, entretanto, é mais flexível. A raiz de um documento não é necessariamente a mesma ou uma outra. Caso dois elementos compartilham a mesma raiz de um documento, a DTD deve declarar quais os elementos a constituem.

XML vai mais além, ou seja, é permitido que um documento qualquer seja formado por múltiplos pequenos pedaços de documentos XML localizados nos locais mais remotos da rede. O *parser* é responsável pela montagem correta destes fragmentos. Desta forma, documentos podem conter outros documentos, que por sua vez podem conter outros, do ponto de vista da aplicação existe somente um arquivo completo. Este processo é realizado no lado do cliente.

Para que uma entidade externa passe a fazer parte de um documento, devemos declará-la da seguinte maneira: `<!ENTITY name SYSTEM "URI">`, de maneira mais precisa, `<!ENTITY CAPITULO SYSTEM "http://atlas.ucpel.tche.br/~cmachado/capituloum.xml">`. A inclusão é feita de maneira semelhante ao exemplo anterior através do código da entidade `&CAPITULO;`.

A figura 4.3 abaixo, tem o objetivo de esclarecer a possibilidade de inclusão de arquivos XML no interior de outros arquivos XML.

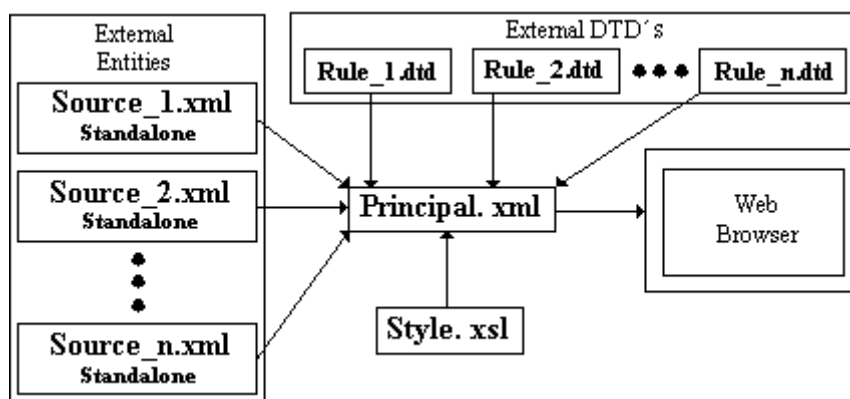


FIGURA 4.3 – Entidades gerais externas

4.9 Inclusão de dados não-XML

Existem uma infinidade de tipos de dados e a maioria não é XML. Desta maneira deve-se prover mecanismos para que os mais diferentes formatos de dados possam ser incluídos normalmente em nossas aplicações.

De acordo com [HAR99], este tema é controverso estando ainda em discussão.

O XML permite três construções geralmente usadas para trabalhar com estes tipos de dados, tais como: *notations*, *unparsed external entities* e *processing instructions*.

Notations – O primeiro problema quando utilizamos arquivos não-XML, diz respeito em identificar e informar a aplicação XML como ler e mostrar ao usuário o respectivo dado. Em HTML, por exemplo, no momento em que o *tag* IMAGE é encontrado portando arquivos do tipo GIF ou JPEG, naturalmente o *browser* sabe como lidar com estes, mas como fazer quando o formato é do tipo EPS ou TIFF. Em XML pela sua característica de livre criação de *tags*, outros fatores importantes devem ser citados, por exemplo, em um tag de nome CONTEUDO, pode estar contido uma imagem ou outro recurso multimídia como um arquivo de som no formato MP3, WAV ou MIDI, desta maneira deve-se prover os recursos para o adequado processamento e disponibilização destes arquivos.

Unparsed external entities – XML não é o formato ideal para todo o tipo de dado, particularmente dados não-texto. Um exemplo interessante e nada prático, seria codificar em XML uma figura no formato *bitmap*, armazenando cada bit, observe exemplo: <PIXEL X="121" Y="68" COLOR="BABAFA">.

Em uma típica página Web, inclui arquivos tipo GIF, JPEG, *Java applets*, *ActiveX*, vários tipos de sons, etc. Em XML todos estes arquivos não-XML são chamados de *Unparsed entity*, por que o processador do arquivo não procurará entender estes arquivos, apenas ficará ciente de sua existência. Observe exemplo abaixo:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
  <!DOCTYPE DOCUMENT [
    <!ENTITY aacute "&#225;";>
    <!ENTITY Eacute "&#201;";>
    <!ENTITY Ecirc "&#202;";>
    <!ENTITY rho "&#961;";>
    <!ENTITY UFRGS "UFRGS - Universidade Federal do Rio Grande do Sul">
    <!ENTITY IN "Conceitos b&aacute;sicos de eletricidade">
    <!ENTITY LOGO SYSTEM "logo_tec.gif" NDATA GIF>
    <!NOTATION GIF SYSTEM "image/gif">
    <!ELEMENT DOCUMENT (TITLE, CONTEUDO)>
    <!ELEMENT CONTEUDO (TITULO, ITEM, LAST_MODIFIED, IMAGEM)>
    <!ELEMENT TITLE (#PCDATA)>
    <!ELEMENT CAPITULO (#PCDATA)>
    <!ELEMENT ITEM (#PCDATA)>
    <!ELEMENT LAST_MODIFIED (#PCDATA)>
    <!ELEMENT IMAGE EMPTY>
    <!ATTLIST IMAGE SOURCE ENTITY #REQUIRED>
  ]>
<LICAO>
  <TITLE>&IN;</TITLE>
  <CONTEUDO>
    <CAPITULO>2000 &UFRGS;</CAPITULO>
    <ITEM>RESISTIVIDADE EL&Eacute;TRICA - UNIDADE (&rho;)</ITEM>
    <LAST_MODIFIED>Outubro 16, 2000</LAST_MODIFIED>
    <IMAGEM SOURCE="LOGO" />
  </CONTEUDO>
</LICAO>
```

LISTAGEM 4.3 – Inserção de arquivos não-XML

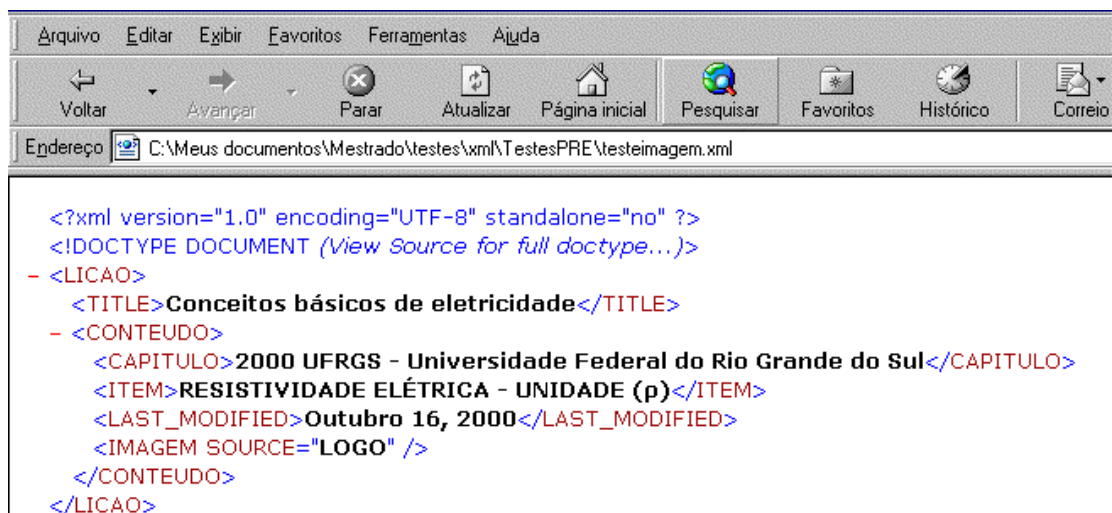


FIGURA 4.4 – Inserção de arquivos não-XML

4.10 Reconhecendo arquivos não-XML

Em termos estruturais conseguimos fazer com que uma saída gerada por um estado ou uma transição do *Hyper-Automaton* fique ciente de que neste momento qualquer recurso não-XML, como imagens, sons ou outro tipo de arquivo, poderá compor o documento a ser mostrado ao usuário. Na listagem abaixo se defini através do código da DTD exemplo, os detalhes da figura um, esquematizada acima.

Observe o exemplo abaixo, pois neste consta o código da DTD das entidades externas compostas dos fragmentos que comporão a saída em um determinado instante. Cabe-se reforçar que neste artigo os aspectos de visualização do conteúdo não serão abordados, sendo assunto para um próximo trabalho.

```

<!NOTATION JPEG SYSTEM "image/jpeg">
<!NOTATION GIF SYSTEM "image/gif">
<!NOTATION WAV SYSTEM "sound/wav">
<!NOTATION MPEG SYSTEM "audio/mpeg">
<!NOTATION MIDI SYSTEM "sequencia MIDI/midi">
<!ENTITY FRAG2 SYSTEM "frags/frag2.gif" NDATA GIF>
<!ENTITY FRAG3 SYSTEM "frags/frag3.mpeg" NDATA MPEG>
<!ENTITY FRAG4 SYSTEM "frags/frag4.gif" NDATA GIF>
<!ENTITY FRAG5 SYSTEM "frags/frag5.midi" NDATA MIDI>
<!ENTITY FRAG6 SYSTEM "frags/frag6.gif" NDATA GIF>
<!ENTITY FRAG7 SYSTEM "frags/frag7.jpeg" NDATA JPEG>
<!ENTITY FRAG8 SYSTEM "frags/frag8.wav" NDATA WAV>

```

LISTAGEM 4.4 – Definição de entidades externas (nucleo.dtd)

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="estilo2.xsl"?>
<!DOCTYPE LICA0 [
  <!ENTITY % NUCLEO_DTD SYSTEM "nucleo.dtd">
  %NUCLEO_DTD;
  <!ENTITY omega "&#937;">
  <!ENTITY aacute "&#225;">
  <!ENTITY Eacute "&#201;">
  <!ENTITY Ecirc "&#202;">
  <!ENTITY UFRGS "UFRGS - Universidade Federal do Rio Grande do Sul">
  <!ENTITY IN "Conceitos b&aacute;sicos de eletricidade">
  <!ELEMENT LICA0 (TITLE, CONTEUDO)>
  <!ELEMENT CONTEUDO (CAPITULO, ITEM, IMAGEM, SOM, LAST_MODIFIED)>
  <!ELEMENT TITLE (#PCDATA)>
  <!ELEMENT CAPITULO (#PCDATA)>
  <!ELEMENT ITEM (#PCDATA)>
  <!ELEMENT LAST_MODIFIED (#PCDATA)>
  <!ELEMENT IMAGEM EMPTY>
  <!ELEMENT SOM EMPTY>
  <!ATTLIST IMAGEM FONTE ENTITY #REQUIRED>
  <!ATTLIST SOM FONTE ENTITY #REQUIRED>
]>
<LICA0>
  <TITLE>&IN;</TITLE>
  <CONTEUDO>
    <CAPITULO>2000 &UFRGS;</CAPITULO>
    <ITEM>RESIST&Ecirc;NCIA          EL&Eacute;TRICA          -          UNIDADE:
OHM(&omega;)</ITEM>
    <IMAGEM FONTE= "FRAG2" />
    <SOM FONTE= "FRAG3" />
    <LAST_MODIFIED>Outubro 16, 2000</LAST_MODIFIED>
  </CONTEUDO>
</LICA0>

```

LISTAGEM 4.5 – Reconhecendo arquivos externos

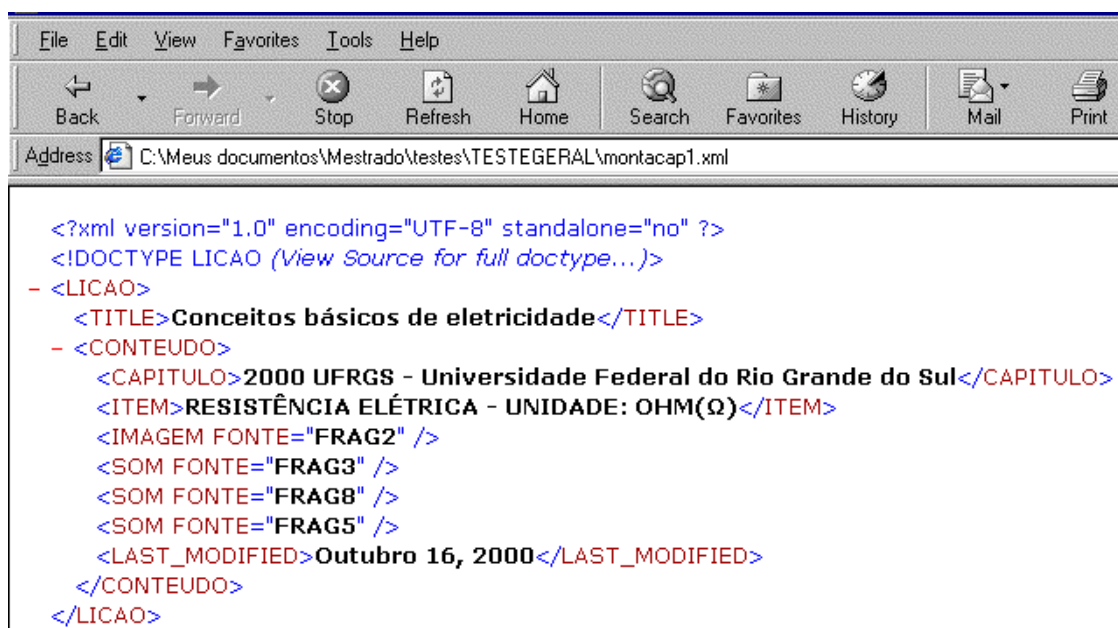


FIGURA 4.5 – Reconhecendo arquivos externos

4.11 Conclusões

A utilização de regras para estabelecer padrões de elaboração de documentos XML orientados ao ensino, possibilitará uma total flexibilidade para a construção adequada de material instrucional pelas mais diferentes pessoas e nos mais diferentes assuntos que possa ser disponibilizados via *Web*.

Embora seja menos direta a utilização do XML para inclusão de recursos não-XML do que o conhecido HTML, esta disposição de entidades externas, organiza de sobremaneira e acima de tudo padroniza nossas construções de acordo com a *World Wide Web Consortium*.

Cabe-se ressaltar que a utilização de entidades externas ainda encontra-se em fase de desenvolvimento, podendo seu uso trazer funcionamento inadequado e imprevisível as aplicações.

Em nosso trabalho, limitamos até o presente momento, utilizar os formatos de arquivos mais comuns, como podemos observar nos exemplos.

Devido ao comportamento não regular dos principais *browsers* hoje no mercado, alguns testes poderão não surtir efeito, pois até agora, não estudamos a causa de possíveis *bugs* oriundos do processamento do XML e seus aplicativos nestes navegadores.

A partir deste ponto, nossa pesquisa toma o rumo dos elementos necessários para possibilitar ao usuário utilizar os recursos multimídia suportados pelo XML e seus aplicativos. Os estudos serão voltados para a flexibilização da visualização do conteúdo e o correto processamento dos mais variados tipos de arquivos de acordo com a finalidade específica, assim sendo, permitir que seja disposta corretamente todas aplicações multimídia suportadas e necessárias a um curso na *Web* através de seu *browser*, desde que este suporte os recursos XML.

5 Flexibilização e adequação do formato de saída ao conteúdo disponibilizado no sistema de Hyper-Automaton

Este artigo [MAC01] foi submetido e publicado na categoria de artigo completo na *International Conference on Internet Computing IC'2001*, ocorrido de 25 a 28 de Junho 2001 na cidade de Las Vegas nos Estados Unidos.

Um número de fatores estão recentemente contribuindo para tornar o uso da Internet como um ambiente para a criação de aplicações. A Internet atualmente é a mais explorada do que qualquer outro sistema de computação na história e que continua a crescer rapidamente. Novas tecnologias, incluindo redes de comunicação de dados de alta velocidade e uso de programas adequados que delas façam o melhor uso, prometem torná-la de muito maior utilização como suporte à distribuição de *software* e outros propósitos. Essa conferência busca explorar as tecnologias envolvidas para possibilitar o avanço da Internet, bem como, o envolvimento de aplicações que fazem uso desta tecnologia. Esse evento de conceito internacional torna-se um dos maiores fóruns para cientistas, engenheiros, e pesquisadores do mundo apresentarem seus últimos resultados, teorias, desenvolvimentos e aplicações.

5 Flexibilização e adequação do formato de saída ao conteúdo disponibilizado no sistema de Hyper-Automaton

5.1 Resumo

Este trabalho dá prosseguimento a estudos anteriores pesquisando e buscando oferecer uma forma mais flexível e inteligente de disponibilização de material de cursos para a Internet que utilizam o sistema de navegação de construções formais conhecidas como Autômatos Finitos Determinísticos com Saída. O estudo baseado nas evoluções tecnológicas da Internet sugere que o conteúdo instrucional adequadamente estruturado em XML, armazenado e utilizado no sistema *Hyper-Automaton*, incorpore os benefícios da aplicação de folhas de estilo em cascata e/ou avançadas, obedecendo a um importante critério da estilização padrão e moderna de documentos, consoantes com as necessidades e benefícios tecnológicos do XML. O trabalho a seguir, tratará das formas de disponibilizar conteúdo visualmente, descrevendo, comparando e exemplificando as mais importantes características, de acordo com uma visão aplicada e amparada bibliograficamente.

5.2 Introdução

Este artigo dá prosseguimento ao trabalho Modelagem de Cursos na *Web* Utilizando Sistemas Formais [MAJ00], [MEN99] e tem como objetivo propor alternativas com respeito à formatação da *interface* com o usuário através da utilização do XML e de suas aplicações.

Cursos na *Web* Utilizando Sistemas Formais utilizam construções formais conhecidas como Autômatos Finitos Determinísticos [MEN00]. Foi introduzido pelo grupo o conceito de "cursos são autômatos" como uma estrutura que permite fácil implementação, criação de material hipermídia independente do autômato, ao mesmo tempo que encoraja o reuso de páginas *Web* em vários cursos, os quais podem ser construídos com enfoques diferentes, diminuindo a redundância na criação de páginas [MAJ00], [MEN99], [MAJ98].

O objetivo geral do modelo *Hyper-Automaton* centra-se no estudo da aplicação do formalismo de Autômatos Finitos com Saída (Máquina de *Mealy* e Máquina de *Moore*) como um modelo estrutural para a organização de hiperdocumentos instrucionais, em especial de Cursos na *Web*. O modelo é inspirado por pesquisas clássicas na área de hiperdocumentos e recentes iniciativas na *WWW*, com especial enfoque no desenvolvimento de sistemas de hipertexto onde a base de hiperdocumentos é projetada de forma independente da estrutura de controle da aplicação hipermídia, e suporta algumas facilidades como a composição de estruturas hierárquicas, especificação de vários conjuntos de *links* sobre um mesmo corpo de hiperdocumentos e objetos separados da estrutura de navegação.

Cada autômato define um curso e consiste de um conjunto de hiperdocumentos independentes, os quais podem pertencer a outros cursos. A função de transição funciona como ligação lógica entre os hiperdocumentos e a função de saída compõe as páginas. O resultado final é a estrutura básica de páginas e *links* de hipertexto em um site na *Web*. O modelo leva a um alto grau de modularização do material instrucional, apresentando as seguintes vantagens: facilidade de reuso de páginas em diversos cursos, com eliminação da redundância; independência dos hiperdocumentos da estrutura do

autômato, cuja alteração não influi nas páginas e vice-versa; permite que qualquer usuário crie *links* de e para qualquer documento; facilidade de implementação e manutenção; interface gráfica simples e direta; elaboração de seqüências instrucionais com enfoques específicos e capazes de oferecer estudo individualizado; operações categoriais fornecem um esquema de composição de cursos que permite a construção de novos cursos sobre cursos já existentes através de procedimentos de alto nível [MAJ00], [MEN99].

O sistema *Hyper-Automaton* constitui-se de um sistema semi-automatizado para o suporte a cursos na Web (com base em uma arquitetura cliente/servidor e *interface* desenvolvida em HTML) através da aplicação de conceitos inerentes à Ciência da Computação, em especial da Teoria de Autômatos, tecnologia de Hipermídia e Teoria das Categorias, reunindo os benefícios de cada uma. Embora a ênfase utilizada para a validação do modelo seja a implementação de sistemas para o ensino a distância, os resultados obtidos se aplicam para sistemas de hipertexto como um todo.

Nosso trabalho na fase atual, está centrado nas várias possibilidades de formatação do conteúdo disponibilizado, culminando em uma adequada interface com o usuário, onde, a flexibilização da formatação do conteúdo obtido na função de saída do autômato será uma aplicação que remonte o documento nas partes desejadas e com as características mais adequadas relacionadas às necessidades do usuário e do curso em questão. O estudo proposto trará uma maior flexibilidade ao já existe, pois no estado atual, o que se apresenta é uma única e rígida possibilidade composta de fragmentos em HTML.

Por considerações que excluem explicitamente qualquer conotação comercial, os trabalhos aqui desenvolvidos foram avaliados utilizando o Internet Explorer 5.5, sendo este, o único *browser* que a partir da versão 4, suporta os recursos XML padrão.

5.3 Visão do sistema adequada a estilos

O diagrama detalhado da figura 5.1 ilustra as partes do sistema e serve para que tenhamos uma boa idéia do que estamos atualmente desenvolvendo, girando em torno da forma adequada, correta e padronizada de visualização dos documentos corretamente estruturados em XML, bem-formados ou válidos contra uma DTD [LIG99], [HAR99]. O estudo de regras para o Sistema de *Hyper-Automaton* foi alvo de abordagem em trabalhos anteriores.

Estas alterações objetivarão principalmente flexibilização da função de saída do *Hyper-Automaton* através da aplicação de múltiplos estilos para o material disponibilizado, bem como a atualização tecnológica e do sistema de acordo com os benefícios já descritos em trabalhos anteriores relacionados ao XML [MAC00], [MAC00a]. Através da utilização de *templates* XSL e/ou folhas de estilos em cascata (CSS) [LIE99], [BOS98], obteremos as formas de visualização e *layout* do referido curso on-line.

A partir deste momento este trabalho descreverá uma das mais importantes características do XML no tocante ao estudo das maneiras possíveis de visualização do conteúdo previamente estruturado obedecendo firmemente às normas de padronização estabelecidas pela *World Wide Web Consortium (W3C)*.

Conhecidas como Folhas de estilo, estas aplicações estão intimamente ligadas aos nossos objetivos, pois estabelecem formas para a manipulação de documentos, seja por simples inserção de critérios de formatação direta ou pela transformação da árvore do documento XML [ADL00].

As folhas de estilo avançadas possibilitam várias maneiras de visualização do

mesmo conteúdo, através de adequados comandos de formatação através de expressões XPath, culminando na visualização adequada em navegadores *WWW* que suportem a apresentação de arquivos XML e não-XML que normalmente compõe documentos exibidos na *Web*.

Pormenorizadas na figura abaixo, estas características indubitavelmente farão parte e serão amplamente utilizadas na nova proposta de implementação sobre o sistema de Hyper-Automaton.

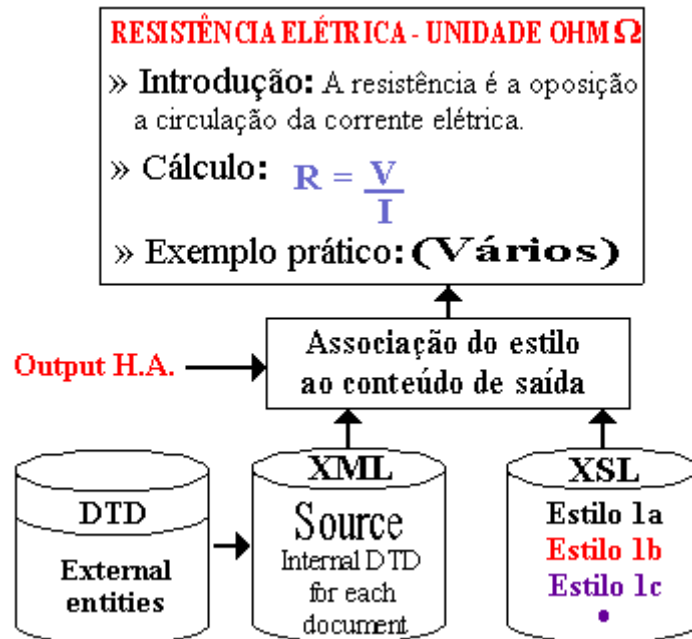


FIGURA 5.1 – Elementos gerais do sistema

5.4 Folhas de Estilos em cascata

Representam um dos novos recursos mais interessantes feitas à HTML 4.0 e Web. Foi introduzida pela W3C com o intuito de incorporar consistência a *Web*, pois promete uma grande melhora ao estado atual, de certa forma caótico [LIE99a]. As folhas de estilo permitem que os autores especifiquem os elementos de layout e projeto de todo um site da *Web*, tais como fontes, cores, recuos e o controle preciso sobre o modo como os elementos aparecem em uma página da *Web*. O mais interessante e importante é que nem browsers mal configurados ou caprichos dos usuários podem estragar a exibição de documentos da *Web* que dependem dos estilos.

Segundo bibliografia [LIE99a], folhas de estilos pertencem ao ramo da *markup* avançada, pois são complexas e difíceis.

Uma folha de estilo define as informações de projeto e *layout* dos documentos. Em geral, as folhas de estilos também especificam as fontes, cores, recuos, *kerning*, *leading*, margens e até mesmo as dimensões da página de qualquer documento que as invoque.

No mundo da editoração, as folhas de estilos são indispensáveis para fatores específicos de *layout*, pois todos os participantes de um projeto podem trabalhar em seus próprios sistemas, independentemente dos outros membros da equipe. Quando as peças de um projeto são reunidas o produto final resulta em uma aparência consistente, por que um modelo comum garante uma definição comum para o documento final.

A consistência é uma característica altamente desejável nos produtos finais tanto

nos documentos impressos quanto eletrônicos (*on-line*).

O sistema de folhas de estilo é mais um importante item do sistema proposto, pois este é um elemento indispensável do trinômio conteúdo, regras e visualização.

As folhas de estilo em cascata é uma versão da norma CSS, conhecida como CSS1, mas é tão completa que todas as mudanças futuras tenderão ser pequenas [LIE99a].

Um dos recursos fundamentais da CSS1 é sua suposição de que várias folhas de estilos relacionadas podem ser colocadas em *cascata*, o que significa que os autores podem anexar folhas de estilos preferidas aos documentos da Web, e os leitores podem associar suas próprias folhas pessoais de estilo àqueles mesmos documentos. Isso permite corrigir deficiências humanas ou tecnológicas, como por exemplo o ajuste de pontos para impressão ou limitações locais de resolução ou área de exibição.

Basicamente, a CSS contém um conjunto de regras para resolver conflitos de estilo que surgem quando se aplicam várias folhas de estilos ao mesmo documento. Como os conflitos podem surgir, algum método de resolução é essencial para fazer o conteúdo de um documento aparecer de forma adequada no vídeo de um usuário.

Para que folhas de estilos não acabem por atrapalhar definições pessoais de visualização, especialmente aquelas pessoas portadoras de deficiências visuais, depois que todas as folhas de estilos são referenciadas e suas alterações são carregadas na memória, o *browser* resolve os conflitos aplicando a definição com o maior peso e ignorando as outras definições.

Um documento que anexa somente folhas de estilo em cascata traz uma importante desvantagem quando se trata de flexibilização do formato de saída. Estas funcionam como se o documento XML original fosse “escaneado”, disponibilizando na tela do usuário exatamente como este fora elaborado. Em outras palavras, não é permitida a alteração na estrutura da árvore do documento XML original. A figura 5.2 ilustra o funcionamento de uma folha de estilo em cascata em um documento XML.

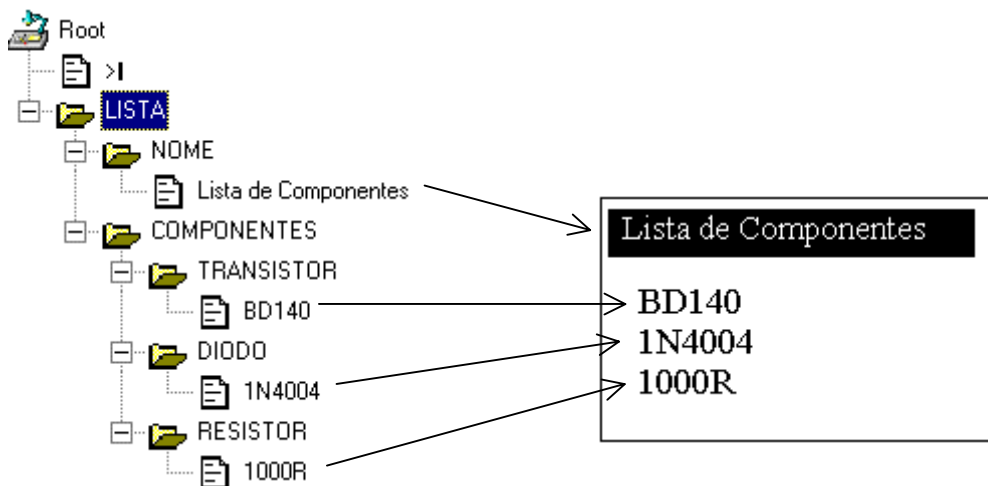


FIGURA 5.2 – Aplicação de CSS

5.5 A XSL padrão

XSL significa *eXtensible Stylesheet Language* sendo que no seu estágio inicial de desenvolvimento este nome foi escolhido para descrever sua especificação. Desde o princípio alguns fatores tenderam a ficar um pouco mais complicadas e atualmente XSL padrão é uma mistura de três padrões: XSL, XSLT e XPath. As respectivas regras destes três padrões podem ser descritos isoladamente. É possível descrever e exemplificar a

maneira de como estes três padrões funcionam juntos para executar uma tarefa de formatação de um documento XML.

O termo original XSL é agora usado de maneira mais limitada, descrevendo como especificar estilos de saída. O padrão XSL define uma aplicação do XML que deliberadamente quebra a regra das estruturas de documentos XML [BRA00]. Nesta aplicação um elemento é utilizado para descrever a aparência de seu conteúdo, ao invés, do seu significado.

A XSLT (*XSL Transformations*) é uma linguagem XML de transformação. Utilizando XSLT, é possível converter documentos XML em um outro formato de dados, incluindo muitas outras linguagens de formatação que usam controles de renderização. Esta linguagem foi parte do XSL padrão, mas agora começa ter sua própria padronização.

A XSLT padrão especifica o formato da folha de estilo do documento e inclui definições para o mapeamento de regras. Um documento XSLT é atualmente um documento XML que está em conformidade com a XSLT DTD [BRA00].

Hoje em dia, o uso mais freqüente de XSLT é orientado para converter um documento XML para outro formato de dado reconhecido pelos navegadores *Web*.

XSLT pode ser usada para vários propósitos além de formatação. Documento XML pode simplesmente ser transformado em outro documento XML de diferente estrutura de acordo com uma outra DTD. Desta maneira, o elemento contido pode ser reusado, reposicionado, sorteado, separado ou unido com outro conteúdo, e transformado de acordo com outros atributos ou valores.

Um número de interessantes benefícios partem do fato de que folhas de estilo XSLT são documentos XML. Primeiro, é possível usar uma DTD para validar a folha de estilo e o documento XML. Segundo, folhas de estilo podem ser armazenadas em um repositório, onde podem ser manipuladas e identificadas conforme versão. Terceiro, pode-se construir folhas de estilo que irão formatar outra folha de estilo para apresentação ou impressão.

Com referência a XPath, XSLT pode transformar um documento XML que está conforme uma DTD em outro documento XML conforme com uma outra DTD. De uma maneira simples, utilizando-se técnicas mostradas abaixo, tags podem ser renomeados de acordo com as definições constantes na DTD. Por exemplo, um elemento 'Paragrafo' pode ser mudado para elemento 'Para' ou ainda 'P'. Mas XSLT possui outras importantes características que permitem que conteúdos sejam movidos, copiados e especifica formatação que pode ser aplicado aos elementos quando eles aparecem um dado local na estrutura de um documento. Este tipo de análise avançada e de manipulação de dados são executadas através da utilização de uma linguagem de expressão.

A proposta original da XSL a incluía como uma linguagem, mas a XSLT padrão agora referencia um outro padrão separado chamado XPath. Este padrão define um propósito geral para interrogar as estruturas de documentos XML. É também usada como uma linguagem de consulta e para ligações de avançados esquemas de hiperlinks. Esta linguagem não é baseada em XML, mas possui uma sintaxe compacta que tem sido desenvolvida para um número variado de circunstâncias. A seguinte expressão identifica o elemento 'Paragrafo' que possui um atributo 'Tipo' o valor 'Importante' em um elemento capítulo:

```
capitulo // paragrafo [@tipo='Importante']
```

As expressões da linguagem fornecem maneiras para que habilmente possamos

interrogar estruturas XML.

