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

Organização e Armazenamento de Conteúdo Instrucional no Ambiente AdaptWeb Utilizando XML

por

MARÍLIA ABRAHÃO AMARAL

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

Prof. Dr. José Palazzo M. de Oliveira
Orientador

Profa. Dra. Maria Angélica O. Camargo Brunetto
Co-Orientadora

Porto Alegre, dezembro de 2002.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Amaral, Marília Abrahão

Organização e Armazenamento de Conteúdo Instrucional no Ambiente AdaptWeb Utilizando XML / por Marília Abrahão Amaral – Porto Alegre: PPGC da UFRGS, 2002.

101 p. : il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2002. Orientador: Oliveira, José Palazzo M. de.

1. EAD 2. Ambiente Hipermídia 3. XML 4. Adaptabilidade I. Oliveira, José Palazzo M. de, II. Título

Agradecimentos

À minha família, principalmente aos meus pais, pelo apoio em todos os momentos,
Ao meu marido Eduardo, pelo amor e companheirismo,
Ao meu orientador e minha Co-orientadora, pela paciência e dedicação,
Aos amigos e amigas do Projeto AdaptWeb,
A todos os meus amigos, principalmente à Lúcia e ao Jorge, um casal maravilhoso,
e, finalmente,
A Deus, por tudo.

Sumário

Lista de Abreviaturas

AHA	Adaptive Hypermedia for All
API	Application Programming Interface
DDL	Data Definition Language
DOM	Document Object Model
DTD	Document Type Definition
HTML	Hyper Text Markup Language
RMD	Required Markup Declaration
SGML	Standard Generalized Markup Language
SH	Sistemas Hipermídia
SHA	Sistemas Hipermídia Adaptativos
STI	Sistemas Tutores Inteligentes
XML	Extensible Markup Language
XSL	Extensible StyleSheet Language
W3C	World Wide Web Consortium
WWW	World Wide Web

Lista de Figuras

Lista de Tabelas

Resumo

O uso da Internet como ferramenta de ensino tem se tornado cada vez mais freqüente. A recente popularização da Internet vem permitindo o desenvolvimento de ambientes de ensino-aprendizagem baseados na *Web*. Os principais recursos explorados para fins educacionais são hipertexto e hipermídia, que proporcionam uma grande gama de elementos para o instrutor que pretende utilizar a WWW. Este trabalho está inserido no desenvolvimento do ambiente AdaptWeb (Ambiente de Ensino e Aprendizagem Adaptativo para a *Web*), que visa o desenvolvimento de um ambiente de educação a distância. A arquitetura do ambiente é composta por quatro módulos entre eles o módulo de Armazenamento de dados que armazena todos os dados provenientes da fase de Autoria utilizando XML (*Extensible Markup Language)*. Na etapa de Autoria é feita a inserção de todos os dados relativos a disciplina que deseja disponibilizar, estes dados serão armazenados temporariamente em uma representação matricial em memória. A entrada de dados do módulo de Armazenamento de Dados é esta representação matricial que serve então como base para a geração dos arquivos XML, que são utilizados nas demais etapas do ambiente. Para a validação dos arquivos XML foram desenvolvidas DTD (*Document Type Definition)* e também foi implementado um analisador de documentos XML, utilizando a API (*Application Programming Interface)* DOM (*Document Object Model*), para efetuar a validação sintática destes documentos. Para conversão da representação matricial em memória foi especificado e implementado um algoritmo que funciona em conformidade com as DTD especificadas e com a sintaxe da linguagem XML.

Palavras-Chave: EAD, Ambiente Hipermídia, XML, Adaptabilidade, *Word Wide Web*

TITLE: “Organization and Storage of instutional conteudo in AdaptWeb using XML”

Abstract

The use of the Internet as teaching tool has became more frequent. The recent popularization of the Internet comes allowing the development of set teaching-learning environment in the Web. The main resources explored for educational purposes are hipertext and hypermidia, that provide a great range of elements for the instructor that intends to use WWW. This work is inserted in the development of the environment AdaptWeb, that seeks the development of a distance learning environment. The architecture of the environment is composed by for modules among them the module of Storage of data that stores every data coming of the Authory using XML (Extensible Markup Language). In the stage of Authory it is made the insertion of every data of the discipline that wants to show, these data will be stored temporarily in a matricial representation in memory. The entrance of data of the module of Storage is this matricial representation that is important for the generation of the files XML, that are used in the other stages of the environment. For the validation of the files XML was developed DTD (Document Type Definition) and an parser of XML documents was also implemented, using API (Application Programming Interface) DOM (Document Object Model), to make the syntactic validation of these documents. For conversion of the matricial representation in memory was specified and implemented an algorithm that works in conformity with specified DTD and with the syntax of the language XML.

Keywords: Distance Learning Environment, Hypermidia, XML, Adaptive, Word Wide Web

1 Introdução

O crescente desenvolvimento da tecnologia de redes de computadores, com o aperfeiçoamento dos meios de comunicação, dos protocolos e das técnicas de processamento distribuído impulsionam o uso de sistemas distribuídos com finalidades de ensino-aprendizagem [GIR99].

Adicionalmente, a recente popularização da Internet vem permitindo o desenvolvimento de ambientes de ensino-aprendizagem via Web. Estes ambientes têm sido a preocupação e a meta de professores que perceberam a vantagem da utilização do ambiente de rede como ferramenta de apoio ao ensino.

Em uma primeira instância, a utilização da Internet como ferramenta de ensino restringia-se a *sites* para disponibilização de material didático. Este tipo de experiência, por mais elementar que pareça, deu início a uma importante área de pesquisa, a área de Ensino à Distância ou EAD. Com isto foram definidos componentes alternativos para os ambientes de ensino-aprendizagem baseados na *Web*, tais como: livros eletrônicos, bases de exercícios, fórum de conversação (*chat*), vídeos e outros elementos [BRU2001].

Um aspecto importante nos sistemas de ensino-aprendizagem para *Web* é a questão da adaptabilidade destes com respeito ao perfil do aluno, pois o aluno possui um papel de extrema importância, porém ainda existem dificuldades para se desenvolver sistemas hipermídia adaptativos para a *Web*.

Sistemas hipermídia adaptativos provêm acesso personalizado de maneira automática às informações hipermídia. A maioria dos sistemas hipermídia adaptativos possui adaptabilidade no que tange a navegação e a apresentação do conteúdo. A estruturação dos *links* ou sua apresentação pode ser diferente para cada grupo de usuário. O conteúdo das páginas também pode ser diferenciado. Os sistemas hipermídia adaptativos que possuem seu conteúdo disponibilizado na *Web* tornam-se mais complexos pelo dinamismo inerente a *Web*, que permite uma quantidade maior de usuários com perfis distintos acessando a mesma informação [WU2001].

A utilização de grupos de usuários com características comuns para divisão dos perfis de usuário pode ser considerada como uma forma de adaptabilidade. Em sistemas hipermídia adaptativos baseado na *Web*, prover ao usuário um ambiente com características específicas voltadas para o perfil de seu grupo auxilia tanto o instrutor, que poderá tornar seu material mais organizado e de fácil manutenibilidade, como o aluno, que terá o foco de seu estudo mais centrado [BRU98].

Existem alguns sistemas hipermídia adaptativos que apresentam conceitos de adaptabilidade tanto na apresentação do conteúdo como na navegação, entre eles pode ser citado o AHA (*Adaptive Hypermedia for All*). O AHA tem como meta a criação de um ambiente simples para construção de *websites* que se adaptem ao usuário. A arquitetura do AHA é dividida em 3 partes: modelo de domínio, modelo de usuário e modelo de adaptação [BRA2001].

O modelo de domínio descreve o domínio da aplicação, utilizando páginas e conceitos definidos pelo autor. O modelo do usuário consiste em identificações e preferências relativas ao usuário. O modelo de adaptação consiste na definição de um conjunto de regras, relacionamentos e definições do sistema. Todos elementos destes três módulos compõem o comportamento de adaptabilidade do AHA [BRA2002].

As páginas referentes ao modelo de domínio, as informações sobre o modelo do usuário e o processo de manipulação de conteúdo instrucional adaptativo no AHA,

são implementadas através de documentos XML (*Extensible Markup Language)* [W3C2000]. Isto porque XML é uma linguagem extensível, possuindo uma alta flexibilidade para descrição e representação de dados que necessitam de informações agregadas, com mais flexibilidade que HTML (*Hyper Text Markup Language*) [MAC98] por não possuir um *tagset* (conjunto de marcações) fixo. Oferece ainda a estruturação de documentos de forma completamente independente da apresentação. Portanto documentos XML possuem uma descrição própria, permitindo que o autor defina sua própria estrutura de informação [BRA2002].

XML também é utilizado no desenvolvimento desta dissertação. Este trabalho está inserido no desenvolvimento do ambiente AdaptWeb (Ambiente de Ensino e Aprendizagem Adaptativo para a *Web*), um ambiente hipermidia adaptativo para *Web*, que utiliza conceitos de adaptabilidade na navegação e na apresentação, para disponibilização de um mesmo conteúdo à aprendizes de diferentes grupos, bem como em diferentes mídias.

O AdaptWeb é composto por quatro módulos distintos [AMA2002]: (1) Autoria, (2) Armazenamento de dados em XML, foco deste trabalho, (3) Adaptação do conteúdo baseado no modelo do aluno e (4) Interface adaptativa, que estão detalhados no capítulo quatro deste documento.

A maioria dos ambientes hipermídia possuem algumas fases que precedem a disponibilização do conteúdo instrucional, entre elas está a fase de modelagem conceitual também conhecida como pré-autoria e a fase de autoria [CAM96]. No AdaptWeb todo o processo de desenvolvimento de uma disciplina se inicia com o módulo de autoria. Entre o módulo de autoria e a disponibilização do material para o aluno existem os módulos listados anteriormente.

O processo de armazenamento de conteúdo instrucional no AdaptWeb é realizado após o módulo de autoria. A utilização de XML é relevante para o processo de organização e padronização dos dados provenientes da autoria, bem como para prover mecanismos para a disponibilização de diferentes apresentações de um mesmo conteúdo.

O uso da linguagem XML é de fundamental importância para o AdaptWeb, pois esta é um recurso rico para a manutenção de organização em estrutura hierárquica. As próprias *tags* de marcação de conteúdo, podem estabelecer uma estruturação lógica dos dados. XML ainda permite que o conteúdo seja mantido separadamente do *layout* de apresentação, e com a utilização de folhas de estilo XSL (*Extensible StyleSheet Language*) ou outras formas de apresentação de conteúdo é possível a geração de novas apresentações para o mesmo documento XML. O módulo de Armazenamento fornece os dados, em XML, para os próximos módulos, Adaptação do Conteúdo Baseado no Perfil do Usuário e Interface Adaptativa.

O presente trabalho trata da utilização de XML neste Ambiente de Ensino e Aprendizagem Adaptativo para a *Web* e está divido da seguinte maneira: o Capítulo 2 aborda os Sistemas Hipermídia Adaptativos para *Web*, seus conceitos, e exemplo de aplicações já desenvolvidas. O Capítulo 3 descreve o uso de XML para a organização de conteúdo instrucional. As características e particularidades do ambiente AdaptWeb são descritas no Capítulo 4. O Capítulo 5 detalha o módulo de armazenamento de dados utilizando XML, com seus algoritmos, e na seqüência o Capítulo 6 relata um estudo de caso desenvolvido. A conclusão do presente trabalho é apresentada no Capítulo 7.

2 Sistemas Hipermídia Adaptativos para *web*

Nos últimos anos a popularidade dos Sistemas Hipermídia (SH) vem aumentando principalmente por estes serem aplicações que auxiliam no direcionamento do acesso a informação. Dentro do conceito de SH, os Sistemas Hipermídia Adaptativos (SHA) são uma nova tendência de pesquisa cuja meta é aumentar a funcionalidade dos SH tornando-os personalizados, ou seja, adaptados às necessidades do usuário [BRU96].