Até o presente momento pode-se observar que XSLT, com a assistência de expressões XPath, oferece uma maneira ideal para transformar um documento XML para outro documento XSL, pronto para ser formatado para apresentação. O exemplo abaixo demonstra uma regra XSLT que mapeia um elemento 'paragrafo' no documento fonte para um parágrafo em negrito no formato XSL. Observe que somente quando o parágrafo está no interior de um elemento 'capitulo' e tiver um valor de atributo tipo 'importante' será aplicada a regra. Após a listagem do exemplo, a figura 5.3 busca definir o contexto destes três padrões. A figura 5.4 demonstra o processo para que documentos XML possam ser visualizados.

```
<template match="capitulo//paragrafo[@tipo='importante']">  
  <fo:block font-style='bold'>  
    <apply-templates font-style='bold'>  
  </fo:block>  
</template>
```

LISTAGEM 5.1 – Exemplo de XSLT

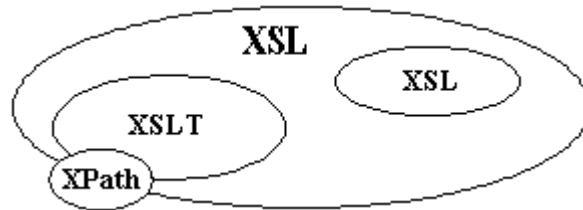


FIGURA 5.3 – Contexto dos padrões

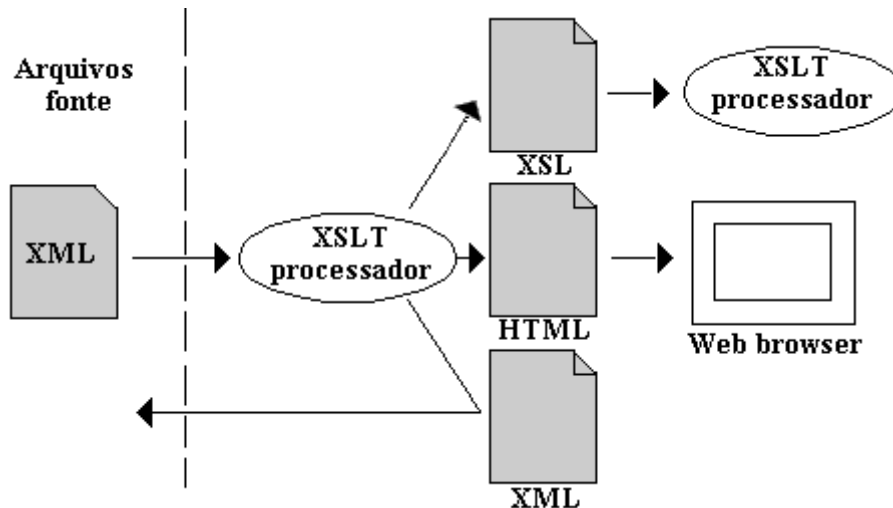


FIGURA 5.4 – Processo de visualização [BRA00]

5.6 Características da linguagem e exemplos práticos

A estrutura do documento XML fonte, exemplo abaixo, identifica um arquivo resumido de informações corretamente estruturadas de um curso *on-line*. Podemos observar quando a listagem a aplicação XSL é acessada a partir da referência feita no arquivo fonte XML. A aplicação torna-se responsável pela aplicação da folha de estilo, pois é possuidora de expressões XPath, fazendo pesquisas na árvore do arquivo fonte disponibilizando na tela do navegador os dados pela folha de estilo acessada.

Structure	Values
CURSO	
IDENTIFICACAO	
DESCURSO	
NOME	Eletronica Basica
CODIGO_CURSO	EB01
LISTA	
ALUNO	
NOME	Jorge Silva
CODIGO_ALUNO	0120dR
ENDERECO	
RUA	Rua Sete, 232
CEP	96000-100
BAIRRO	CENTRO
TELEFONE	225.5656
IDADE	28
NOTA	8,0
PAGAMENTO	
MARCO	Pago
ABRIL	Debito
MAIO	Debito
ALUNO	
ALUNO	

FIGURA 5.5 – Estrutura de um documento

Abaixo pode-se observar um estilo associado que gera um saída específica ao documento fonte na figura 5 indicado, a seguir, consta a listagem da folha de estilo XSL, com expressões XPath, que anexada ao documento fonte, permite a visualização dos dados requeridos e formatados. Esta folha de estilo é capaz de pesquisar a estrutura XML disponibilizando na tela os alunos e seus respectivos códigos cujos quais permanecem em débito no mês de abril, juntamente com a disponibilização de sua nota, indicada em vermelho se reprovado. Na mesma página, encontramos uma outra tabela cuja função é listar os alunos que moram em um determinado bairro.

Cabe-se salientar que se procurou automatizar ao máximo a formatação de saída do documento, onde a *markup* HTML inserida possui a única finalidade de proporcionar um *layout* adequado na tela para as informações de saída.

Um das grandes vantagens do XML é o poder de identificação de cada parte da informação, aspecto pelo qual vem a tornar fácil a formatação desta quando da aplicação de folhas de estilo avançadas com expressões XPath [LIG99], [HAR99], [MOU99].


Finalmente como saída para o navegador tem-se a tela mostrada abaixo:

Electronica Basica EB01 - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites History Mail Print Edit MicroPortal

Address C:\meus documentos\Mestrado\Vestres\TESTEGERAL\aluno.xml



Electronica Basica

EB01

ACOMPANHAMENTO DE ALUNOS

Nome	Codigo	Nota	Pagamentos
Cesar Gomes	8d20Az	5,5	AbriL_Debite
Jorge Silva	0120dR	5,0	AbriL_Debite
Luiz Santos Silva	2120gR	6,5	AbriL_Debite

Nome	Endereco
Emerson Gaudino	Rua Dumont, 23A / Laranjal / 96017-100
Luiz Santos Silva	Rua Nove, 552 / Laranjal / 96010-100

FIGURA 5.6 – Saída para alunos


```

<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<HTML xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<HEAD>
<TITLE>
<xsl:for-each select="CURSO/IDENTIFICACAO/DESCURSO">
<xsl:value-of select="NOME"/>
<xsl:value-of select="CODIGO_CURSO"/>
</xsl:for-each>
</TITLE>
</HEAD>
<BODY bgcolor="#FCF2AF">
<TABLE align="center">
<tr>
<td>
<IMG SRC="frags/frag10.gif" HEIGHT="110" WIDHT="107" HSPACE="20"
BORDER="1"/>
</td>
<td align="center">
<xsl:for-each select="CURSO/IDENTIFICACAO/DESCURSO">
<b><font face="Verdana, Arial, Helvetica, sans-serif">
<xsl:value-of select="NOME"/></font></b><p/>
<font face="Arial, Helvetica, sans-serif" color="#800000">
<xsl:value-of select="CODIGO_CURSO"/><br/>ACOMPANHAMENTO DE ALUNOS
</font>
</xsl:for-each>
</td>
<tr>
<td align="center">
<table border="1">
<tr>
<th>Nome</th>
<th>Codigo</th>
<th>Nota</th>
<th>Pagamentos</th>
</tr>
<xsl:for-each select="CURSO/LISTA/ALUNO" order-by="+ NOME">
<xsl:if match=".[PAGAMENTO/ABRIL='Debito']">
<tr>
<td><xsl:value-of select="NOME"/></td>
<td><xsl:value-of select="CODIGO_ALUNO"/></td>
<xsl:choose>
<xsl:when match=".[NOTA &lt; '6']">
<td><font color="red"><xsl:value-of select="NOTA"/></font></td>
</xsl:when>
<xsl:otherwise>
<td><font color="blue"><xsl:value-of select="NOTA"/></font></td>
</xsl:otherwise>
</xsl:choose>
<td><u>Abril: </u><xsl:value-of select="PAGAMENTO/ABRIL"/></td>
</tr>
</xsl:if>
</xsl:for-each>
</table>
<table border="1">
<tr>
<th>Nome</th>
<th>Endereco</th>
</tr>
<xsl:for-each select="CURSO/LISTA/ALUNO" order-by="+ NOME">
<xsl:if match=".[ENDERECO/BAIRRO='Laranjal']">

```

```

<tr>
<td><xsl:value-of select="NOME"/></td>
<td><xsl:value-of select="ENDERECO/RUA"/> /
<xsl:value-of select="ENDERECO/BAIRRO"/> /
<xsl:value-of select="ENDERECO/CEP"/>
</td>
</tr>
</xsl:if>
</xsl:for-each>
</table>
</td>
</tr>
</TABLE>
</BODY>
</HTML>
</xsl:template>
</xsl:stylesheet>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<HTML xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<HEAD>
<TITLE>
<xsl:for-each select="CURSO/IDENTIFICACAO/DESCURSO">
<xsl:value-of select="NOME"/>
<xsl:value-of select="CODIGO_CURSO"/>
</xsl:for-each>
</TITLE>
</HEAD>
<BODY bgcolor="#FCF2AF">
<TABLE align="center">
<tr>
<td>
<IMG SRC="frags/frag10.gif" HEIGHT="110" WIDHT="107" HSPACE="20"
BORDER="1"/>
</td>
<td align="center">
<xsl:for-each select="CURSO/IDENTIFICACAO/DESCURSO">
<b><font face="Verdana, Arial, Helvetica, sans-serif">
<xsl:value-of select="NOME"/></font></b><p/>
<font face="Arial, Helvetica, sans-serif" color="#800000">
<xsl:value-of select="CODIGO_CURSO"/><br/>ACOMPANHAMENTO DE ALUNOS
</font>
</xsl:for-each>
</td>
</tr>
<tr>
<td align="center">
<table border="1">
<tr>
<th>Nome</th>
<th>Codigo</th>
<th>Nota</th>
<th>Pagamentos</th>
</tr>
<xsl:for-each select="CURSO/LISTA/ALUNO" order-by="+ NOME">
<xsl:if match=".[PAGAMENTO/ABRIL='Debito']">
<tr>
<td><xsl:value-of select="NOME"/></td>
<td><xsl:value-of select="CODIGO_ALUNO"/></td>
<xsl:choose>
<xsl:when match=".[NOTA &lt; '6']">
<td><font color="red"><xsl:value-of select="NOTA"/></font></td>

```

```

</xsl:when>
<xsl:otherwise>
<td><font color="blue"><xsl:value-of select="NOTA"/></font></td>
</xsl:otherwise>
</xsl:choose>
<td><u>Abril: </u><xsl:value-of select="PAGAMENTO/ABRIL"/></td>
</tr>
</xsl:if>
</xsl:for-each>
</table>
<table border="1">
<tr>
<th>Nome</th>
<th>Endereco</th>
</tr>
<xsl:for-each select="CURSO/LISTA/ALUNO" order-by="+ NOME">
<xsl:if match=". [ENDERECO/BAIRRO='Laranjal']">
<tr>
<td><xsl:value-of select="NOME"/></td>
<td><xsl:value-of select="ENDERECO/RUA"/> /
<xsl:value-of select="ENDERECO/BAIRRO"/> /
<xsl:value-of select="ENDERECO/CEP"/>
</td>
</tr>
</xsl:if>
</xsl:for-each>
</table>
</td>
</tr>
</tr>
</TABLE>
</BODY>
</HTML>
</xsl:template>
</xsl:stylesheet>

```

LISTAGEM 5.2 – Folha de estilo XSL para *aluno.xml*

Mais um exemplo das funcionalidades da manipulação de estruturas XML válidas e bem-formadas, associadas à estilos é o fato do professor ter a possibilidade de escolher através de entradas pré-determinadas os dados adequados para que sejam exibidas na tela de um navegador, aquilo que em um dado momento seja do seu real interesse.

O próximo exemplo sugere que sobre uma base XML validada contra a uma DTD interna e externa, listada abaixo, com dados sobre exercícios dos mais variados e com os mais diversos níveis de dificuldade, possam ser exibidos aqueles que possuem grau de dificuldade baixo.

A idéia deste exemplo vem do fato do professor ao longo de sua carreira acumular exercícios de toda ordem e estes serem aplicados no momento correto durante o aprendizado dos seus alunos.

Primeiramente abaixo encontra-se listado o arquivo XML e sua DTD interna.

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="estiloexerc3.xsl"?>

<!DOCTYPE lista [
  <!ENTITY % NUCLEO_DTD SYSTEM "nucleo.dtd">
  %NUCLEO_DTD;

```

```

<!ENTITY omega "Ω">
<!ENTITY aacute "á">
<!ENTITY Eacute "É">
<!ENTITY Ecirc "Ê">
<!ENTITY atilde "ã">
<!ENTITY ccdil "ç">
<!ENTITY oacute "ó">
<!ENTITY iacute "í">
<!ENTITY CEFET "CEFET RS - Centro Federal de Educa&ccdil;&atilde;o
Tecnol&oacute;gica">
  <!ENTITY CONT "Conceitos b&aacute;sicos de eletricidade">

  <!ELEMENT lista (cabecalho, capitulo+)>
  <!ELEMENT cabecalho (instituicao, conteudo, descricao, atualizacao,
logo)>
  <!ELEMENT instituicao (#PCDATA)>
  <!ELEMENT conteudo (#PCDATA)>
  <!ELEMENT descricao (#PCDATA)>
  <!ELEMENT atualizacao (#PCDATA)>
  <!ELEMENT logo EMPTY>
  <!ATTLIST logo nome CDATA #REQUIRED>
  <!ATTLIST logo y CDATA #REQUIRED>
  <!ATTLIST logo x CDATA #REQUIRED>

  <!ELEMENT capitulo (exercicio+)>
  <!ATTLIST capitulo id CDATA #REQUIRED>

  <!ELEMENT exercicio (numero, nivel, utilizacao+, enunciado, image+,
resposta)>
  <!ELEMENT image EMPTY>
  <!ATTLIST image nome CDATA #REQUIRED>
  <!ATTLIST image y CDATA #REQUIRED>
  <!ATTLIST image x CDATA #REQUIRED>
]>

<lista>
  <cabecalho>
    <instituicao>&CEFET;</instituicao>
    <conteudo>&CONT;</conteudo>
    <descricao>Exerc&iacute;cios</descricao>
    <atualizacao>17.9.2001</atualizacao>
    <logo nome="frag10.gif" y="110mm" x="107mm" />
  </cabecalho>

  <capitulo id="1">

    <exercicio>
      <numero>1.</numero>
      <nivel>baixo</nivel>
      <utilizacao>geral</utilizacao>
      <enunciado>Calcule a resistencia total do circuito:</enunciado>
      <image nome="circ1.gif" y="60mm" x="40mm" />
      <resposta>x</resposta>
    </exercicio>

    <exercicio>
      <numero>2.</numero>
      <nivel>medio</nivel>
      <utilizacao>geral</utilizacao>
      <enunciado>Calcule a intensidade de corrente em R1: </enunciado>
      <image nome="circ2.gif" y="60mm" x="40mm" />

```

<resposta>x</resposta>
</exercicio>

<exercicio>
<numero>3.</numero>
<nivel>baixo</nivel>
<utilizacao>trabalho</utilizacao>
<enunciado>Calcule a ddp sobre R4:</enunciado>
<image nome="circl.gif" y="60mm" x="40mm" />
<resposta>x</resposta>
</exercicio>

<exercicio>
<numero>4.</numero>
<nivel>medio</nivel>
<utilizacao>geral</utilizacao>
<enunciado>Calcule a intensidade de corrente em R3:</enunciado>
<image nome="circl.gif" y="60mm" x="40mm" />
<resposta>x</resposta>
</exercicio>

<exercicio>
<numero>5.</numero>
<nivel>baixo</nivel>
<utilizacao>geral</utilizacao>
<enunciado>Calcule a resistencia total do circuito:</enunciado>
<image nome="circl.gif" y="60mm" x="40mm" />
<resposta>x</resposta>
</exercicio>

<exercicio>
<numero>6.</numero>
<nivel>medio</nivel>
<utilizacao>prova</utilizacao>
<enunciado>Calcule a potencia eletrica fornecida pela fonte:</enunciado>
<image nome="circl.gif" y="60mm" x="40mm" />
<resposta>x</resposta>
</exercicio>

<exercicio>
<numero>7.</numero>
<nivel>medio</nivel>
<utilizacao>geral</utilizacao>
<enunciado>Calcule a resistencia total do circuito:</enunciado>
<image nome="circ2.gif" y="60mm" x="40mm" />
<resposta>x</resposta>
</exercicio>

<exercicio>
<numero>8.</numero>
<nivel>alto</nivel>
<utilizacao>geral</utilizacao>
<enunciado>Calcule a resistencia total do circuito:</enunciado>
<image nome="circ2.gif" y="60mm" x="40mm" />
<resposta>x</resposta>
</exercicio>

<exercicio>
<numero>9.</numero>
<nivel>alto</nivel>

```

    <utilizacao>prova</utilizacao>
    <enunciado>Calcule a intensidade de corrente em R4</enunciado>
    <image nome="circ2.gif" y="60mm" x="40mm" />
    <resposta>x</resposta>
  </exercicio>

<exercicio>
  <numero>10.</numero>
  <nivel>alto</nivel>
  <utilizacao>trabalho</utilizacao>
  <enunciado>Calcule a potencia eletrica em R3:</enunciado>
  <image nome="circ2.gif" y="60mm" x="40mm" />
  <resposta>x</resposta>
</exercicio>

</capitulo>
</lista>

```

LISTAGEM 5.3 – Arquivo de dados (exercicios.xml)

Abaixo encontra-se listado o arquivo XSL com expressões Xpath, para que sejam feitas manipulações no arquivo de dados.

```

<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

<xsl:template match="/">
<HTML xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<HEAD>
<TITLE>
<xsl:for-each select="lista/cabecalho">
<xsl:value-of select="instituicao"/>
<xsl:value-of select="titulo"/>
</xsl:for-each>
</TITLE>
</HEAD>
<BODY bgcolor="#FBF2CF">
<table align="center">
<tr>
<td align="center">
<xsl:for-each select="lista/cabecalho" >
<b><font face="Verdana, Arial, Helvetica, sans-serif">
<xsl:value-of select="instituicao"/></font></b><p/>
<font face="Arial, Helvetica, sans-serif" color="#800000">
<xsl:value-of select="conteudo"/><br/>
<xsl:value-of select="descricao"/>
<xsl:value-of select="atualizacao"/></font>
</xsl:for-each>
</td>
<td>
<xsl:for-each select="lista/cabecalho" >
<xsl:value-of select="image"/>
<IMG SRC="frags/frag10.gif" HEIGHT="110" WIDHT="107" HSPACE="20"
BORDER="1"/>
</xsl:for-each>
</td>
</tr>
</table>
<table border="1" align="center">
<tr>
<th>Exercicio</th>

```

```

<th>Figura</th>
</tr>
<xsl:for-each select="lista/capitulo/exercicio" order-by="+ number">
<xsl:choose>
<xsl:when match=".[nivel='baixo']">
<tr>
<td><xsl:value-of select="numero"/><xsl:value-of
select="enunciado"/></td>
<td><xsl:value-of select="image/@Circ2.gif"/><IMG
SRC="frags/Circ2.gif" HEIGHT="134" WIDHT="236" HSPACE="20"
BORDER="1"/></td>
</tr>
</xsl:when>
</xsl:choose>
</xsl:for-each>
</table>
</BODY>
</HTML>
</xsl:template>
</xsl:stylesheet>

```

LISTAGEM 5.4 – Folha de estilo XSL para exercicios.xml

Como resultado do processamento do documento XML, conjuntamente com suas regras de formação corretamente descritas na DTD e o estilo padrão XSL associado, tem-se como resultado a saída abaixo:

CEFET RS - Centro Federal de Educação Tecnológica

Conceitos básicos de eletricidade
Exercícios 17.9.2001

Exercicio	Figura
1. Calcule a resistencia total do circuito:	
3. Calcule a ddp sobre R4:	
5. Calcule a resistencia total do circuito:	

FIGURA 5.7 – Saída para exercícios

5.7 Conclusões

Neste trabalho procurou-se demonstrar com exemplos práticos que serão adequadamente reformulados de acordo com nossas necessidades no decorrer da construção das nossas aplicações, o poder de formatação e pesquisa sobre estruturas XML.

Notou-se que através a utilização do XML e de suas aplicações, consegue-se flexibilizar o trato com documentos no que tange especificamente a disponibilização aos olhos do usuário de informações que este julgar relevante.

Muito ainda devemos explorar com os recursos XML / XSL / Xpath / DTD / XMLSchema para tornar a manipulação da informação estruturada cada vez mais fácil, pois nossos objetivos estão centrados em oferecer flexibilização ao sistema tanto quanto nas facilidades do trato das informações instrucionais armazenadas, quanto aos aspectos de visualização destas, bem como, na adição cada vez mais de outros recursos no sistema Hyper-Automaton, figura 5.1.

Problemas ainda persistem, como a inserção de arquivos que fazem uso de *plug-ins*, como aplicações VRML, cuja solução não parece ainda clara, também podemos citar a manipulação de imagens, pois apesar de serem reconhecidas pelo arquivo fonte XML, estas não podem ser usadas para acesso através da utilização de *value-of select*, sendo assim, a maneira tradicional do HTML ainda persiste [CAG99], [BRA00].

Embora sendo o XSL uma aplicação do XML em franco desenvolvimento, muito ainda está para ser feito no sentido de tornar mais fácil a utilização de *templates* HTML para geração de informação facilmente processada pelos atuais navegadores.

6 Implementando o novo sistema Hyper-Automaton

Este artigo foi submetido ao *International Symposium on Knowledge Management* – Simpósio Internacional da Gestão do Conhecimento, promovido pela Pontifícia Universidade Católica de Curitiba – PUC PR e Centro Internacional de Tecnologia de Software – CITS, que ocorrerá de 19 a 21 de agosto de 2002 na cidade de Curitiba.

Em sua quinta edição, o ISKM consolida-se como o evento de referência destas áreas, atraindo contribuições de alto nível dos mais diversos setores.

O artigo descreve a parte final da pesquisa de reestruturação do Sistema Hyper-Automaton que na sua parte inicial previa somente aspectos a serem estudados e implementados visando a flexibilização do sistema para uma melhora dos aspectos visualização do seu conteúdo. Na verdade, o que acabou acarretando, foi a inevitável reestruturação completa de sistema, dando-lhe grande capacidade, atualizando-o tecnologicamente e aumentando as perspectivas de trabalhos futuros para a sua própria melhoria.

Na descrição dos aspectos de implementação acabamos por abordar outros assuntos que antes não foram previstos, como: XMLSchema, Java, Java Server Pages, Servlets e objetos de ensino.

Tal trabalho fecha de maneira prática esta fase de estudo através de uma implementação, embora em fase inicial, um protótipo do novo sistema.

Até o presente momento temos a aprovação em primeira fase deste trabalho, ou seja, aceitação do resumo re-enquadrado na área temática de Tecnologia da Informação como Suporte a Gestão do Conhecimento.

6 Implementando o novo sistema Hyper-Automaton

6.1 Resumo

Este trabalho finaliza a primeira grande parte do estudo de uma reestruturação plena no Sistema Hyper-Automaton, nome dado ao gerenciador de Hipertextos desenvolvido pelo grupo de métodos formais e informática teórica da Universidade Federal do Rio Grande do Sul – Brasil. Tal implemento amplamente utilizado como processador de hipertextos instrucionais está baseado no formalismo dos autômatos finitos com saída sendo utilizado para o auxílio as aulas servindo como importante ferramenta de apoio ao ensino. Esta pesquisa remonta totalmente a máquina de estados que até então gerenciava rígidos fragmentos HTML para uma versão atual, fazendo uso da tecnologia XML e Java que são modernas e poderosas aplicações que vão de encontro aos nossos interesses onde é buscado a flexibilização total do material disponibilizado, suporte a recursos de links não convencionais, renderização perfeita de ampla gama de caracteres e fórmulas matemáticas, suporte a arquivos multimídia e facilidades futuras de edição de conteúdo, estilos e regras.

6.2 Introdução

Este artigo dá prosseguimento ao trabalho Modelagem de Cursos na Web Utilizando Sistemas Formais [MAJ00], [MEN99] e tem como objetivo propor alternativas com respeito à formatação da interface com o usuário através da utilização do XML e de suas aplicações.

Cursos na Web Utilizando Sistemas Formais, utilizam construções formais conhecidas como Autômatos Finitos Determinísticos [MEN99]. Foi introduzido pelo grupo o conceito de "cursos são autômatos" como uma estrutura que permite fácil implementação, criação de material hipermídia independente do autômato, ao mesmo tempo que encoraja o reuso de páginas Web em vários cursos, os quais podem ser construídos com enfoques diferentes, diminuindo a redundância na criação de páginas [MAJ00], [MEN99], [MAJ98].

O objetivo geral do modelo Hyper-Automaton centra-se no estudo da aplicação do formalismo de Autômatos Finitos com Saída (Máquina de Mealy e Máquina de Moore) como um modelo estrutural para a organização de hiperdocumentos instrucionais, em especial de Cursos na Web. O modelo é inspirado por pesquisas clássicas na área de hiperdocumentos e recentes iniciativas na WWW, com especial enfoque no desenvolvimento de sistemas de hipertexto onde a base de hiperdocumentos é projetada de forma independente da estrutura de controle da aplicação hipermídia, e suporta algumas facilidades como a composição de estruturas hierárquicas, especificação de vários conjuntos de links sobre um mesmo corpo de hiperdocumentos e objetos separados da estrutura de navegação.

Cada autômato define um curso e consiste de um conjunto de hiperdocumentos independentes, os quais podem pertencer a outros cursos. A função de transição funciona como ligação lógica entre os hiperdocumentos e a função de saída compõe as páginas. O resultado final é a estrutura básica de páginas e links de hipertexto em um site na Web. O modelo leva a um alto grau de modularização do material instrucional, apresentando as seguintes vantagens: facilidade de reuso de páginas em diversos cursos, com eliminação da redundância; independência dos hiperdocumentos da estrutura do autômato, cuja alteração não influi nas páginas e vice-versa; permite que qualquer usuário crie links de e para qualquer documento; facilidade de implementação e

manutenção; interface gráfica simples e direta; elaboração de seqüências instrucionais com enfoques específicos e capaz de oferecer estudo individualizado; operações categoriais fornecem um esquema de composição de cursos que permite a construção de novos cursos sobre cursos já existentes através de procedimentos de alto nível [MAJ00], [MEN99].

O Sistema Hyper-Automaton está baseado na arquitetura cliente-servidor e tem-se hoje como objetivo buscar o aprimoramento tecnológico do sistema de disponibilização do material instrucional na Web desenvolvido pelo grupo de pesquisa. Estas mudanças tornarão o sistema capaz de realizar operações mais sofisticadas e condizentes com as novas necessidades dos cursos on-line, como personalização de conteúdo, funcionalidades de suporte ao aluno, e até mesmo operações aritméticas entre cursos. Este presente estudo está focado na melhoria de duas partes do sistema Hyper-Automaton: armazenamento dos cursos e objetos de ensino e apresentação destes elementos na Web. Este trabalho põe em prática a avaliação feita em trabalhos anteriores sobre a linguagem de marcação XML, onde buscou-se conhecer e testar suas reais potencialidades, utilizar os mecanismos e suas características que possibilitam soluções mais sofisticadas no que se refere a disponibilização de objetos de ensino na Web, estruturação de documentos, entre outras funcionalidades que estão sendo cogitadas com a utilização desta nova linguagem.

Finalmente com a inclusão de outras poderosas linguagens de programação dotadas de versatilidade, recursos e compatíveis dentro dos nossos paradigmas previamente estabelecidos, postos agora em prática, acabamos por potencializar de sobremaneira as aplicações iniciais, tornando o sistema mais poderoso e com condições de importantes implementações atuais e futuras.

6.3 XML e Java no H.A.

6.3.1 Regras: DTD para XML Schema, revisão de conceitos

Como fortemente abordado nos trabalhos anteriores [MAC01a], quando se estava no princípio da pesquisa, a DTD era o único elemento consistente, fortemente explorado, suportado pelos principais browser hoje disponíveis e com farto material bibliográfico que a tornava como a principal linguagem de expressão de regras de construção de documentos XML. Durante a evolução do trabalho, mais especificamente em maio de 1999, a W3C publicou em seu site, www.w3c.org, o *W3C Schema Working Draft*. Neste período estávamos satisfeitos e seguros com o uso e poder da DTD em nossas aplicações, pois esta havia sido dominada de acordo com as nossas necessidades naquela época.

Conjuntamente com o desenvolvimento prático das aplicações, tornou-se a necessária inclusão de outros tipos de dados, atualização tecnológica do sistema com a manutenção das suas características originais anteriormente estabelecidas em [MAJ00]. Com a crescente maturidade do XML, reforçado pelo surgimento de farto material de apoio em livros ou na WWW, migramos naturalmente para o uso de XML Schema como linguagem de estabelecimento de padrões de construção de regras para o documentos instrucionais em XML no H.A.

Um *schema* é um modelo de descrição estrutural da informação, é um termo originado do mundo do banco de dados para a descrição da estrutura dos dados em tabelas relacionais. No contexto do XML, um *schema* descreve um modelo para uma determinada classe de documentos. O modelo descreve o possível arranjo dos tags e texto em um documento válido. Um *schema* deve também ser visto como um

vocabulário padrão comum aceito para uma aplicação particular que envolva troca de documentos. Tal afirmação feita pelo autor vai de encontro ao que pretendemos em um futuro bem próximo relacionado ao envio de material didático como exercícios, dicas, gabaritos de respostas, conteúdo, etc.

O XML herdou a *Document Type Definitions* (DTD's) do SGML que na verdade é um mecanismo de *schema* para o antigo SGML [COS01]. Desta maneira a modernização de transações, obrigou que um novo padrão fosse criado para oferecer significativas vantagens até então ausentes nas definições anteriores sobre estabelecimento de regras sobre documentos XML.

Significativas vantagens além daquelas anteriormente descritas que nos levaram as esta escolha foram:

- Possuir a mesma sintaxe XML;
- Permitir muitas maneiras de especificar tipos de dados;
- Suportar pelo menos 44 tipos de dados contra 10 da DTD;
- Pode ser definido múltiplos elementos com o mesmo nome, mas em diferentes contextos;
- Pode ser definido elementos substitutos, por exemplo: livro é substituído por publicação.
- Criação de outros tipos de dados;

Apesar do grande aumento da expressividade do XMLSchema, por se tratar de um *Draft* (rascunho), a sua especificação se encontra em evolução, de forma que alguns aspectos tratados no DTD ainda não foram incorporados pelo *Schema*. Uma das funcionalidades mais importantes ainda ausente é a definição de entidades. Como colocado por [MAC01a], as entidades são essenciais por permitirem que se utilize caracteres especiais fora do padrão, por permitirem a inclusão de dados não-xml (como imagens, vídeos, etc), por permitem o reuso de passagens de texto e documentos, entre outras facilidades. No caso do Hyper Automaton caracteres especiais serão utilizados tanto para acentuação quanto para símbolos matemáticos, logo o uso de entidades é essencial.

A solução é utilizar as duas formas de definição nos seus pontos fortes, já que a especificação de documentos através de DTD e XMLSchema não são auto-exclusivas, isto é, podem ser usadas conjuntamente (no caso de ambigüidades o validador escolhe de qual versão considerar as definições). Desta forma, os documentos XML utilizados no Hyper-Automaton utilizam XMLSchema para definir os tipos de dados e a estrutura dos documentos, enquanto o DTD é utilizado essencialmente para a utilização de caracteres especiais e inclusão/referência a arquivos externos.

6.3.2 A utilização do Java no H.A.

As contribuições de Java como linguagem e ambiente de programação são aplaudidas igualmente por desenvolvedores e pesquisadores. Jamais uma linguagem reuniu rapidamente tantas características favoráveis além do apoio de instituições importantes, estando prestes a se tornar um padrão para o desenvolvimento com objetos [DEI01].

Algumas destas características são [CES01], [RIC01]:

- Java oferece quase todo o potencial de C++ mas de uma forma bem uniforme, muito clara e sem idiosincrasias.

- É mais simples e mais fácil de ser compreendida que C++.
- Foi projetada com base em 10 anos de experiência de C++ aproveitando suas melhores características.
- Oferece tratamento de exceções hierárquicas, essencial para a robustez de sistemas complexos; memória dinâmica é gerenciada automaticamente, diminuindo o potencial para erros.
- Muitas características estão incluídas diretamente na linguagem/API padronizada, e não em bibliotecas ou ferramentas externas. Isso simplifica seu aprendizado e uso, além de garantir a portabilidade: gerenciamento de *threads* para programação concorrente, gráficos e interfaces gráficas, conexão em rede, facilidades cliente-servidor, polimorfismo, gerenciamento de compilação de módulos (classes).
- Programas Java podem ser executados em todas as plataformas significativas sem necessidade do código-fonte ou recompilação.
- A uniformidade dos conceitos de Java permite a integração em aplicações de conceitos modernos como componentes, invocação remota, reflexão/introspecção, validação e conectividade a bancos de dados.
- O potencial de Java se aplica em várias direções, de aplicações convencionais ou distribuídas na Internet a programas embeded.
- Programas Java podem ser interpretados ou executados
- A maioria dos conceitos em Java são mais simples, concisas e elegantes que os equivalentes em C++, facilitando o aprendizado e a manutenção, além de reduzir o risco de erros
- Ao contrário de C++, quase todas as facilidades de Java foram desenvolvidas simultaneamente e de forma integrada, compondo um todo muito mais uniforme e consistente
- Java dispensa várias características de C++ cuja complexidade nem sempre justificava eventuais benefícios. Em alguns casos alternativas mais elegantes e seguras foram introduzidas (como herança por interfaces)
- Orientação a objetos já é uma tecnologia e metodologia consagrada e estabelecida. Java é a melhor linguagem para programação OO.
- Nunca uma linguagem teve tanto esforço de padronização nem tantas instituições importantes colaborando para um mesmo objetivo

6.3.3 XML e Java juntos no H.A.

A tecnologia Java e a XML preenchem, de fato, duas funções distintas. Enquanto a tecnologia Java é uma linguagem de programação orientada para objetos, ou seja, uma plataforma, o XML é uma maneira formal de definir, armazenar e intercambiar dados estruturados. O XML não possui a complexidade necessária para manipular os dados, faz-se necessário uma linguagem de programação com a tecnologia Java permitindo de maneira efetiva a troca de informações entre sistemas diferentes. Em outras palavras Java possibilita a manipulação adequadas dos dados para que as aplicações façam sentido e ganhem poder [AVI01].

A implantação de aplicativos baseados na tecnologia Java, que aproveitam as sinergias do código reutilizável e portátil desta plataforma e dos dados reutilizáveis e portáteis da XML, ajudam a simplificação a atualização e a manutenção do sistema H.A.

6.4 Definição do objeto elementar

Por volta de 1994 o termo “Objeto de Ensino” foi introduzido na área de Tecnologia da Instrução: recursos de ensino quebrados em componentes modulares para mais tarde serem combinados por instrutores, estudantes e até mesmo por computadores em estruturas maiores dariam suporte ao ensino [WIL00].

Uma das questões discutidas ao se tratar de objetos de ensino é a noção de granularidade, mais especificamente como serão manipulados os objetos de ensino para a formação de objetos mais complexos. Atualmente pode se dividir essa discussão em duas correntes: A primeira determina o nível de granularidade de um objeto de ensino conforme o grau de composição de objetos menores para compor objetos de ensino maiores. A segunda e mais recente leva em consideração a coesão do objeto de ensino: quanto mais centrados em um único conceito, menor sua granularidade [WIL00].

Por acreditar que o reuso de material instrucional é potencializado pelo uso de objetos com maior coesão, os esforços foram voltados para a definição do objeto elementar de ensino: o menor fragmento de material instrucional que não seja definido em termos de outros objetos de ensino. Desta forma, é possível medir mais precisamente a complexidade de um curso, já que se leva em conta a mensagem de ensino e não o número de objetos que compõem essa mensagem.

Os fragmentos de HTML utilizados na versão anterior do H.A. já foram construídos segundo essa noção de coesão, mas por se tratar de arquivos HTML esses objetos eram o menor nível de abstração manipuláveis no sistema. Já que o nosso objetivo era utilizar e manipular esses mesmos objetos de ensino de forma mais inteligente e com maior poder de reuso, na nova proposta de implementação esses objetos foram analisados internamente, de tal forma que se identificasse os elementos que os constituíssem, como textos, imagens, listas, definições, exemplos, etc.

Com base nessa análise, foi elaborada uma definição em XMLSchema de objeto elementar de ensino que traduz para XML os elementos antes em marcação HTML. A definição do objeto foi especificada de tal maneira que seus componentes internos e externos (textos, figuras, arquivos externos) fossem facilmente identificados e manipulados. Abaixo segue o arquivo XMLSchema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <xsd:element name="object" type="typObj" minOccurs="0"
maxOccurs="unbounded"/>
  <xsd:complexType name="typObj" content="elementOnly">
    <xsd:sequence>
      <xsd:element name="metadata" minOccurs="0">
        <xsd:complexType content="elementOnly">
          <xsd:element name="title" type="xsd:string" minOccurs="0"
maxOccurs="1"/>
          <xsd:element name="name" type="xsd:string" minOccurs="0"
maxOccurs="1"/>
          <xsd:element name="author" type="xsd:string"
minOccurs="0" maxOccurs="1"/>
          <xsd:element name="description" type="typText"
minOccurs="0" maxOccurs="1"/>
          <xsd:element name="date-created" type="xsd:date"
minOccurs="0" maxOccurs="1"/>
          <xsd:element name="last-modified" type="xsd:date"
minOccurs="0" maxOccurs="1"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```

    <xsd:element name="content">
      <xsd:complexType content="elementOnly">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="text" type="typText" minOccurs="0"
maxOccurs="1"/>
          <xsd:element name="definition" type="typDef"
minOccurs="0" maxOccurs="1"/>
          <xsd:element name="theorem" type="typThm" minOccurs="0"
maxOccurs="1"/>
          <xsd:element name="example" type="typEx" minOccurs="0"
maxOccurs="1"/>
          <xsd:element name="observ" type="typText" minOccurs="0"
maxOccurs="1"/>
          <xsd:element name="HTML" type="anything" minOccurs="0"
maxOccurs="1"/>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="anything">
  <xsd:any minOccurs="0" maxOccurs="unbounded" namespace="##any"/>
</xsd:complexType>
<xsd:complexType name="typEx" content="elementOnly">
  <xsd:sequence>
    <xsd:element name="title" type="xsd:string" minOccurs="1"
maxOccurs="unbounded"/>
    <xsd:element name="exBody" type="typText"/>
    <xsd:element name="observ" type="typText" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="typThm" content="elementOnly">
  <xsd:sequence>
    <xsd:element name="title" type="xsd:string" minOccurs="1"
maxOccurs="unbounded"/>
    <xsd:element name="thmBody" type="typThmBody"/>
    <xsd:element name="observ" type="typText" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="typDef" content="elementOnly">
  <xsd:sequence>
    <xsd:element name="title" type="xsd:string" minOccurs="1"
maxOccurs="unbounded"/>
    <xsd:element name="defBody" type="typDefBody"/>
    <xsd:element name="observ" type="typText" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="typText" content="mixed">
  <xsd:choice>
    <xsd:element name="bold" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="term" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="image" type="typImage" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="list" type="typList" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:any minOccurs="0"/>
  </xsd:choice>

```

```

        </xsd:choice>
    </xsd:complexType>
    <xsd:complexType name="typThmBody" content="elementOnly">
        <xsd:choice minOccurs="1" maxOccurs="unbounded">
            <xsd:element name="statement" type="typText" minOccurs="0"
maxOccurs="unbounded"/>
            <xsd:element name="proof" type="typText" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:choice>
    </xsd:complexType>
    <xsd:complexType name="typDefBody" content="elementOnly">
        <xsd:choice minOccurs="1" maxOccurs="unbounded">
            <xsd:element name="text" type="typText" minOccurs="0"
maxOccurs="unbounded"/>
            <xsd:element name="comp" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:choice>
    </xsd:complexType>
    <xsd:complexType name="typImage" content="empty">
        <xsd:attribute name="path" type="xsd:string"/>
        <xsd:attribute name="file" type="xsd:string"/>
        <xsd:attribute name="caption" type="xsd:string"/>
        <xsd:attribute name="width" type="xsd:number"/>
        <xsd:attribute name="height" type="xsd:number"/>
    </xsd:complexType>
    <xsd:complexType name="typList" content="elementOnly">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="item" type="typText" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:choice>
    </xsd:complexType>
</xsd:schema>

```

LISTAGEM 6.1 – XMLSchema do objeto elementar

Este é o primeiro esboço da definição do objeto atômico de ensino, e de maneira alguma deve ser considerado como a versão final. A estruturação através de XML permite que se faça inclusões e modificações na definição de forma ordenada e controlada. Além disso é possível a existência de mais de uma versão de definição: cada objeto indicaria qual a versão que deveria ser usada na sua validação, por exemplo.

Note que entre os elementos existe o <HTML>, para ser usado pelos cursos legados e aqueles que não querem utilizar as funcionalidades avançadas possíveis com a boa estruturação em XML.

O objeto de ensino contendo a definição formal da máquina de Mealy é reproduzido abaixo:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Object SYSTEM
"http://localhost:8080/examples/jsp/hypno/config/obj_entities.dtd">
<object xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="
http://localhost:8080/examples/jsp/hypno/config/obj-ens-schema-
new.xsd">
    <metadata>
        <title>Máquina de Mealy</title>
    </metadata>
    <content>
        <definition>
            <title>Máquina de Mealy</title>

```



```

    <defBody>
      <text>Uma Máquina de Mealy  $M = (Q, \Sigma, \delta, q_0, F, \Delta)$ 
      Finito Determinístico com as seguintes associadas:
      transições. É representada por uma 6-tupla:
       $M = (Q, \Sigma, \delta, q_0, F, \Delta)$ 
      <list>
        <item> $\Sigma$ : alfabeto de símbolos de
        entrada;</item>
        <item> $Q$  conjunto de estados possíveis do
        autômato o qual é finito;</item>
        <item> $\delta$ : função de transição:
         $Q \times \Sigma \rightarrow Q \times \Delta$ ; a qual é uma
        função parcial;</item>
        <item> $q_0$  estado inicial do autômato tal que  $q_0 \in Q$ 
        é elemento de  $Q$ ;</item>
        <item> $F$  conjunto de estados finais tal que  $F \subseteq Q$ 
        é contido em  $Q$ ;</item>
        <item> $\Delta$ : alfabeto de símbolos de
        entrada</item>
      </list>
    </text>
  </defBody>
  <observ>
    Protanto, as componentes  $\Sigma$ ,  $Q$ ,  $q_0$  e  $F$  como no
    Autômato Finito Determinístico.
  </observ>
</definition>
</content>
</object>

```

LISTAGEM 6.2 – Objeto de ensino da máquina de Mealy

A idéia principal do sistema do Hyper-Automaton é que estes objetos de ensino sejam armazenados em um ou mais repositórios públicos, onde os acadêmicos se responsabilizassem pela manutenção e integridade dos arquivos de ensino considerados públicos. Os metadados armazenados com as informações auxiliam no controle e na busca de objetos de ensino nos repositórios.

6.5 Armazenamento, recuperação e dinamicidade

6.5.1 XML e banco de dados relacional no H.A.

Uma das principais características deste sistema é a diferença que é feita entre estrutura do curso e objetos de ensino. O autômato que define um curso é uma estrutura formal pré-estabelecida e neste, como anteriormente definido em [MAJ00], estabelece toda a estrutura navegacional deste. Quanto aos objetos elementares de ensino que fazem parte de um curso, carregam uma dinâmica considerável, além de possuírem seu próprio esquema estrutural os objetos elementares sofrem intensas alterações estruturais atualmente manuais, podendo ser renomeados, bipartidos, reestruturados e sofrerem alterações no seu esquema.

Conforme explicado no parágrafo acima, no desenvolvimento de nossas aplicações, estamos utilizando o XML para o armazenamento de dados instrucionais e o banco de dados para armazenar a estrutura do curso.

A significativa vantagem do armazenamento de dados em XML é a facilidade para a edição de dados em um simples editor de textos, mais simples do que em uma

complexa ferramenta de banco de dados. Arquivos XML são mais apropriados de serem trocados e carregados por clientes, tornando-se mais fácil à transmissão de dados a um site através da utilização de FTP [CHA00].

A maior vantagem abstrata do XML é que, sendo este um formato hierárquico do que um relacional, ele pode ser usado de uma maneira muito mais direta para projetar estruturas de objetos de ensino de acordo com as nossas necessidades. Não é necessário usar um editor de relacionamento de entidades para normalizar seu esquema. Se existe um elemento que contém um outro elemento, pode-se representá-lo diretamente no formato.

De acordo com os nossos propósitos de manipulação de informações estruturadas em XML para fins instrucionais on-line, duas das quais abordados em trabalhos anteriores, são utilizadas, três técnicas para acessar dados a partir de documentos XML, que são:

- Xpath – É usado o processador XPath para localizar os elementos no arquivo XML através da especificação do caminho e condições;
- XSL – É usado o processador XSL para transformar XML para HTML;
- Java Server Pages (JSP) – São elaboradas classes que utilizam técnicas determinadas para carregar dados.

6.5.2 A utilização de Java Server Pages

A ligação do aluno (cliente) com servidor (cursos e fragmentos), é realizado através de uma implementação Java denominado por JSP, similar em funcionamento ao ASP da Microsoft, aplicação na qual oferece todo o dinamismo que torna possível a implementação do projeto em questão.

A utilização do JSP está intimamente ligada ao fato deste ser programado em Java, manipular com facilidade documentos XML, além de ser disponível abertamente no servidor de páginas Java chamado TomCat-Jakarta.

A tecnologia JSP é uma das mais poderosas, fáceis de usar, por trata-se de uma ferramenta fundamental no desenvolvimento de aplicações Web. O JSP combina HTML e XML com Java Servlets (*server application extension*) e a tecnologia de JavaBeans para criar ambientes altamente produtivos para o desenvolvimento e uso confiáveis, interativas, de alta performance, independentes de plataforma [BER00].

Um servlet trabalha da seguinte maneira:

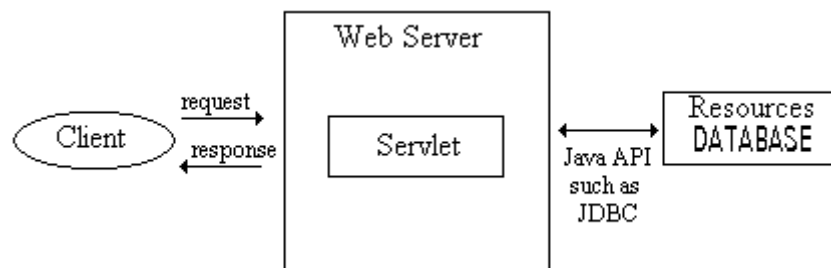


FIGURA 6.1 – Funcionamento de uma requisição JSP

A página JSP é traduzida dentro de uma Java Servlet e executada no servidor. As declarações JSP no interior de uma página JSP torna-se parte do servlet gerado a partir da página JSP. O servlet resultante é executado no servidor, tornando-o invisível ao

usuário.

O fluxo básico é o seguinte:

1. O cliente envia uma requisição para o servidor;
2. O servidor instancia o servlet e cria uma concorrência (*thread*) para o processo. O servlet é carregado na primeira requisição, ele permanece carregado até o encerramento da seção.
3. O servidor envia a informação requisitada para o servlet;
4. O servlet constrói uma resposta e passa para o servidor;
5. O servidor envia a resposta de volta ao cliente.

O servlet constrói dinamicamente a resposta a partir da requisição do cliente, adiciona dados a partir de outras fontes caso necessário.

Servlets são invocados no sistema H.A. a partir de uma explícita referência na URL de uma aplicação Web, que indica o ponto de localização (diretório) do servlet, estes podem estar localizados em qualquer diretório na máquina onde a aplicação está sendo executada de maneira que os arquivos de classes possam ser encontrados e facilmente manipulados por ftp.

6.6 Sistema de cursos H.A.

O Sistema Hyper-Automaton é um sistema de armazenamento e disponibilização de material instrucional na Web, acessível através de uma URL, onde o conteúdo apresentado é separado de sua estrutura de navegação [MAJ00]. Como discutido anteriormente, esta definição de quais páginas mostrar e que links disponibilizar em quais estados e em quais cursos estão definidos no banco de dados. Os objetos de ensino propriamente ditos podem estar em qualquer diretório do servidor (de preferência em um mesmo diretório por motivos de segurança) ou mesmo em qualquer endereço acessível pela Internet, em outros servidores, repositório de dados, etc.

O Sistema de Hyper-Automaton é responsável por processar a máquina de estados através de consultas ao banco de dados e devolver para o browser um arquivo (HTML, PDF ou mesmo WML) obtido através da aplicação de arquivos de transformação XSL a um arquivo XML criado a partir da anexação de objetos de ensino a sua estrutura.

A nova versão do Sistema Hyper-Automaton foi implementada conforme especificado em [MAJ00] na forma de uma classe Java para a sua utilização no Servidor de Páginas Java chamado TomCat-Jakarta.

Para ser utilizada na Internet, essa classe é instanciada em uma página JSP. Essa página JSP é armazenada no servidor WEB, de modo que para cada pessoa que acessa a página uma instância distinta do Hyper-Automaton é criada na memória do Servidor e associada àquela sessão.

Como qualquer máquina de estados, o seu funcionamento ocorre mediante parâmetros de entrada. No caso do Hyper-Automaton esses parâmetros são três: um curso, um estado e uma palavra de entrada. No exemplo abaixo o curso acessado é aquele de identificação 1, no estado 3 e com a palavra de entrada identificada pelo número inteiro 7.

<http://teia/webapps/hypno/hypno.jsp?course=1&state=3&word=7>

Normalmente os cursos disponíveis terão páginas iniciais independentes do Hyper-

Automaton, com apontadores para as várias seções de seus cursos, servindo de referência rápida aos estudantes. Esses pequenos portais de cursos são importantes já que não faz sentido que os usuários tenham que decorar longas URLs, cheias de variáveis e parâmetros.

Uma vez instanciado um objeto Hyper-Automaton, a página JSP repassa os parâmetros (*course*, *state* e *word*) para o objeto através de tags específicos. O parâmetro *course* é sempre obrigatório, já que é a partir do seu valor que se define a função de transição a ser aplicada sobre as entradas. Caso não se forneça algum valor para o curso, o Hyper-Automaton pára por indefinição.

Após fornecer os parâmetros a página JSP faz a chamada do método `showPage()` do objeto instanciado. Esse método é o processamento da máquina de estados propriamente dita, podendo ser dividido em quatro etapas:

1ª Etapa: Iniciar parâmetros , aplicar a função de transição e descobrir a identificação da palavra de saída;

2ª Etapa: Anexar ao documento XML de saída os objetos de ensino;

3ª Etapa: Anexar ao documento XML de saída informações sobre a navegação do curso, como links, referências cruzadas, etc.,

4ª Etapa: Transformar e apresentar o documento XML de saída na forma de apresentação, seja HTML, WAP, PDF, etc.;

1ª Etapa: Iniciação e Função de Transição

As primeiras linhas deste método fazem uma pequena verificação dos parâmetros vindos da página JSP. Caso seja a primeira vez que este método é chamado depois da instanciação, ou se o curso atual mudou desde a última vez que foi chamada, então algumas informações pertinentes ao curso são coletadas do banco de dados e armazenadas em propriedades do objeto, como nome do Curso, Tipo do Curso (Moore, Mealy), estado inicial do curso, entre outras. Além do curso geralmente é passado também como parâmetro um estado e uma palavra de entrada. Caso o estado não seja fornecido a máquina vai automaticamente para o estado inicial e continua o processamento

Uma vez definido o estado atual do Hyper-Automaton, um documento virtual XML de saída é criado: o documento *Page*. Atualmente ele possui os seguintes elementos: `<metadata>`, `<content-set>` e `<input-words>`. Assim uma página é composta de uma série de metadados (nome do curso, autor do curso, etc), um multi-conjunto ordenado de objetos de ensino (inicialmente vazio) e um conjunto de palavras de entrada (inicialmente vazio -transformado em um conjunto de links no final do processamento).

Vejamos um exemplo:

```
<page>
  <metadata>
    <course>
      <name>Linguagens Formais & Autômatos</name>
      <author>Paulo Blauth Menezes</author>
    </course>
  </metadata>
  <content-set/>
  <input-words/>
</page>
```

Após a criação deste do documento *Page*, é executada uma consulta no banco de dados para descobrir o novo estado da máquina conforme o estado atual e a palavra de entrada fornecidos. A consulta executada é mostrada abaixo:

```
"SELECT T.TRANS_DEST_ID, T.TRANS_SAIDA_ID
      FROM ALFA_ENTRADA AE, TRANSICAO T
      WHERE AE.ALFAENTR_ID = T.ALFAENTR_ID AND T.CURSO_ID =
this.course
      AND T.TRANS_ORIGEM_ID = this.state AND T.ALFAENTR_ID =
this.inputWord"
```

LISTAGEM 6.4 - Consulta ao banco de dados

O novo estado atual é o valor da coluna *T.TRANS_DEST_ID* do resultado da consulta. Se o curso for Mealy, a coluna *TRANS_SAIDA_ID* terá como valor o identificador da palavra de saída definida para a transição.

2ª Etapa: Palavra de Saída

Nesta etapa do método *showPage()* descobriremos quais os objetos de ensino que fazem parte da palavra de saída do autômato do curso através de uma consulta de Banco de Dados. Esses objetos armazenados em documentos XML serão lidos e seus conteúdos anexados ao elemento *<content-set>* do documento *Page*, apresentado na etapa anterior como o multi-conjunto ordenado de objetos de ensino.

Caso o Curso em questão seja de Mealy, a identificação da palavra de saída é utilizada na consulta ao Banco de Dados que relaciona (os caminhos para) os objetos de ensino. Caso seja Moore, a identificação da palavra de saída corresponde ao campo *SAIDA_ID* na tabela *ESTADO*, por isso ela é acrescentada na consulta correspondente.

No resultado de ambas as consultas, cada linha representa um objeto de ensino, onde a coluna *OBJETO_DIR* corresponde ao path do arquivo, e *OBJETO_ARQ* o nome do arquivo XML do objeto de ensino propriamente dito. Abaixo seguem as consultas:

Mealy:

```
"SELECT O.OBJETO_DIR, O.OBJETO_ARQ
      FROM OBJETO O, SAIDA_OBJETO SO
      WHERE O.OBJETO_ID = SO.OBJETO_ID AND SO.SAIDA_ID = this.outputId
      ORDER BY SO.SDAOBJ_ORDEM"
```

Moore:

```
"SELECT O.OBJETO_DIR, O.OBJETO_ARQ
      FROM OBJETO O, SAIDA_OBJETO SO, ESTADO E
      WHERE O.OBJETO_ID = SO.OBJETO_ID AND SO.SAIDA_ID = E.SAIDA_ID
      AND E.ESTADO_ID = this.state e E.CURSO_ID = this.course
      ORDER BY SO.SDAOBJ_ORDEM"
```

LISTAGEM 6.5 - Consulta ao banco de dados da palavra de saída

Após a consulta, cada objeto é lido, transformado em uma árvore DOM e seu elemento raiz é anexado ao elemento *<content-set>* do documento *Page*. Para uma saída que possuiu o objeto de ensino apresentado anteriormente como palavra de saída, obtém-se o seguinte resultado:

```

<page>
  <metadata>
    ...
  </metadata>
  <content-set>
    <object xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Arquivos de programas\Apache Tomcat
4.0\webapps\examples\jsp\hypno\config\obj-ens-schema-new.xsd">

      <metadata>
        <title>Autômato Finito com Saída</title>
      </metadata>
      <content>
        <text>
          O conceito básico de Autômato Finito possui
aplicações restritas, pois a informação de saída é limitada e
binária aceita/rejeita. Sem alterar a classe de linguagens
reconhecidas, é possível estender a
definição de Autômato Finito incluindo a
geração de uma palavra de saída. As saídas
podem ser associadas às transições
(Máquina de Mealy) ou aos estados (Máquina de Moore). Em
ambas as máquinas, a saída não pode ser lida,
ou seja, não pode ser usada como memória auxiliar, e
segue:
          <list>
            <item>é definida sobre um alfabeto especial,
denominado Alfabeto de Saída (pode ser igual ao alfabeto de
entrada);</item>
            <item>a saída é armazenada em uma fita
independente da de entrada;</item>
            <item>a cabeça da fita de saída move uma
célula para direita a cada símbolo gravado;</item>
            <item>o resultado do processamento do Autômato Finito
é o seu estado final (condição de aceita/rejeita)
e a informação contida na fita de saída.</item>
          </list>
          As Máquinas de Mealy e Moore são
modificadas sobre o Autômato Finito
Determinístico. A extensão da definição
prevendo as facilidades de Não-Determinismo e Movimentos
Vazios, é sugerida como exercício.
        </text>
      </content>
    </object>
  </content-set>
</input-words/>
</page>

```

LISTAGEM 6.6 – Resultado da consulta

3ª Etapa: Palavras de entrada

Mais uma consulta é feita no Banco de Dados para se descobrir as palavras de entrada que definem transições para outros estados a partir do estado atual. As identificações e as palavras em si são anexadas ao documento *Page*, mais precisamente ao elemento `<input-words>`. São os elementos `<input>` contidos em `<input-words>` que serão transformados em elementos `<A>` no caso de HTML, por exemplo. Abaixo segue

a consulta SQL para busca das palavras de entrada:

```
"SELECT AE.ALFAENTR_PALAVRA, AE.ALFAENTR_ID
FROM ALFA_ENTRADA AE, TRANSICAO T
WHERE AE.ALFAENTR_ID = T.ALFAENTR_ID
AND T.TRANS_ORIGEM_ID = this.state AND T.CURSO_ID =
this.course
ORDER BY AE.ALFAENTR_ORDEM"
```

LISTAGEM 6.7 – Consulta ao banco de dados da palavra de entrada

No exemplo abaixo são definidas duas transições a partir do estado atual 2: uma para a próxima página, outra para o próximo capítulo.

```
<page>
  <metadata>
    ...
  </metadata>
  <content-set>
    ...
  </content-set>
  <input-words>
    <input>
      <word id="3">Anterior</word>
      <currState>2</currState>
      <currCourse>1</currCourse>
    </input>
    <input>
      <word id="1">Pr&ocirc;ximo</word>
      <currState>2</currState>
      <currCourse>1</currCourse>
    </input>
  </input-words>
</page>
```

LISTAGEM 6.8 - Transições de estado

O documento *Page* está completo para ser transformado e formatado para apresentação.

4ª Etapa: XML transformado

A última etapa do método é que diferencia a nova versão em XML da implementação em Perl: uma vez formado o documento *Page*, ao invés de simplesmente apresentar os objetos de ensino concatenados na saída na forma de HTML, um arquivo de transformação particular a cada curso é aplicado ao documento *Page* através de um objeto transformador (*Transformator*), lhe dando forma, sabor e personalização na sua apresentação.

Cada curso tem o seu próprio arquivo de transformação XSL que define a estrutura da página de saída: cabeçalhos, menus, cores, layout, etc. O responsável pelo curso poderá escolher entre vários layouts padrões para seu curso e ainda terá a opção de desenvolver o seu próprio layout. Essa personalização não era possível na versão anterior pois a formatação e diagramação eram inerentes aos fragmentos HTML utilizados.

Um arquivo de transformação padrão foi elaborado e será amplamente divulgado

para auxiliar e incentivar a criação de novos arquivos de transformações para os objetos de ensino utilizados no Sistema Hyper-Automaton. Deste modo qualquer pessoa que domine a linguagem de Transformação XSL poderá definir comportamentos próprios para os elementos que compõem o documento *Page*. Para aqueles que não conhecem XSL, mas gostariam de algum nível de personalização, serão oferecidos alguns Templates padrões.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" version="1.0" encoding="iso-8859-1"
indent="yes"/>

  <xsl:template match="*" />

  <xsl:template match="object">
    <xsl:apply-templates select="metadada|content" />
  </xsl:template>

  <xsl:template match="content">
    <xsl:apply-templates select="definition|theorem|text|example" />
  </xsl:template>

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

  <xsl:template match="title" mode="definition">
    <xsl:apply-templates />
  </xsl:template>

  <xsl:template match="text|statement|observ|exBody">
    <xsl:apply-templates />
  </xsl:template>

  <xsl:template match="example">
    <p>Example: <xsl:apply-templates select="title"
mode="example" /></p>
    <p><xsl:apply-templates select="exBody" /></p>
  </xsl:template>

  <xsl:template match="definition">
    <p>Definição: <xsl:apply-templates select="title"
mode="definition" /></p>
    <p><xsl:apply-templates select="defBody" /></p>
    <p align="left">[]</p>
    <p><xsl:apply-templates select="observ" /></p>
  </xsl:template>

  <xsl:template match="image">
    <p><xsl:element name="img">
      <xsl:attribute name="src">
        <xsl:value-of select="@path"/><xsl:value-of select="@file"/>
      </xsl:attribute>
      <xsl:attribute name="width">
        <xsl:value-of select="@width"/>
      </xsl:attribute>
      <xsl:attribute name="height">
        <xsl:value-of select="@height"/>
      </xsl:attribute>
    </p>
  </xsl:template>
</xsl:stylesheet>
```



```

    </xsl:element></p>
    <xsl:if test="not (@caption='') ">
      <p>Figura: <xsl:value-of select="@caption"/></p>
    </xsl:if>
  </xsl:template>

  <xsl:template match="theorem">
    <p>Teorema:<xsl:apply-templates select="title"/></p>
    <p><xsl:apply-templates select="thmBody"/></p>
    <p><xsl:apply-templates select="observ"/></p>
  </xsl:template>

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

  <xsl:template match="thmBody">
    <p><xsl:apply-templates select="statement"/></p>
    <p>Prova:<br/><br/><xsl:apply-templates select="proof"/></p>
  </xsl:template>

  <xsl:template match="list">
    <ul>
      <xsl:for-each select="item">
        <li><xsl:apply-templates/></li>
      </xsl:for-each>
    </ul>
  </xsl:template>

  <xsl:template match="bold">
    <b><xsl:apply-templates/></b>
  </xsl:template>
</xsl:stylesheet>

```

LISTAGEM 6.9 - Templates padrão

Na implementação atual, o documento padrão de transformação deve ser explicitamente importado no arquivo de transformação (XSL) específico de cada curso. Desta forma o objeto transformador só irá considerar as definições do documento padrão de transformação se este não houver uma definição para o mesmo elemento no documento específico de transformação do curso.

A codificação a seguir mostra um trecho do arquivo de Transformação XSL de um determinado curso. Em particular, é definido onde ficarão os links de navegação e onde estará o conteúdo do curso propriamente dito:

```

<xsl:template match="page">
  <HEAD>
    <TITLE><xsl:value-of select="name"/></TITLE>
  </HEAD>
  <BODY>
    <TABLE WIDTH="100%" HEIGHT="100%" BORDER="1" STYLE="border-width:
1px 1px 1px 1px">
      <TR>
        <TD WIDTH="150" VALIGN="TOP">
          <xsl:apply-templates select="input-words" mode="left"/>
        </TD>
        <TD VALIGN="TOP">
          <xsl:apply-templates select="content-set"/>

```

```

        <xsl:apply-templates select="input-words"/>
    </TD>
</TR>
</TABLE>
</BODY>
</xsl:template>

```

LISTAGEM 6.10 – Transformação XSL para curso

O resultado dessa aplicação é a página HTML que é mostrada no browser do internauta (fig.6.2).

Um outro curso pode utilizar os mesmos objetos de ensino mas como um arquivo de transformação de XSL diferente. A saída HTML, figura 6.3, foi criada a partir do mesmo objeto de ensino sendo utilizado por outro curso. Esse por sua vez utiliza um arquivo XSL com maiores detalhes de estilo e com outro formato de diagramação. Note que além do link Próximo, para este curso-exemplo a seguir existe também um link para a página anterior.

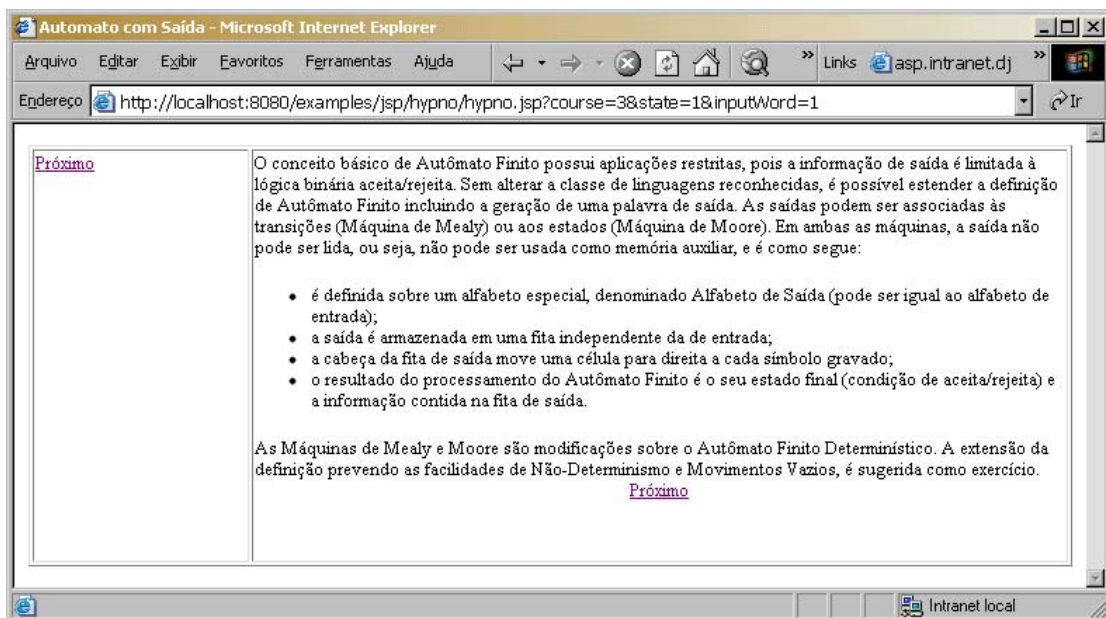


FIGURA 6.2 – Saída no *browser* para uma determinada XSL

Na versão atual do sistema, toda a transformação XML e XSL é feita no servidor. Isso significa que o servidor deve retornar ao usuário as informações já em formato html. Hoje em dia alguns browsers já suportam XML e são capazes de usar arquivos XSL para transformações. Isso significa que podemos aliviar a carga do servidor mandando para o cliente o documento page e o xsl gerado, deixando o processamento de transformação para a máquina do cliente caso o browser suporte XML – como Internet Explorer 5.5 e 6.0. Deve ser levado em consideração que desta forma o usuário tem como enxergar o objeto page através da função ver código fonte do browser. Caso algumas informações sejam confidenciais (como em um teste eletrônico), essa opção poderia causar transtornos.