Sistemas Hipermídia Adaptativos provêm acesso personalizado de maneira automática às informações hipermídia. A maioria dos SHA possui adaptabilidade no que tange a navegação e a apresentação do conteúdo. A estruturação ou a apresentação dos *links* pode ser diferente para cada grupo de usuário, bem como o conteúdo das páginas. Grande parte dos sistemas hipermídia adaptativos possui seu conteúdo disponibilizado na *Web*, o que os torna mais complexos pelo dinamismo inerente a *Web*, ou seja, pode-se ter uma quantidade maior de usuários com perfis diferentes acessando a mesma informação [WU2001].

A utilização de grupos de usuários com características comuns para divisão dos perfis de usuário pode ser considerada uma forma de adaptabilidade. Em sistemas hipermídia adaptativos voltados para Educação à Distância, prover ao usuário um ambiente com características específicas para o perfil de seu grupo auxilia tanto o instrutor, que pode tornar seu material mais organizado e de fácil manutenibilidade, como o aluno, que tem o foco de seu estudo mais centrado [BRU2001].

Este capítulo explora os tipos de adaptabilidade aplicáveis a sistemas hipermídia. Tem como objetivo também descrever alguns SHA já desenvolvidos, analisando como estes provêm adaptabilidade.

2.1 Adaptabilidade no ensino à distância

Segundo [BRU96a], os hipermídias são um conjunto de nós, onde cada nó é uma página, estas são conectadas por *links* e contêm alguma informação local além de um conjunto de *links* para outras páginas relacionadas. Estes *links* podem aparecer juntamente com o conteúdo da página ou como um menu em um local separado do conteúdo.

Os sistemas hipermídia também podem oferecer um índice ou um mapa global que provê acessibilidade para todos os *links* do conjunto de páginas. O que pode ser adaptado nestes sistemas são os conteúdos das páginas e os *links*, índices e mapas de navegação. De acordo com [BRU96] a adaptabilidade dos conteúdos instrucionais das páginas de um *site* é chamada de adaptabilidade de apresentação, e a adaptabilidade dos *links*, índices e mapas é chamada de adaptabilidade de suporte a navegação.

Nas próximas seções deste capítulo são exploradas estas duas formas de adaptabilidade bem como suas particularidades.

2.2 O que pode ser adaptado em Sistemas Hipermídia Adaptativos

De modo geral a adaptabilidade de apresentação pode resolver o problema dos sistemas hipermídia que são utilizados por diferentes classes de usuários, enquanto que a adaptabilidade de suporte a navegação é utilizada para prover algum tipo de suporte navegacional e prevenir que o usuário se perca no hiperespaço.

2.2.1 Adaptabilidade de Apresentação

A adaptabilidade de apresentação está intimamente ligada com a organização de conteúdo. A idéia das várias técnicas de apresentação adaptativas é permitir que o conteúdo de uma determinada página seja adaptado para um usuário em particular levando em consideração o seu conhecimento atual, suas metas de aprendizagem e outras características relevantes. Por exemplo, um usuário mais qualificado pode ter acesso a informações mais detalhadas e avançadas sobre um determinado conceito do que um usuário de nível iniciante que deve receber informações básicas para compreender o assunto em questão [BRU96a], [BRU96].

2.2.2 Adaptabilidade no Suporte a Navegação

A idéia das técnicas de adaptabilidade no suporte à navegação é auxiliar o usuário a encontrar seus caminhos no hiperespaço, prevenindo que o usuário se perca. Existem várias técnicas de suporte à navegação. Segundo [BRU96] são cinco as principais técnicas de apresentação de *links* adaptativos: *percurso orientado, ou direct guidance, sorting, hiding, annotation* e *map adaptation*.

Percurso Orientado

A técnica percurso orientado (em inglês *direct guidance*) é uma das mais simples no que tange o suporte à navegação adaptativa. Pode ser aplicada em qualquer tipo de sistema auxiliando na decisão de qual o melhor *link* a ser visitado pelo usuário. Esta decisão é tomada de acordo com as metas do usuário e outros parâmetros apresentados no modelo do usuário que consiste em identificações e preferências relativas ao usuário. Para prover esta técnica, o sistema pode salientar o *link* indicado como o melhor para o usuário. Um problema encontrado é que ela não provê suporte para os usuários que não utilizarem suas sugestões de navegação. É uma técnica útil, porém deve ser empregada com outras tecnologias de suporte a navegação [BRU99a].

Adaptive ordering

A idéia da *adaptive ordering* ou *sorting* é ordenar todos os *links* de uma página em particular de acordo com o modelo do usuário e alguns critérios como relevância do *link*, por exemplo, quanto mais próximo do topo mais relevante é o *link*. Esta técnica é utilizada extensivamente para índices e páginas de conteúdo, e nunca pode ser utilizada em *links* contextuais e mapas [BRU96].

Um outro problema com a técnica *adaptive ordering* é que esta mantém a ordem dos *links* instável: podem existir modificações na ordem dos *links* a cada vez que

o usuário acessa uma página. Ao mesmo tempo, algumas pesquisas mostram que a ordem estável do menu de opções é importante para usuários novatos, entretanto esta tecnologia é útil para aplicações de recuperação de informação. Alguns experimentos mostram que esta técnica pode reduzir significativamente o tempo de navegação em aplicações de Recuperação de Informações, em que cada página pode possuir muitos *links* não contextualizados, ou seja *links* que são independentes do conteúdo instrucional da página, que aparecem como uma lista, um conjunto de botões ou um menu *pop-up* [BRU96a].

Hiding

Esta é uma das técnicas mais freqüentemente utilizada para suporte a navegação adaptativa. Sua idéia está em restringir o espaço de navegação escondendo *links* para páginas não relevantes. Uma página pode ser considerada irrelevante por várias razões: por exemplo, se esta não está relatada na meta atual do usuário, ou se esta apresenta materiais que o usuário ainda não está preparado para entender.

Superficialmente, *hiding* é considerada uma das mais simples técnicas a ser implementada e a mais óbvia. Ela protege o usuário da complexidade do hiperespaço e reduz sua sobrecarga cognitiva. Esta técnica tem uma larga aplicabilidade: pode ser utilizada com todos os tipos de *links* não contextualizados, índices ou mapas, através da omissão de botões ou itens do menu, e também pode ser utilizada com *links* contextualizados, que estão localizados no texto da página, através da transferência de *links* evidentes para texto normal. É uma técnica mais transparente para o usuário, que parece ser mais estável do que a *adaptive ordering*, pois os *links* são adicionados, de maneira incremental, porém não são removidos ou reordenados [BRU96].

Adaptive annotation

Esta técnica tem como idéia o aumento de *links* com alguma forma de comentários ou sugestões que podem dizer ao usuário mais sobre o estado atual dos nós. Esta anotação pode prover informação na forma textual ou na forma visual com diferentes ícones, cores, ou tamanho de fontes. *Link Annotation* ou *Adaptive annotation* é uma técnica efetiva do suporte a navegação em hipermídia [BRU96].

Geralmente *annotation* é considerada uma técnica mais poderosa do que *hiding*, pois a segunda pode distinguir apenas entre dois estados dos nós (relevante ou irrelevante), enquanto que a primeira pode possuir vários níveis de relevância. *Annotation* não reduz a sobrecarga cognitiva tanto quanto *hiding*, porém a tecnologia *hidding* pode ser facilmente simulada pela tecnologia de anotação utilizando o recurso de esmaecer os *links* não relevantes. O uso deste processo pode diminuir a sobrecarga cognitiva em alguns domínios, porém os *links* esmaecidos continuam visíveis apesar de estarem protegidos e não permitirem ao usuário a construção de mapas mentais [BRU96].

Map adaptation

Esta técnica permite a adaptabilidade dos mapas de navegação modificando sua estrutura, restringindo o número de *links* e conseqüentemente o número de possibilidades de navegação [HEN2000]. As outras tecnologias citadas anteriormente também podem ser utilizadas para adaptar mapas de hipermídias, porém apenas esta altera a forma ou a estrutura do mapa [BRU96].

Estas são algumas das mais clássicas técnicas para suporte a navegação adaptativa. Existem outras técnicas para esta finalidade, porém as cinco citadas nesta seção são base para o desenvolvimento de novas. Percurso Orientado, ou *Direct guidance, sorting, hiding, annotation* e *map adaptation* podem ser utilizadas de forma individual ou em conjunto, pois uma técnica não anula a outra.

2.3 Características adicionais que podem ser utilizadas para prover adaptação

Além das adaptabilidades de suporte à navegação e de apresentação existem outros aspectos que podem ser adaptados, estes consideram quais características o sistema pode adaptar, podendo haver conteúdos distintos para usuários diferentes ou conteúdos distintos para um mesmo usuário em momentos diferentes. Segundo [BRU96] foram identificadas cinco características que são utilizadas por sistemas hipermídia adaptativos existentes para tratar o usuário como uma pessoa individual. São elas: metas, conhecimento, *background*, hiperespaço, experiência e preferências.

2.3.1 Conhecimento

Uma das mais importantes características do usuário para os sistemas hipermídia adaptativos é o conhecimento deste sobre o assunto representado. O conhecimento é uma característica variável e particular ao usuário, pois os sistemas que a utilizam devem reconhecer as modificações no estado do conhecimento do usuário e devem, de acordo com estas informações, atualizar o modelo do usuário.

O conhecimento do usuário sobre um determinado assunto freqüentemente é representado por um modelo denominado modelo de *overlay*. O modelo de *overlay* é baseado no modelo estrutural do domínio. Geralmente o modelo estrutural do domínio é representado através de uma rede de conceitos do domínio. Os conceitos são relacionados entre si e formam uma rede semântica que representa a estrutura deste domínio. Estes conceitos podem receber diferentes nomes em diferentes sistemas, tais como: tópicos, elementos do conhecimento e objetos, porém todos são considerados como pedaços elementares do domínio total [BRU96].

Grande parte dos sistemas utiliza uma organização avançada do domínio do conhecimento, com uma série de conceitos que representam diferentes tipos de elementos do conhecimento ou objetos, e vários tipos de *links* que representam diferentes relacionamentos entre estes conceitos.

A idéia deste modelo de *overlay* é representar o conhecimento dos usuários de maneira individual. Para cada conceito do modelo de domínio, um modelo de *overlay* armazena alguns valores com uma estimativa sobre o nível de conhecimento do usuário sobre este conceito. Este valor pode ser binário (sabe – não sabe), um valor qualitativo (bom – médio – pouco), ou um valor qualitativo, como uma probabilidade do usuário saber o conceito.

Um modelo de *overlay* de um usuário pode representar um conjunto de pares "conceito-valor", sendo um destes pares para cada conceito do domínio. Modelos de *overlay* são poderosos e flexíveis, eles podem de maneira independente medir o conhecimento do usuário em diferentes conceitos. Foram originariamente desenvolvidos na área de Sistemas Tutores Inteligentes (STI) e Modelagem do estudante [BRU96]. Em

muitos STI o modelo do estudante é apenas um modelo de *overlay* do conhecimento do estudante. Como resultado, na área de interfaces adaptativas, um modelo de *overlay* do conhecimento do estudante é em alguns casos chamado de modelo do estudante.

Algumas vezes um simples estereótipo do modelo do usuário é utilizado para representar o seu conhecimento. Podem existir estereótipos para classificar o conhecimento do aluno com relação a um determinado assunto, tais como: iniciante, intermediário e avançado. Alguns modelos podem necessitar de mais classes com estereótipos, por exemplo, para verificar o nível do usuário com relação ao conhecimento de inglês e o conhecimento de português. O usuário pode ser iniciante com relação ao requisito inglês, porém pode ser considerado avançado com relação ao requisito português.

2.3.2 Metas do usuário

Meta do usuário ou tarefa do usuário é uma característica relacionada com o contexto de trabalho deste no hipermídia, como uma entidade individual. Dependendo do tipo do sistema, pode ser uma meta de trabalho, uma descoberta, uma resolução de problema ou uma meta de aprendizado (para sistemas de ensino como o AdaptWeb). Em todos estes casos a meta é a resposta a seguinte questão: “Por qual razão o usuário utiliza o sistema hipermídia e o que ele realmente deseja alcançar?”. A resposta desta pergunta traz explicitamente qual o objetivo do usuário com relação à utilização do hipermídia [BRU96].