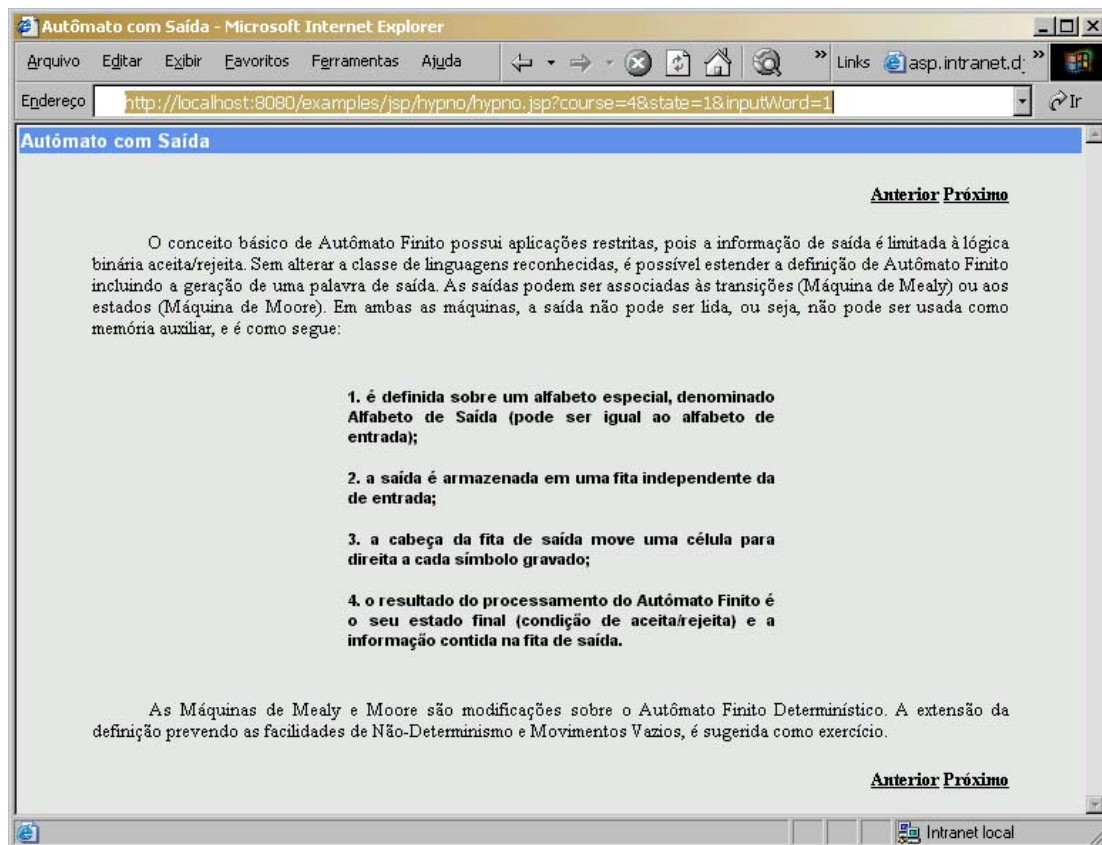


FIGURA 6.3 – Saída no browser para outra determinada XSL

6.7 Comparação com o sistema anterior

6.7.1 Organização de arquivos

Na versão anterior cada curso tinha seu próprio diretório onde ficavam os seus três arquivos de definição: curso.fl, alfabeto.fl e estados.fl. Além disso todos os fragmentos HTML disponíveis eram armazenados em apenas um diretório do servidor. Isto ocasionava um inconveniente: a listagem dos objetos disponíveis para inclusão no curso era muito longa e sem qualquer tipo de ordenação, tornando muito exaustiva a procura dos arquivos desejados. Na nova versão os fragmentos poderão ficar em qualquer repositório acessível pela web. Não existem mais os arquivos correspondentes aos cursos, já que agora são armazenados em banco de dados.

6.7.2 Manutenção facilitada: uma máquina de estados que incorpora as duas funcionalidades

Enquanto na versão em Perl o acesso a máquina de Moore e Mealy eram feitos através de dois sistemas diferentes, na nova versão em JSP a funcionalidade das duas máquinas foram incorporadas em um só sistema. Assim, a manutenção se torna muito mais fácil e o acesso ao sistema se torna único, facilitando o acesso aos cursos.

6.7.3 Acentuação e símbolos

No exemplo de curso utilizado no sistema anterior usavam-se muitos caracteres

especiais, como Omega, União, Sigma, etc. Por se tratar de um livro eletrônico, a preocupação com a compatibilidade entre os diversos sistemas operacionais levou a utilização de pequenas imagens Gif para a representação dos símbolos [FED99].

Embora tenha solucionado parcialmente o problema, o uso de Gifs resultou num engessamento, pois eles sempre tinham a mesma cor e o mesmo tamanho: existia uma figura para o símbolo sigma, e outra para o sigma sublinhado, por exemplo. Com XML é possível especificar o uso de uma codificação e o uso de entidades: palavras começadas com ‘&’ e terminadas com ‘;’, como em Σ, que representa “ Σ ”. Na nova versão temos em nossas mãos todos os caracteres codificados universalmente e com a possibilidade de sublinhado, caixa alta e toda as cores que forem necessárias.

6.7.4 Reuso inteligente

O primeiro sistema do Hyper-Automaton já utilizava o conceito de reuso através da disponibilização de material instrucional na forma de fragmentos de HTML. Os cursos eram criados através da definição dos fragmentos utilizados em cada estado do curso. O material instrucional era apresentado com o mesmo layout em todos os cursos. Através do uso de XML e XSL, o conteúdo do material de ensino foi separado da sua forma de apresentação. Isso significa que um mesmo objeto de ensino é apresentado conforme a diagramação do curso que o utiliza.

6.7.5 Expansibilidade

Como a criação da página de saída (normalmente HTML) depende exclusivamente dos arquivos XML e XSL sendo aplicados, e sendo possível estender a definição do objeto de ensino elementar para suportar mais elementos quando for necessário, é possível que se invente novas funcionalidades que darão personalização e suporte ao conteúdo sendo mostrado. Graças a estruturação dos documentos em XML, é infundável as funções que podem ser adicionadas ao sistema do Hyper-Automaton. Basta adicionar novas regras de transformação no XSL do curso em questão que as novas funcionalidades estarão disponíveis nas páginas do curso. Caso a novidade seja bastante útil e inovadora, é possível que se generalize a nova funcionalidade incluindo a transformação correspondente no XSL padrão. Na grande maioria dos casos não será preciso modificar a aplicação: estendemos o documento XLS de transformação e a própria aplicação se torna mais poderosa. Na versão anterior seria praticamente impossível implementar novas facilidades sem ter que rever a estrutura das implementações da máquina de moore e mealy.

6.8 Conclusões e perspectivas futuras

6.8.1 Administrativo Web para criação de cursos

Os cursos e documentos desenvolvidos durante a implementação do sistema foram criados editando-se arquivos em editores de texto e criando-se entradas no banco de dados diretamente. Uma vez que o sistema está bem especificado, seja pelo ER, seja pelos XMLSchema, já é possível se pensar em um sistema *front-end* de criação de cursos. Neste sistema (que provavelmente será implementado usando-se JSP) proverá ao professor uma interface amigável de estruturação de conteúdo.

6.8.2 Funções avançadas: índice remissivo, busca, etc.

Todas as funcionalidades que serão acrescentadas ao sistema serão pela adição de elementos específicos no objeto de ensino e no próprio objeto página. Assim como o glossário, que utiliza o elemento <term>, a criação de índices de palavras relacionadas será facilmente implementada através da definição de novas regras de aplicação no XSL padrão, outras funcionalidades serão estendidas. A vantagem é que todos os cursos que já utilizarem esses elementos terão as novas facilidades incorporadas automaticamente.

6.8.3 Software para a criação de objetos

Outro sistema interessante é o aquele que irá auxiliar o professor a criar seus próprios objetos de ensino, através de *wizards*, templates e até mesmo a partir do material didático em formato digital já utilizado pelo professor. Este sistema fará com que se aproveite ao máximo as funções avançadas implementadas no Sistema do Hyper-Automaton.

6.8.4 Login e conteúdo sensível a usuário

Existem casos em que o professor só deseja que seu material fique visível somente para seus alunos. Para tanto é preciso que se crie um mecanismo de autenticação de usuário e senha. Existem cursos onde o professor deseja que seja mostrado em etapas: mostrar links para o capítulo 2 somente após o aluno ter visitado certas páginas do capítulo 1. Outro caso seria o acesso a liberação de acesso a somente partes do curso.

6.8.5 Log de utilização

É interessante para os criadores dos cursos saberem como os seus cursos estão sendo utilizados. Através do uso de *logs*, é possível identificar pontos entradas, ponto de saídas, páginas mais e menos visitas pelos alunos. Desta forma o professor tem a oportunidades de tomar decisões que possam melhorar a eficiência do seu curso: como melhorar explicações, incluir mais exemplos, eliminar páginas redundantes, etc.

6.8.6 Cursos sensíveis ao aplicativo

O sistema permite através da aplicação de diferentes arquivos de transformação gerar saídas diferenciadas para cada curso definido no sistema. É interessante também que se possa utilizar o mesmo curso para ser utilizado nos mais diversos tipos de aplicativos, não apenas em navegadores para web, mas sim em celulares wap, em palmtops, etc. No momento é possível associar apenas um arquivo de transformação para cada curso. No futuro cada curso possuirá diversos arquivos de transformação: o hyper-automaton irá identificar o aplicativo sendo utilizado para o acesso e adaptará o conteúdo dos cursos segundo arquivos de transformações definidos para o curso.

7 Conclusões e trabalhos futuros

Este capítulo final da dissertação está dividido em duas seções que apresentam as conclusões do trabalho realizado, uma visão do sistema antes e depois das alternativas estudadas e propostas, trabalhos futuros baseado nas novas perspectivas de expansão tecnológicas do Sistema *Hyper-Automaton* baseado no XML.

7.1 Conclusão

A produção de material instrucional para a *WWW* é uma tarefa dispendiosa. Para justificar sua construção é necessário ter-se a habilidade de poder utilizá-lo em muitas circunstâncias e dentro de cursos produzidos por diferentes professores. De forma similar, os componentes do *courseware* associados devem ser reutilizáveis sob os mais diferentes aspectos como em conteúdo e forma de visualização sempre que possível.

O Sistema *Hyper-Automaton* na sua forma original utilizava o HTML como linguagem de hipertexto para a elaboração dos fragmentos monolíticos do material hipermídia (fig. 7.1), embora funcionando perfeitamente, trazia, por necessidade, a implementação de novas funções que estendessem sua funcionalidade atual e fornecessem suporte para aplicações futuras tais como: intercâmbio de dados entre aplicações, alteração da formatação do *layout* de saída sem alterar a documento original, oferecer outros recursos de *links*, estabelecer regras de construção de documentos, facilitar a busca sobre o material armazenado, melhorar a manipulação dos documentos através da utilização de *tags* personalizados, utilizar uma linguagem padrão que não fizesse uso de recursos proprietários, facilitar a elaboração de conteúdo por parte do professor.

A figura 7.2, embora simplificada, traz uma visão das novas características propostas no sistema

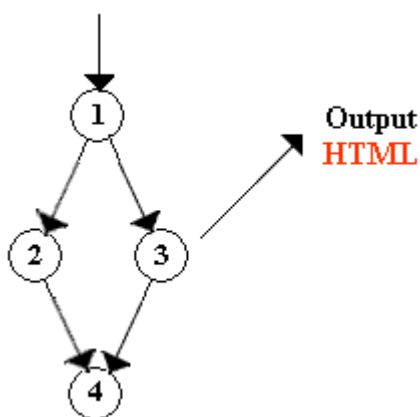


FIGURA 7.1 – Sistema H.A na sua forma original

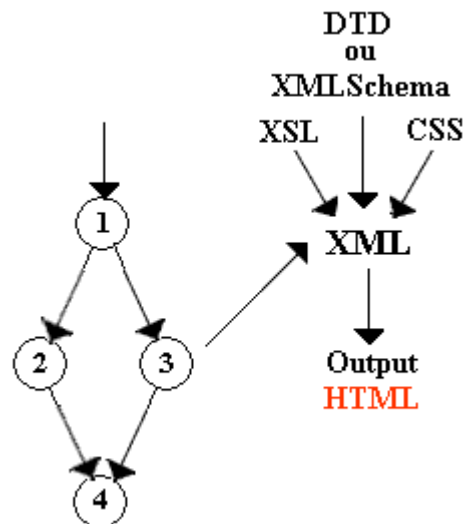


FIGURA 7.2 – Sistema H.A na forma atual

Dessa forma, este trabalho apresentou uma alternativa de implementação e modelagem de sistemas sobre um sistema já existente, amplamente comparado, testado e validado conforme [MAJ00], atribuindo-lhe uma série de benefícios imediatos e futuros listados a seguir.

A tabela 7.1 resume as diferenças entre os sistemas, sendo os benefícios seguintes:

- Os documentos instrucionais passam a ser legíveis tanto ao ser humano como à máquina, pois com a adição do XML cada *tag* passa a ser descrito conforme o seu conteúdo tornando fácil para que tanto ferramentas de busca quanto seres humanos tenham a capacidade de identificá-lo;
- As aplicações são desenvolvidas sobre uma linguagem de *markup* padrão denominado XML que está sendo desenvolvido pelas principais empresas em nível mundial, tais como: Adobe, Oracle, IBM, Sun, XEROX; associadas a um órgão padronizador denominado *W3C (World Wide Web Consortium)*;
- Separação das partes física, lógica dos aspectos de visualização que trouxeram um importante fator de organização e poder de manipulação de aspectos em separado.
- A fragmentação da informação passa a ser maior, aumentando as possibilidades de reuso e manipulação.
- O sistema continua oferecendo suporte a aplicações multimídia convencionais e as derivadas de aplicações XML, assim sendo, as aplicações atuais podem ser executadas, desde que em navegadores portadores dos *plug-ins* apropriados, quanto as derivadas, tais como VML (*Vector Markup Language*), VoxML, SMIL (*Synchronized Multimedia Integration Language*), CDF (*Channel Definition Format*), MusicML, somente nas versões mais modernas dos navegadores Internet Explorer [HAR99];
- O conteúdo instrucional passa a ser construído sobre um padrão de elaboração. Com a utilização de regras de formatação, será possível o estabelecimento de normas de construção para que desenvolvedores de conteúdo instrucional obedeçam a fatores organizacionais de elaboração;
- Diminuição do tráfego de informações na rede a partir da possibilidade do uso de outros padrões de codificação de caracteres, pois anteriormente os caracteres não codificáveis eram adicionados em forma de imagens padrão GIF;
- Utilização de folhas de estilo avançadas padrão *XSL* que atribuíram uma série de

benefícios para a flexibilização do *layout* de saída do sistema podendo realizar pesquisas na estrutura do documento XML, disponibilizando informações pré-determinadas em qualquer seqüência e acima de tudo, realizando manipulações de tomada de decisão em nível de folha de estilo, fator até então não conseguido no sistema atual com a utilização de folhas de estilo em cascata ou em codificação direta em HTML;

- Suporte a utilização de folhas de estilo em cascata, deste modo o XML aceita a inclusão de folhas de estilo em cascata (CSS), se bem que limitada, ainda pode ser útil para imediatas aplicações;
- Utilização de pesquisa sobre documentos. Pelo fato da marcação ser auto-descritível torna-se útil, pois com grande precisão consegue-se resultados na pesquisa em árvore de documentos XML o que acarreta a implementação de ferramentas tais como glossários e aplicações tipo “saiba mais sobre...”, fig. 7.3

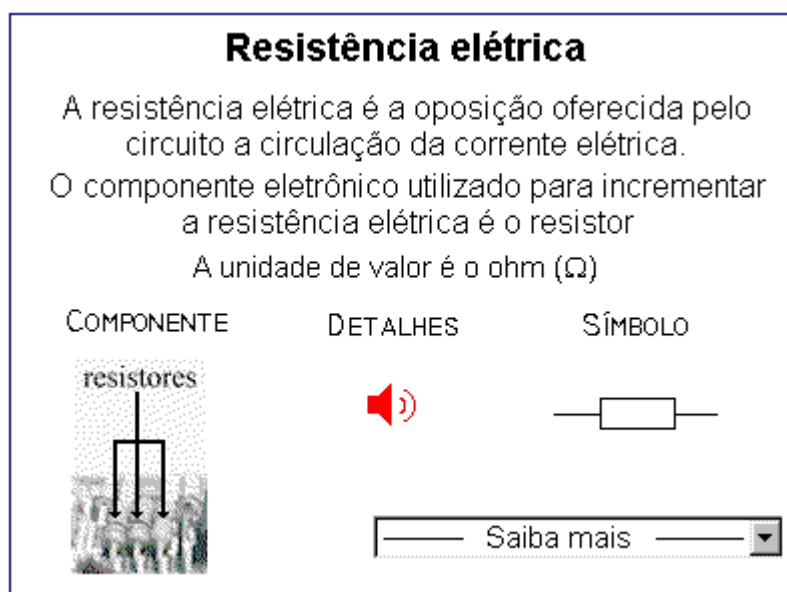


FIGURA 7.3 – Exemplo de saída do sistema H.A.

TABELA 7.1 – Diferenças entre os sistemas HA e XHA

Característica	HA	XHA	Vantagem
Flexibilização	Concatenação HTML	Concatenação XML	Aumento da possibilidade de criação de objetos e controle
Função de saída	Única: Concatenação HTML	Várias: Aplicação de folhas de estilo	Aplicação de múltiplos estilos ao conteúdo de saída
Representação de caracteres	Imagens	Indicado no padrão de codificação	Diminuição do trabalho de criação e tráfego na rede
Possibilidades de implementação	HTML	XML	Ampliação de recursos padronizados existentes ou em criação.

Organização geral	Arquivos	BD	Aumento da performance o organização do sistema
Informação sobre objetos	Não possibilita	Metadados	
Performance	HTML + Perl	XML + Java	Perl – Interpretada Java - Compilada

7.2 Trabalhos futuros

Nesta seção estão definidos os parâmetros e idéias para trabalhos futuros sobre o sistema *Hyper-Automaton*.

Esta dissertação está inserida no contexto de um grupo de pesquisa com trabalhos paralelos na área explorada, os quais se utilizam ou utilizarão dos resultados aqui obtidos.

Com a inclusão da tecnologia XML e suas aplicações no Sistema *Hyper-Automaton*, abrem-se novos campos de pesquisa até então não explorados pelo grupo, que serão listados e explicados nos mesmos moldes do item anterior.

Como principal trabalho a ser desenvolvido com a inclusão de outras áreas da informática, torna-se fundamental o desenvolvimento de aplicações nas quais possam mapear para os *tags* em estruturas previamente definidas um conteúdo instrucional qualquer, cuja fonte, poderia ser um documento de hipertexto tendo por objetivo a facilidade de inclusão de material no sistema.

Como importante e intenso trabalho o qual está sendo desenvolvido por um grupo, consta da implementação de ferramentas de edição de conteúdo para os cursos desenvolvidos orientados ao sistema, como também, a criação de editores para a definição de estilos, cuja função será de facilitar a construção da interface para a visualização do conteúdo. Todas as aplicações descritas neste parágrafo terão por desafio serem dirigidas a usuários leigos, desta maneira, as facilidades deverão ser implantadas sem fazer com que o sistema perca suas características originais, ao passo que, o usuário não tenha como restrição a obrigação de possuir conhecimentos de funcionamento intrínseco do sistema.

Através da exploração de recursos de SGBD, uma ferramenta poderá ser criada com a finalidade de realizar a exportação de um curso armazenado. Conjuntamente com um *applet* o referido aluno poderia *off-line* acompanhar o curso, fazer exercícios propostos ou manusear um determinado material.

Faz-se necessário, em disciplinas de cunho tecnológico ou matemáticas, onde as expressões numéricas ou literais são abundantes, a utilização correta de uma aplicação XML orientada a tipo de dados puramente matemáticos, tal aplicação chama-se MathML e esta ainda não foi amplamente explorada em nossas aplicações, pelo fato de que os atuais navegadores ainda não processarem corretamente os conteúdo do seus *tags*.

Para finalizar, serão desenvolvidas aplicações que a partir de estruturas XML gerem formas monolíticas com saídas multiplataforma que são muito utilizadas quando se deseja alta rigidez de formatação e/ou a imunidade total de programas maliciosos, também conhecidos por “vírus”. Essas saídas são normalmente conhecidos por documentos padrão PDF ou PS.

Anexo 1 Artigo WIE'2000

Este artigo [MAC 00] foi submetido e publicado na categoria de artigo completo no I Workshop Informática e Educação: uma Nova Abordagem Educacional, promovido pela Faculdade de Educação – FAED conjuntamente com o Grupo de Estudos e Pesquisa em Software Educacional – GEPESE nos dias 9 e 11 de Novembro de 2000 na Universidade de Passo Fundo (UPF).

Este evento tem por objetivo divulgar, refletir e discutir questões, consideradas prioritárias no desenvolvimento e na pesquisa de softwares educacionais e seus reflexos quanto ao uso da informática na educação e teve por clientela envolvida professores, alunos e profissionais da área de informática e educação.

O artigo faz uma análise preliminar das características peculiares do XML para sua utilização no sistema de Hyper-Automaton para ensino a distância, descrevendo algumas maneiras de construção de documentos XML, regras de estrutura e maneiras de disponibilização do conteúdo armazenados e estruturados.

Neste capítulo são descritos as limitações do HTML e os motivos pelos quais o XML passa a ser introduzido em nossas aplicações, logo após são abordados os conceitos iniciais do XML e os problemas enfrentados com sua utilização.

O estudo comparativo referente às linguagens de hipertexto [MAC99], também serviu como forte embasamento para a elaboração deste artigo, bem como, tornou clara a adoção do XML como linguagem de hipertexto utilizada em nossos desenvolvimentos que culminaram nesta dissertação.

O trabalho individual citado no parágrafo anterior, não foi acrescentado neste trabalho, mas está à disposição dos interessados na biblioteca desta instituição.

I WORKSHOP EM INFORMÁTICA NA EDUCAÇÃO:
UMA NOVA ABORDAGEM EDUCACIONAL
PASSO FUNDO-RS-BRASIL
NOV.2000

CARACTERÍSTICAS E VANTAGENS NA ESTRUTURAÇÃO DE DOCUMENTOS EM XML NA WWW PARA EAD

César Costa Machado
Prof. Paulo B. Menezes

Universidade Federal do Rio Grande do Sul – Instituto de Informática
Av. Bento Gonçalves, 9500. Caixa Postal 15064. CEP 91501-970.
Porto Alegre – RS – Brasil
Tel.: +55(51)316-6161 Fax: +55(51)319-1576
E-mail: cmachado@atlas.ucpel.tche.br; blauth@inf.ufrgs.br

RESUMO

Este trabalho apresenta uma forma de estruturação e visualização do conteúdo do material utilizado em cursos disponibilizados na WWW utilizando-se construções formais conhecidas como Autômatos Finitos Determinísticos com Saída. O estudo baseado nas evoluções tecnológicas da internet sugere que o sistema atual incorpore os benefícios da estruturação moderna de documentos, consoantes com as necessidades e benefícios tecnológicos, bem como, faça uso de importantes recursos derivados das linguagens de marcação referentes as possibilidades de formatação do conteúdo disponibilizado visualmente. O objetivo deste trabalho é reestruturar o curso existente, que embora funcionando perfeitamente à base de fragmentos HTML, possa ser reconstruído perante estas novas tecnologias, mais adequadamente para o formato XML e folhas de estilo XSL e CSS.

Palavras-chave: WWW, XML, autômatos finitos, educação à distância

CARACTERÍSTICAS E VANTAGENS NA ESTRUTURAÇÃO DE DOCUMENTOS EM XML NA WWW PARA EAD

Prof. Paulo Blauth Menezes
Universidade Federal do Rio Grande do Sul
Porto Alegre – RS – Brasil
e-mail: blauth@inf.ufrgs.br

César Costa Machado
Universidade Católica de Pelotas/
CEFET-RS
Pelotas – RS – Brasil
e-mail: cmachado@atlas.ucpel.tche.br

RESUMO

Este trabalho apresenta uma forma de estruturação e visualização do conteúdo do material utilizado em cursos disponibilizados na WWW utilizando-se construções formais conhecidas como Autômatos Finitos Determinísticos com Saída. O estudo baseado nas evoluções tecnológicas da internet sugere que o sistema atual incorpore os benefícios da estruturação moderna de documentos, consoantes com as necessidades e benefícios tecnológicos, bem como, faça uso de importantes recursos derivados das linguagens de marcação referentes as possibilidades de formatação do conteúdo disponibilizado visualmente. O objetivo deste trabalho é reestruturar o curso existente, que embora funcionando perfeitamente à base de fragmentos HTML, possa ser reconstruído perante estas novas tecnologias, mais adequadamente para o formato XML e folhas de estilo XSL e CSS.

Palavras-chave: WWW, XML, autômatos finitos, educação à distância

1. Introdução

O ensino a distância é uma das áreas de grande desenvolvimento e que gera inúmeras fontes de pesquisa nas universidades brasileiras. Na Universidade Federal do Rio Grande do Sul (UFRGS), mais especificamente, desenvolvemos muitas aplicações voltadas a esta área.

Este trabalho tratará das principais características da linguagem de marcação de documentos estruturados na Web, a *eXtensible Markup Language (XML)*, que oferece significativos avanços em termos de disponibilização e visualização de documentos dos mais diferentes formatos, principalmente orientadas ao ensino auxiliado por computador.

2. O HTML e suas limitações

Um dos importantes serviços que a Internet oferece é a conhecido como World Wide Web - WWW, onde vários recursos disponíveis na Internet podem ser acessados.

A WWW suporta inúmeros recursos multimídia e é o local destinado para a publicação dos mais variados tipos de informações já imaginados, com inúmeras possibilidades de disposição de textos, imagens dos mais diferentes formatos, sons e outros recursos

também associados a hyperlinks, que permitem inicializar outras funções como serviços, aplicativos ou redirecionar a navegação para outro endereço eletrônico [1].

Atualmente a ferramenta mais comum utilizada pelos desenvolvedores afim de disponibilizar conteúdo na Web que suportam satisfatoriamente bem os recursos de hiperlinks, é o bem conhecido HTML (*HyperText Markup Language*), bem como, outras linguagens de hipertexto derivadas desta, que encapsulam scripts oferecendo recursos adicionais poderosos de grande valia.

Embora o HTML tenha sido definido apartir do SGML (*Standard Generalized Markup Language*), através da apropriação de *tags* adequados com a finalidade de suporte entre plataformas para a apresentação de páginas Web, esta começa a dar sinais de fraqueza, vítima do seu próprio sucesso, pois sua facilidade de compreensão, manipulação e alta flexibilidade, torna-se evidente [2,3].

Problemas decorrentes destas facilidades começam a surgir e, como se não bastasse, grandes empresas desenvolvedoras de browsers passaram a criar implementações conhecidas como proprietárias, destinadas a navegadores Web específicos, portanto não reguladas pela W3C (*World Wide Web Consortium*), órgão este que busca a tão sonhada padronização na WWW.

Onde a HTML falha? [3]

- Exibir páginas extremamente longas, fornecer melhor controle sobre aparência das páginas, suportar vários tipos de hyperlinks, distribuir uma variedade cada vez maior de informações em intranets bem como na Internet.

O XML (*eXtensible Markup Language*), torna-se adequado a tipos de documentos que requerem formatações que não se encaixam no molde do HTML. Eis alguns exemplos: livros, transações financeiras, manuais técnicos, fórmulas químicas, registros médicos, catálogos de registros para museus

3. O XML e sua importância

XML é um texto baseado em linguagem de markup (*tags*) que está se tornando um rápido padrão de troca de dados na Web. Em parte é similar a HTML, pois pode-se

identificar dados usando tags (identificadores inclusos, como estes: <...>).

Os tags XML identificam (explicitam) o que o dado significa, melhor do que o HTML, pois preocupa-se somente como os dados serão dispostos visualmente quando vistos pelo usuário, como por exemplo: este dados será exibido em negrito, itálico, na cor vermelha, etc., (...); um tag XML age como um campo de nome, ele permite identificar com um rótulo em um dado, tornando o documento muito mais claro ao desenvolvedor (por exemplo: <capitulo>...</capitulo>).

Na mesma maneira que é definido os nomes dos rótulos dos dados, é também definido a estrutura do documento, existe uma liberdade para que seja usado qualquer tag XML que faça sentido para uma certa aplicação, obviamente, desta maneira para múltiplas aplicações pode-se utilizar o mesmos dados XML [3,4]. É nesta característica de um tag conter outros tags que dá ao XML a habilidade de representar hierarquicamente a estrutura de dados de qualquer documento que o torna de fácil acesso.

Em comparação com o HTML, espaços em branco são essencialmente irrelevantes, desta maneira, pode-se formatar o dado com legibilidade, processá-lo facilmente, e ainda, procurar e localizar conjuntos de dados, pois como já explicado, XML identificada o dado contido, tornando fácil a localização destes pelo usuário ou qualquer implementação de pesquisa de dados.

Há um número de razões para a larga aceitação do padrão XML. Abaixo estão listadas as mais importantes:.

Texto de fácil clareza:

Como o XML não é um formato binário, pode-se criar e editar arquivos apartir de editores de textos, bem como ferramentas de desenvolvimento em ambientes visuais. Este é de fácil depuração e torna-se apropriado para de armazenamento de pouca quantidade de dados. Nesta mesma visão, o XML “front end” de um banco de dados torna possível e eficiente o armazenamento de grande volume de dados. Então o XML permite escalabilidade para muitas aplicações apartir de pequenas adaptações nos arquivos, permitir o gerenciamento de grandes volumes de dados [5].

Identificação de dados:

XML identifica com clareza o tipo de dados que este condiciona, e não como este é disposto na tela de um monitor de vídeo ou em outro dispositivo que permita visualização, pois sua markup identifica a informação e a separa em partes, tornando mais fácil o processamento por um programa de e-mail, por exemplo: um *address book* teria a possibilidade de procurar mais facilmente a pessoa em particular e extrair a informação de endereço do resto do corpo da mensagem. Resumindo: pelo fato de poder-se identificar pequenas partes de informação, estes podem ser utilizados por diferentes aplicações [5,6].

Estilização:

Quando a visualização é um fator importante, a folha de estilo padrão XSL, permite que se defina como o dado deverá ser apresentado [5,7,8]. Por exemplo, a folha de estilo (*stylesheet*) para:

<to>you@yourAddress.com</to>, pode informar:

4. Começar em nova linha.
5. Mostrar "To:" em negrito, seguido por um espaço.
6. Mostra o destino.

O que produz: **To:** you@yourAddress

Naturalmente que mesma tarefa poderia ser feita em HTML, mas neste formato, não haveria a possibilidade destes dados serem processados por programas de busca ou programas para a extração de endereços eletrônicos ou semelhantes. O mais importante é que este padrão é independente do estilo, o que permite que para cada documento seja aplicado os mais variados formatos de visualização, podendo aplicar diferentes folhas de estilos para produzir como saída documentos em postscript, TEX, PDF, ou algum novo formato que ainda não tenha sido inventado [8].

Reusabilidade:

Um dos melhores aspectos de documentos em XML é que estes podem ser compostos por vários módulos. Até então igual ao HTML, mas com a grande diferença, com XML pode-se ligá-lo a outros documentos. É permitido assim que se possa acessar uma parte de um documento sem que seja necessário o cliente carregá-lo totalmente. Pode-se construir um documento em módulos, onde estes podem estar em qualquer local no mundo, como em um trabalho cooperativo, onde resultados devem ser trocados e atualizados via Web, assim sendo, este documentos será constituído como os pedaços de

uma peça, facilitando sua atualização [3].

Linkability

Graças ao HTML, a habilidade de definir *links* entre documentos é tido como uma necessidade. O XML possui a habilidade de manipulá-los das mais diferentes formas, isto é, permite que seja definido *links* duplex, multi-alvo, links expandidos e links entre dois documentos existentes que estão definidos em um terceiro [7].