Esta é uma das características que mais se altera, pois ela pode modificar de seção para seção e freqüentemente pode modificar várias vezes em uma mesma seção de trabalho. Em alguns sistemas é interessante distinguir metas locais ou de baixo nível, as quais podem ser modificadas com mais freqüência, e as metas ou tarefas de alto nível que são mais constantes. Por exemplo, em sistemas educacionais a meta de aprendizagem é considerada uma meta de alto nível, pois é estável, enquanto que a meta de resolução de problemas é tratada como de baixo nível, pois esta segunda é alterada de um problema educacional para outro em várias ocasiões dentro de uma mesma seção [KOB2001].

2.3.3 *Background* e Experiência

Background e experiência são duas características que parecem ser similares ao conhecimento do usuário, porém funcionalmente elas são diferentes.

Toda a informação que o usuário precisa e que foi adquirida fora do sistema hipermídia é considerada *background* do usuário. Isto inclui a profissão do usuário, experiências adquiridas no trabalho ou em áreas relacionadas assim como o ponto de vista do usuário e sua perspectiva sobre um determinado assunto. O background, em alguns sistemas, é incluído no modelo do usuário, e utilizado na adaptabilidade de apresentação ou de navegação [BRU96].

Experiência está relacionada com a familiaridade do usuário com a estrutura do hiperespaço e quão facilmente ele pode navegar no sistema. Não é o mesmo que conhecimento sobre o domínio. Algumas vezes, o usuário que já é considerado familiar com o domínio, entretanto não está familiarizado com a estrutura do hiperespaço. O contrário também é válido, o usuário pode estar familiarizado com a estrutura do hiperespaço, porém pode não ter um conhecimento profundo do assunto [BRU96].

2.3.4 Preferências

Esta característica está relacionada com as preferências que o usuário possui no sistema. Por diferentes razões o usuário pode preferir um *link* a outro, ou uma parte da página a outra.

Preferências do usuário diferem dos outros componentes do modelo do usuário em vários aspectos. As preferências do usuário podem ser deduzidas pelo sistema. O usuário pode informar ao sistema suas preferências de maneira direta, isto é, através de questionários, ou então de maneira indireta, através de ações que se repetem durante a execução de determinadas tarefas no sistema [BRU96].

2.4 Ambientes Hipermídia Adaptativos

Uma aplicação hipermídia considerada clássica oferece os mesmos conteúdos e os mesmos conjuntos de *links* para todos os usuários. Os ambientes hipermídia adaptativos tornam possível a disponibilização de um mesmo material para usuários diferentes sem que o autor necessite programar esta adaptabilidade [BRU99].

Ambientes hipermídia adaptativos constroem um modelo baseado nas metas, preferências e conhecimento de cada usuário, e utilizam as informações deste modelo em todas as interações com o usuário para se adaptar as necessidades deste. Por exemplo, um estudante que utiliza um sistema hipermídia adaptativo terá uma apresentação que é adaptada especificamente para o seu conhecimento sobre determinado assunto e terá ainda a sugestão de um conjunto de *links* que serão mais relevantes nos procedimentos do estudante em momentos futuros na utilização do sistema [BRU2001].

Para estruturar e armazenar todas as informações sobre o modelo do aluno e sobre o conteúdo instrucional existem alguns sistemas hipermídia que utilizam XML [W3C2000], pois esta é uma meta-linguagem que permite a criação de marcações para propósitos específicos. Além disto XML mantém uma separação entre o conteúdo a ser exibido e a sua apresentação.

Esta seção descreve alguns sistemas hipermídia adaptativos que utilizam XML no seu processo de manipulação das informações instrucionais.

2.4.1 AHA! (Adaptive Hypermedia for All)

O AHA (*Adaptive Hypermedia for All*) é um exemplo de sistema hipermídia adaptativo proposto por [BRA98]. Baseia-se em [BRU96] utilizando o conceito de modelo de usuário. O modelo do usuário no AHA! possui alguns aspectos sobre o usuário como suas preferências, e um modelo de *overlay* que mostra como o usuário relaciona as páginas e os conceitos do modelo de domínio. No AHA! o modelo do usuário consiste em um conjunto de pares atributo-valor para cada conceito ou página.

O modelo do usuário é atualizado a cada vez que um usuário visita uma página, assim pode ser utilizado para determinar como a apresentação da 'próxima' página deve ser adaptada e quais recursos devem ser fornecidos ao usuário [BRA2002].

Segundo [BRA2002] as funções principais do AHA! são:

- Para cada usuário existe um modelo de usuário que representa como este se relaciona com as páginas e os conceitos do *site.* A cada vez que o usuário visita uma página um conjunto de regras determina como este modelo deve ser atualizado. Esta atualização ocorre por meio de atributos do usuário que são pré-definidos pelo autor. Cada página ou conceito também possui atributos relacionados. Então é feita uma comparação entre os atributos da página e os escolhidos pelo autor e através da análise do resultado desta comparação o modelo do usuário é atualizado;
- Cada página contém zero ou mais fragmentos que são condicionalmente incluídos na apresentação, esta é a técnica de apresentação adaptativa que o AHA! utiliza;
- Para cada página há um conjunto de regras. Isto é utilizado para a adaptabilidade de navegação. O AHA! usa diferentes cores para *links* a fim de indicar diferentes situações, como por exemplo que um *links* está desabilitado. Dependendo da escolha das cores pode ser configurado para utilizar as técnicas *link hiding* (com *links* indesejáveis em preto) ou *link annotation* (com todos os *links* em cores visíveis e diferentes).

O AHA! utiliza XML e HTML para o armazenamento do conteúdo das páginas e das regras de adaptabilidade associadas a estas páginas. [BRA2002a].

Além de utilizar XML no conteúdo a ser disponibilizado, o AHA! possui o modelo do usuário armazenado em um arquivo XML. Assim XML é utilizado para prover adaptabilidade na navegação e na apresentação.

2.4.2 Adaptive Intelligent Hypermedia using XML

De acordo com [BON2000] o domínio do conhecimento é organizado através de uma estrutura de nós e relacionamentos entre estes. Os nós e seus relacionamentos estão definidos utilizando um conjunto de *tags* implementados com XML. Nesta definição os nós incluem:

- tópicos do domínio de conhecimento;
- conceitos, que definem palavras chaves associadas a um tópico específico;
- explicações, que são textos utilizados para simplificar tópicos complexos;
- condições que indicam um pré-requisito didático. Em particular estas condições definem que tipo de pré-requisitos um estudante necessita para acessar um determinado nó.

Um exemplo de como os tópicos são organizados em uma estrutura de grafos de acordo com [BON2000] pode ser visto na figura 2.1:

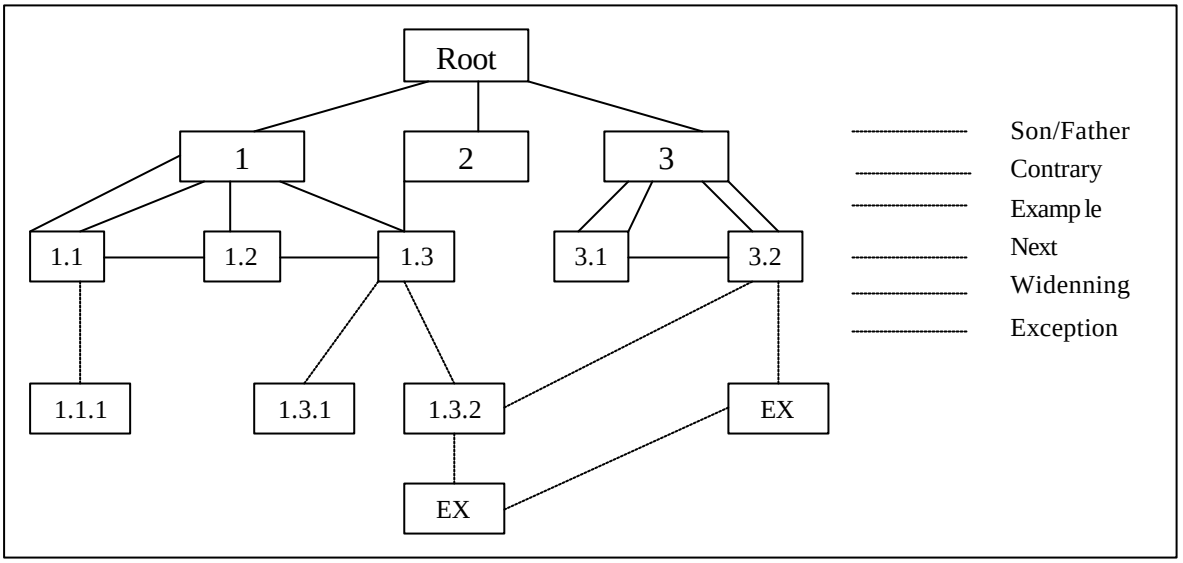

FIGURA 2.1 –Exemplo de estruturação do domínio do conhecimento

Os relacionamentos mostrados na figura 2.1 são:

- *Son/father*: relacionamento bi-direcional entre dois nós que define uma hierarquia;
- *Next*: relacionamento entre um conceito e o próximo conceito que deve ser aprendido pelo estudante.
- *Contrary*: relacionamento bi-direcional entre um conceito e o antônimo deste. Por exemplo, o conceito "programação *top-down*" é o conceito oposto de "programação *bottom-up*" e vice-versa.
- *Exemple*: relacionamento entre um conceito e um exemplo que o explica.
- *Widening:* relacionamento existente entre um conceito e um item de estudo deste conceito, mais detalhado que se encontra em profundidade.
- *Exception*: relacionamento entre um conceito e uma exceção ou relacionamento entre um conceito e um caso particular deste.

O estudante pode navegar no grafo resultante seguindo todos os tipos de relacionamentos e definindo assim seu caminho pessoal de aprendizagem. A figura 2.1 mostra um relacionamento simples de uma estrutura de domínio de conhecimento apresentado a um estudante em um certo momento. Um elemento XML corresponde a cada nó ou a cada relacionamento. Os elementos XML consistem de uma *tag* de início e fim com atributos opcionais e descrição em *hypertexto* de cada nó, que pode ser escrita em HTML [BON2000]. A linguagem XML é detalhada no capítulo 3.

Um exemplo de código XML utilizado para estruturar este domínio de conhecimento é descrito na figura 2.2:

```
<TOPIC>
        …
        Descrição em HTML do Tópico
        …
        <CONDITION USER_LEVEL="3">
                <EXPLANATION>
                        …
                        Descrição de explanação em HTML
                        …
                </EXPLANATION>
        </CONDITION>
</TOPIC>.
```

FIGURA 2.2 – Código XML para estruturar domínio de conhecimento

Um agrupamento *condition* (*tag* de início e fim) é associado a cada relação e define o tipo de exigências que o estudante precisa para seguir entre dois nós relacionados. O *link* só esta disponível se o modelo do estudante satisfizer a todas as *condition*.

Não são aplicadas condições à relação *son/father* e o *link* associado sempre aparece, sem exigências. Todos os outros *links* podem ser condicionados, e em particular estes podem ter valores diferentes em função do estado do modelo do estudante. Isto significa que mais de um próximo *link* pode ser associado a um certo nodo e cada próximo *link* é válido a partir de um conjunto diferente de exigências [BON2000].

Atualmente é utilizado um esquema de condição indicando um requisito didático. Na figura 2.2 a condição está descrita na *tag* <CONDITION USER LEVEL="3">, esta condição está relacionada com o nível de conhecimento do aluno. Neste caso um valor para o nível do usuário é requerido. Os estudantes que se possuem o nível do usuário igual a 3 no modelo de usuário estão aptos a visualizar a explanação sobre o tópico em questão. As condições podem possuir mais de um termo, utilizando para isto operador *and* ou operador *or*. Compondo termos de condições com operadores de *and* e *or* é possível obter exigências didáticas complexas [BON2000].

.

3 XML para Estruturação de Conteúdo Instrucional

A disseminação de páginas disponibilizadas na *World Wide Web* tornou o uso do HTML *(Hyper Text Markup Language)* popular. Estas páginas desenvolvidas em HTML evoluem a cada dia com o desenvolvimento de novas aplicações e tecnologias. Os *sites* se encontram cada vez mais dinâmicos, personalizados e interativos, tentando oferecer mais facilidades para o usuário. Entretanto, aspectos como estruturação e manutenção de conteúdo destes *sites* ficam cada vez mais complexos devido à limitação das *tags* da linguagem HTML [ABI99].