Fácil Processamento

Com uma regular e consistente notação, torna fácil construir programas para processar dados XML. Por exemplo, em HTML por sua exagerada flexibilidade [5], as regras de construção de documentos não são rígidas, permitindo em que algumas vezes tags sejam suprimidos ou subtendidos. Isto torna difícil o processamento das informações e leva a resultados inesperados. Mas em XML, o tag deve obrigatoriamente possuir o terminador. Estas restrições obedecidas, torna o documento elaborado nesta linguagem de *markup* bem formado (document well-formed). De outro modo o processador XML não estaria apto a ler os dados contidos em tal documento. Para concluir existem softwares adequados para analisar a estrutura de um documento XML, estes programas são denominados *parsers*.

4. Proposta de aplicação

A proposta de aplicação que será alvo de estudos, visa dar prosseguimento ao trabalho Modelagem de Cursos na Web Utilizando Sistemas Formais [11,12] e tem como objetivo propor alternativas com respeito à formatação da interface com o usuário através da utilização do XML e de suas aplicações descritas nos itens anteriores.

Cursos na Web Utilizando Sistemas Formais, utilizam construções formais conhecidas como Autômatos Finitos Determinísticos [13]. Nós introduzimos o conceito de "cursos são autômatos" como uma estrutura que permite fácil implementação, criação de material hipermídia independente do autômato, ao mesmo tempo que encoraja o reuso de páginas Web em vários cursos, os quais podem ser construídos com enfoques diferentes, diminuindo a redundância na criação de páginas [10,11,12].

Este projeto já é realidade e cria um sistema semi-automatizado para o suporte do

ensino de Informática Teórica através da aplicação de conceitos inerentes à Ciência da Computação, em especial da Teoria de Autômatos e Teoria das Categorias, juntamente à tecnologia de hiperdocumentos, reunindo os benefícios de ambas [11].

O objetivo geral do modelo Hyper-Automaton centra-se no estudo da aplicação do formalismo de Autômatos Finitos com Saída (Máquina de Mealy e Máquina de Moore) como um modelo estrutural para a organização de hiperdocumentos instrucionais, em especial de Cursos na Web. O modelo é inspirado por pesquisas clássicas na área de hiperdocumentos e recentes iniciativas na WWW, com especial enfoque no desenvolvimento de sistemas de hipertexto onde a base de hiperdocumentos é projetada de forma independente da estrutura de controle da aplicação hipermídia, e suporta algumas facilidades descritas no Modelo Dexter como a composição de estruturas hierárquicas, especificação de vários conjuntos de links sobre um mesmo corpo de hiperdocumentos e objetos separados da estrutura de navegação [14]. Cada autômato define um curso e consiste de um conjunto de hiperdocumentos independentes, os quais podem pertencer a outros cursos. A função de transição funciona como ligação lógica entre os hiperdocumentos e a função de saída compõe as páginas. O resultado final é a estrutura básica de páginas e links de hipertexto em um site na Web. O modelo leva a um alto grau de modularização do material instrucional, apresentando as seguintes vantagens: facilidade de reuso de páginas em diversos cursos, com eliminação da redundância; independência dos hiperdocumentos da estrutura do autômato, cuja alteração não influi nas páginas e vice-versa; permite que qualquer usuário crie links de e para qualquer documento; facilidade de implementação e manutenção; interface gráfica simples e direta; elaboração de seqüências instrucionais com enfoques específicos e capaz de oferecer estudo individualizado; operações categoriais fornecem um esquema de composição de cursos que permite a construção de novos cursos sobre cursos já existentes através de procedimentos de alto nível [11, 12].

O sistema Hyper-Automaton constitui-se de um sistema semi-automatizado para o suporte a cursos na Web (com base em uma arquitetura cliente/servidor e interface desenvolvida em HTML) através da aplicação de conceitos inerentes à Ciência da Computação, em especial da Teoria de Autômatos, tecnologia de Hipermídia e Teoria das Categorias, reunindo os benefícios de ambas. Embora a ênfase utilizada para a validação do modelo seja a implementação de sistemas para o ensino a distância, os

resultados obtidos se aplicam para sistemas de hipertexto como um todo.

Nosso trabalho na fase atual, está centrado nas várias possibilidades de formatação do conteúdo disponibilizado, culminando em uma adequada interface com o usuário, onde, a flexibilização da formatação do conteúdo obtido na função de saída do autômato será uma aplicação que remonte o documento nas partes desejadas e com as características mais adequadas relacionadas as necessidades do usuário e do curso em questão. O estudo proposto trará uma maior flexibilidade ao já existe, pois no estado atual, o que se apresenta é uma única e rígida possibilidade composta de fragmentos em HTML.

O diagrama em blocos que ilustra as partes do sistema na figura abaixo, inclui o que estamos desenvolvendo atualmente. A novidade está em estruturar em XML todos as informações do curso e utilizar templates XSL afim de visualização e lay-out do referido curso on-line. Tais alterações objetivarão principalmente a atualização tecnológica do sistema de acordo com os benefícios já descritos do XML.

5. Dificuldades encontradas

As principais dificuldades encontradas até este ponto da pesquisa é a incompatibilidade entre diversos browsers, pois somente apartir da versão 4.0 do browser Internet Explorer da Microsoft, suporta a utilização de *Extensive Style Sheets (XSL)*. Outro aspecto importante é escassa bibliografia à respeito, justificada pelo recém desenvolvimento e utilização destas tecnologias.

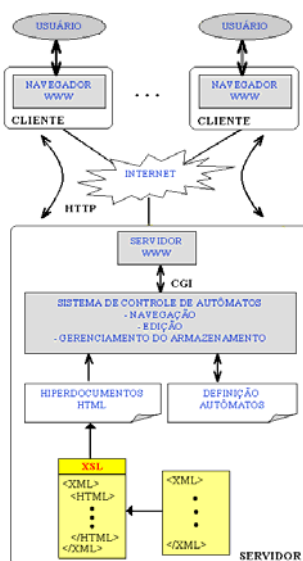


Figura 1. Estruturação em blocos do sistema

6. Referências

- [1] LYNCH, PATRICK J.; HORTON SARAH. Web Style Guide. 1999.
- [2] GOLDFARB, CHARLES F. The SGML Handbook, Clarendon Press, Oxford, 1990, pág. 5.
- [3] LIGHT, RICHARD. Iniciando em XML. Makron Books do Brasil Editora Ltda. 1999.
<http://www.w3.org/TR/REC-xml>
- [4] BRAY, T. PAOLI, C. M.; Extensible Markup Language (XML) 1.0 Specification. Fevereiro 1998.
- [5] HAROLD, ELLIOTTE, R. XML Bible. IDG Books Worldwide. Ago. 1999.
- [6] TIDWELL, DOUG. Tutorial: Introduction to XML. IBM. Jul. 1999.
<http://www.ibm.com/developerWorks>
- [7] ARMSTRONG, ERIC. Working with XML. Version 1, Update 5 -- 9 Aug 1999.
- [8] ADLER, S. (IBM); DEATCH, S. (Adobe); GUTENTAG, E. (Sun). Extensible Stylesheet Language (XSL) – Version 1.0. Draft 27 March 2000.
<http://www.w3.org/TR/2000/WD-xsl-20000327/>
- [9] MOULTIS, NATANYA P., KIRK, CHERYL; XML Black Book. 1999
- [10] Júlio P. Machado, Paulo F. B. Menezes. *Sistemas de Gerenciamento para o Ensino a Distância*. In Proceedings of V Congresso Internacional de Educação a Distância. São Paulo - Brasil, 1998. <http://www.abed.org.br>
- [11] MACHADO, Júlio Henrique Araújo Pereira. Hyper-Automaton: Hipertextos e Cursos na Web Utilizando Autômatos Finitos com Saída. Porto Alegre: CPGCC da UFRGS. 2000. (Dissertação de mestrado).
- [12] MENEZES, Paulo Blauth, MACHADO, Júlio Henrique Araújo Pereira. Web courses are automata: a categorial framework. In: II WORKSHOP DE MÉTODOS FORMAIS, 1999, Florianópolis. II Workshop On Formal Methods. Florianópolis: UFSC, Instituto de Informática da UFRGS, 1999. p.79-88.
- [13] Paulo F. B. Menezes. *Linguagens Formais e Autômatos*. Segunda Edição. Editora Sagra-Luzzatto. Brasil, 1998.
- [14] F. Halasz, M. Schwartz. The Dexter HyperText Reference Model. Communications of the ACM, 1994, v.37, n.2, p.30-39.

Bibliografia Consultada

1. Paulo F. B. Menezes, A. Sernadas, J. F. Costa. *Nonsequential Automata Semantics for a Concurrent Object-Based Language, Revised and Extended Version*. In Proceedings of the First USA-Brazil Workshop on Formal Foundations of Software Systems. Rio de Janeiro - Brasil, 1998. Electronic Notes in Theoretical Computer Science n.14. <http://www.elsevier.nl/locate/entcs/volume1.html>
2. MORAIS, Carlos Tadeu Q de, MENEZES, Paulo Blauth, MACHADO, Júlio Henrique Araújo Pereira. Study of rooms for the teaching mediated by computers. In: INTERNATIONAL CONFERENCE ON ENGINEERING AND COMPUTER EDUCATION, 1998, Rio de Janeiro. International Conference On Engineering And Computer Education. Rio de Janeiro: IEEE, 1998. p.395-398.
3. McCormack, C.; Jones D.; Building a Web-Based Education System. Wiley Computer Publishing, 1998.

Anexo 2 Artigo ISKM/DM'2000v2

Este artigo [MAC 00a] foi submetido e publicado na categoria de artigo completo no *International Symposium on Knowledge Management / Document Management* – Simpósio Internacional da Gestão do Conhecimento e Gestão Documentos, promovido pela Pontifícia Universidade Católica de Curitiba – PUC PR e Centro Internacional de Tecnologia de Software – CITS, ocorrido em 26 a 29 de Novembro de 2000 na cidade de Curitiba.

Em sua quarta edição, o ISKM/DM consolida-se como o evento de referência destas áreas, atraindo contribuições de alto nível dos mais diversos setores. Esta edição do ISKM/DM, contou com 150 participantes de 50 diferentes organizações. Foram apresentados 22 trabalhos técnicos e 8 relatos de experiências, além de 5 palestras convidadas.

O artigo faz uma análise mais aprofundada do XML orientado sua aplicação ao sistema de Hyper-Automaton. Outras abordagens importantes da linguagem são colocadas, conjuntamente com testes e seus respectivos resultados, bem como, início do estudo da aplicação correta de folhas de estilos e pesquisa na árvore estrutural do documento objetivando a disponibilização do conteúdo armazenados em XML.

Neste capítulo são descritos os conceitos iniciais do gerenciamento do armazenamento de documentos, regras estruturais para a elaboração padrão de documentos e introdução ao uso de folhas de estilos, ao final, listamos os desafios pelos quais temos que enfrentar com as novas alterações no sistema visando adequá-lo as novas necessidades.

Utilização do XML no Sistema Hyper-Automaton

César Costa Machado¹, Júlio P. Machado², Roges Grandi², P. Blauth Menezes²

cmachado@atlas.ucpel.tche.br

{jhapm, roges, blauth}@inf.ufrgs.br

1 - Universidade Católica de Pelotas /
CEFET-RS

Félix da Cunha, 412. CEP 96010-000.

Pelotas - RS

2 - Universidade Federal do Rio Grande do Sul

Av. Bento Gonçalves, 9500 - Bloco IV.

Caixa Postal: 15064 CEP 91501-970.

Porto Alegre - RS

Resumo

Este trabalho dá prosseguimento a estudos anteriores buscando oferecer uma forma de estruturação e visualização do conteúdo do material utilizado em cursos disponibilizados na WWW utilizando-se construções formais conhecidas como Autômatos Finitos com Saída. O estudo baseado nas evoluções tecnológicas da Internet sugere que o sistema atual incorpore os benefícios da estruturação moderna de documentos, consoantes com as necessidades e benefícios tecnológicos, bem como, incorpore importantes recursos derivados das linguagens de marcação referentes às possibilidades de formatação do conteúdo disponibilizado visualmente. O objetivo deste trabalho é esquematizar e descrever, de acordo com uma visão prática, o que é necessário dentro das inúmeras possibilidades do XML a fim de apontar caminhos para a reestruturação e inclusão do XML e XSL no sistema Hyper-Automaton.

Palavras-chave

WWW, XML, Autômatos finitos, Gerenciamento de hiperdocumentos, Cursos na web.

Introdução

Este artigo dá prosseguimento ao trabalho Modelagem de Cursos na Web Utilizando Sistemas Formais [18, 1, 2] e tem como objetivo propor alternativas com respeito à modelagem de hiperdocumentos e formatação da interface com o usuário através da utilização do XML e de suas aplicações.

Cursos na Web baseados no modelo Hyper-Automaton [1,2,18] utilizam construções formais conhecidas como Autômatos Finitos Determinísticos [3]. Introduzimos o conceito de "cursos são autômatos" como uma estrutura que permite fácil implementação, criação de material hipermídia

independente do autômato, ao mesmo tempo que encoraja o reuso de páginas Web em vários cursos, os quais podem ser construídos com enfoques diferentes, diminuindo a redundância na criação de páginas.

Este projeto já é realidade e constitui-se de um ambiente semi-automatizado para o suporte do ensino de Informática Teórica (<http://teia.inf.ufrgs.br>) através da aplicação de conceitos inerentes à Ciência da Computação, em especial da Teoria de Autômatos e Teoria das Categorias, juntamente à tecnologia de Hiperdocumentos, reunindo os benefícios de ambas [1].

O objetivo geral do modelo Hyper-Automaton centra-se no estudo da aplicação do formalismo de Autômatos Finitos com Saída (Máquina de Mealy e Máquina de Moore) [3] como um modelo estrutural para a organização de hiperdocumentos instrucionais, em especial de Cursos na Web. O modelo é inspirado por pesquisas clássicas na área de hiperdocumentos e recentes iniciativas na WWW, com especial enfoque no desenvolvimento de sistemas de hipertexto onde a base de hiperdocumentos é projetada de forma independente da estrutura de controle da aplicação hipermídia, e suporta algumas facilidades descritas no Modelo Dexter [5] como a composição de estruturas hierárquicas, especificação de vários conjuntos de links sobre um mesmo corpo de hiperdocumentos e objetos separados da estrutura de navegação.

Cada autômato define um curso e consiste em um conjunto de hiperdocumentos independentes, os quais podem pertencer a outros cursos. A função de transição funciona como ligação lógica entre os hiperdocumentos e a função de saída compõe as páginas. O resultado final é a estrutura básica de páginas e links de hipertexto em um site na Web. O modelo leva a um alto grau de modularização do material instrucional, apresentando as seguintes vantagens: facilidade de reuso de páginas em diversos cursos, com eliminação da redundância; independência dos hiperdocumentos da estrutura do autômato, cuja alteração não influi nas páginas e vice-versa; permite que qualquer usuário crie links de e para qualquer documento; facilidade de implementação e manutenção; interface gráfica simples e direta; elaboração de seqüências instrucionais com enfoques específicos e capaz de oferecer estudo individualizado; operações categoriais fornecem um esquema de composição de cursos que permite a construção de novos cursos sobre cursos já existentes através de procedimentos de alto nível [1,2,18].

Este trabalho está centrado nas várias possibilidades de formatação do conteúdo disponibilizado, culminando em uma adequada interface com o usuário, onde, a flexibilização da formatação do conteúdo obtido na função de saída do autômato será uma aplicação que remonte o documento nas partes desejadas, com características mais adequadas às necessidades do usuário e do curso em questão. O estudo proposto trará uma maior flexibilidade ao sistema já existente pois, no estado atual, o que se apresenta é uma única possibilidade composta de fragmentos em HTML (concatenação de páginas).

VISÃO DO SISTEMA

O diagrama em blocos, que ilustra as partes do sistema (fig.1), apresenta em destaque a extensão ao sistema Hyper-Automaton discutida neste trabalho. A novidade está em estruturar os conteúdos (hiperdocumentos) em XML obedecendo às definições preestabelecidas para os documentos (DTD) [6,7] e utilização de templates XSL e/ou folhas de estilos em cascata (CSS) [8,9] a fim de visualização e layout

do referido curso on-line. Estas alterações objetivarão principalmente a atualização tecnológica do sistema de acordo com os benefícios relacionados ao XML e descritos na seções seguintes.

A partir deste momento, este trabalho descreverá outras importantes características do XML [9] intimamente ligadas aos nossos objetivos, explicitando sua utilização conforme descrito na fig.1. Características estas que serão amplamente utilizadas na nova proposta de implementação sobre o sistema Hyper-Automaton.

Observando a fig.1 (direita), procura-se detalhar de forma gráfica o que deverá ser implementado. As características de cada bloco serão detalhadas na seqüência.

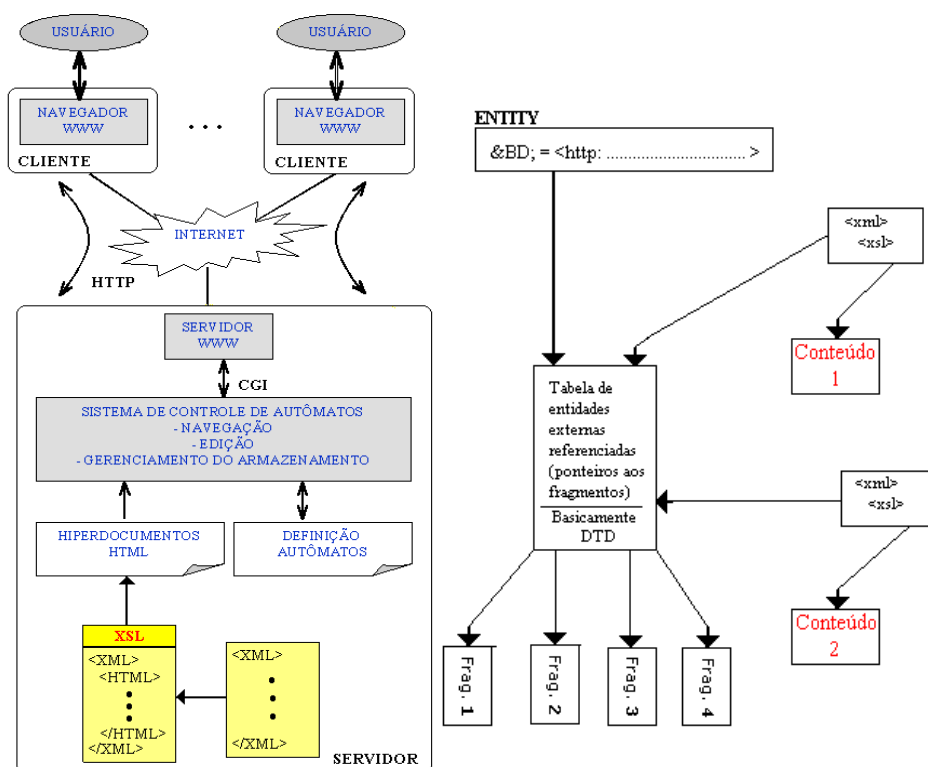


Figura 1 - Estrutura em blocos do sistema

Conforme o modelo de autômatos com saída utilizado, no nível mais baixo do sistema têm-se fragmentos de informação. Estes são considerados unidades atômicas se levarmos em consideração o conceito de alfabeto de saída. Um fragmento pode vir a ser um parágrafo de um texto, uma imagem, um vídeo, etc. A unidade de apresentação é uma página *Web* (documento *HTML*) chamada de palavra de saída no universo dos autômatos (na terminologia de hipermídia o conceito é equivalente ao termo nodo). Uma página é, então, construída sobre esses fragmentos e quais construtores são possíveis depende da implementação do sistema. Na versão atual do Hyper-Automaton, cada página é uma seqüência linear de fragmentos estáticos. Novos construtores podem ser definidos a partir da extensão da função de saída dos autômatos com saída. Um benefício imediato da utilização de hiperdocumentos marcados com XML, é a criação de funções de saída que constróem automaticamente um novo hiperdocumento (como um índice, ou resumo de um livro) a partir de consultas XSL. Como a função de saída também é responsável pela finalização das páginas *Web* a serem visualizadas por um navegador, um uso imediato do XSL é permitir que a função determine aspectos de *layout* das páginas para os conteúdos (fontes, alinhamento, cores,

etc.) utilizando-se de folhas de estilo. Além disso, através de um modelo de usuário mantido em banco de dados, é possível permitir aos usuários armazenarem opções pessoais para a visualização dos conteúdos.

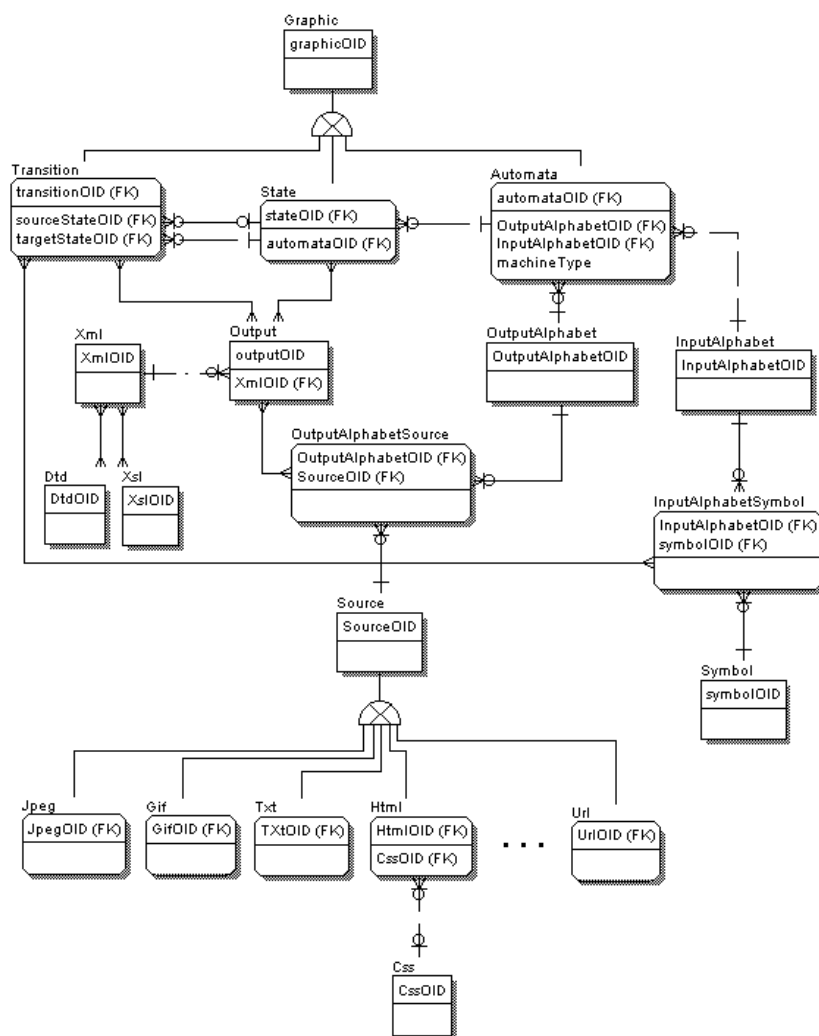


Figura 2 - Diagrama entidade-relacionamento do sistema

GERENCIAMENTO DO ARMAZENAMENTO

Uma última abordagem a considerar sobre os formatos para os cursos realizados através do Hyper-Automaton é seu armazenamento persistente em banco de dados. Foi elaborado um diagrama entidades-relacionamentos (DER) a partir do diagrama de classes do Hyper-Automaton, que foi gerado através de mapeamento do paradigma da orientação a objetos para o paradigma relacional [15, 16, 17]. Como podemos observar (fig.2), existem várias tabelas relacionadas à formatação dos documentos apresentados via Web. Em relação aos conceitos do Hyper-Automaton, os documentos disponibilizados aos usuários são funções de saída sobre um alfabeto.

As saídas, (tabela *Output*), especializadas em documentos GIF, JPEG, etc., associam-se, portanto, a alfabetos de saída (tabela *OutputAlphabetSource*), que as vinculam aos hyper-automata, e um formato que as define, conforme os conceitos de XML (tabela *Xml*). Cada documento XML armazenado pode se relacionar, por sua vez, a declarações de tipo de documento (tabela *Dtd*) e a códigos XSL (tabela

XsI). Ainda relacionado ao tema formatação, podemos relacionar estilos diretamente aos documentos HTML armazenados através da tabela CSS.

XML (EXTENSIBLE MARKUP LANGUAGE)

XML é um texto baseado em linguagem de marcação que esta se tornando rapidamente um padrão de troca de dados na Web. Em parte é similar à HTML, pois pode-se identificar dados utilizando marcações (identificadores inclusos).

As marcações XML identificam (explicitam) o que o dado significa de forma melhor que o HTML, o qual preocupa-se somente como os dados serão dispostos visualmente quando acessados pelos usuários. Uma marcação XML age como um campo de nome, permitindo identificar com um rótulo um determinado dado (por exemplo, "<capítulo>...</capítulo>"), tornando o documento muito mais claro ao desenvolvedor e aos programas de manipulação (denominados de parsers).

Da mesma maneira que é definido os nomes dos rótulos dos dados, é também definida a estrutura do documento. Existe uma liberdade para que seja utilizado qualquer marcação XML que faça sentido para uma certa aplicação e, desta maneira, para múltiplas aplicações, pode-se utilizar os mesmos dados XML. Além disso, a característica de uma marcação conter outras marcações fornece a habilidade ao XML de representar hierarquicamente a estrutura de dados de qualquer documento.

Um outro aspecto fundamental dos documentos XML é que estes podem ser compostos por vários módulos. É permitido assim que se possa acessar uma parte de um documento sem que seja necessário ao cliente carregá-lo totalmente. Assim sendo, documentos podem ser constituídos por peças separadas, as quais podem estar localizadas em qualquer repositório na rede, facilitando a atualização e reutilização de hiperdocumentos [6].

A DTD (Data Type Definition)

A DTD significa, em português, declaração de tipo de documento. Esta tem por função indicar as regras que o documento XML está seguindo. Como se pode observar, é o centro do esquema da fig.1 (direita).

As regras contidas na DTD podem estar inclusas em um documento XML, chamado de subgrupo interno, ou em outra unidade, referenciadas pelo URF, chamado de subgrupo externo. É oportuno ressaltar que nada impede que tenhamos em um mesmo documento XML os dois tipos de DTD's.

A importância da utilização de documentos XML em acordo com uma DTD é fundamental quando é necessário construir documentos obedecendo a padrões corretos de elaboração física e/ou lógica. Desta maneira os documentos elaborados podem ser:

- Documentos inválidos: é quando um documento XML não é validado contra uma DTD associada ao documento, não existindo elementos e atributos declarados. Assim sendo, nenhuma verificação é feita para assegurar que cada elemento contenha os subelementos que a DTD informa que deveriam

conter.

- Documentos válidos: o documento XML válido deve obedecer a todas as regras. Este só não deve ter uma DTD, mas cada elemento deve estar em conformidade com as regras que a DTD contém. Este é o modelo no qual os documentos XML geralmente serão criados e atualizados. As regras de validade do XML são uma camada adicional às regras para serem bem formados. Todo documento XML válido deve ser também bem formado.

Os documentos XML podem fazer referência a recursos não-XML que estão fora do documento, como por exemplo: arquivos de imagem, clipes de vídeo, clipes de áudio, arquivos de processador de texto e applets Java [6]. Para que se possa vincular estes arquivos externos em um documento é necessário que estejam vinculados a uma entidade externa, convenientemente declarada na DTD, por exemplo:

```
<!ENTITY fig.estrela SYSTEM "/Images/estrela.gif" NDATA GIF>
```

Desta maneira, entidades terão função fundamental em nosso sistema, pois é nesta parte da DTD que todos os arquivos pertencentes a uma página do curso on-line serão "apontados". Para enfatizar melhor o uso de entidades, veja a lista de possibilidades [6,7]:

- Representar caracteres que não são padrão no seu documento XML;
- Funcionar como abreviação para frases freqüentemente utilizadas;
- Manter partes de marcação que podem aparecer em mais de um documento XML;
- Manter seções ou capítulos de um amplo documento XML;
- Representar recursos que não são XML.

Folhas de Estilo (Style Sheets)

A noção de folhas de estilo é complementar a estrutura de documentos. Documentos contêm conteúdos e estrutura, e as folhas de estilo descrevem como os documentos devem ser apresentados. Esta apresentação é uma necessidade para tornar independente do dispositivo o conteúdo do documento, isto é, todas as características necessárias para o conteúdo ser apresentado de maneira correta em um dispositivo específico é informada na folha de estilo, pois isto simplifica o gerenciamento do documento, uma vez que um único documento pode esta associado a muitas folhas de estilo [11].

Por exemplo, se um documento XML utiliza elementos como: autor, nome e e-mail (fig.3), não haveria modo de dizer como este conteúdo deveria ser apresentado.

```
Markup:
    <autor>
      <nome>Irwing Cosow</nome>
      <email>cosow@w3.org</email>
    <autor>
```

Figura 3 - CSS em XML

CSS foi desenvolvido em 1994 pela CERN com a meta de criar uma linguagem de folha de estilo para a Web que fornecesse ao autor um maior controle de estilos sobre documentos HTML. Em 1996, CSS1 (o primeiro nível de CSS) tornou-se uma recomendação da W3C [11]. Em 1997, CSS1 passou a ser suportada pelos principais navegadores, incluindo o Netscape Navigator e o Microsoft Internet Explorer, bem como várias ferramentas de autoria [10].

CSS utiliza regras declarativas para anexar estilos aos documentos. No exemplo abaixo, uma simples regra informa que todos os elementos "P", da classe "definição" são exibidos com o texto em vermelho com um fundo branco:

```
P.Definição {
    color: red;
```

CSS1 suporta screen-based formating, incluindo fontes, cores, e layout [8,10]. Antes das folhas de estilo existirem, autores de conteúdo para a Web criavam imagens de texto para construir fontes (no caso de notação matemática) e cores convenientes. Isto resultou que a maior parte da largura de banda e usado não para texto e sim para elementos gráficos. Além do mais, folhas de estilo tem o potencial de aumentar significativamente a performance da rede de comunicação dados, como concluído por um recente estudo de como a tecnologia afeta a performance da rede [10,12].

Utilizando-se folhas de estilo ao invés de imagens, permite-se acessibilidade, característica de grande valia em sistemas de ensino a distância. Como exemplos: um sintetizador de voz pode ler um texto codificado em HTML para um usuário cego; o texto pode também ser apresentado em algum dispositivo que permita a leitura em braille; notação matemática com fontes não-padrão podem ser visualizadas em diversos navegadores.

O próximo nível da CSS, a CSS2, surgiu em 1998 [9] e, de acordo com a W3C, fortalece a acessibilidade a Web através da adição de conceitos de multimídia em folhas de estilo. Por exemplo, uma folha de estilo pode aplicar no documento, caso o dispositivo suportar, recursos sonoros.

Folhas de Estilo Avançadas

A Extensible Stylesheet Language (XSL), que está sendo definida pela W3C, oferece um passo

adiante neste conceito, pelo fato desta estar apta a transformar a estrutura do documento. Por exemplo, uma folha de estilo XSL pode automaticamente gerar uma tabela de conteúdos (ou índice automático) através da extração dos títulos dos capítulos de um documento. O uso de XSL para transformar os dados estruturados em XML, traz grandes benefícios para a área de publicação de documentos nos mais variados formatos.

Muitas páginas Web misturam dados declarativos (como HTML, XML e CSS) com programas executáveis (como scripts e applets), pois desenvolvedores de conteúdo são motivados a usá-los para permitir efeitos especiais de apresentação (como menus especiais pop-up). O mais importante é o fato de que até agora não foi levado em consideração pelos desenvolvedores os custos e benefícios antes confiados aos scripts e applets para mostrar a informação. Os custos incluem [10] :

- **Acessibilidade:** conteúdo misturado com programa, torna este escondido aos sites de busca (search engines), e torna difícil se não impossível, converter este conteúdo em outro formato.
- **Manteneabilidade:** dados declarativos são mais fáceis de manter sua longevidade comparados aos programas.
- **Independência do equipamento:** muitos scripts são incompatíveis entre navegadores.

Com o desenvolvimento das folhas de estilo, espera-se que os efeitos mais comuns de apresentação estejam fazendo parte de regras declaradas no estilo. Por exemplo, a CSS2 inclui a funcionalidade de alteração na cor de um elemento ao passar do cursor do mouse, tal efeito somente era possível através da utilização de scripts.

O mais importante é que o padrão XML é independente do estilo, o que permite que para cada documento seja aplicado os mais variados formatos de visualização, podendo ser aplicado diferentes folhas de estilo para produzir documentos de saída diversos [13].

Comparação entre CSS e XSL

XSL é freqüentemente comparada a CSS como uma maneira de aplicar diferentes formatos para as marcações XML. Entretanto esta comparação é um pequeno engano. CSS lê cada elemento XML como se este fosse "escaneado" (fig.4) no documento e aplica estilos em ordem. Em outras palavras, CSS não altera a estrutura do documento XML, ela somente muda a aparência visual para cada elemento. Se for colocado um nome no topo de um documento XML, CSS irá colocar este nome no mesmo local, ao menos que seja explicitado outra posição absoluta. Além do mais, CSS irá tratar cada marcação de um dado tipo exatamente da mesma maneira.

Do ponto de vista dos desenvolvedores, pode-se alcançar uma maior flexibilidade usando uma combinação das três tecnologias: XML contém os dados, CSS (na forma interna ou externa) provê o estilo ao documento (manipulação da apresentação), enquanto o XSL é usado para modificar a estrutura do documento. Através da separação destes pedaços, o controle sobre alteração dos dados começa a ser total e os benefícios são significativos [13].

Para concluir, CSS não pode filtrar, reordenar dados, adicionar texto ou subordinar a estruturas HTML. Alguns destes problemas, podem ser resolvidos através da utilização de ambientes DHTML (Dynamic HTML), mas esta tecnologia é caracterizada por implementações proprietárias.

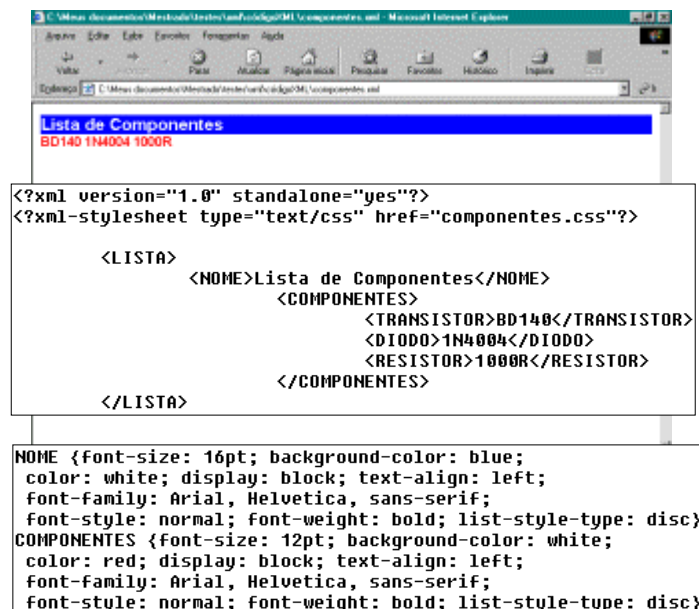


Figura 4 - Utilização do XML com CSS

XSL permite solução complementar para a formatação de XML. Diferente de CSS, o qual aplica informação de estilo para cada nodo XML que for encontrado na seqüência, XSL efetivamente repõe uma informação com a outra, independentemente da ordem que a informação encontra-se estruturada. Outras características de grande importância é a transformação que o XSL pode fazer em um documento XML tornando-o em outro documento de estrutura diferente como XML, HTML, texto, SQL [13,14]. Diferentemente de outra linguagem de transformação, XSL possui o benefício de ser escrita em XML, o que significa que o mesmo parser que pode manipular dados XML pode também manipular XSL.

XSL consiste em uma série de templates que podem ser usados para encontrar informações em um documento. Estas templates aplicam padrões ao fluxo de informações de entrada que as transformam em um outro fluxo de saída, o qual pode conter código HTML.

XSL possui um número de ferramentas para fazer comparações condicionais, ordenamento e executar operações em grupo. Assim sendo, fica justificado que a saída do processamento XSL, está de longe, desvinculada da seqüência em que as marcações aparecem no documento XML. Por exemplo, consideraremos o seguinte código XSL que manipula a criação de um "nome":

```
<xsl: template match="nome">
  <h1><xsl:value-of/></h1>
```

Esta simples template XSL irá chamar em qualquer momento o processador XSL a encontrar a marcação "<nome>" (neste caso somente será um). Quando houver a ocorrência, o parser XSL irá, de

acordo com o texto da marcação "<xsl:value-of/>", colocá-lo entre duas marcações "<h1>" na saída da informação.

O exemplo final (fig.5, 6 e 7), um pouco mais completo, ilustra com listagens dos códigos XML/XSL, exemplificando como ficaria a saída para o documento, através da aplicação da folha de estilo em questão. Pode-se ainda combinar XML/XSL com JavaScript para criar complexas aplicações [13,14].

```
<?xml version="1.0"?>
<?xml:stylesheet type="text/xsl" href="dados.xsl"?>
<ender ano="2000">
  <nome>Cesar C. Machado</nome>
  <residencial>
    <rua>Vila Real, 1160</rua>
    <cep>96083-370</cep>
    <tel>278.8704</tel>
    <estado>RS</estado>
```

Figura 5 - Arquivo fonte XML

```
<xsl:template match="/">
  <HTML xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <HEAD>
  <TITLE>
  <xsl:for-each select="ender">
  <xsl:value-of select="@ano"/>
  </xsl:for-each>
  - Dados pessoais atualizados
  </TITLE>
  </HEAD>
  <BODY>
  <xsl:for-each select="ender">
  <H1>
  <xsl:value-of select="@ano"/>
  Dados pessoais atualizados
  </H1>
  <H2>
  <xsl:value-of select="nome"/>
```

Figura 6 - Arquivo fonte XSL

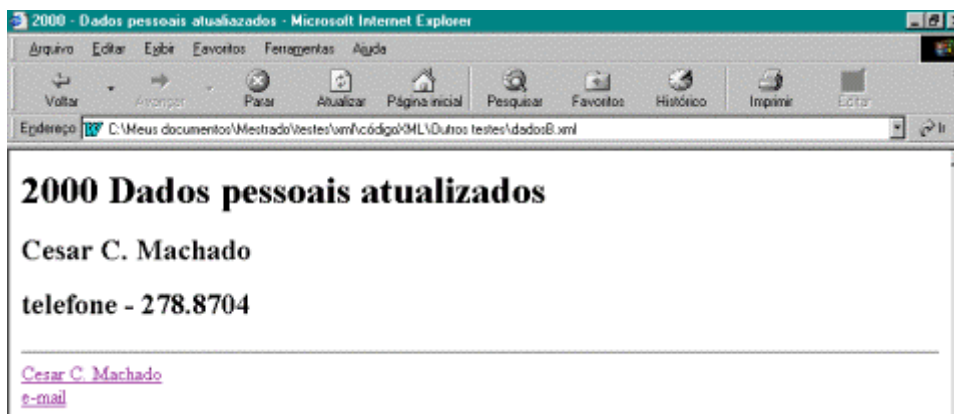


Figura 7 - Resultado da visualização em navegador

CONCLUSÕES

Este artigo apresentou as principais características da linguagem de marcação de documentos estruturados na Web, a eXtensible Markup Language (XML), a qual oferece significativos avanços em termos de disponibilização e visualização de documentos e a sua utilização como extensão ao sistema Hyper-Automaton.

Todas as características tratadas neste trabalho, fazem parte da implementação para a inclusão da tecnologia XML, pois como se pode observar há o envolvimento de vários módulos com diferentes implementações, isto é, estrutura dos documentos em marcações apropriadas, elaboração correta da DTD e folhas de estilos XSL.

O núcleo central deste trabalho compreende uma DTD que funcionará basicamente com ponteiros a fragmentos de documentos existentes na hiperbase do curso na Web, os quais são alvo de folhas de estilo XSL, que de posse de uma DTD interna referenciada a este núcleo, terá então a capacidade de formatação visual dos aspectos necessários para a visualização dos conteúdos no navegador do usuário. O projeto neste estágio, encontra-se em fase de estudos para a construção adequada desta DTD.

As características básicas do XML e XSL identificadas neste artigo, como a estrutura clara e de fácil depuração e processamento, a capacidade de identificação de dados, o poder de estilização de conteúdos, a reusabilidade de definições e a manipulação de links, trazem benefícios adicionais ao sistema Hyper-Automaton, entre os quais: possibilidade de implementação de ferramentas de busca sobre hiperdocumentos; maior estruturação das informações disponibilizadas em hiperdocumentos e facilidades para a manipulação e gerenciamento dos dados; flexibilização da função de saída; múltiplas formatações de hiperdocumentos para um mesmo conteúdo; capacidade do usuário em alterar a formatação de saída dos hiperdocumentos de acordo com preferências pessoais; flexibilização do processamento de hiperdocumentos em operações de composição; capacidade de formatação de caracteres matemáticos.

BIBLIOGRAFIA

- [1] Machado, Júlio P. Hyper-Automaton: Hipertextos e Cursos na Web Usando Autômatos Finitos com Saída. Porto Alegre: PPGC da UFRGS, 2000. (Dissertação de mestrado).
- [2] Menezes, P. Blauth; Machado, Júlio P. Web Courses Are Automata: a categorial framework. Anais do II Workshop de Métodos Formais, 1999, Florianópolis. Florianópolis: UFSC, Instituto de Informática da UFRGS, 1999. p.79-88.
- [3] Menezes, P. Blauth. Linguagens Formais e Autômatos. Terceira Edição. Editora Sagra-Luzzatto, 2000.
- [4] Machado, Júlio P.; P. Blauth Menezes. Sistemas de Gerenciamento para o Ensino a Distância. Anais do V Congresso Internacional de Educação a Distância. São Paulo: ABED, 1998. <http://www.abed.org.br>
- [5] Halasz, F.; Schwartz, M. The Dexter HyperText Reference Model. Communications of the ACM, 1994, v.37, n.2, p.30-39.
- [6] Light, Richard. Iniciando em XML. Makron Books do Brasil, 1999.

- [7] Harold, Elliotte R. XML Bible. IDG Books Worldwide, 1999.
- [8] Lie, Håkon W.; Bos, B. Cascading Style Sheets, Level1 W3C Recommendation. 17 dez. 1996, revisão 11 jan. 1999. <http://www.w3.org/TR/REC-CSS1>
- [9] Bos, B; Lie, Håkon W.; Liley, C; Jacobs, I. Cascading Style Sheets, Level 2 Specification. Maio 1998. <http://www.w3.org/TR/REC-CSS2>
- [10] Lie, Håkon W.; Saarela, Janne. Multipurpose Web Publishing Using HTML, XML and CSS. Communications of the ACM, 1999,v.42, n.10.
- [11] World Wide Web Consortium (W3C). Setembro 1999. <http://www.w3.org>
- [12] Nielsen, H.; Gettys, J.; Baird-Smith, A.; Prud'Hommeaux, E.; Lie, H.; Lilley, C. Network Performance Effects of HTTP/1.1, CSS1, and PNG. Proceedings of ACM SIGCOMM'97, 1997.
- [13] Cagle, Kurt. Transform Your Data With XSL. 1999. <http://www.xmlmag.com/upload/free/features/xml/1999/01/01win99/kc2win99/kcswin99.asp>
- [14] Moulitis, Natanya P.; Kirk, Cheryl. XML Black Book. 1999.
- [15] Ambler, Scott W. Mapping Objects to Relational Databases. Julho 2000. <http://www.AmbySoft.com/mappingObjects.pdf>
- [16] Lermen, Alessandra de Lucena. A Mapping Framework from ODMG to Object/Relational DBMS. Porto Alegre: PPGC da UFRGS, 1999.
- [17] Silberschatz, Abraham et al. Database System Concepts. 3th Ed. McGraw-Hill, 1997.
- [18] Machado, Júlio P.; Morais, Carlos T.Q. de; Menezes, P. Blauth; Reis, Ricardo A.L. Structuring Web Course Pages as Automata: revising concepts. Proceedings of RIAO'2000 Recherche d'Informations Assistee par Ordinateur, 2000, Paris. Paris: Centre de Hautes Etudes Internationales d'Informatique Documentaires, Center for the Advanced Study of Information Systems, 2000, v.1, p.150-159.

Anexo 3 Artigo DEXA'2001

Este artigo [MAC01a] foi submetido e publicado na categoria de artigo completo no *Fifth International Query Processing and Multimedia Issues in Distributed Systems Workshop QPMIDS'2001* em conjunto com a *12th International Conference on Data and Expert Systems Application DEXA'01* e *IEEE Computer Society*, ocorrido de 3 a 4 de Setembro de 2001 na cidade de Munich na Alemanha.

Este workshop visa reunir pesquisadores, desenvolvedores, usuário e demais interessados na área de Processamento de Consultas e Gerenciamento de Multimídia em Sistemas Distribuídos. Avanços recentes na área do gerenciamento de multimídia tornam o uso de processamento de consultas e recuperação do conteúdo multimídia mais crítica em sistemas distribuídos. Efetivos e eficientes métodos de gerenciamento de processamento de consultas sobre dados multimídia (imagem, vídeo, som e texto) são necessários para habilitar de maneira rápida e com alta qualidade o acesso a estes tipos de dados para bancos de dados distribuídos.

Nessa direção, este artigo descreve de maneira prática e com exemplos previamente testados com forte ênfase na construção de regras corretas e rígidas que determinarão o nível de flexibilidade sobre estruturas XML armazenadas pertencentes a cursos na *Web* com conteúdo multimídia. Tal desenvolvimento traz uma análise de como estas regras poderão ser aplicadas no Sistema Hyper-Automaton para que tenhamos um formalismo correto na estruturação de documentos XML sempre obedecendo o conceito de documentos válidos e bem formados, pois a padronização e não ocorrência de erros de processamento são fundamentais.

Definition and Application of Rules for the Adequate Designing of XML Documents for the Hyper-Automaton System

César Costa Machado, Gustavo L. Federizzi, P. Blauth Menezes

cmachado@atlas.ucpel.tche.br

glf.ez@terra.com.br

blauth@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul

Av. Bento Gonçalves, 9500 - Bloco IV.

Caixa Postal: 15064 CEP 91501-970.

Porto Alegre – RS

ABSTRACT

This work continues previous studies seeking for offering a form of structuring and visualization of the content of courses available on the WWW from the usage of formal constructions known as Deterministic Finite Automata with Output. The study based on the Internet technological evolution suggests that the nowadays system incorporates the benefits of the modern document structuring, along with the necessities and technological benefits, as well as, uses important resources arose from the markup languages which refer to the possibilities of formatting of the visually available content. The main goal of this work is to schematize and describe the rules according to a bibliographically supported and applied view. Such rules which, in a general meaning, are optionally part of a XML document, in the Hyper-Automaton System which we developed, will be undoubtedly indispensable and permanently integrating elements.

Keywords: WWW, XML, DTD, distance education, remote courses, finite automata, course modeling.

1. INTRODUCTION

This work continues the Web Courses Modeling Using Formal Systems [1,2] work and has as goal to provide alternatives regarding the formatting of the user interface using XML and its applications.

Web Courses Using Formal Systems use formal constructions known as Deterministic Finite Automata [3]. The group introduced the concept of “courses are automata” as a structure that allows easy implementation, creation of hypermedia material independent from automata, along with the encouragement of reutilization of Web pages on several courses, which can be built in different views, reducing the redundancy during the creation of the pages [1,2,4].

The general goal of the Hyper-Automaton model is focused on the study of the application of the formalism of Finite Automata with Output (Mealy Machine and Moore Machine) as a structural model for the arrangement of instructional hyperdocuments, specially Web Courses. The model is inspired by classical researches in the area of hyperdocuments and recent initiatives on the WWW, with special attention to the development of hypertext systems where the hyperdocuments basis is designed independently of the hypermedia application control structure, and supports some facilities as the composing of hierarchical structures, specifications of several link groups over a common hyperdocument body and objects apart from the navigation structure.

Each automata defines a course and is composed of a group of independent hyperdocuments, which can be part of other courses. The transition function works as a logical link between the hyperdocuments and the exit function composes the pages. The final result is the basic structure of hypertext links and pages on a Web site. The model leads to a high level of modularization of the instructional material, presenting the following advantages: facility on the reutilization of pages in several courses, exhausting the redundancies; hyperdocument independency from the

automata structure, that, on the presence of any change, does not influence the pages and vice-versa; allows that any user to create links from and to any document; facility of implementation and maintenance; direct and simple graphic interface; elaboration of instructional sequences with specific focus and capable of offering individualized study; categorical operations provide a course composition scheme which allows the construction of new courses over already existing ones, through high level procedures [1, 2].

The Hyper-Automaton system constitutes of a semi-automated system to support Web courses (based on a client/server architecture and HTML developed interface) through the application and concepts inherent in Computer Science, specially Automata Theory, Hypermedia technology and Category Theory, uniting the benefits of them. Although the used emphasis to the validation of the model is the implementation of systems for distance learning, the obtained results are applied to hypertext systems as a whole.

Our work on the actual stage is focused on the several possibilities of formatting of the available content, culminating in a adequate user interface, where, the elasticity of the content formatting obtained from the exit function of the automata will be an application which remounts the document on the desired parts and with the most adequate characteristics related to the user needs and the course itself. The proposed study will bring a bigger flexibility to the already existing one, since nowadays, the presented one is a rigid and lone possibility formed of fragments of HTML.

Due to considerations that explicitly exclude any commercial connotation, the works developed here were evaluated using Internet Explorer 5.5, since it is the only internet browser, from the version 4 on, which supports standard XML resources.

2. SYSTEM VIEW

The block diagram illustrates the parts of the system, figure 1, includes what we are developing nowadays. The new aspect is relative to structuring the contents written in XML according to the previous established definitions for documents (DTD) [5, 6] and the usage of XSL templates and/or cascade style sheets (CSS) [7, 8] with the objective of visualization and lay-out of the referenced on-line course.

These altering will have as goal mainly the technological updating of the system according to the benefits already written on previous works related to XML [9, 10].

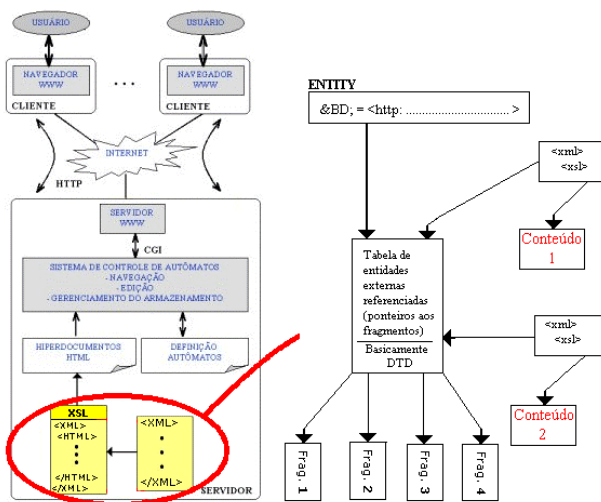


Figure 1 - General Scheme

From this moment on this work will describe one of the most important features of XML regarding the establishment of construction rules. Known as Data Type Definition (DTD), these rules are intimately connected to our objectives, for they establish criteria of elaboration and linkage to non-XML or other types of files which usually form normally exhibited documents on the Web. These files on our system will be referred to as fragments.

Downplayed on the following figure, these characteristics undoubtedly will be part and will be widely used on the new proposal of implementation over the Hyper-Automaton System.

3. DTD (Data Type Definition)

According to [6], XML is described as a *meta-markup language*, a language elaborated to describe markup languages, as for instance: MathML, VML, among others. This way the DTD is the rules establishment place which the application will follow to.

DTD is the center of the scheme on figure 1. It has as function to indicate the rule that the XML document is following. More specifically, the DTD relates a list of elements, attributes, notations, and entities contained on a document, as well as the relations among them. The DTD specifies a set of structural rules of a document. For instance, a DTD determines that a *page* document (of a course, for instance) has exactly one son *title*, one or more son *authors* and must not contain a *subtitle*.

DTDs can be included on the file that contains the description of the document or linked to an external URL.

External DTDs can be shared by different documents and Web sites. External DTDs determine that documents elaborated in any time and place, obey the rules which they objective to establish structure, clearness, and legibility patterns.

A practical example that can be cited is the fact that several people can be together developing documents related to a common Web course. They may be geographically separated, working individually in distinct chapters, but by the fact they are inserting contents that obey the structural rules specified in a DTD, at the moment that the document union occurs as a whole, there will be a structural pattern of the referred course.

Thus, a DTD describes how the different elements of a page are arranged without direct manipulation of these data. The DTD allows that the structure of the document is visualized separated from the data. This means that independently of the most exaggerated forms of styles applied to the related document, the document will always be immune, apart from the altering that it may come to suffer, in other words, painting a house without changing the basic structure. The DTD is on a level that the user does not perceive, but where applications as JavaScripts, CGI, data bank and other softwares can use it [6].

The XML documents may reference non-XML resources that are outside the document, as for instance: image files, video clips, audio clips, text processor files, and Java applets [5]. In order to attach these external files to a document it is necessary that they are attached to an external entity, conveniently declared on the DTD.

This way entities will have a primordial function on our system, for it is on this part of the DTD that all the belonging files will be "pointed". In order to better emphasize the usage of entities, see the possibilities list [5]:

- To represent characters that are not standard on its XML document;
- To function as a abbreviation for frequently used sentences;
- To keep markup parts that may appear on more than one XML document;
- To keep sections or chapters of a wide XML document;
- To organize its DTD into logical units;
- To represent resources that are not XML.

4.11 DOCUMENT TYPE DECLARATIONS

The document type declaration specifies the DTD which is used by the document and it appears on the prologue of a document, after the XML declaration, but before the element root. This must contain the definition of the type of document or the URL that identifies the file where the definition of the type of document is located. In our case we will use two DTDs, normally referred to as internal and external DTD. The internal will establish structural rules to the construction of the document. This one is not obligatory. The external will work as a pointer to the fragments that are part of the composition of the pages of the stored course.

4.11 ELEMENT DECLARATIONS

Each tag used on a valid XML document must be declared as a element on the DTD. The declaration of a element defines the name and the possible content of the element. The list of contents is sometimes called of content specification [6]. This specification uses a simple grammar to precisely specify what is allowed and what is not in a document. The following figure shows the amount of occurrences of an element on a document:

Symbol	<u>NAME</u>	Meaning
?	Interrogation	Optional (zero or one)
*	Asterisk	Zero or more
+	Plus	One or more

Observe the following example, a few more rules can be used in our specifications:

```
<?xml version="1.0" standalone="yes">
<?xml-stylesheet type="text/css" href="standard.css"?>
<!DOCTYPE fragmentoum [
  <!ELEMENT FRAGMENTO (INDENTFRAG, TITULO, CONTEUDO, IMAGEM*, SOM*)>
  <!ELEMENT INDENTFRAG (#PCDATA)>
  <!ELEMENT TITULO (#PCDATA)>
  <!ELEMENT CONTEUDO (#PCDATA)>
  <!ENTITY IMAGEM SYSTEM "logo.gif" NDATA GIF>
  <!ENTITY SOM SYSTEM "regra.mp3" NDATA MP3>
] >
```

It can be noticed on the above list that we are denoting a fragment, for its simplicity, it was associated a cascade style sheet [11, 12]. Our idea is to apply the style sheet on the way out of the page from the document to the user, aiming to stretch the most the formatting of the relevant aspects to the wishes of whoever makes use of the referred course.

On the fourth line an important restriction that relates the sequence on which the items that form a fragment should be composed is imposed, that is: INDENTFRAG, TITULO e CONTEUDO; must exist only once, IMAGEM* e SOM*, zero or more times on a document.

The elements INDENTFRAG, TITULO e CONTEUDO, are made of conventional characters except markup, that is why the definition #PCDATA.

Finally the elements IMAGEM and SOM, binary entities, which require a special treatment on the specification and whose occurrence is not frequent, will be completely disengaged and pointed by the documents that include them. The fact of being connected and not included on the XML document reinforces the idea of maximum flexibility. This is a little disadvantage of XML in relation to HTML, for on the second language, the insertion of sounds and images is a little more direct.

4.11 ENTITIES

A XML document must contain data and declaration of many different sources and files. This way the data must be directed into a data bank, CGI scripts, or even a different data source. The items where the pieces of a XML are stored are called entities. The reference to these entities has as goal to load them into the main XML document.

The storage unit that contains the XML declarations, the DTD, and the element root is called *entity document*. However, the root element and its descendants can also contain referenced entities pointing to additional data that must be included on the document [6, 13].

There are two types of entities: internal and external. The internal entities are defined totally inside the entity document. The external entities relate their contents to a different source located by a URL. The main document includes only the reference to the URL where the content resides.

A practical usage of internal entity is when there is excessive repetition of a part of text in several places on the document. Another advantage is the possibility of easy updating of the referenced contents. Observe the following example:

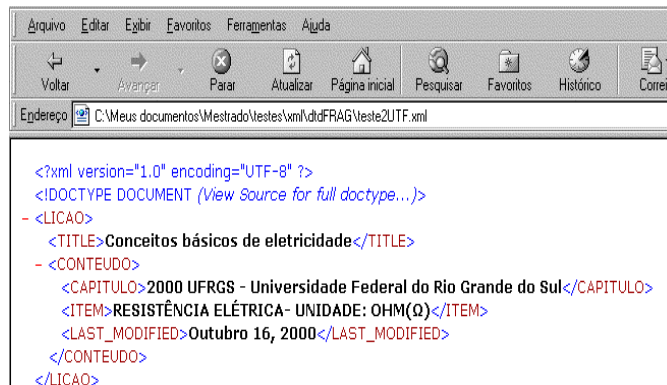
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DOCUMENT [
  <!ENTITY omega "&#937;">
  <!ENTITY aacute "&#225;">
  <!ENTITY Eacute "&#201;">
  <!ENTITY Ecirc "&#202;">
  <!ENTITY UFRGS "UFRGS - Universidade Federal do Rio Grande do Sul">
  <!ENTITY IN "Conceitos b&aacute;sicos de eletricidade">

  <!ELEMENT DOCUMENT (TITLE, CONTEUDO)>
  <!ELEMENT CONTEUDO (TITULO, ITEM, LAST_MODIFIED)>
  <!ELEMENT TITLE (#PCDATA)>
  <!ELEMENT CAPITULO (#PCDATA)>
  <!ELEMENT ITEM (#PCDATA)>
  <!ELEMENT LAST_MODIFIED (#PCDATA)>
] >
<LICAO>
  <TITLE>&IN;</TITLE>
  <CONTEUDO>
    <CAPITULO>2000 &UFRGS;</CAPITULO>
    <ITEM>RESIST&ecirc;NCIA EL&Eacute;TRICA - UNIDADE: OHM(&omega;)</ITEM>
```

```

<LAST_MODIFIED>Outubro 16, 2000</LAST_MODIFIED>
</CONTEUDO>
</LICAO>

```



Observing the example above, we can notice that there was no concern about the visual formatting yet, it was not applied in this code any style sheet.

Another important aspect is the definition of the pattern of character encoding. For this example it was chosen UTF-8, supporting besides the ASCII pattern another series of characters, making possible the correct accentuation for the grammar surveillance.

4.11 EXTERNAL GENERAL ENTITIES

External entities are data located outside the main file which contain the root element / entity document. External entities allow the inclusion of these external (entities) files making possible that a XML document is consisted of several other independent files [5, 6, 13].

Documents that use only internal entities, clearly remount the HTML model. The complete text document is made available in one file only. Images, applets, sounds, and other non-HTML data must be linked, however the entire text is present. Naturally the HTML model presents some problems, such as the difficulty of including dynamic information on the file, possible only through CGI, Java applets, data bank software, etc. The HTML model allows a static document only. A solution for this problem would be the usage of frames, but they behave in an irregular and confuse way for most users [6].

Part of this problem is the fact that a HTML document does not fit into another in a natural manner. Every HTML document must have only one BODY, and no more. The servers enable HTML fragments only and never an entire valid document within another.

However, XML is more flexible. The root of a document is not necessarily the same or another. If two elements share the same root of a document, the DTD must declare which elements constitute it.

XML goes further beyond, it is allowed that any document is formed by multiple small pieces of XML documents located on the most distant places of the network. The parser is responsible by the correct mounting of these fragments. This way, documents can contain other documents, that can contain other documents. From the application point of view there is only one complete file. This process is taken care on the client side.

In order to make an external entity part of a document, we must declare it like this: `<!ENTITY name SYSTEM "URI">`, more precisely, `<!ENTITY CAPITULO SYSTEM "http://atlas.ucpel.tche.br/~cmachado/capituloum.xml">`. The inclusion is made similarly to the following example through the entity code `&CAPITULO;`.

4.11 NON-XML DATA INCLUSION

There is an infinity of types of data and most of them are not XML. Thus, we must provide mechanisms so that the most different data formats can be included normally in our applications.

According to [6], this theme is controversial, and so, it is still being discussed.

The XML allows three constructions generally used to function with these types of data, such as: *notations*, *unparsed external entities* e *processing instructions*.

Notations – The first problem when we use non-XML files, regards identifying and informing the XML application how to read and show the respective data to the user. Using HTML, for instance, at the moment the tag IMAGE is found, thus GIF or JPEG, naturally the internet browser knows how to handle it, but what to do when the format is

EPS or TIFF. Regarding XML, because of its characteristics of free tag creation, other important factors should be cited, for instance, a tag named CONTEUDO, can contain an image or a multimedia resource, as a sound file (MP3, WAV, or MIDI), this way one must provide the resources for the adequate processing and availability of these files.

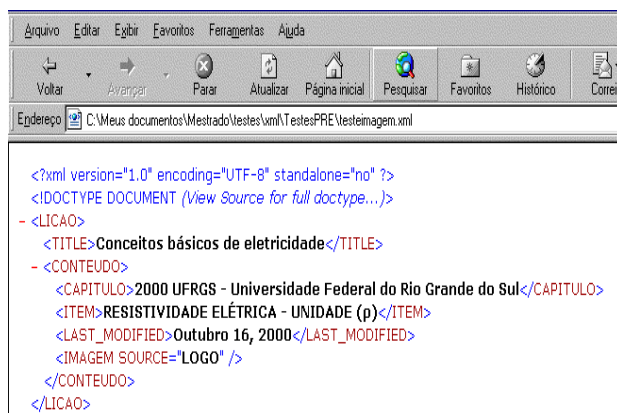
Unparsed external entities – XML is not the ideal format for every type of data, particularly non-text data. An interesting example, and not practical at all, would be to encode into XML a figure on the bitmap format storing every bit, observe the example: <PIXEL X="121" Y="68" COLOR="BABAFA">.

On a typical Web page, GIF, JPEG, Java Applets, ActiveX, several types of sounds and many other types of files are included. Using XML all these non-XML files are called *Unparsed entity*, because the file processor will not try to understand these files, it will only be aware of their existence. Observe the example below:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
  <!DOCTYPE DOCUMENT [
    <!ENTITY aacute "á">
    <!ENTITY Eacute "É">
    <!ENTITY Ecirc "Ê">
    <!ENTITY rho "ρ">
    <!ENTITY UFRGS "UFRGS - Universidade Federal do Rio Grande do Sul">
    <!ENTITY IN "Conceitos b&aacute;sicos de eletricidade">
    <!ENTITY LOGO SYSTEM "logo_tec.gif" NDATA GIF>
    <!NOTATION GIF SYSTEM "image/gif">
    <!ELEMENT DOCUMENT (TITLE, CONTEUDO)>
    <!ELEMENT CONTEUDO (TITULO, ITEM, LAST_MODIFIED, IMAGEM)>
    <!ELEMENT TITLE (#PCDATA)>
    <!ELEMENT CAPITULO (#PCDATA)>
    <!ELEMENT ITEM (#PCDATA)>
    <!ELEMENT LAST_MODIFIED (#PCDATA)>
    <!ELEMENT IMAGE EMPTY>
    <!ATTLIST IMAGE SOURCE ENTITY #REQUIRED> ]>
<LICAO>
  <TITLE>&IN;</TITLE>
  <CONTEUDO>
    <CAPITULO>2000 &UFRGS;</CAPITULO>
    <ITEM>RESISTIVIDADE EL&Eacute;TRICA - UNIDADE (&rho;)</ITEM>
    <LAST_MODIFIED>Outubro 16, 2000</LAST_MODIFIED>
    <IMAGEM SOURCE="LOGO" />
  </CONTEUDO>
</LICAO>
```

4.11 RECOGNIZING NON-XML FILES

Regarding structural terms we were able to make that one exit generated by a state or a transition of the Hyper-Automaton is aware that at this moment any non-XML resource, like images, sounds, or other type of file, may compose the document to be shown to the user. On the code list below it is defined, through the code of the example DTD, the details of the figure one schematized above.