Na tentativa de minimizar limitações do HTML, a W3C (*World Wide Web Consortium*) propôs o padrão XML para atuar na descrição e troca de dados na Web [ROY2001]. Este padrão foi criado para descrever o conteúdo dos documentos, e projetado para ser "diretamente utilizável na Internet", conforme definição do Grupo de Trabalho XML do W3C. XML produz documentos que podem ser facilmente descritos, criados e manipulados, e também transferidos e visualizados na Internet, trazendo consigo toda a estrutura e descrição de seu conteúdo [W3C2000].

Neste capítulo são descritas características da linguagem XML, abordando sua sintaxe, semântica e propriedades específicas como DTD, XML *Schema* entre outras. Também é explorado o uso de XML para estruturação de conteúdo instrucional.

3.1 A Linguagem XML

Uma linguagem de marcação utiliza rótulos ou marcas (*tags*) para descrever o conteúdo e estruturar informações contidas em um documento. O HTML *(Hyper Text Markup Language)* é uma linguagem de marcação baseada em SGML (*Standard Generalized Markup Language*) [ISO86], uma metalinguagem padrão definida antes mesmo do surgimento da própria *Web* e que especifica gramáticas para linguagens de marcação de documentos.

SGML permite descrever a gramática utilizada em um documento especificando o conjunto de *tags* usados no documento e o que essas *tags* representam [BRY97]. Documentos HTML reúnem um subconjunto de *tags* (*tagset*) reduzido, em conformidade com a especificação SGML, ou seja, um conjunto fixo de *tags* com utilização e significado definidos [TEI2001].

XML (*Extensible Markup Language*) também é considerada uma linguagem de marcação, voltada para a descrição de documentos. Foi projetada para ser um subconjunto de SGML, porém com a função específica de publicação de dados na *Web*, como o HTML. Não possui uma gramática fixa, como o HTML, mas sim extensível, voltada à descrição e marcação, apropriada para a representação de dados e documentos cuja essência é fundamentada na capacidade de agregar informações [W3C2000].

XML foi criada como uma SGML menos complexa, usada para descrever o conteúdo dos documentos, e projetada para ser utilizada na Internet, conforme definição do Grupo de Trabalho XML do *World Wide Web Consortium* (W3C).

Oferece mais flexibilidade que HTML convencional para definir *tags*, pois mudanças na HTML requerem uma atualização do padrão, ou seja, como HTML possui um *tagset* fixo, seria necessária a inserção ou exclusão de *tags* direto no conjunto de *tags* padrão, gerando uma nova versão da linguagem. Já XML é extensível, permitindo que os autores de documentos *web* criem suas próprias *tags*. Isso possibilita uma rica

troca de informações entre *sites web*, desde que, tanto o remetente quanto o destinatário, usem as mesmas definições de *tags*.

XML é uma linguagem que estabelece regras gerais, as quais documentos em conformidade com XML devem respeitar, porém não é uma gramática fixa, como o HTML. Por esta característica nota-se que XML possibilita estender o conjunto de *tags* original, permitindo então que autores ou comunidades criem seus próprios conjuntos de marcas, definidos para melhor caracterizarem os documentos de interesse [W3C2000], [PIM2000].

Foi desenvolvida para ser altamente expressiva, de fácil aprendizado e implementação. Segundo [BOS98], XML deve ser utilizada principalmente em quatro amplas categorias:

- Aplicações para Web que acessem bancos de dados heterogêneos.
- Aplicações que distribuam a carga de processo do servidor para o cliente.
- Aplicações que exijam visões diferentes dos mesmos dados, dependendo do cliente.
- Aplicações voltadas para necessidades individuais de usuários.

As duas últimas categorias são de grande importância para o ambiente AdaptWeb, visto que os conteúdos instrucionais terão usuários com diferentes perfis tendo visões distintas do mesmo documento.

3.2 Documento XML

Cada documento XML possui uma estrutura física e uma estrutura lógica. No âmbito da estrutura física o documento é composto por unidades chamadas de entidades. Uma entidade pode referenciar outras entidades, tendo assim suas inclusões no documento XML. Um documento XML se inicia com uma entidade raiz. Com relação a estrutura lógica, o documento é composto de declarações, elementos, comentários, caracteres de referência e instruções de processamento e de tipo de documento (DTD) [W3C2000].

3.2.1 Estruturas Lógicas em Documentos XML

Dentre as estruturas lógicas de um documento XML as principais são: declaração XML, a qual é preparada e geralmente compreendida por ferramentas compatíveis com XML; DTD ou o XML*Schema*, que é a base de documentos válidos; e o documento XML propriamente dito, ou instância de documento, onde o conteúdo é definido pelas *tags* [LAN98]. Estes três tipos de estruturas lógicas são formados por elementos e atributos que funcionam como estruturas lógicas mínimas.

3.2.1.1 Elementos e Atributos

Um elemento pode ser definido, de maneira geral, como qualquer cadeia de caracteres que esteja entre os caracteres delimitadores < e >, excluindo os comentários e a seção CDATA. Existem elementos de abertura e de fechamento. Os de abertura sempre se iniciam com o caractere < seguido do nome do elemento, analogamente, os

de fechamento começam com os caracteres </ também seguidos do mesmo nome utilizado no elemento de abertura [PIM2000]. Um exemplo de elemento é:

<NOMETAG> Conteúdo da *Tag* </NOMETAG>

Todo elemento de abertura deve estar associado a um elemento de fechamento.

Os nomes de elemento são *case sensitive* (diferenciam letras maiúsculas de minúsculas), devem começar por letras ou pelo caractere "_" (*underline*). Os demais caracteres podem ser letras, dígitos, hífens, pontos e novamente o caractere "_", porém não são aceitos espaços em branco [PIM2000].

Outra alternativa de manipulação de elementos é o elemento vazio. Ele assume tanto a função de elemento de abertura como de fechamento. Deve ser iniciado também pelo caractere <, porém deve terminar com /> [PIM2000]. O elemento NOME representado acima fica com a seguinte sintaxe se for elemento vazio:

<NOMETAG *conteúdo da tag* />

Todo documento XML começa com um elemento raiz, que não pode ser usado em outro lugar do documento. A *tag* <HTML> é um exemplo de elemento raiz em documentos HTML, abrindo e fechando cada documento. Os elementos dentro do elemento raiz devem ser organizados, de forma que cada elemento deve estar dentro do seu elemento pai e não devem ocorrer fechamentos cruzados de *tags* [W3C2000].

Estes elementos descritos até agora podem ou não conter um ou mais atributos, que são pares nome-valor separados pelo caractere =. São aplicadas as mesmas regras de nomenclatura de elementos para os atributos. O conteúdo dos atributos deve estar delimitado por aspas [W3C2000].

A informação que descreve o elemento e seu conteúdo é chamada de atributo. Cada atributo pode conter um ou mais valores, dependendo de sua definição, mas não podem conter sub-atributos [TEI2001]. Eles são definidos dentro das *tags* de início. Um exemplo de atributo é:

<curso nomecurso="Computação">

O elemento curso possui o atributo nomecurso que neste exemplo está sendo valorado com Computação.

3.2.1.2 Declarações XML

Os documentos XML devem começar com uma declaração XML, que especifica a versão XML que está sendo utilizado. O número de versão "1.0" deve ser utilizado para indicar que o documento em questão está em conformidade com esta versão da especificação XML [W3C2000]. Esta definição consta no documento XML com a seguinte linha:

<?XML version="1.0"?>

Nesta declaração também pode haver a indicação da versão do *character set*, ou seja, conjunto de caractere que será utilizado no documento XML. Ao incluir a definição do conjunto de caracter a linha de declaração é:

<?XML version="1.0" encoding="UTF-8"?>

Segundo [PIM2000] todas as aplicações XML devem suportar, pelo menos, as codificações UTF-8 e UTF-6 da ISO 10646.

A declaração XML também pode conter diretrizes sobre a necessidade de processar toda ou uma única parte de uma DTD. O conceito de DTD é explorado na próxima seção. Essas diretrizes são chamadas RMD (*Required Markup Declaration*) e possuem o seguinte formato:

<?XML version=”1.0” encoding=”UTF-8” RMD=”INTERNAL”?>

O valor INTERNAL significa que o processador XML deve ler e processar o subgrupo interno da DTD para interpretar o documento corretamente. O valor NONE significa que o documento pode ser analisado corretamente sem acessar nenhuma DTD. O valor ALL, que é assumido como padrão, significa que subgrupos internos e externos da DTD precisam ser lidos para que o documento seja interpretado corretamente [W3C2000].

Por fim, porém não menos importante que as demais definições da declaração XML, está a definição dos requisitos de marcação. É utilizada a declaração *standalone*, que indica a presença ou não de DTD. Se a declaração *standalone* recebe valor *yes* significa que não há declaração de marcação externa direta ou indireta. O valor *no* significa que marcações externas podem existir.

3.2.1.3 DTD

A DTD (*Document Type Definition*) define a gramática de uma classe de documentos XML. A DTD permite representar a dimensão estrutural de um documento [W3C2000] [PIM2000], pois restringe os elementos e atributos que podem ser utilizados na caracterização do documento, define ordem, encadeamento, aninhamento de elementos entre outras regras. Com a definição de DTD surge o conceito de documento válido e documento bem formado.

O documento válido obedece a todas as regras estabelecidas na DTD (interna e/ou externa) e no cabeçalho do documento. Se o documento XML é considerado válido, o mesmo está em conformidade com a especificação SGML (*Standard Generlized Markup Language*) e também é considerado um documento XML válido. Isso transforma o documento em algo compatível com qualquer meio que interprete XML [TEI2001], [W3C2000], [LIG99].

Um documento XML válido inicia com a definição da DTD, que pode conter o seguinte [LIG99]:

- Descrição de regras estruturais que a marcação no documento deve seguir;
- Declaração de recursos externos (entidades externas) que podem formar parte do documento;
- Declaração de recursos internos (entidades internas) que podem ser requeridas pelo documento;
- Declaração de tipos de recursos que não são XML (notações e entidades de dados binários) que podem ser encontradas no documento;

O documento bem formado é o documento perfeitamente estruturado, porém não necessariamente válido. Todos os elementos estão posicionados e declarados de forma correta, mas a estrutura lógica não necessariamente possui regras definidas via DTD [TEI2001].

DTD Externa ou Interna

A DTD pode ser um sub-conjunto externo (como um tipo especial de entidade externa) contendo as declarações de marcações, pode conter estas declarações de marcações diretamente no documento (internas) ou pode assumir estas duas configurações [W3C2000].