Observe the example below because inside this one there is the DTD code of the external entities composed by de fragments that will compose the exit on a determined moment. It is of good will to reinforce that in this work the aspects of content visualization will not be approached, becoming subject for a next work.

nucleo.dtd

```

<!NOTATION JPEG SYSTEM "image/jpeg">
<!NOTATION GIF SYSTEM "image/gif">
<!NOTATION WAV SYSTEM "sound/wav">
<!NOTATION MPEG SYSTEM "audio/mpeg">
<!NOTATION MIDI SYSTEM "sequencia MIDI/midi">

<!ENTITY FRAG2 SYSTEM "frags/frag2.gif" NDATA GIF>
<!ENTITY FRAG3 SYSTEM "frags/frag3.mpeg" NDATA MPEG>
<!ENTITY FRAG4 SYSTEM "frags/frag4.gif" NDATA GIF>
<!ENTITY FRAG5 SYSTEM "frags/frag5.midi" NDATA MIDI>
<!ENTITY FRAG6 SYSTEM "frags/frag6.gif" NDATA GIF>
<!ENTITY FRAG7 SYSTEM "frags/frag7.jpeg" NDATA JPEG>
<!ENTITY FRAG8 SYSTEM "frags/frag8.wav" NDATA WAV>

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="estilo2.xsl"?>
<!DOCTYPE LICAO [
  <!ENTITY % NUCLEO_DTD SYSTEM "nucleo.dtd">
  %NUCLEO_DTD;
  <!ENTITY omega "&#937;";>
  <!ENTITY aacute "&#225;";>
  <!ENTITY Eacute "&#201;";>
  <!ENTITY Ecirc "&#202;";>
  <!ENTITY UFRGS "UFRGS - Universidade Federal do Rio Grande do Sul">
  <!ENTITY IN "Conceitos b&aacute;sicos de eletricidade">
  <!ELEMENT LICAO (TITLE, CONTEUDO)>
  <!ELEMENT CONTEUDO (CAPITULO, ITEM, IMAGEM, SOM, LAST_MODIFIED)>
  <!ELEMENT TITLE (#PCDATA)>
  <!ELEMENT CAPITULO (#PCDATA)>
  <!ELEMENT ITEM (#PCDATA)>
  <!ELEMENT LAST_MODIFIED (#PCDATA)>
  <!ELEMENT IMAGEM EMPTY>
  <!ELEMENT SOM EMPTY>
  <!ATTLIST IMAGEM FONTE ENTITY #REQUIRED>
  <!ATTLIST SOM FONTE ENTITY #REQUIRED>
]>
<LICAO>
  <TITLE>&IN;</TITLE>
  <CONTEUDO>
    <CAPITULO>2000 &UFRGS;</CAPITULO>
    <ITEM>RESIST&Ecirc;NCIA EL&Eacute;TRICA - UNIDADE: OHM(&omega;)</ITEM>
    <IMAGEM FONTE= "FRAG2" />
    <SOM FONTE= "FRAG3" />
    <LAST_MODIFIED>Outubro 16, 2000</LAST_MODIFIED>
  </CONTEUDO>
</LICAO>

```

```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE LICAO (View Source for full doctype...)>
- <LICAO>
  <TITLE>Conceitos básicos de eletricidade</TITLE>
  - <CONTEUDO>
    <CAPITULO>2000 UFRGS - Universidade Federal do Rio Grande do Sul</CAPITULO>
    <ITEM>RESISTÊNCIA ELÉTRICA - UNIDADE: OHM(Ω)</ITEM>
    <IMAGEM FONTE="FRAG2" />
    <SOM FONTE="FRAG3" />
    <SOM FONTE="FRAG8" />
    <SOM FONTE="FRAG5" />
    <LAST_MODIFIED>Outubro 16, 2000</LAST_MODIFIED>
  </CONTEUDO>
</LICAO>

```


4.11 CONCLUSIONS

The usage of rules to establish standards of XML document elaboration focused on teaching will allow a total flexibility to the adequate construction of instructional material by the most different people and in the most different subjects that may become available on the Web.

Although the usage of XML for the inclusion of non-XML is not so direct as the HTML, the external entity disposal greatly organizes and, above all, creates a standard for constructions according to the *World Wide Web Consortium*.

It is proper to say that the usage of external entities is still being developed, and maybe their usage will bring inadequate and unpredictable characteristics to the applications.

In our work we have limited up to the present moment, the usage of the most common file formats, as we can observe on the examples.

Due to the not regular behavior of the main internet browsers on the market nowadays, some tests may not become effective, for up to now, we have not studied the cause of possible *bugs* originated from the XML processing and its applications on this browsers.

From this point on, our research goes in a direction toward the necessary elements to allow the user enjoy the multimedia resources supported by XML and its applications. The studies will regard the stretching of the content visualization and the correct processing of the many types of files according to the specific goal, so being, allow that every supported multimedia application necessary to a Web course is available through your Web browser, since it supports the XML resources.

8. BIBLIOGRAPHICAL REFERENCES

- [1] Machado, Júlio Henrique Araújo Pereira. Hyper-Automaton: **Hipertextos e Cursos na Web Utilizando Autômatos Finitos com Saída**. Porto Alegre: CPGCC da UFRGS. 2000. (Dissertação de mestrado).
- [2] Menezes, Paulo Blauth, Machado, Júlio Henrique Araújo Pereira. **Web courses are automata: a categorial framework**. In: II WORKSHOP DE MÉTODOS FORMAIS, 1999, Florianópolis. II Workshop On Formal Methods. Florianópolis: UFSC, Instituto de Informática da UFRGS, 1999. p.79-88.
- [3] Paulo F. B. Menezes. **Linguagens Formais e Autômatos**. Segunda Edição. Editora Sagra-Luzzatto. Brasil, 1998.
- [4] Júlio P. Machado, Paulo F. B. Menezes. **Sistemas de Gerenciamento para o Ensino a Distância**. In Proceedings of V Congresso Internacional de Educação a Distância. São Paulo - Brasil, 1998. <http://www.abed.org.br>
- [5] Light, Richard. **Iniciando em XML**. Makron Books do Brasil Editora Ltda. 1999.
- [6] Harold, Elliotte, R. **XML Bible**. IDG Books Worldwide. Ago. 1999.
- [7] Lie, Håkon W, Bos, Ha B. **Cascading Style Sheets, level1** W3C Recommendation 17 Dec. 1996, revised 11 Jan. 1999. <http://www.w3.org/TR/REC-CSS1>
- [8] Bos, B; Lie, H.W.; Liley, C; Jacobs, I. **Cascading Style Sheets, level 2 (CSS2) Specification**. Maio 1998. <http://www.w3.org/TR/REC-CSS2>
- [9] Machado, César C.; Menezes, P.B. **Características e vantagens na estruturação de documentos em XML na WWW para EAD** In. I Workshop em Informática na Educação, Passo Fundo-RS, 2000
- [10] Machado, César C.; Machado, J.H.P.; Grandi, R.; Menezes, P.B.; **Utilização do XML no Sistema Hyper-Automaton** In. International Symposium on Knowledge Management / Document Management - ISKM/DM, Curitiba-PR,2000
- [11] Lie, Håkon Wiun; Saarela, Janne. **Multipurpose Web Publishing Using HTML, XML and CSS**. Communications of the ACM. Outubro 1999/Vol.42, nº.8
- [12] Nielsen, H.; Gettys, J.; Baird-Smith, A.; Prud'Hommeaux, E.; Lie, H.; e Lilley, C. **Network Performance Effects of HTTP/1.1, CSS1, and PNG**. IN Proceedings of ACM SIGCOMM'97 (Canes, France, 1997).
- [13] Moulitis, Natanya P., Kirk, Cheryl; **XML Black Book**. 1999
- [14] Adler, S; Berglund, A.; **Extensible Stylesheet Language (XSL) Versão 1.0** - W3C. Março 2000.
- [15] Cagle, Kurt; **Transform Your Data With XSL**. 1999.

Anexo 4 Artigo IC'2001

Este artigo [MAC01] foi submetido e publicado na categoria de artigo completo na *International Conference on Internet Computing IC'2001*, ocorrido de 25 a 28 de Junho 2001 na cidade de Las Vegas nos Estados Unidos.

Inúmeros fatores estão recentemente contribuindo para tornar o uso da Internet como um ambiente para a criação de programas. A Internet atualmente é mais explorada do que qualquer outro sistema de computação na história que continua a crescer rapidamente. Novas tecnologias, incluindo redes de comunicação de dados de alta velocidade e uso de programas adequados que delas façam o melhor uso, prometem torná-la de muito maior utilização como suporte à distribuição de software e outros propósitos. Essa conferência busca explorar as tecnologias envolvidas para possibilitar o avanço da Internet, bem como, o envolvimento de aplicações que fazem uso desta tecnologia. Este evento de conceito internacional torna-se um dos maiores fóruns para cientistas, engenheiros, e pesquisadores do mundo apresentarem seus últimos resultados, teorias, desenvolvimentos e aplicações.

Como este artigo foi reproduzido na íntegra no capítulo 5 desta dissertação em anexo consta o *draft paper* enviado para o evento.

Draft Paper

Keywords:

WWW, XML, XSL, Finite Automata, Hyper-documents Management, Web courses.

This work continues former studies reaching for a more intelligent and flexible way of displaying material from courses for the Internet that uses the formal constructions navigating system known as Deterministic Finite Automata with Output. The study based on the technologies evolution of the Internet suggests that the instructional material well structured in XML, used by today's systems, incorporates the benefits of the usage of cascade style sheets and advanced style sheet systems, ruled by important criteria of modern styling and formatting of the visually available content, along with the necessities and technological benefits of XML. The main goal of this work is to schematize and describe the rules according to a bibliographically supported and applied view. Such rules, which in a general meaning are optionally part of a XML document, in the Hyper-Automaton System, which we developed, will be undoubtedly indispensable and permanently integrating elements. This work continues the Web Courses Modeling Using Formal Systems work and has as goal to provide alternatives regarding the formatting of the user interface using XML and its applications.

Web Courses Using Formal Systems use formal constructions known as Deterministic Finite Automata. The group introduced the concept of "courses are automata" as a structure that allows easy implementation, creation of hypermedia material independent from automata, along with the encouragement of reutilization of Web pages on several courses, which can be built in different views, reducing the redundancy during the creation of the pages.

The general goal of the Hyper-Automaton model is focused on the study of the application of the formalism of Finite Automata with Output (Mealy Machine and Moore Machine) as a structural model for the arrangement of instructional hyper documents, especially Web Courses. The model is inspired by classical researches in the area of hyper documents and recent initiatives on the WWW, with special attention to the development of hypertext systems where the hyper documents basis is designed independently of the hypermedia application control structure, and supports some facilities as the composing of hierarchical structures, specifications of several link groups over a common hyper document body and objects apart from the navigation structure.

Each automaton defines a course and is composed of a group of independent hyper documents, which can be part of other courses. The transition function works as a logical link between the hyper documents and the exit function composes the pages. The final result is the basic structure of hypertext links and pages on a Web site. The model leads to a high level of modularization of the instructional material, presenting the following advantages: facility on the reutilization of pages in several courses, exhausting the redundancies; hyper document independency from the automata structure, that, on the presence of any change, does not influence the pages and vice-versa; allows that any user to create links from and to any document; facility of implementation and maintenance; direct and simple graphic interface; elaboration of instructional sequences with specific focus and capable of offering individualized study; categorical operations provide a course composition scheme which allows the construction of new courses over already existing ones, through high level procedures.

The Hyper-Automaton system constitutes of a semi-automated system to support Web courses (based on a client/server architecture and HTML developed interface) through the application and concepts inherent in Computer Science, especially Automata Theory, Hypermedia technology and Category Theory, uniting the benefits of them. Although the used emphasis to the validation of the model is the implementation of systems for distance learning, the obtained results are applied to hypertext systems as a whole.

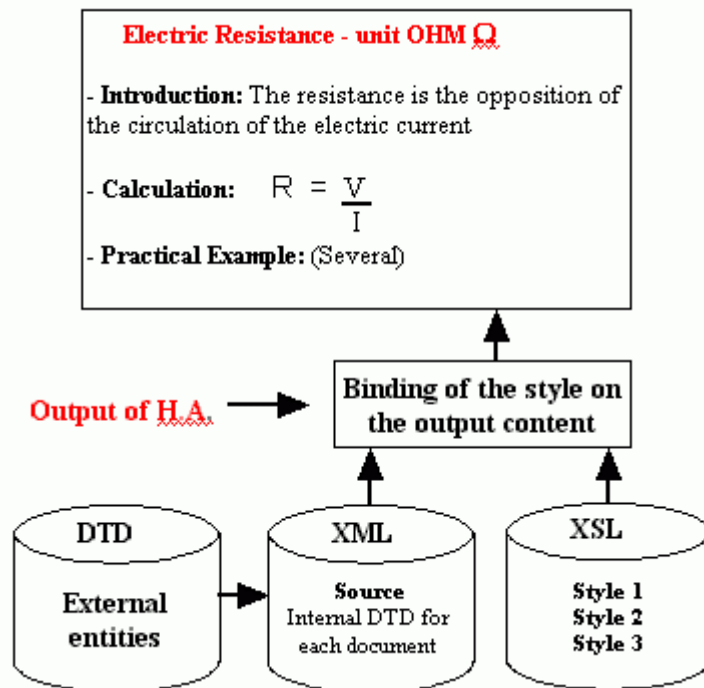


Figure1: Dynamic Binding of Style

Our work on the actual stage is focused on the several possibilities of formatting of the available content, culminating in a adequate user interface, where, the elasticity of the content formatting obtained from the exit function of the automata will be an application which remounts the document on the desired parts and with the most adequate characteristics related to the user needs and the course itself. The proposed study will bring a bigger flexibility to the already existing one, since nowadays, the presented one is a rigid and lone possibility formed of fragments of HTML.

The diagram detailed in figure 1 illustrates the system's parts and gives us a good idea of what is being developed, working on a adequate, correct and standard way of displaying well structured XML, well formed or valid against a DTD. The study of the rules for the Hyper-Automaton System was the main issue discussed in former works.

These changes will mainly objectify flexibly of the output function of the Hyper-Automaton through the application of multiple styles on the material displayed, as well as the technological update of the system in accordance with the benefits described in previous works related to XML. Through the use of XSL templates and/or Cascade Style Sheets (CSS), we will obtain the forms of visualization and layout of the on-line courses.

This work will describe one of the most important features of XML related to the study of the possible ways of visualization of content previously structured obeying the norms of standardization established by the *World Wide Web Consortium (W3C)*.

Known as Cascade Style Sheets, these applications are closely linked to our objectives, because they establish forms for the document manipulation, either for simple insertion of criteria of direct formatting or for the hashing of the tree of document XML.

The Advanced Style Sheets (XSL) make possible multiple ways of visualization of the same content, through the commands of formatting and XPath expressions, resulting in the adequate visualization for WWW browsers that support XML and not-XML documents that commonly composes web documents.

Detailed in the figure above, these features certainly will be part and will be used widely in new the proposal of implementation on the Hyper-Automaton system.

In the considered system, the style sheet will be annexed to the archive source XML on the fly, that is, the user will visualize the content - output function of the Hyper Automaton system - through its choice and based in criteria, such as the size of source, colors, type of content being displayed, etc.

Bibliographical references

- Machado, Júlio Henrique Araújo Pereira. Hyper-Automaton: **Hipertextos e Cursos na Web Utilizando Autômatos Finitos com Saída**. Porto Alegre: CPGCC da UFRGS. 2000.
- Menezes, Paulo Blauth, Machado, Júlio Henrique Araújo Pereira. **Web courses are automata: a categorial framework**. In: II WORKSHOP DE MÉTODOS FORMAIS, 1999, Florianópolis. II Workshop On Formal Methods. Florianópolis: UFSC, Instituto de Informática da UFRGS, 1999. p.79-88.
- Harold, Elliotte, R. **XML Bible**. IDG Books Worldwide. Ago. 1999.
- Adler, S; Berglund, A.; **Extensible Stylesheet Language (XSL) Versão 1.0** - W3C. Março 2000.
- Bos, B; Lie, H.W.; Liley, C; Jacobs, I. **Cascading Style Sheets, level 2 (CSS2) Specification**. Maio 1998. <http://www.w3.org/TR/REC-CSS2>
- Machado, César C.; Menezes, P.B. **Características e vantagens na estruturação de documentos em XML na WWW para EAD** In. I Workshop em Informática na Educação, Passo Fundo-RS, 2000
- Machado, César C.; Machado, J.H.P.; Grandi, R.; Menezes, P.B.; **Utilização do XML no Sistema Hyper-Automaton** In. International Symposium on Knowledge Management / Document Management - ISKM/DM, Curitiba-PR,2000
- Machado, César C., Federizzi, G. L., Menezes, P. B. **Definition and Application of Rules for the Adequate Designing of XML Documents for the Hyper-Automaton System**. In: The 5th World Multiconference on Systemics, Cybernetics and Informatics, 2001, Orlando. SCI 2001 / ISAS 2001 Proceedings. 2001.
- Bradley, Neil; **The XSL Companion**. 2000.

Anexo 5 Resumo ISKM'2002

Este artigo foi submetido ao *International Symposium on Knowledge Management* – Simpósio Internacional da Gestão do Conhecimento, promovido pela Pontifícia Universidade Católica de Curitiba – PUC PR e Centro Internacional de Tecnologia de Software – CITS, que ocorrerá de 19 a 21 de agosto de 2002 na cidade de Curitiba.

Em sua quinta edição, o ISKM consolida-se como o evento de referência destas áreas, atraindo contribuições de alto nível dos mais diversos setores.

O artigo descreve a parte final da pesquisa de reestruturação do Sistema Hyper-Automaton que na sua parte inicial previa somente aspectos a serem estudados e implementados visando a flexibilização do sistema para uma melhora dos aspectos visualização do seu conteúdo. Na verdade, o que acabou acarretando, foi a inevitável reestruturação completa de sistema, dando-lhe grande capacidade, atualizando-o tecnologicamente e aumentando as perspectivas de trabalhos futuros para a sua própria melhoria.

Na descrição dos aspectos de implementação acabamos por abordar outros assuntos que antes não foram previstos, como: XMLSchema, Java, Java Server Pages, Servlets e objetos de ensino.

Tal trabalho fecha de maneira prática esta fase de estudo através de uma implementação, embora em fase inicial, um protótipo do novo sistema.

Até o presente momento temos a aprovação em primeira fase deste trabalho, ou seja, aceitação do resumo re-enquadrado na área temática de Tecnologia da Informação como Suporte a Gestão do Conhecimento.

Implementando o Novo Sistema Hyper-Automaton

Gustavo L. Federizzi¹, César C. Machado^{1,2}, Paulo B. Menezes¹

¹Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
– Porto Alegre – RS – Brasil

²Centro Federal de Educação Tecnológica de Pelotas (CEFET-RS)
- Pelotas – RS – Brasil

glf.ez@terra.com.br, machado@cefetr.rs.tche.br, blauth@inf.ufrgs.br

Resumo. *Este trabalho finaliza a primeira grande parte do estudo de uma reestruturação plena no Sistema Hyper-Automaton, nome dado ao gerenciador de Hipertextos desenvolvido pelo grupo de métodos formais e informática teórica da Universidade Federal do Rio Grande do Sul - Brasil. Tal implemento amplamente utilizado como processador de hipertextos instrucionais está baseado no formalismo dos autômatos finitos com saída sendo utilizado para o auxílio as aulas servindo como importante ferramenta de apoio ao ensino. Esta pesquisa remonta totalmente a máquina de estados que até então gerenciava rígidos fragmentos HTML para uma versão atual, fazendo uso da tecnologia XML e Java que são modernas e poderosas aplicações que vão de encontro aos nossos interesses onde é buscado a flexibilização total do material disponibilizado, suporte a recursos de links não convencionais, renderização perfeita de ampla gama de caracteres e fórmulas matemáticas, suporte a arquivos multimídia e facilidades futuras de edição de conteúdo, estilos e regras.*

Palavras-chave: WWW, XML, Java, Objetos de ensino, Gerenciamento de hipertexto.

1. Introdução

Este artigo da prosseguimento ao trabalho Modelagem de Cursos na Web Utilizando Sistemas Formais [MAJ00], [MEN99] e tem como objetivo propor alternativas com respeito à formatação da interface com o usuário através da utilização do XML e de suas aplicações.

Cursos na Web Utilizando Sistemas Formais, utilizam construções formais conhecidas como Autômatos Finitos Determinísticos [MEN99]. Foi introduzido pelo grupo o conceito de "cursos são autômatos" como uma estrutura que permite fácil implementação, criação de material hipermídia independente do autômato, ao mesmo tempo que encoraja o reuso de páginas Web em vários cursos, os quais podem ser construídos com enfoques diferentes, diminuindo a redundância na criação de páginas [MAJ00], [MEN99], [MAJ98].

O objetivo geral do modelo Hyper-Automaton centra-se no estudo da aplicação do formalismo de Autômatos Finitos com Saída (Máquina de Mealy e Máquina de Moore) como um modelo estrutural para a organização de hiperdocumentos instrucionais, em especial de Cursos na Web. O modelo é inspirado por pesquisas clássicas na área de hiperdocumentos e recentes iniciativas na WWW, com especial enfoque no desenvolvimento de sistemas de hipertexto onde a base de hiperdocumentos é projetada de forma independente da estrutura de controle da aplicação hipermídia, e suporta algumas facilidades como a composição de estruturas hierárquicas, especificação de vários conjuntos de links sobre um mesmo corpo de hiperdocumentos e objetos separados da estrutura de navegação.

Cada autômato define um curso e consiste de um conjunto de hiperdocumentos

independentes, os quais podem pertencer a outros cursos. A função de transição funciona como ligação lógica entre os hiperdocumentos e a função de saída compõe as páginas. O resultado final é a estrutura básica de páginas e links de hipertexto em um site na Web. O modelo leva a um alto grau de modularização do material instrucional, apresentando as seguintes vantagens: facilidade de reuso de páginas em diversos cursos, com eliminação da redundância; independência dos hiperdocumentos da estrutura do autômato, cuja alteração não influi nas páginas e vice-versa; permite que qualquer usuário crie links de e para qualquer documento; facilidade de implementação e manutenção; interface gráfica simples e direta; elaboração de seqüências instrucionais com enfoques específicos e capaz de oferecer estudo individualizado; operações categoriais fornecem um esquema de composição de cursos que permite a construção de novos cursos sobre cursos já existentes através de procedimentos de alto nível [MAJ00], [MEN99].

O Sistema Hyper-Automaton está baseado na arquitetura cliente-servidor e tem-se hoje como objetivo buscar o aprimoramento tecnológico do sistema de disponibilização do material instrucional na Web desenvolvido pelo grupo de pesquisa. Estas mudanças tornarão o sistema capaz de realizar operações mais sofisticadas e condizentes com as novas necessidades dos cursos on-line, como personalização de conteúdo, funcionalidades de suporte ao aluno, e até mesmo operações aritméticas entre cursos. Este presente estudo está focado na melhoria de duas partes do sistema Hyper-Automaton: armazenamento dos cursos e objetos de ensino e apresentação destes elementos na Web. Este trabalho põe em prática a avaliação feita em trabalhos anteriores sobre a linguagem de marcação XML, onde buscou-se conhecer e testar suas reais potencialidades, utilizar os mecanismos e suas características que possibilitam soluções mais sofisticadas no que se refere a disponibilização de objetos de ensino na Web, estruturação de documentos, entre outras funcionalidades que estão sendo cogitadas com a utilização desta nova linguagem [MAC00a], [MAC01].

Finalmente com a inclusão de outras poderosas linguagens de programação dotadas de versatilidade, recursos e compatíveis dentro dos nossos paradigmas previamente estabelecidos, postos agora em prática, acabamos por potencializar de sobremaneira as aplicações iniciais, tornando o sistema mais poderoso e com condições de importantes implementações atuais e futuras.

2. Regras: DTD para XMLSchema, revisão de conceitos

Como fortemente abordado nos trabalhos anteriores [MAC01a], quando se estava no princípio da pesquisa, a DTD era o único elemento consistente, fortemente explorado, suportado pelos principais browser hoje disponíveis e com farto material bibliográfico que a tornava como a principal linguagem de expressão de regras de construção de documentos XML. Durante a evolução do trabalho, mais especificamente em maio de 1999, a W3C publicou em seu site, www.w3c.org, o *W3C Schema Working Draft*. Neste período estávamos satisfeitos e seguros com o uso e poder da DTD em nossas aplicações, pois esta havia sido dominada de acordo com as nossas necessidades.

Conjuntamente com o desenvolvimento prático das aplicações, tornou-se a necessária inclusão de outros tipos de dados, atualização tecnológica do sistema com a manutenção das suas características originais anteriormente estabelecidas em [MAJ00]. Com a crescente maturidade do XML, reforçado pelo surgimento de farto material de apoio em livros ou na WWW, migramos naturalmente para o uso de XML Schema como linguagem de estabelecimento de padrões de construção de regras para o documentos instrucionais em XML no H.A.

Segundo [COS01], um *schema* é um modelo de descrição estrutural da informação, é um termo originado do mundo do banco de dados para a descrição da estrutura dos dados em tabelas relacionais. No contexto do XML, um *schema* descreve um modelo para uma determinada classe de documentos. O modelo descreve o possível arranjo dos tags e texto em um documento válido. Um *schema* deve também ser visto como um vocabulário padrão comum aceito para uma aplicação particular que envolva troca de documentos. Tal afirmação feita pelo autor vai de encontro ao que pretendemos em um futuro bem próximo relacionado ao envio de material didático como exercícios, dicas, gabaritos de respostas, conteúdo, etc.

O XML herdou a *Document Type Definitions* (DTD's) do SGML que na verdade é um mecanismo de *schema* para o antigo SGML [COS01]. Desta maneira a modernização de transações, obrigou que um novo padrão fosse criado para oferecer significativas vantagens até então ausentes nas definições anteriores sobre estabelecimento de regras sobre documentos XML.

Significativas vantagens além daquelas anteriormente descritas que nos levaram a esta escolha foram: possuir a mesma sintaxe XML; permitir muitas maneiras de especificar tipos de dados; suportar pelo menos 44 tipos de dados contra 10 da DTD; poder ser definido múltiplos elementos com o mesmo nome, mas em diferentes contextos; poder ser definido elementos substitutos, por exemplo: livro é substituído por publicação e, finalmente, poder criar-se outros tipos de dados.

Apesar do grande aumento da expressividade do XMLSchema, por se tratar de um *Draft* (rascunho), a sua especificação se encontra em evolução, de forma que alguns aspectos tratados no DTD ainda não foram incorporados pelo *Schema*. Uma das funcionalidades mais importantes ainda ausente é a definição de entidades. Como colocado por [COS01], as entidades são essenciais por permitirem que se utilize caracteres especiais fora do padrão, por permitirem a inclusão de dados não-xml (como imagens, vídeos, etc), por permitem o reuso de passagens de texto e documentos, entre outras facilidades. No caso do Hyper-Automaton caracteres especiais serão utilizados tanto para acentuação quando para símbolos matemáticos, logo o uso de entidades é essencial.

A solução é utilizar as duas formas de definição nos seus pontos fortes, já que a especificação de documentos através de DTD e XMLSchema não são auto-exclusivas, isto é, podem ser usadas conjuntamente (no caso de ambigüidades o validador escolhe de qual versão considerar as definições). Desta forma, os documentos XML utilizados no Hyper-Automaton utilizam XMLSchema para definir os tipos de dados e a estrutura dos documentos, enquanto o DTD é utilizado essencialmente para a utilização de caracteres especiais e inclusão/referência a arquivos externos.

A figura 1, procura exemplificar de maneira mais clara as definições descritas no parágrafo anterior.

3. XML e Java juntos no H.A.

A tecnologia Java e a XML preenchem, de fato, duas funções distintas. Enquanto Java é uma linguagem de programação orientada para objetos, ou seja, uma plataforma, o XML é uma maneira formal de definir, armazenar e intercambiar dados estruturados. O XML não possui a complexidade necessária para manipular os dados, faz-se necessário uma linguagem de programação com a tecnologia Java permitindo de maneira efetiva a troca de informações entre sistemas diferentes. Em outras palavras Java possibilita a manipulação adequadas dos dados para que as aplicações façam sentido e ganhem poder [AVI01].

A implantação de aplicativos baseados na tecnologia Java, que aproveitam as sinergias do código reutilizável e portátil desta plataforma e dos dados reutilizáveis e portáteis da XML, ajudam a simplificação a atualização e a manutenção do sistema H.A.

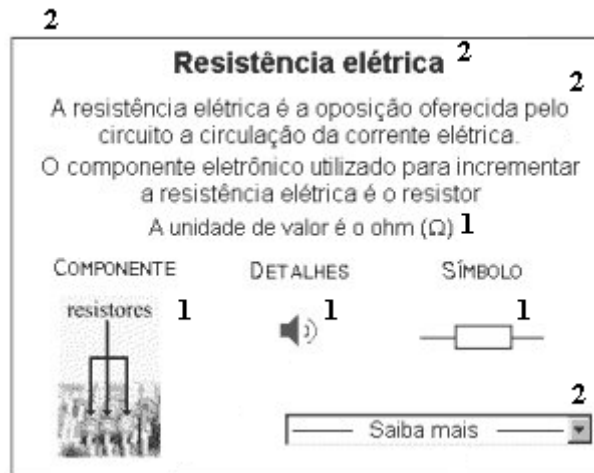


Figura 1 – Definições DTD (1) / XMLSchema (2)

3. Definição do objeto elementar

Por volta de 1994 o termo “Objeto de Ensino” foi introduzido na área de Tecnologia da Instrução: recursos de ensino quebrados em componentes modulares para mais tarde serem combinados por instrutores, estudantes e até mesmo por computadores em estruturas maiores dariam suporte ao ensino [WIL00].

Uma das questões discutidas ao se tratar de objetos de ensino é a noção de granularidade, mais especificamente como serão manipulados os objetos de ensino para a formação de objetos mais complexos. Atualmente pode se dividir essa discussão em duas correntes: A primeira determina o nível de granularidade de um objeto de ensino conforme o grau de composição de objetos menores para compor objetos de ensino maiores. A segunda e mais recente leva em consideração a coesão do objeto de ensino: quanto mais centrados em um único conceito, menor sua granularidade [WIL00].

Por acreditar que o reuso de material instrucional é potencializado pelo uso de objetos com maior coesão, os esforços foram voltados para a definição do objeto elementar de ensino: o menor fragmento de material instrucional que não seja definido em termos de outros objetos de ensino. Desta forma, é possível medir mais precisamente a complexidade de um curso, já que se leva em conta a mensagem de ensino e não o número de objetos que compõem essa mensagem.

Os fragmentos de HTML utilizados na versão anterior do H.A. já foram construídos segundo essa noção de coesão, mas por se tratar de arquivos HTML esses objetos eram o menor nível de abstração manipuláveis no sistema. Já que o nosso objetivo era utilizar e manipular esses mesmos objetos de ensino de forma mais inteligente e com maior poder de reuso, na nova proposta de implementação esses objetos foram analisados internamente, de tal forma que se identificasse os elementos que os constituíssem, como textos, imagens, listas, definições, exemplos, etc.

Com base nessa análise, foi elaborada uma definição em XMLSchema de objeto elementar de ensino que traduz para XML os elementos antes em marcação HTML. A definição do objeto foi especificada de tal maneira que seus componentes internos e externos (textos, figuras, arquivos externos) fossem facilmente identificados e manipulados, conforme exemplificado na figura 1. Desta maneira em nosso protótipo a

definição do objeto de ensino abrange um grande número de itens pré-estabelecidos possibilitando ao desenvolvedor do material instrucional criar o seu conteúdo de maneira flexível, em outras palavras, este esquema de objeto de ensino age como uma “forma”, nos quais os elementos constantes obedecem a critérios de especificação.

A idéia principal do sistema do Hyper-Automaton é que estes objetos de ensino sejam armazenados em um ou mais repositórios públicos, onde os acadêmicos se responsabilizassem pela manutenção e integridade dos arquivos de ensino considerados públicos. Os metadados armazenados com as informações auxiliam no controle e na busca de objetos de ensino nos repositórios.

5. Armazenamento, recuperação e dinamicidade

Uma das principais características deste sistema é a diferença que é feita entre estrutura do curso e objetos de ensino. O autômato que define um curso é uma estrutura formal pré-estabelecida e neste, como anteriormente definido em [MAJ00], estabelece toda a estrutura navegacional deste. Quanto aos objetos elementares de ensino que fazem parte de um curso, carregam uma dinâmica considerável, além de possuírem seu próprio esquema estrutural os objetos elementares sofrem intensas alterações estruturais atualmente manuais, podendo ser renomeados, bipartidos, reestruturados e sofrerem alterações no seu esquema.

Conforme explicado no parágrafo acima, no desenvolvimento de nossas aplicações, estamos utilizando o XML para o armazenamento de dados instrucionais e o banco de dados para armazenar a estrutura do curso.

A significativa vantagem do armazenamento de dados em XML é a facilidade para a edição de dados em um simples editor de textos, mais simples do que em uma complexa ferramenta de banco de dados. Arquivos XML são mais apropriados de serem trocados e carregados por clientes, tornando-se mais fácil à transmissão de dados a um site através da utilização de FTP [CHA01].

A maior vantagem abstrata do XML é que, sendo este um formato hierárquico do que um relacional, ele pode ser usado de uma maneira muito mais direta para projetar estruturas de objetos de ensino de acordo com as nossas necessidades. Não é necessário usar um editor de relacionamento de entidades para normalizar seu esquema. Se existe um elemento que contém um outro elemento, pode-se representá-lo diretamente no formato.