A sintaxe de uma declaração de DTD possui o seguinte formato:

<!DOCTYPE NOME TIPO-IDENTIFICADOR IDENTIFICADOR [DECLARAÇÕES-SUBGRUPO-INTERNO]>

Abaixo segue um exemplo de declaração de DTD externa, apontando para o arquivo Estrutura_Topico.dtd:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE material SYSTEM "Estrutura_Topico.dtd">
```

Os documentos que têm o atributo de declaração XML *standalone* igual a *yes* ou não possuem DTD, devem fornecer as declarações de marcação no subsistema interno, como neste exemplo:

```
<?xml version=”1.0” standalone=”yes”?>
<!DOCTYPE material[
<!ELEMENT material(conteudo+)>
<!ELEMENT conteudo (#PCDATA)>
<!ATTLIST material matDisc CDATA>]>
```

Na DTD são definidos os elementos e atributos relacionados a estes. No exemplo acima foi definido um elemento material e um elemento conteúdo. O elemento material possui um atributo matDisc. Para que as regras do documento XML fiquem claras, na DTD devem ser especificados os tipos dos atributos e elementos, bem como os indicadores de ocorrência destes. A sintaxe e particularidades destas características são explanadas a seguir.

Indicadores de ocorrência

Os elementos podem conter um recurso chamado indicador de ocorrência para definir o número de vezes que um grupo ou elemento pode ser utilizado. Existem quatro tipos de indicadores de ocorrência, com diferentes graus de opção, são eles:

- sem símbolo, define como único e requerido;
- +, define como requerido e repetível (uma ou mais vezes);
- *, define como opcional e repetível (zero ou mais vezes);
- ?, define como opcional e não repetível (uma ou zero vezes).

O exemplo abaixo mostra a declaração do elemento conceito que é composto por conteúdo e curso. Conteúdo é considerado único e requerido, por tal razão não

possui marcador algum; curso é repetível e requerido, para isto é utilizado o símbolo '+'.

<!ELEMENT conceito (conteudo,curso+)>

Tipos de Atributos

Os atributos possuem tipos pré-definidos. Os tipos de atributos são declarados na DTD e podem ter um dos seguintes valores [LIG99]:

- CDATA – para atributo do tipo string;
- ID – para identificadores. Um identificador é considerado um nome exclusivo no documento;
- IDREF - é um ponteiro para um atributo do tipo ID. Um elemento com um ID de valor referenciado deve ser incluído em algum lugar do documento XML;
- IDREFS – é uma lista de IDREF, separada por espaços;
- ENTITY – deve ser o nome de uma entidade externa. Funciona como um ponteiro para uma entidade externa;
- ENTITIES – é uma lista de ENTITY, separada por espaços;
- NMTOKEN – é essencialmente uma palavra sem espaços;
- NMTOKENS – é uma lista de NMTOKEN, separada por espaços.

Os atributos também podem possuir na sua declaração uma definição para especificar como o processador XML deve proceder no caso de não informação do atributo em questão. Existem três alternativas para este controle [W3C2000]:

- #REQUIRED – significa que o atributo é obrigatório;
- #IMPLIED – o processador avisa à aplicação que nenhum valor foi pré-fixado para este atributo, ou seja, ele é opcional;
- #FIXED – significa que o atributo deve possuir o mesmo dado definido no valor padrão.

3.2.1.4 XML Schema

O objetivo do XML *Schema* é definir uma classe de documentos XML, assim como definir o termo "instância do documento", que é freqüentemente utilizado para designar um documento XML que está em conformidade com um esquema em particular [W3C2001].

XML *Schema* oferece recursos para a definição da estrutura e para imposição de restrições ao conteúdo de documentos XML, provendo assim um superconjunto dos recursos disponibilizados por *Document Type Definitions* (DTD) [PIM2001].

O XML *Schema* descreve a estrutura legal, conteúdo e restrições dos documentos XML. A idéia do XML *Schema* é similar ao da DDL (*Data Definition Language*) dos bancos de dados relacionais. Nos bancos de dados relacionais, a DDL é usada para a criação do banco de dados, estabelecer regras e restrições para as tabelas criadas. Com o XML *Schema* é possível criar os documentos XML especificando uma

estrutura válida, restrições e os tipos de dados para os elementos e atributos que compõe um documento [ROY2001].

Componentes do XML Schema

Assim como uma DTD, um XML *Schema* pode definir elementos que contêm subelementos ou atributos, estes são ditos terem tipos complexos, enquanto que elementos contendo apenas cadeias de caracteres são ditos terem tipos simples. Alguns elementos possuem atributos e os atributos sempre são considerados tipos simples [W3C2001].

Um XML *Schema* consiste de um elemento e uma variedade de subelementos, os quais permitem especificar com mais rigor o conteúdo possível nas instâncias dos documentos correspondentes. Estes elementos podem ser dos tipos element, complexType e simpleType [PIM2001].

Cada um dos elementos de um esquema possuem o prefixo xsd:, este é associado com o XML *namespace* através da declaração "xmlns:xsd="http:// www.w3c.org/XML/Schema" que consta como atributo na definição do elemento *Schema*. O prefixo xsd: é utilizado por convenção para denotar o *namespace* do XML *Schema*, porém pode ser utilizado qualquer prefixo. O mesmo prefixo, e por conseqüência a mesma associação também podem aparecer nos nomes de tipos simples. Como exemplo tem-se o uso do prefixo com o tipo de dados string:

xsd:string

O propósito da associação é identificar elementos e tipos simples como pertencentes ao vocabulário da linguagem XML *Schema*, e não apenas ao vocabulário do esquema do autor [W3C2001].

Existe uma diferença básica entre os tipos complexos e simples, uma vez que apenas os tipos complexos permitem que os elementos contenham outros elementos ou atributos associados [W3C2001].

Há também outra distinção importante entre a definição que cria novos tipos, tanto simples como compostos, e aquelas declarações que permitem o uso de instâncias de elementos ou atributos com nomes e tipos específicos. Tipos complexos novos são definidos através do elemento xsd:complexType, de modo geral estas definições contêm um conjunto de declarações, referências a elementos e declarações de atributos. As declarações não são tipos em si e sim associações entre o nome correspondente e as restrições impostas pelo esquema. Elementos e atributos são declarados com os elementos xsd:element e xsd:attribute respectivamente [W3C2001], [PIM2001].

Um tipo simples e sem elementos pode ser declarado da seguinte maneira: <xsd:simpleType name="material">. Já atributo pode ser declarado assim:

<xsd: attribute name="nomedisciplina">

A declaração de um elemento complexo pode ser desenvolvida no formato a seguir:

```
<xsd: complexType name="conceito">
    <xsd: element name="indice" type="xsd:id">
    <xsd: element name="conteudo" type= "xsd:string">
</xsd: complexType>
```

O elemento conceito é dito tipo complexo e é composto pelo elemento índice que é do tipo *indexType* e pelo elemento conteúdo que é do tipo *string*.

Uma outra forma de declaração de elementos é utilizar um elemento já existente que faz referência a um elemento declarado em algum ponto do esquema. Regra geral, o nome de um atributo precisa referenciar um elemento global declarado no primeiro nível do esquema, e não como parte de um tipo complexo. Tanto elementos como atributos podem ser declarados globalmente [PIM2001].

XML *Schema* permite definição de número de ocorrências de um elemento, para isto são utilizados os atributos para valor mínimo *minOccurs* e máximo *maxOccurs* que devem ser inteiros e positivos. No caso de *maxOccurs* este também pode assumir o valor *unbounded* para indicar que nenhum limite é especificado. O valor *default* de *minOccurs* é 1; quando o elemento *maxOccurs* não é declarado, seu valor coincide com o de *minOccurs*. Quando ambos são omitidos o elemento deve ocorrer apenas uma vez [PIM2001].

Atributos ocorrem uma ou nenhuma vez. O atributo *use* indica o fato de deste ser requerido (*required*) ou opcional (*optional*), se é definido como opcional então o atributo *value* pode assumir um valor *fixed* ou *default*. Um exemplo de atributo pode ser visto a seguir: <attribute name=”laboratório” type=”string“ use=”required”/>

Além de todas estas definições XML *Schema* também possui um conjunto de tipos simples que podem ser utilizados diretamente nas declarações de elementos e atributos. Alguns tipos simples são: string, *boolean, integer, float*, decimal, positiveinteger, time, dateTime entre outros especificados em [W3C2001]. Existe a possibilidade de criação de novos tipos simples derivados dos citados acima, estes devem possuir nomes distintos dos valores que derivam e podem impor restrições aos valores definidos originariamente através da utilização de especificações denominadas *facets.*

As demais características de XML *Schema* podem ser exploradas em [W3C2001].

3.2.2 Estruturas Físicas em Documentos XML

A estrutura física de um documento XML consiste de uma ou mais unidades de armazenamento. Estas unidades são denominadas entidades (*entities)*; todas estas entidades possuem conteúdo e são identificadas através de um nome da entidade.

As declarações de entidades permitem associar um nome com algum outro fragmento de conteúdo. Esse outro conteúdo pode ser uma parte de texto normal, um fragmento de uma DTD ou uma referência a um arquivo externo que contém dados do tipo texto ou binário.

Declarações de entidades quando referenciam um arquivo externo, especificam que porções do documento em questão estão contidas em outros documentos e devem ser incluídas quando do processamento do documento [PIM2000].

Segundo [LIG99], as entidades são utilizadas nas seguintes situações:

- Para representar caracteres que não são padrão no seu documento XML;
- Funcionar como abreviação de frases freqüentemente usadas;
- Manter partes da marcação que podem aparecer em mais de um documento XML;

- Organizar a DTD em unidades lógicas;
- Representar recursos que não são XML.

Existem três tipos de entidades: internas, externas e de parâmetro. No decorrer deste capítulo são explorados estes três tipos.

3.2.2.1 Entidades Internas

As entidades internas associam um nome com uma cadeia de caracteres ou texto literal. Se uma entidade é definida como interna então não há separação física de armazenamento [W3C2000]. Estes mecanismos permitem a definição de atalhos para textos freqüentemente digitados.

XML possui algumas entidades internas pré-definidas que são utilizadas para representar alguns caracteres da tabela ASCII. Algumas dessas entidades e suas referências que são geralmente representadas pelo símbolo & seguido do nome e ponto e vírgula, são:

TABELA 3.1 – Entidades pré-definidas para documento XML

Entidade	Caracter
&amp;	&
&lt;	<
&gt;	>
&apos;	'
&quot;	"

Podem ser definidas outras entidades internas que não sejam estas pré-estabelecidas pelo XML. Como exemplo pode-se criar uma entidade da seguinte maneira:

<!ENTITY UFRGS “Universidade Federal do Rio Grande do Sul”>

Utilizando &UFRGS; em qualquer lugar do documento XML é inserido o texto “Universidade Federal do Rio Grande do Sul” no local indicado, permitindo a definição de um atalho no texto. Além disto o uso de entidades internas facilita na manutenibilidade do texto XML, pois se o texto que é associado a entidade ocorre várias vezes no documento XML e este necessita de uma alteração basta modificar a indicação ENTITY que esta é visualizada no documento todo.

3.2.2.2 Entidades Externas

As entidades externas associam um nome com o conteúdo de um outro arquivo. Estas entidades permitem ao documento XML referenciar o conteúdo de um outro arquivo, contendo texto ou dados binários [PIM2000]. Se a entidade contiver texto, o conteúdo do arquivo externo é inserido no ponto de referência e analisado como parte do documento XML. Dados binários não são analisados e podem somente ser referenciados em um atributo; eles são usados para referenciar figuras e outro conteúdo que não seja XML.

As entidades externas de um documento XML podem ser referenciadas do seguinte modo:

<!ENTITY arquivo SYSTEM “/documentos/adaptweb.xml”>

O uso de &arquivo; insere o conteúdo do arquivo /documentos/adaptweb.xml no local referenciado no texto. O processador XML analisa o conteúdo deste arquivo como se ele ocorresse literalmente no local.

3.2.2.3 Entidades de Parâmetro

A entidade de parâmetro (ou entidades parametrizadas) ocorre somente na declaração de tipo de documento (DTD) [TEI2001]. Uma declaração deste tipo de entidade é identificada por “% ” (símbolo de porcentagem seguido de espaço em branco) antes do nome da entidade. O sinal de porcentagem também é utilizado em referências para entidades de parâmetro, ao invés do & (E comercial).

Entidades de parâmetro são imediatamente expandidas na declaração de tipo de documento e seu texto de substituição é parte da declaração, onde as referências a entidades normais não são expandidas. Entidades de parâmetro não são reconhecidas no corpo de um documento XML e sim no corpo da DTD [W3C2000].

A utilização de uma entidade em DTD (entidade de parâmetro) pode ser definida assim:

<!ENTITY % EXentidade “(#PCDATA | citacao)”>
<!ELEMENT partetexto (%EXentidade;)>

A entidade EXentidade quando referenciada na DTD é expandida com o conteúdo (#PCDATA | citação).

3.3 XML e Organização de Conteúdo Instrucional

Nos últimos anos a *World Wide Web* vem sendo utilizada em larga escala como base para atividades de ensino-aprendizagem. A organização do conteúdo instrucional utilizado nestes cursos baseados na *Web* é uma fase importante nos projetos de implementação de ensino à distância. XML surge como uma nova ferramenta para prover esta organização. Por separar o conteúdo da apresentação cria uma estrutura mais independente permitindo trabalhar melhor com conceitos como adaptabilidade.

Esta seção apresenta a utilização de XML para organização de conteúdo instrucional.

3.3.1 Como estruturar conteúdo utilizando XML

Para ser dinamicamente processado o domínio de um curso à distância deve ser estruturado. Segundo [BON2000] uma boa forma de organização de conteúdo para ensino a distância é o uso de tópicos e a associação de cada tópico com uma página *Web*, como se estes fossem nós de uma estrutura hierárquica. Estes podem ser caracterizados por vários elementos, tais como: título, palavras-chave, explicações, exercícios, exemplos, etc.

Esta organização permite também uma fácil representação da estrutura do domínio do conhecimento através de um conjunto de nós elementares interligados por relacionamentos [BRU96].

Estes nós relacionados formam uma rede semântica, eles podem ter nomes diferentes em sistemas diferentes, tais como: conceitos, tópicos, elementos do conhecimento, objetos e etc., porém em todos os casos eles são pedaços elementares do domínio de conhecimento. Em sistemas orientados a conceito, que é o caso do ambiente descrito neste trabalho, estes nós são denominados conceito.

Um exemplo deste tipo de representação, onde cada nó é um conceito, é dado abaixo:

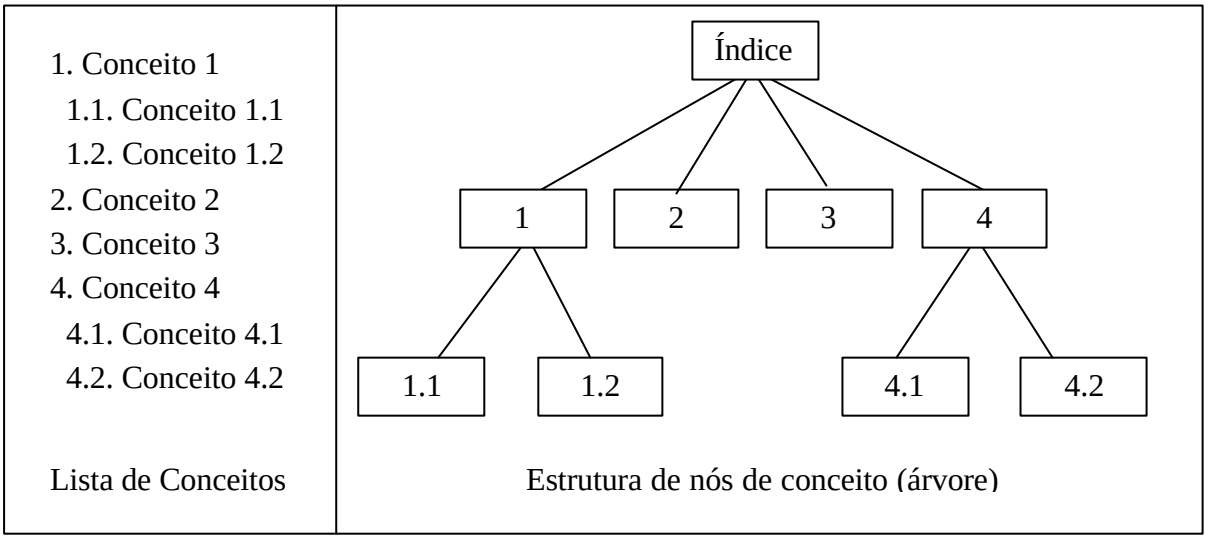

FIGURA 3.1 – Organização de um mesmo conteúdo em lista de conceitos e em Árvore de conceitos

Como pode ser observado na figura 3.1, o uso desta estrutura de nós leva a uma organização hierárquica dos elementos. Segundo [MUR98], organizações hierárquicas são praticamente universais e são muito utilizadas para organizar estruturação de conteúdo. Tanto aprendizes como autores entendem facilmente as representações baseadas em hierarquia.

XML é adequado a este tipo de desenvolvimento. As próprias *tags* de marcação do conteúdo dos documentos, podem estabelecer uma estrutura lógica, permitindo assim uma organização hierárquica que na prática é visualizada através de *tags* que contenham outras *tags*. Um elemento XML, indicado por uma *tag* de início e uma de fim, corresponde a um nó da árvore de conteúdo, ou seja, a um conceito do curso em questão.

A representação utilizando XML da figura 3.1 pode ser visualizada a seguir:

```
<indice>
        <Conceito 1>
                        <Conceito 1.1>   .............   </ Conceito 1.1>
                        < Conceito 1.2>  .............   </ Conceito 1.2>
        </Conceito 1>
        < Conceito 2>
                .......
        </Conceito 2>
        < Conceito 3>
                .......
        </ Conceito 3>
        < Conceito 4>
                        < Conceito 4.1>  .............   </ Conceito 4.1>
                        < Conceito 4.2>  .............   </ Conceito 4.2>
        </Conceito 4>
</indice>
```

FIGURA 3.2. – Esquema de código XML da representação em árvore

A *tag* indice delimita a raiz do documento XML. Cada par de *tag* <Conceito N> e </ Conceito N> representa um conceito na estrutura de itens, ou na árvore da figura 3.1. A *tag* <Conceito 1.1> é encontrada dentro da <Conceito 1>, portanto assim está representada a estrutura hierárquica de organização do curso.

Além desta facilidade para representação/organização de documentos com níveis de hierarquia, XML também separa o conteúdo da apresentação. Através do uso das folhas de estilo é possível a geração de novas apresentações para o mesmo documento XML. Assim, o arquivo XML só armazena conteúdo e detalhes sobre a apresentação podem ser configurados através de mecanismos como folhas de estilo.

4 Ambiente AdaptWeb

O AdaptWeb é um ambiente hipermídia adaptativo para *Web* que utiliza conceitos de adaptabilidade com a finalidade de disponibilizar um mesmo conteúdo para aprendizes de grupos distintos. Este ambiente propõe dois níveis de adaptabilidade: no conteúdo a ser apresentado, onde a estrutura do conteúdo é adaptada ao perfil do usuário e na navegação, apresentando ou não informações ao usuário com base em seu ambiente tecnológico, que será informado pelo usuário por intermédio de um questionário, e nos pré-requisitos estabelecidos pelo autor [AMA2002].

Os conteúdos instrucionais do ambiente são modelados através de uma estrutura hierárquica orientada pelos conceitos onde são estabelecidos critérios de pré-requisitos. Esta estrutura é definida durante a fase de autoria e posteriormente armazenada em documentos XML [AMA2002].

Os documentos XML resultantes da etapa de autoria devem passar por um processo de filtragem, antes de serem apresentados ao aluno. Os filtros acontecem dinamicamente durante a interação do aluno no ambiente e obedecem aos critérios de adaptação representados no modelo deste aluno. O conhecimento, a formação e o ambiente de trabalho do aluno são as principais características representadas neste modelo.

Neste contexto, a linguagem XML desempenha um importante papel, visto que um documento XML também possui uma estrutura hierárquica e seu conteúdo é mantido separado dos elementos de apresentação. Desta maneira, através da API (*Application Programming Interface*) DOM (*Document Object Model*), torna-se possível acessar e manipular este documento, gerando diferentes apresentações para o mesmo documento XML [DOM98], pois as próprias *tags* de marcação de conteúdo, podem estabelecer uma estrutura lógica, o nível de interatividade e de dificuldade de aprendizagem de cada conteúdo instrucional.

A partir do(s) documento(s) XML, são geradas diferentes apresentações de uma disciplina, baseado no curso do aluno.

4.1 Módulos em desenvolvimento

Estão em desenvolvimento quatro módulos para dar funcionalidade total ao ambiente, sendo eles: (1) Autoria [FRE2002]; (2) Armazenamento em XML, descrita neste trabalho; (3) Adaptação do conteúdo baseado no modelo do aluno [MAR2002]; (4) Interface adaptativa [GAS2002].

4.2 Descrição dos módulos

O ambiente AdaptWeb é composto por quatro módulos que estão descritos nas próximas seções. A figura 4.1 representa a arquitetura do ambiente AdaptWeb. Nas próximas seções deste capítulo os módulos deste ambiente estão descritos levando em consideração suas entrada, saída e seu funcionamento [AMA2002].

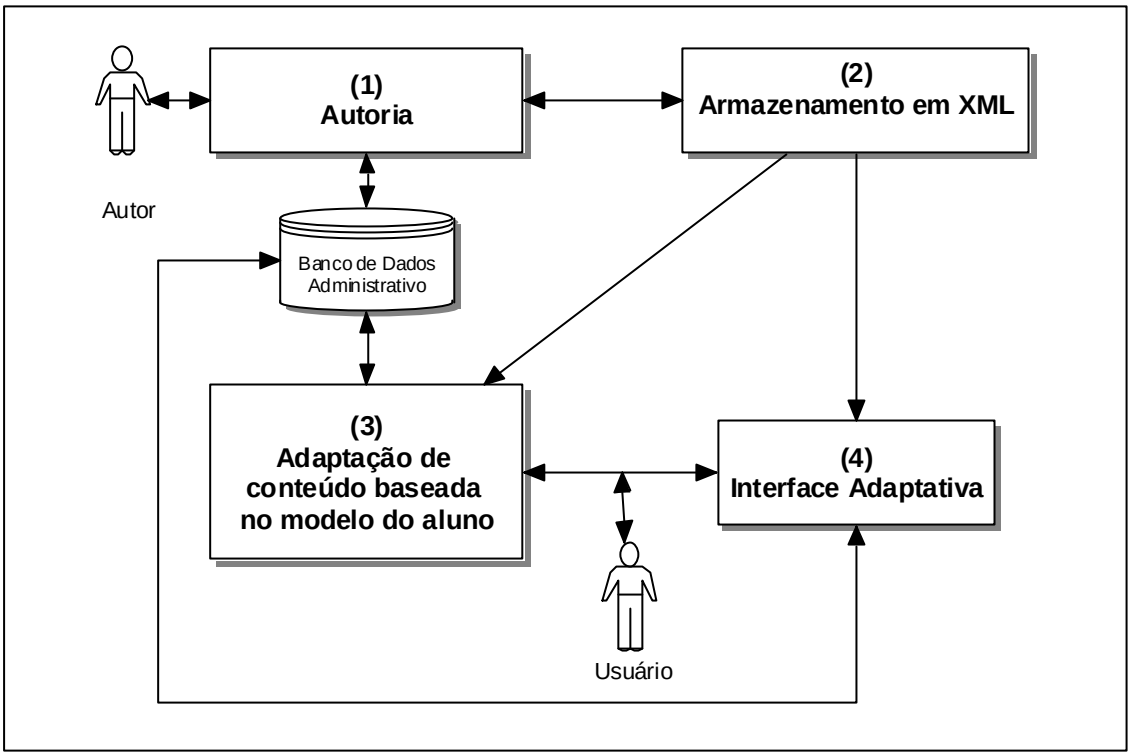

FIGURA 4.1 – Arquitetura do Ambiente AdaptWeb

4.2.1 Autoria / Estruturação do conteúdo

O módulo de autoria consiste na estruturação e organização do conteúdo instrucional a ser disponibilizado para o usuário. Para estruturação do conteúdo o autor tem como base uma sistemática para pré-autoria e uma ferramenta de autoria. A sistemática de pré-autoria auxilia na organização da estrutura geral da disciplina e na identificação dos arquivos relacionados a esta estrutura. A ferramenta de autoria possibilita a criação da estrutura de conceitos organizada através da sistemática de pré-autoria e associa os arquivos de conceito, exemplos, exercícios e material complementar a cada conceito em uma única estrutura adaptada para os diferentes cursos.

4.2.1.1 Sistemática para pré-autoria

Nesta fase o autor organiza e estrutura o conteúdo programático para ser disponibilizado da melhor maneira para o aluno. Para organização do conteúdo durante a fase de pré-autoria, o autor necessita identificar os dados de entrada, que são utilizados na ferramenta de autoria, onde deve possuir para cada conceito, um arquivo HTML referente ao conceito e outros arquivos associados a este que são classificados como exemplos, exercícios e material complementar.

A sistemática para pré-autoria consiste em várias etapas, aonde o autor irá:

- Definir os objetivos da disciplina;
- Organizar os conceitos de forma hierárquica através de um editor de texto, onde estrutura o conteúdo a ser disponibilizado para o aprendiz, independente do público alvo;

- Organizar o conteúdo instrucional e identificar os arquivos existentes para cada conceito, classificando-os como arquivos de conceito, exemplos, exercícios e material complementar.
- Editar os arquivos referentes a cada conceito que foram identificados na etapa anterior. Durante a edição dos arquivos é definido o nome definitivo dos arquivos através de critérios fornecidos pela sistemática de autoria, que visa facilitar a identificação dos mesmos durante o uso do estruturador de conteúdo.
- Na última etapa, o autor faz uso da ferramenta de autoria para criar a estrutura de conteúdo organizado na fase de pré-autoria e relacionar os arquivos pertencentes a cada conceito adaptados a cada curso de acordo com o perfil do usuário.

4.2.1.2 Módulo de Autoria Adaptativa

Para ter acesso ao módulo de autoria o autor deve solicitar permissão ao administrador do ambiente AdaptWeb. Após a liberação, o mesmo pode ter acesso ao módulo de autoria e então criar a estrutura de conceitos organizada através da sistemática de pré-autoria, adaptadas a diferentes perfis de alunos.

Através da ferramenta de autoria o autor pode:

- Cadastrar os cursos;
- Cadastrar as disciplinas;
- Especificar para quais cursos deseja disponibilizar a disciplina;
- Criar a estrutura de conteúdo organizado através da sistemática de pré-autoria e associar a esta estrutura os conteúdos referentes a cada conceito. Para associar o conteúdo a cada conceito o autor deve fazer uso das tabelas de conceito, exemplos, exercícios e materiais complementares, organizadas através da sistemática de autoria, descritas em [FRE2002].

Durante a manutenção do conceito o autor deve informar um arquivo de conceito (obrigatoriamente em formato HTML), descrição do conceito, abreviação, palavras-chave, lista de pré-requisitos e também para quais cursos deseja disponibilizar este conceito. Para exemplos, exercícios e material complementar o autor pode informar uma lista de arquivos, podendo especificar também o público alvo. Para exemplos e exercícios, deve informar a descrição, nível de complexidade (Sem Classificação, Fácil, Médio, Complexo), para quais cursos deseja disponibilizar e o arquivo referente a este. Para material complementar deve informar a descrição, para quais cursos deseja disponibilizar o material complementar e o arquivo referente ao material complementar.

Ao término do processo de autoria, os dados referentes a estrutura do conteúdo da disciplina estão armazenados em uma estrutura de dados matricial em memória, onde cada linha da matriz contém informações referentes a um conceito.

4.2.1.3 Estrutura de dados resultante da fase de Autoria

Ao finalizar a etapa de autoria todos os dados definidos pelo autor são armazenados em uma estrutura de dados em memória do tipo matriz. Abaixo segue a descrição da estrutura de dados resultante da fase de autoria, contendo as informações inseridas pelo autor:

? Dados descritores do conceito:

- Identificador do conceito: cada conceito é identificado por um número que corresponde ao seu nível na estrutura. Exemplo: 1, 1.1, 1.2, 1.3, 2. 2.1;
- Nome do conceito: armazena o nome do conceito de forma completa;
- Abreviação do conceito: armazena o nome do conceito de forma resumida;
- Palavra-chave: armazena uma relação de palavras-chave separadas por virgula;
- Pré-requisito: armazena uma lista com os pré-requisitos referente ao conceito;

? Estrutura do conceito:

- *Arquivo principal:* Matriz que armazena dados sobre o arquivo principal de conceito, tais como: nome do arquivo principal de conceito, nome dos arquivos que estão associados ao principal, lista de pré-requisitos e identificador que corresponde a classificação do ambiente tecnológico do arquivo associado.
- *Curso:* Matriz que armazena os identificadores dos cursos e seus respectivos *status*. Este *status* sinaliza se o curso está ou não relacionado ao conceito. Caso o curso seja selecionado seu valor é TRUE; caso contrário FALSE;
- *Exemplos:* Matriz que armazena exemplos relacionados ao conceito, contendo: identificação do exemplo, nível de complexidade (Sem Classificação, Fácil, Médio, Complexo), cursos que estão relacionados ao exemplo com identificador e *status* destes, arquivos associados ao exemplo e seus respectivos identificadores de classificação do ambiente tecnológico.
- *Exercícios:* Matriz que armazena exercícios relacionados ao conceito, contendo: identificador do exercício, descrição, nível de complexidade (Sem Classificação, Fácil, Médio, Complexo), cursos que estão relacionados ao exercício com identificador e status destes, arquivos associados ao exercício e seus respectivos identificadores de classificação do ambiente tecnológico.
- *Material Complementar:* Matriz que armazena material complementar relacionado ao conceito, contendo: identificador, cursos que estão relacionados ao material complementar com identificador e status destes, arquivos associados ao material complementar e seus respectivos identificadores de classificação do ambiente tecnológico.

Esta matriz contendo dados descritores do conceito e a estrutura do conceito é a entrada de dados para a etapa de armazenamento em XML, servindo como base para a

geração dos arquivos utilizados nas demais fases do ambiente AdaptWeb. Detalhes sobre esta estrutura de dados podem ser adquiridos em [FRE2002].

4.2.2 Armazenamento em XML

Nesta seção será descrito o objeto deste trabalho, o módulo de armazenamento em XML. Este módulo é responsável pela organização dos arquivos fornecidos pelo autor, através do módulo de autoria, para XML. Também é nesta fase que os dados da autoria são organizados para posterior utilização no ambiente.

A entrada de dados da fase de armazenamento é a estrutura de dados em memória proveniente da fase de autoria, onde todos os dados definidos pelo autor estão armazenados. Esta representação é a entrada de dados para a etapa de armazenamento em XML, servindo como base para a geração dos arquivos utilizados nas demais etapas do ambiente, a fim de ser apresentado ao usuário adaptado ao seu perfil.

A saída de dados da presente fase é composta pelos arquivos XML. Existe um arquivo XML para a estrutura dos conceitos e um arquivo XML para organizar o conteúdo de cada conceito cadastrado pelo autor.

Para criação destes arquivos XML foram definidas duas DTD (*Document Type Definition*) [W3C2000]. Uma delas retrata a estrutura de conceitos definida pelo autor, já que na fase de autoria todos os conceitos de uma disciplina devem conter suas devidas informações, tais como: descrição, número, lista de pré-requisitos, palavras-chave relacionadas, entre outros elementos. A outra DTD tem por finalidade descrever os conteúdos armazenados para cada conceito. Esses conteúdos educacionais estão divididos em: conceito, exemplos, exercícios e material complementar.

4.2.2.1 Algoritmo gerador dos arquivos XML

Através da criação das DTD é possível ter a definição formal dos tipos de documentos que foram manipulados durante o processo de disponibilização do conteúdo instrucional. Com base nestas DTD foi definido e implementado um algoritmo para conversão da representação em memória gerada na fase de autoria em arquivos XML [W3C2000].

Este algoritmo cria um arquivo XML para cada disciplina com sua respectiva estrutura de conceitos, a qual define as características de cada conceito da disciplina. O conteúdo instrucional é armazenado em outro arquivo XML, dividido em conceitos, exemplos, exercícios e material complementar, como exposto na figura 4.2.

O algoritmo também contempla o tratamento dos dados recebidos da fase de autoria. Os arquivos XML só podem ser gerados se os dados da autoria, armazenados em memória, estiverem todos inseridos. Caso exista um elemento da autoria que não esteja cadastrado, como uma descrição de conceito, por exemplo, o algoritmo sinaliza o erro e então não conclui o processo de geração dos arquivos XML. Assim que o autor atualizar os elementos que estão faltando o algoritmo faz uma nova validação na estrutura gerada pela autoria e só com os dados corretos gera os arquivos XML. Estes elementos que devem estar inseridos na autoria são definidos nas DTD como elementos obrigatórios, ou seja *required.*

Na figura 4.2 é visualizado o esquema de geração de arquivos XML que tem como entrada as DTD de estrutura de conceitos e de conteúdo do conceito. São gerados então um arquivo XML contendo a estrutura geral da disciplina, que é totalmente

orientada a conceito, e outros documentos XML referentes aos demais conteúdos relacionados ao conceito, como: exemplos, exercícios e material complementar.

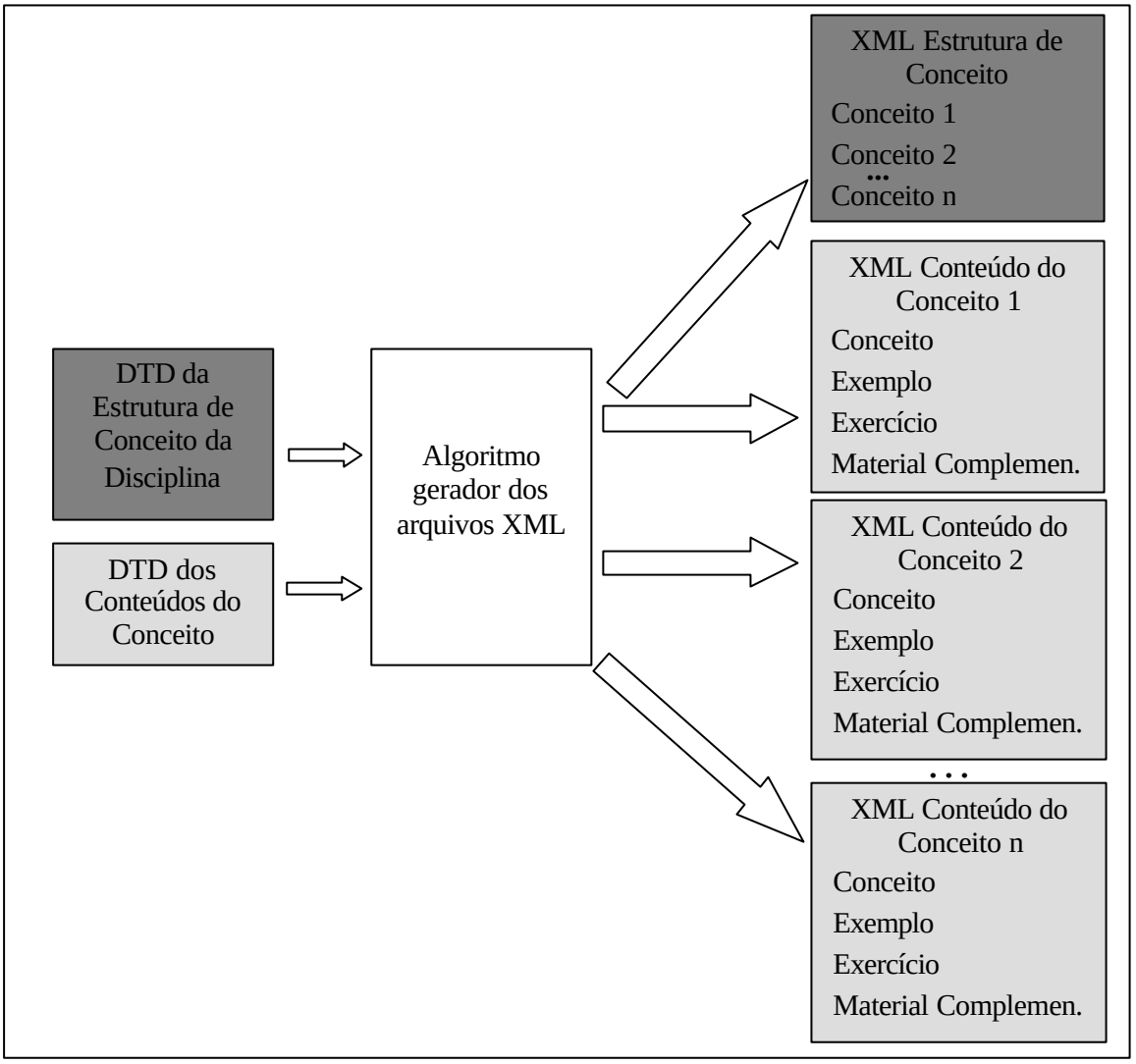

FIGURA 4.2 – Esquema de geração de arquivos XML

Esta organização permite uma estruturação dos dados de forma hierárquica, pois sempre existe um único arquivo XML com a estrutura de conceitos da disciplina e tantos arquivos XML com conteúdo quantos forem os conceitos definidos. Para exemplificar: se o autor inserir dez conceitos na ferramenta de autoria então são criados onze documentos XML através do algoritmo, um para armazenar toda a estrutura de conceitos, como um índice, e um arquivo XML para cada um dos dez conceitos.

Todo o processo de geração de arquivos XML sempre é validado através de um *parser* (analisador) que percorre os documentos. Este *parser* utiliza a API (*Application Programming Interface)* DOM (*Document Object Model)* [DOM98] para o processo de validação. Os documentos XML só são gravados no servidor se forem validados pelo *parser*.

A API DOM foi utilizada como padrão de interface entre os documentos XML e o algoritmo aqui descrito, pois foi confirmada como uma recomendação da W3C em 1998 e é amplamente utilizada como uma interface neutra de plataforma e linguagem, que provê formas de acesso para os programas e *scripts* acessarem e atualizarem

dinamicamente os conteúdos, as estruturas e os estilos dos documentos XML [ROY2001].

O modelo DOM representa um documento XML como uma árvore, cujos nós são elementos, texto, etc. O processador XML gera a árvore na memória, e a entrega ao *parser* para compará-la as regras de criação de documento descritas nas DTD [SEL2001].

No Capítulo 5 é feita uma descrição mais detalhada do desenvolvimento do módulo de armazenamento de dados em XML, que é a contribuição desta dissertação.

4.2.3 Adaptação do conteúdo baseado no modelo do aluno

Os módulos abordados anteriormente dizem respeito ao processo de autoria do conteúdo instrucional, ou seja, como os conteúdos existentes são organizados, estruturados, incluídos e finalmente armazenados. O módulo de adaptação de conteúdo e o de interface adaptativa são responsáveis em recuperar e adaptar devidamente estes conteúdos a fim de apresentá-los aos alunos [MAR2002].

Estes conteúdos instrucionais disponibilizados pelo AdaptWeb precisam ser adaptados dinamicamente de acordo com o modelo do aluno. Este modelo representa o relacionamento entre o aluno e a estrutura de conceitos resultantes do processo de autoria, levando em consideração o curso, a preferência pelo modo de navegação e o ambiente de trabalho do aluno, que é adquirido através de um questionário respondido pelo próprio aluno.

A primeira vez que o aluno acessa o ambiente AdaptWeb, deve se cadastrar e solicitar matrícula nas disciplinas de seu interesse associadas ao seu curso. Desta maneira, assim que o professor liberar sua matrícula, este está apto a interagir no ambiente. Durante o início da interação o aluno passa pelo processo de *login*. A seguir seleciona a disciplina/curso que desejar, o modo de navegação de sua preferência, que pode ser tutorial ou livre e o tipo de conexão de rede que se encontra.

Portanto, antes de dar início à interação entre o aluno e o conteúdo instrucional, são coletadas informações a respeito da identificação deste aluno, da disciplina e do curso que deseja interagir, do modo de navegação escolhido e do tipo de conexão de rede usada naquele instante de tempo. Devido a necessidade de coletar estas informações do aluno a fim de construir seu modelo, o módulo de adaptação de conteúdo fornece recursos para que o aluno possa:

- Cadastrar-se;
- Solicitar matrículas nas disciplinas de seu interesse associadas ao seu curso;
- Alterar ou excluir seu cadastro;
- Selecionar o modo de navegação livre ou tutorial;
- Definir o tipo de conexão de rede;
- Assistir a um curso no ambiente de ensino AdaptWeb.

Além disso, o módulo de adaptação do conteúdo faz o processo de filtragem dos documentos XML a fim de possibilitar a adaptabilidade na apresentação do conteúdo instrucional, das mídias apropriadas e da interface. Este processo de adaptação ocorre na etapa descrita a seguir.

4.2.3.1 Filtros no documento XML conteúdo do conceito