De acordo com os nossos propósitos de manipulação de informações estruturadas em XML para fins instrucionais on-line, duas das quais abordados em trabalhos anteriores, são utilizadas, três técnicas para acessar dados a partir de documentos XML, que são:

- Xpath - É usado o processador XPath para localizar os elementos no arquivo XML através da especificação do caminho e condições;
- XSL - É usado o processador XSL para transformar XML para HTML;
- Java Server Pages (JSP) - São elaboradas classes que utilizam técnicas determinadas para carregar dados.

5.1 A Utilização de Java Server Pages

A ligação do aluno (cliente) com servidor (cursos e fragmentos), é realizado através de uma implementação Java denominado por JSP, similar em funcionamento ao ASP da

Microsoft, aplicação na qual oferece todo o dinamismo que torna possível a implementação do projeto em questão.

A utilização do JSP está intimamente ligada ao fato deste ser programado em Java, manipular com facilidade documentos XML, além de ser disponível abertamente no servidor de páginas Java chamado TomCat-Jakarta.

A tecnologia JSP é uma das mais poderosas, fáceis de usar, por tratar-se de uma ferramenta fundamental no desenvolvimento de aplicações Web. O JSP combina HTML e XML com Java Servlets (*server application extension*) e a tecnologia de JavaBeans para criar ambientes altamente produtivos para o desenvolvimento e uso confiáveis, interativas, de alta performance, independentes de plataforma [DEI01].

Um servlet trabalha da seguinte maneira:

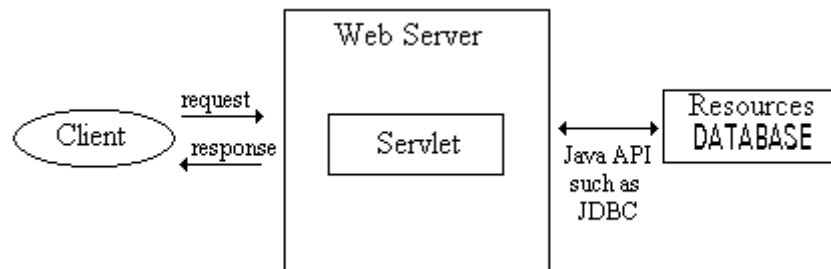


Figura 2 – Funcionamento de uma requisição JSP

A página JSP é traduzida dentro de uma Java Servlet e executada no servidor. As declarações JSP no interior de uma página JSP torna-se parte do servlet gerado a partir da página JSP. O servlet resultante é executado no servidor, tornando-o invisível ao usuário.

O fluxo básico é o seguinte:

6. O cliente envia uma requisição para o servidor;
7. O servidor instancia o servlet e cria uma concorrência (*thread*) para o processo. O servlet é carregado na primeira requisição, ele permanece carregado até o encerramento da seção.
8. O servidor envia a informação requisitada para o servlet;
9. O servlet constrói uma resposta e passa para o servidor;
10. O servidor envia a resposta de volta ao cliente.

O servlet constrói dinamicamente a resposta a partir da requisição do cliente, adiciona dados a partir de outras fontes caso necessário.

Servlets são invocados no sistema H.A. a partir de uma explícita referência na URL de uma aplicação Web, que indica o ponto de localização (diretório) do servlet, estes podem estar localizados em qualquer diretório na máquina onde a aplicação está sendo executada de maneira que os arquivos de classes possam ser encontrados e facilmente manipulados por ftp.

6. Sistema de Cursos H.A.

O Sistema Hyper-Automaton é um sistema de armazenamento e disponibilização de material instrucional na Web, acessível através de uma URL, onde o conteúdo apresentado é separado de sua estrutura de navegação [MAJ00]. Como discutido anteriormente, esta definição de quais páginas mostrar e que links disponibilizar em quais estados e em quais cursos estão definidos no banco de dados. Os objetos de ensino propriamente ditos podem estar em qualquer diretório do servidor (de preferência em um mesmo diretório por motivos de segurança) ou mesmo em qualquer endereço

acessível pela Internet, em outros servidores, repositório de dados, etc.

O Sistema de Hyper-Automaton é responsável por processar a máquina de estados através de consultas ao banco de dados e devolver para o browser um arquivo (HTML,PDF ou mesmo WML) obtido através da aplicação de arquivos de transformação XSL a um arquivo XML criado a partir da anexação de objetos de ensino a sua estrutura.

A nova versão do Sistema Hyper-Automaton foi implementada conforme especificado em [MAJ00] na forma de uma classe Java para a sua utilização no Servidor de Páginas Java chamado TomCat-Jakarta.

Para ser utilizada na Internet, essa classe é instanciada em uma página JSP. Essa página JSP é armazenada no servidor WEB, de modo que para cada pessoa que acessa a página uma instância distinta do Hyper-Automaton é criada na memória do Servidor e associada àquela sessão.

Como qualquer máquina de estados, o seu funcionamento ocorre mediante parâmetros de entrada. No caso do Hyper-Automaton esses parâmetros são três: um curso, um estado e uma palavra de entrada. No exemplo abaixo o curso acessado é aquele de identificação 1, no estado 3 e com a palavra de entrada identificada pelo número inteiro 7.

`http://teia/webapps/hypno/hypno.jsp?course=1&state=3&word=7`

Normalmente os cursos disponíveis terão páginas iniciais independentes do Hyper-Automaton, com apontadores para as várias seções de seus cursos, servindo de referência rápida aos estudantes. Esses pequenos portais de cursos são importantes já que não faz sentido que os usuários tenham que decorar longas URLs, cheias de variáveis e parâmetros.

Uma vez instanciado um objeto Hyper-Automaton, a página JSP repassa os parâmetros (*course*, *state* e *word*) para o objeto através de tags específicos. O parâmetro *course* é sempre obrigatório, já que é a partir do seu valor que se define a função de transição a ser aplicada sobre as entradas. Caso não se forneça algum valor para o curso, o Hyper-Automaton pára por indefinição.

Após fornecer os parâmetros a página JSP faz a chamada do método `showPage()` do objeto instanciado. Esse método é o processamento da máquina de estados propriamente dita, podendo ser dividido em quatro etapas:

1ª Etapa: Iniciar parâmetros , aplicar a função de transição e descobrir a identificação da palavra de saída;

2ª Etapa: Anexar ao documento XML de saída os objetos de ensino;

3ª Etapa: Anexar ao documento XML de saída informações sobre a navegação do curso,como links, referências cruzadas, etc.,

4ª Etapa: Transformar e apresentar o documento XML de saída na forma de apresentação, seja HTML, WAP, PDF, etc.;

1ª Etapa: Iniciação e Função de Transição

As primeiras linhas deste método fazem uma pequena verificação dos parâmetros vindos da página JSP. Caso seja a primeira vez que este método é chamado depois da instanciação, ou se o curso atual mudou desde a última vez que foi chamada, então algumas informações pertinentes ao curso são coletadas do banco de dados e armazenadas em propriedades do objeto, como nome do Curso, Tipo do Curso (Moore,

Mealy), estado inicial do curso, entre outras. Além do curso geralmente é passado também como parâmetro um estado e uma palavra de entrada. Caso o estado não seja fornecido a máquina vai automaticamente para o estado inicial e continua o processamento

Uma vez definido o estado atual do Hyper-Automaton, um documento virtual XML de saída é criado: o documento *Page*. Atualmente ele possui os seguintes elementos: `<metadata>`, `<content-set>` e `<input-words>`. Assim uma página é composta de uma série de metadados (nome do curso, autor do curso, etc), um multi-conjunto ordenado de objetos de ensino (inicialmente vazio) e um conjunto de palavras de entrada (inicialmente vazio -transformado em um conjunto de links no final do processamento).

Vejamos um exemplo:

```
<page>
  <metadata>
    <course>
      <name>Linguagens Formais & Autômatos</name>
      <author>Paulo Blauth Menezes</author>
    </course>
  </metadata>
  <content-set/>
  <input-words/>
</page>
```

Após a criação deste do documento *Page*, é executada uma consulta no banco de dados para descobrir o novo estado da máquina conforme o estado atual e a palavra de entrada fornecidos. A consulta executada é mostrada abaixo:

```
"SELECT T.TRANS_DEST_ID, T.TRANS_SAIDA_ID
   FROM ALFA_ENTRADA AE, TRANSICAO T
   WHERE AE.ALFAENTR_ID = T.ALFAENTR_ID AND T.CURSO_ID =
this.course
   AND T.TRANS_ORIGEM_ID = this.state AND T.ALFAENTR_ID =
this.inputWord"
```

O novo estado atual é o valor da coluna `T.TRANS_DEST_ID` do resultado da consulta. Se o curso for Mealy, a coluna `TRANS_SAIDA_ID` terá como valor o identificador da palavra de saída definida para a transição.

2ª Etapa: Palavra de Saída

Nesta etapa do método *showPage()* descobriremos quais os objetos de ensino que fazem parte da palavra de saída do autômato do curso através de uma consulta de Banco de Dados. Esses objetos armazenados em documentos XML serão lidos e seus conteúdos anexados ao elemento `<content-set>` do documento *Page*, apresentado na etapa anterior como o multi-conjunto ordenado de objetos de ensino.

Caso o Curso em questão seja de Mealy, a identificação da palavra de saída é utilizada na consulta ao Banco de Dados que relaciona (os caminhos para) os objetos de ensino. Caso seja Moore, a identificação da palavra de saída corresponde ao campo `SAIDA_ID` na tabela `ESTADO`, por isso ela é acrescentada na consulta correspondente.

No resultado de ambas as consultas, cada linha representa um objeto de ensino, onde a coluna `OBJETO_DIR` corresponde ao path do arquivo, e `OBJETO_ARQ` o nome do arquivo XML do objeto de ensino propriamente dito. Abaixo seguem as consultas:

Mealy:

```

"SELECT O.OBJETO_DIR, O.OBJETO_ARQ
      FROM OBJETO O, SAIDA_OBJETO SO
      WHERE O.OBJETO_ID = SO.OBJETO_ID AND SO.SAIDA_ID =
this.outputId
      ORDER BY SO.SDAOBJ_ORDEM"

```

Moore:

```

"SELECT O.OBJETO_DIR, O.OBJETO_ARQ
      FROM OBJETO O, SAIDA_OBJETO SO, ESTADO E
      WHERE O.OBJETO_ID = SO.OBJETO_ID AND SO.SAIDA_ID =
E.SAIDA_ID
      AND E.ESTADO_ID = this.state e E.CURSO_ID = this.course
      ORDER BY SO.SDAOBJ_ORDEM"

```

Após a consulta, cada objeto é lido, transformado em uma árvore DOM e seu elemento raiz é anexado ao elemento <content-set> do documento *Page*. Para uma saída que possuiu o objeto de ensino apresentado anteriormente como palavra de saída, obtém-se como resultado a composição do documento XML page através da associação dos fragmentos.

3ª Etapa: Palavras de entrada

Mais uma consulta é feita no Banco de Dados para se descobrir as palavras de entrada que definem transições para outros estados a partir do estado atual. As identificações e as palavras em sí são anexadas ao documento *Page*, mais precisamente ao elemento <input-words>. São os elementos <input> contidos em <input-wrods> que serão transformados em elementos <A> no caso de HTML, por exemplo. Abaixo segue a consulta SQL para busca das palavras de entrada:

```

"SELECT AE.ALFAENTR_PALAVRA, AE.ALFAENTR_ID
      FROM ALFA_ENTRADA AE, TRANSICAO T
      WHERE AE.ALFAENTR_ID = T.ALFAENTR_ID
      AND T.TRANS_ORIGEM_ID = this.state AND T.CURSO_ID =
this.course
      ORDER BY AE.ALFAENTR_ORDEM"

```

No exemplo abaixo são definidas duas transições a partir do estado atual 2: uma para a próxima página, outra para o próximo capítulo.

```

<page>
  <metadata>
    ...
  </metadata>
  <content-set>
    ...
  </content-set>
  <input-words>
  <input>
    <word id="3">Anterior</word>
      <currState>2</currState>
      <currCourse>1</currCourse>
  </input>
  <input>
    <word id="1">Próximo</word>
      <currState>2</currState>
      <currCourse>1</currCourse>
  </input>
</input-words>
</page>

```

O documento *Page* está completo para ser transformado e formatado para apresentação.

4ª Etapa: XML transformado

A última etapa do método é que diferencia a nova versão em XML da implementação em Perl: uma vez formado o documento *Page*, ao invés de simplesmente apresentar os objetos de ensino concatenados na saída na forma de HTML, um arquivo de transformação particular a cada curso é aplicado ao documento *Page* através de um objeto transformador (*Transformator*), lhe dando forma, sabor e personalização na sua apresentação.

Cada curso tem o seu próprio arquivo de transformação XSL que define a estrutura da página de saída: cabeçalhos, menus, cores, layout, etc. O responsável pelo curso poderá escolher entre vários layouts padrões para seu curso e ainda terá a opção de desenvolver o seu próprio layout. Essa personalização não era possível na versão anterior pois a formatação e diagramação eram inerentes aos fragmentos HTML utilizados.

Um arquivo de transformação padrão foi elaborado e será amplamente divulgado para auxiliar e incentivar a criação de novos arquivos de transformações para os objetos de ensino utilizados no Sistema Hyper-Automaton. Deste modo qualquer pessoa que domine a linguagem de Transformação XSL poderá definir comportamentos próprios para os elementos que compõem o documento *Page*. Para aqueles que não conhecem XSL, mas gostariam de algum nível de personalização, serão oferecidos alguns Templates padrões. A figura 3 exemplifica a montagem da formatação do objeto de ensino no browser do usuário.

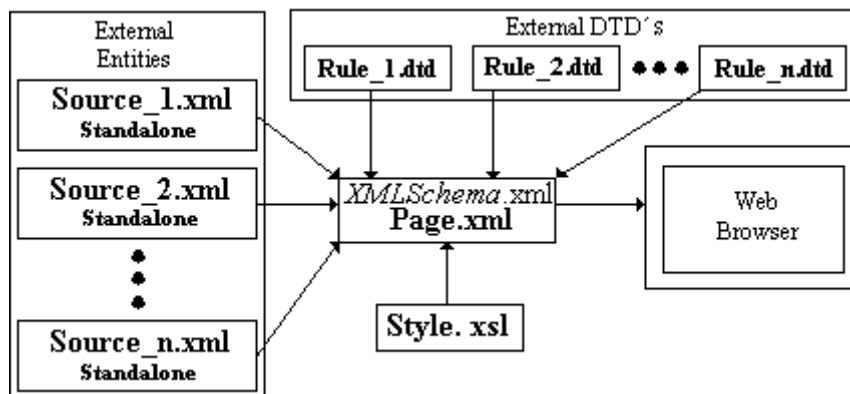


Figura 3 – Partes integrantes do objeto de ensino

O Resultado dessa aplicação é a página HTML que é mostrada no browser do internauta (figura 4).

Um outro curso pode utilizar os mesmos objetos de ensino mas como um arquivo de transformação de XSL diferente.

7. Comparação com o sistema anterior

7.1 Organização de arquivos

Na versão anterior cada curso tinha seu próprio diretório onde ficavam os seus três arquivos de definição: curso.fl, alfabeto.fl e estados.fl. Além disso todos os fragmentos HTML disponíveis eram armazenados em apenas um diretório do servidor. Isto ocasionava um inconveniente: a listagem dos objetos disponíveis para inclusão no curso era muito longa e sem qualquer tipo de ordenação, tornando muito exaustiva a procura dos arquivos desejados. Na nova versão os fragmentos poderão ficar em qualquer

repositório acessível pela web. Não existem mais os arquivos correspondentes aos cursos, já que agora são armazenados em banco de dados.

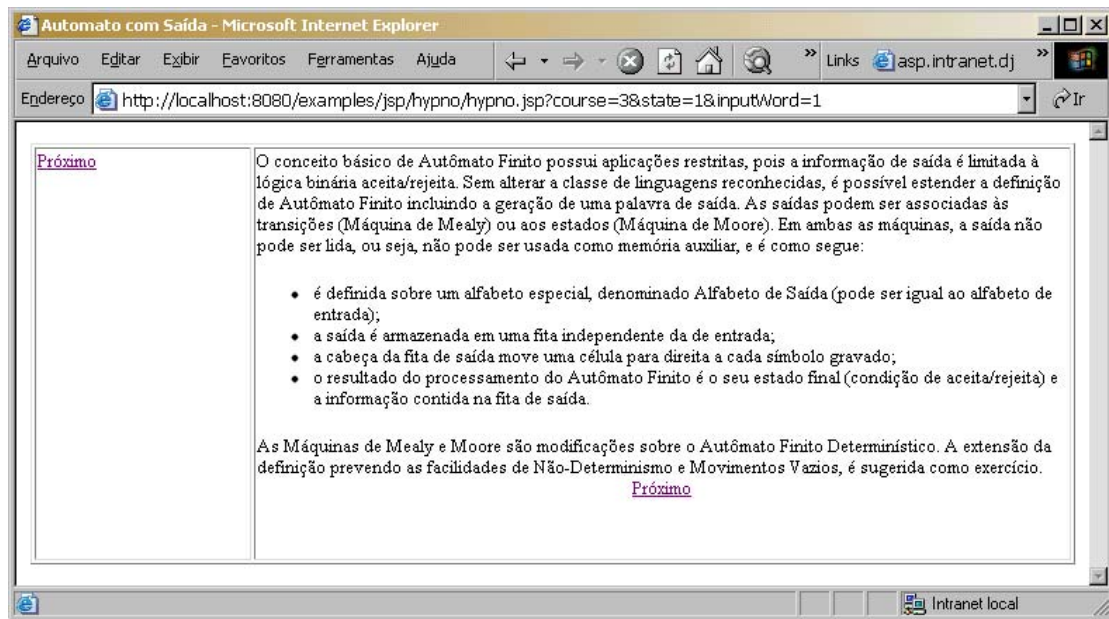


Figura 4 – Formatação de saída para um determinado XSL

7.2 Manutenção facilitada: uma máquina de estados que incorpora as duas funcionalidades

Enquanto na versão em Perl o acesso a máquina de Moore e Mealy eram feitos através de dois sistemas diferentes, na nova versão em JSP a funcionalidade das duas máquinas foram incorporadas em um só sistema. Assim, a manutenção se torna muito mais fácil e o acesso ao sistema se torna único, facilitando o acesso aos cursos.

7.3 Reuso inteligente

O primeiro sistema do Hyper-Automaton já utilizava o conceito de reuso através da disponibilização de material instrucional na forma de fragmentos de HTML. Os cursos eram criados através da definição dos fragmentos utilizados em cada estado do curso. O material instrucional era apresentado com o mesmo layout em todos os cursos. Através do uso de XML e XSL, o conteúdo do material de ensino foi separado da sua forma de apresentação. Isso significa que um mesmo objeto de ensino é apresentado conforme a diagramação do curso que o utiliza.

8. Conclusões e perspectivas futuras

Os cursos e documentos utilizados na implementação do sistema foram desenvolvidos utilizando-se editores de texto e criando-se entradas no banco de dados diretamente. Uma vez que o sistema está bem especificado, seja pelo ER, seja pelos XMLSchema, já é possível se pensar em um sistema front-end de criação de cursos. Neste sistema (que provavelmente será implementado usando-se JSP) proverá ao professor uma interface amigável de estruturação de conteúdo.

Todas as funcionalidades que serão acrescentadas ao sistema serão pela adição de elementos específicos no objeto de ensino e no próprio objeto pagina. Assim como o glossário, que utiliza o elemento <term>, a criação de índices de palavras relacionadas será facilmente implementada através da definição de novas regras de aplicação no XSL

padrão, outras funcionalidades serão estendidas. A vantagem é que todos os cursos que já utilizarem esses elementos terão as novas facilidades incorporadas automaticamente.

Outro sistema interessante é o aquele que irá auxiliar o professor a criar seus próprios objetos de ensino, através de *wizards*, templates e até mesmo a partir do material didático em formato digital já utilizado pelo professor. Este sistema fará com que se aproveite ao máximo as funções avançadas implementadas no Sistema do Hyper-Automaton.

Outras implementações futuras serão desenvolvidas como trabalho de graduação ou pós-graduação possivelmente em conjunto com outras áreas da informática, tais como: login e conteúdo sensível ao usuário, log de utilização e cursos sensíveis ao aplicativo.

Referências

- [AVI2001] AVIRAM, Mariva H. **XML and Java: A powerful combination**. 1999. Disponível em: <http://www.javaworld.com/javaworld/jw-07-1999/jw-07-javaone-xml_p.html>. Acesso em: 26 dez. 2001.
- [COS2001] COSTELLO, Roger L. **Tutorial: XML Technologies Course**. [S.l.]: [s.n.], 2001. 203p. Disponível em: <<http://www.xfront.com/>>. Acesso em: 14 dez. 2001.
- [CHA2001] CHAFFE, A.; **Using XML and JSP together**. 2001. Disponível em: <http://www.javaworld.com/javaworld/jw-03-2000/jw-03-javaone-xml_p.html>. Acesso em: 26 dez. 2001.
- [DEI2001] DEITEL, H.M.; DEITEL, P.J. **Java, como programar**. Porto Alegre: Bookman, 2001. 1201p.
- [MAC2000a] MACHADO, César Costa, MACHADO, Júlio Henrique P, GRANDI, Roges, MENEZES, Paulo Blauth. Utilização do XML no Sistema Hyper-Automaton. In: INTERNACIONAL SYMPOSIUM ON KNOWLEDGE MANAGEMENT / DOCUMENT MANAGEMENT - ISKM/DM, 2000, Curitiba. **Anais...** Curitiba: Ed. Universitária Champagnat, 2000. p.439-455.
- [MAC2001] MACHADO, César C.; FEDERIZZI, Gustavo L.; MENEZES, P. Blauth. Flexibility and Adequacy of the Output Layout in the Hyper-Automaton System. In: INTERNACIONAL CONFERENCE ON INTERNET COMPUTING - IC, 2001, Las Vegas - USA. **Proceedings...** Las Vegas: CSRE Press, 2001. p.1125-1131.
- [MAC2001a] MACHADO, César C.; FEDERIZZI, Gustavo L.; MENEZES, P. Blauth. Definition and Application of Rules for the Adequate Designing of XML Documents for the Hyper-Automaton System. In: INTERNACIONAL WORKSHOP ON DATABASE AND EXPERT SYSTEMS APPLICATIONS - DEXA, 12., 2001, Munich - RDA. **Proceedings...** [S.l.: s.n], 2001.
- [MAJ 98] MACHADO, Júlio P., MENEZES, Paulo F. B. Sistemas de Gerenciamento para o Ensino a Distância. In: CONGRESSO INTERNACIONAL DE EDUCAÇÃO A DISTÂNCIA, 5., 1998, São Paulo. **Anais...** Disponível em: <<http://www.abed.org.br>>. Acesso em: jun. 2000.
- [MAJ2000] MACHADO, Júlio P. **Hyper-Automaton: Hipertextos e Cursos na Web Utilizando Autômatos Finitos com Saída**. 2000. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.
- [MEN99] MENEZES, Paulo Blauth, MACHADO, Júlio Henrique Araújo Pereira. Web courses are Automata: a Categorical Framework. In: WORKSHOP ON FORMAL METHODS, 2., 1999, Florianópolis. **Proceedings...** Florianópolis: SBC, 1999. p.79-88.
- [WIL2000] WILEY, David A. Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. Disponível em: <http://reusability.org/read/chapters/wiley.doc>. Acesso em: fev. 2000.

Bibliografia

- [ADL2000] ADLER, S.; DEATCH, S.; GUTENTAG, E. **Extensible Stylesheet Language (XSL) – Version 1.0**. Draft 27 March 2000. Disponível em: <<http://www.w3.org/TR/2000/WD-xsl-20000327/>>. Acesso em: jan. 2000.
- [AMB2000] AMBLER, Scott W. **Mapping Objects to Relational Databases**. 2000. Disponível em: <<http://www.AmbySoft.com/mappingObjects.pdf> >. Acesso em: jul. 2000.
- [ARM 99] ARMSTRONG, Eric. **Working with XML**. Version 1, Update 5 -- 9 Aug. 1999. Disponível em: <<http://www.xml.com>>. Acesso em: mar. 2000.
- [AVI2001] AVIRAM, Mariva H. **XML and Java: A powerful combination**. 1999. Disponível em: <http://www.javaworld.com/javaworld/jw-07-1999/jw-07-javaone-xml_p.html>. Acesso em: 26 dez. 2001.
- [BOR99] BORONI, Christopher M.; GOOSSEY, Frances W. et al. Trying it All Together: Creating Self-Contained, Animated, Interactive, Web-based Resources for Computer Science Education. **SIGCSE Bulletin**, New Orleans, v.31, n.1, p.7-11, Mar.1999.
- [BOS98] BOS, B; LIE, Håkon W.; LILEY, C.; JACOBS, I. **Cascading Style Sheets, Level 2 Specification**. 1998. Disponível em: <<http://www.w3.org/TR/REC-CSS2>>. Acesso em: mar. 2000.
- [BRA98] BRAY, T.; PAOLI, C. M. **Extensible Markup Language (XML) 1.0 Specification**. 1998. Disponível em: <<http://www.w3.org/TR/REC-xml>>. Acesso em: mar. 2000.
- [BRA2000] BRADLEY, Neil. **The XSL Companion**. [S.l.]: Addison-Wesley, 2000. 317p.
- [CAG99] CAGLE, Kurt. **Transform Your Data With XSL**. 1999. Disponível em: <<http://www.xmlmag.com>>. Acesso em: mar. 2000.
- [CES2001] CESTA, André A. **Tutorial: A Linguagem de Programação JAVA**. Campinas: [s.n.], 2001. 133p.
- [CHA2001] CHAFFE, A. **Using XML and JSP together**. 2001. Disponível em: <http://www.javaworld.com/javaworld/jw-03-2000/jw-03-javaone-xml_p.html>. Acesso em: 26 dez. 2001.
- [COS2001] COSTELLO, Roger L. **Tutorial: XML Technologies Course**. [S.l.]: [s.n.], 2001. 203p. Disponível em: <<http://www.xfront.com/>>. Acesso em: 14 dez. 2001.

- [DEI2001] DEITEL, H.M.; DEITEL, P.J. **Java, como programar**. Porto Alegre: Bookman, 2001. 1201p.
- [FED99] FEDERIZZI, Gustavo L. Compatibilização de Documentos entre Diferentes Plataformas. In: SALÃO DE INICIAÇÃO CIENTÍFICA, 11., 1999. Porto Alegre. **Anais...** Porto Alegre: PROPESQ da UFRGS, 1999.
- [GOL90] GOLDFARB, Charles F. **The SGML Handbook**. Oxford: Clarendon Press, 1990. p.5.
- [HAL94] HALASZ, F.; SCHWARTZ, M. The Dexter HyperText Reference Model. **Communications of the ACM**, New York, v.37, n.2, p.30-39, 1994.
- [HAR99] HAROLD, Elliotte, R. **XML Bible**. Foster City: IDG Books Worldwide, 1999. 1015p.
- [LER99] LERMEN, Alessandra de Lucena. **A Mapping Framework from ODMG to Object/Relational DBMS**. 1999. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.
- [LIE99] LIE, Håkon W.; BOS, B. **Cascading Style Sheets, Level1 W3C Recommendation**. 17 dez. 1996, revisão 11 jan. 1999. <<http://www.w3.org/TR/REC-CSS1>>. Acesso em: abr. 2000.
- [LIE99a] LIE, Håkon W.; SAARELA, Janne. Multipurpose Web Publishing Using HTML, XML and CSS. **Communications of the ACM**, New York, v.42, n.10, 1999.
- [LIG99] LIGHT, Richard. **Iniciando em XML**. São Paulo: Makron Books do Brasil, 1999. 404p.
- [LOH96] LOHUIS, R.A.G. **Computer Mediated Communication in Distance Education**. Disponível em: <<http://wcd.student.etwente.nl/~ronny/literat.htm>>. Acesso em: mar. 2000.
- [LYN99] LYNCH, Patrick J.; HORTON, Sarah. **Web Style Guide**. New Haven: Yale University Press, 1999. 164p.
- [MAC99] MACHADO, César Costa. **Estudo Comparativo Sobre as Ferramentas de Autoria Existentes para a Criação de Documentos para a World Wide Web**. 2000. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.
- [MAC2000] MACHADO, César Costa, MENEZES, Paulo Blauth. Características

e vantagens na estruturação de documentos em XML na WWW para EAD. In: WORKSHOP EM INFORMÁTICA NA EDUCAÇÃO, 1., 2000, Passo Fundo. **Anais...** Passo Fundo: UPF, 2000. p.99-104.

- [MAC2000a] MACHADO, César Costa; MACHADO, Júlio Henrique P.; GRANDI, Roges; MENEZES, Paulo Blauth. Utilização do XML no Sistema Hyper-Automaton. In: INTERNATIONAL SYMPOSIUM ON KNOWLEDGE MANAGEMENT / DOCUMENT MANAGEMENT - ISKM/DM, 2000, Curitiba. **Anais...** Curitiba: Ed. Universitária Champagnat, 2000. p.439-455.
- [MAC2001] MACHADO, César C.; FEDERIZZI, Gustavo L.; MENEZES, P. Blauth. Flexibility and Adequacy of the Output Layout in the Hyper-Automaton System. In: INTERNATIONAL CONFERENCE ON INTERNET COMPUTING - IC, 2001, Las Vegas - USA. **Proceedings...** Las Vegas: CSRE Press, 2001. p.1125-1131.
- [MAC2001a] MACHADO, César C.; FEDERIZZI, Gustavo L.; MENEZES, P. Blauth. Definition and Application of Rules for the Adequate Designing of XML Documents for the Hyper-Automaton System. In: INTERNATIONAL WORKSHOP ON DATABASE AND EXPERT SYSTEMS APPLICATIONS - DEXA, 12., 2001, Munich - RDA. **Proceedings...** [S.l.: s.n], 2001
- [MAJ 98] MACHADO, Júlio P.; MENEZES, Paulo F. B. Sistemas de Gerenciamento para o Ensino a Distância. In: CONGRESSO INTERNATIONAL DE EDUCAÇÃO A DISTÂNCIA, 5., 1998, São Paulo. **Anais...** Disponível em: < <http://www.abed.org.br>>. Acesso em: jun. 2000.
- [MAJ2000] MACHADO, Júlio P. **Hyper-Automaton: Hipertextos e Cursos na Web Utilizando Autômatos Finitos com Saída.** 2000. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.
- [MAJ2000a] MACHADO, Júlio P.; MORAIS, Carlos T.Q. de; MENEZES, P. Blauth; REIS, Ricardo A.L. Structuring Web Course Pages as Automata: Revising Concepts. In: RECHERCHE D'INFORMATIONS ASSISTEE PAR ORDINATEUR, RIAO, 2000, Paris. **Proceedings...** Paris: Centre de Hautes Etudes Internationales d'Informatique Documentaires, Center for the Advanced Study of Information Systems, 2000. v.1, p.150-159.
- [MAR96] MARSHALL, A.D.; HURLEY, S. **Interactive Hypermedia Couseware for the WWW.** [Espanha: s.n.], 1996.
- [MEN2000] MENEZES, Paulo Blauth. **Linguagens Formais e Autômatos.** 2.ed. Porto Alegre: Sagra-Luzzatto. 2000.
- [MEN99] MENEZES, Paulo Blauth, MACHADO, Júlio Henrique Araújo

- Pereira. Web courses are Automata: a Categorical Framework. In: WORKSHOP ON FORMAL METHODS, 2., 1999, Florianópolis. **Proceedings...** Florianópolis: SBC, 1999. p.79-88.
- [MOU99] MOULTIS, Natanya P.; KIRK, Cheryl. **XML Black Book**. [S.l.]: Makron Books, 1999
- [RIC01] RICARTE, Ivan Luiz M. **Programação Orientada a Objetos**. Campinas: [s.n.], 2001. 117p.
- [SCH95] SCHNEIDER, D.; BLOCK, K. **The World Wide Web in Education**. 1995. Disponível em: <<http://tecfa.unige.ch/tecfa/tecfa-research/CMC/andrea95/andrea.txt>>. Acesso em: abr. 1998.
- [SIL97] SILBERSCHATZ, Abraham et al. **Database System Concepts**. 3rd. [S.l.]: McGraw-Hill, 1997.
- [TID99] TIDWELL, Doug. **Introduction to XML: Tutorial**. [S.l.]: IBM, 1999. Disponível em: <<http://www.ibm.com/developerWorks>>. Acesso em: maio 2000.
- [WEB98] WEBER, Taysi et al. Uma Experiência com Hiperdocumentos e Internet no Suporte a Disciplinas de Computação. In. WORKSHOP SOBRE EDUCAÇÃO EM INFORMÁTICA, 6.; CONGRESSO DA SOCIEDADE BRASILEIRA DA COMPUTAÇÃO, 18., 1998, Belo Horizonte. **Anais...** Belo Horizonte: SBC, 1998. p. 532-545.
- [WIL2000] WILEY, David A. **Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy**. Disponível em: <http://reusability.org/read/chapters/wiley.doc>. Acesso em: fev. 2000.