Cada conceito da estrutura hierárquica global possui um arquivo XML de conteúdo associado. Nestes arquivos estão armazenados o conceito teórico, os exemplos, exercícios e materiais complementares associados a este conceito e suas respectivas características, como quais cursos podem resolver um determinado exercício ou se o exercício é fácil, médio ou complexo.

Estes arquivos passam pelo segundo filtro durante a navegação do aluno pelos conceitos, sendo que estes possuem quatro formas de apresentação: conceito (teoria), exemplo, exercício e material complementar. De acordo com a forma e o conceito escolhido, o filtro é executado sobre o respectivo arquivo XML de conteúdo e é gerada dinamicamente a apresentação para o aluno.

Neste filtro o curso e as características da conexão de rede do ambiente de trabalho do aluno também são consideradas como critérios para a geração adaptada do conteúdo das páginas HTML.

4.2.4 Interface Adaptativa e Monitoramento do aluno

A Interface geral foi estruturada visando auxiliar o aluno em sua navegação, proporcionando simplicidade operacional e visão geral das opções de navegação. A Interface foi planejada para exibir todas as formas de apresentação de um mesmo conteúdo, previstas na fase de autoria: os conceitos, os exemplos, os exercícios e o material complementar [GAS2002]. A figura 4.3 mostra a Interface no modo tutorial.

FIGURA 4.3 – Interface no modo Tutorial

O módulo de Interface Adaptativa propõe a adaptabilidade nos dois níveis de adaptação: na navegação e na apresentação do conteúdo propriamente dito. A adaptabilidade de navegação é um recurso muito importante especialmente para

ambientes educacionais, em que o aluno pode ser ajudado pelo ambiente, que toma algumas decisões para uma melhora na aprendizagem do aluno.

Existem várias técnicas e métodos para a implementação da adaptabilidade, e a Interface Geral utiliza-se de técnicas como a orientação global (Mapa) que é uma técnica utilizada para mostrar todo o *hiperespaço*, a condução global (usada para informar o menor caminho), *history list* (menu de conceito) que faz uma lista geral e também quando o usuário entra novamente no ambiente volta onde este havia finalizado, o suporte a navegação que é uma orientação e condução local, pois consiste em auxiliar o usuário a entender seu posicionamento no *hiperespaço* e este pode escolher modificar esta posição. Todas as técnicas são baseadas nas características dos alunos, representadas em seu modelo [BRU99].

O AdaptWeb utiliza quatro aspectos para prover adaptabilidade: a área de conhecimento/formação do aluno, seu conhecimento, seu ambiente de trabalho e modo de navegação. Em especial, o módulo de Interface Adaptativa, utiliza as adaptações já feitas pelo autor e pelos filtros e também adapta de acordo com o conhecimento adquirido do estudante e com suas preferências navegacionais.

O monitoramento do aluno também é mantido pelo módulo de Interface adaptativa, pois a cada vez que o aluno adquirir um novo conhecimento, o módulo ativa uma função que atualiza uma tabela de *log* cadastrando o novo dado.

5 Módulo de Armazenamento em XML para o AdaptWeb

Conforme verificado no Capítulo 4, o Ambiente AdaptWeb é formado por quatro módulos principais que são: Autoria adaptativa de hipermídia educacional, Armazenamento em XML, Adaptação do conteúdo baseado no modelo do aluno e Interface adaptativa.

Cabe ao módulo de armazenamento em XML a organização dos dados fornecidos pelo autor, via módulo de autoria. É a partir desta fase que os conteúdos inseridos pelo autor são utilizados no ambiente, para serem adaptados e apresentados ao usuário.

Este capítulo tem como objetivo descrever o módulo de armazenamento em XML. São abordadas a colocação do problema, a entrada de dados e saída de dados deste módulo, bem como os algoritmos implementados.

5.1 Colocação do Problema

No desenvolvimento do AdaptWeb foram consideradas duas formas de adaptabilidade: na apresentação e na navegação. Segundo [BON2000] a adaptabilidade é uma característica dos hipertextos e hipermídias que permite a adaptação de conteúdo de acordo com as necessidades do usuário. Além disto, Sistemas Hipermídia Adaptativos modificam a apresentação do domínio do conhecimento de acordo com o modelo do aluno, este mecanismo permite personalizar o Hipermídia em termos de navegação [BON2000].

No AdaptWeb, antes da introdução destas características de adaptabilidade, o conteúdo do autor deve ser inserido e organizado. A inserção dos dados do autor é feita através do módulo de autoria, onde os conteúdos instrucionais são modelados através de uma estrutura matricial representando uma seqüência de conceitos que posteriormente são utilizados para organizar o conteúdo de maneira hierárquica nos documentos XML.

Antes de serem apresentados ao aluno, estes documentos XML passam por um processo dinâmico de filtragem, que é realizado durante a interação do aluno com o ambiente. Esta filtragem obedece aos critérios de adaptação representados no modelo deste aluno. O conhecimento, a formação e o ambiente de trabalho do aluno são as principais características representadas neste modelo [MAR2002].

Diante disto, o uso da linguagem XML é de fundamental importância, por ser um recurso rico para manter uma estrutura hierárquica de organização, pois as próprias *tags* de marcação de conteúdo, podem estabelecer uma estruturação lógica dos dados. XML ainda permite que o conteúdo seja mantido separadamente do *layout* de apresentação, facilitando assim a geração de novas apresentações para o mesmo documento XML, portanto as diferentes apresentações de uma disciplina, baseadas no curso do aluno, são geradas a partir dos documentos XML criados na fase de Armazenamento em XML.

5.2 XML e Adaptabilidade no Ambiente AdaptWeb

Como nos ambientes discutidos nas seções 2.4.2 e 2.4.3 [BRA2002] e [BON2000], o AdaptWeb também utiliza XML para armazenamento e organização do

conteúdo instrucional. Porém, de maneira diferente destes os documentos XML do ambiente AdaptWeb não possuem conteúdos em HTML e sim referências aos arquivos de extensão HTML. Isto torna a etapa de autoria mais simples para o autor, pois este não necessita saber XML para utilizar a ferramenta de autoria, basta anexar seus arquivos HTML aos conceitos definidos para a disciplina.

A existência de dois tipos de documento XML cria uma divisão que contribui para a manutenção do ambiente. Um tipo de documento XML armazena a estrutura de conceitos da disciplina, exibindo seus relacionamentos, pré-requisitos existentes e arquivos HTML associados a estes conceitos. O outro tipo de documento XML armazena informações sobre o conteúdo instrucional de cada conceito. Este documento contém as informações sobre o conteúdo, inclusive informações sobre o arquivo HTML de conteúdo.

O documento XML de estrutura dos conceitos organiza as referências a estes arquivos HTML, permite que o armazenamento dos relacionamentos existentes entre os conceitos seja facilmente implementado em uma estrutura hierárquica e separa totalmente o conteúdo da apresentação.

A adaptabilidade de navegação é feita através do arquivo XML que armazena a estrutura de conceitos da disciplina. Através da análise do documento XML é possível estabelecer qual conceito pode ou não ser visitado por um aluno. O uso de XML facilita este processo de filtragem dos conceitos, pois podem ser analisados os conceitos e seus pré-requisitos e quais conteúdos estão disponíveis para cada perfil de usuário.

Uma outra característica do AdaptWeb é a possibilidade de se filtrar as mídias existentes nos conteúdos instrucionais. Assim, dependendo do ambiente tecnológico em que o aluno se encontra, podem ser exibidos conteúdos em determinadas mídias. Esse filtro trabalha com o HTML inserido pelo autor e os eventuais arquivos de mídia associados a este.

O uso de XML para armazenamento de conteúdo instrucional em ambientes hipermídia adaptativos é uma realidade. No AdaptWeb foram utilizados os recursos desta linguagem para se desenvolver documentos que possam ter facilmente aplicadas as técnicas de adaptabilidade mais variadas.

5.3 Entrada de Dados do Armazenamento em XML

O módulo de armazenamento tem como entrada a estrutura de dados em memória proveniente da fase de autoria, onde todos os dados definidos pelo autor estão armazenados. Segue abaixo uma descrição desta estrutura de dados, maiores detalhes podem ser obtidos em [FRE2002].

A estrutura em forma de matriz possui duas divisões de grande importância: os dados descritores dos conceitos e os dados da estrutura de conteúdo do conceito. No ambiente AdaptWeb todo curso é formado por conceitos e cada um deles deve conter os seguintes elementos: arquivo relacionado ao conceito (obrigatoriamente um), exercício (podendo ser zero ou mais), exemplo (podendo ser zero ou mais), material complementar (podendo ser zero ou mais).

A figura 5.1 ilustra como a estrutura em memória gerada no módulo de autoria é organizada. Uma disciplina pode ser formada por vários conceitos (de 1 a n) e cada um pode conter exercício, exemplo e material complementar. Apenas para o conceito deve haver um arquivo principal obrigatório e único.

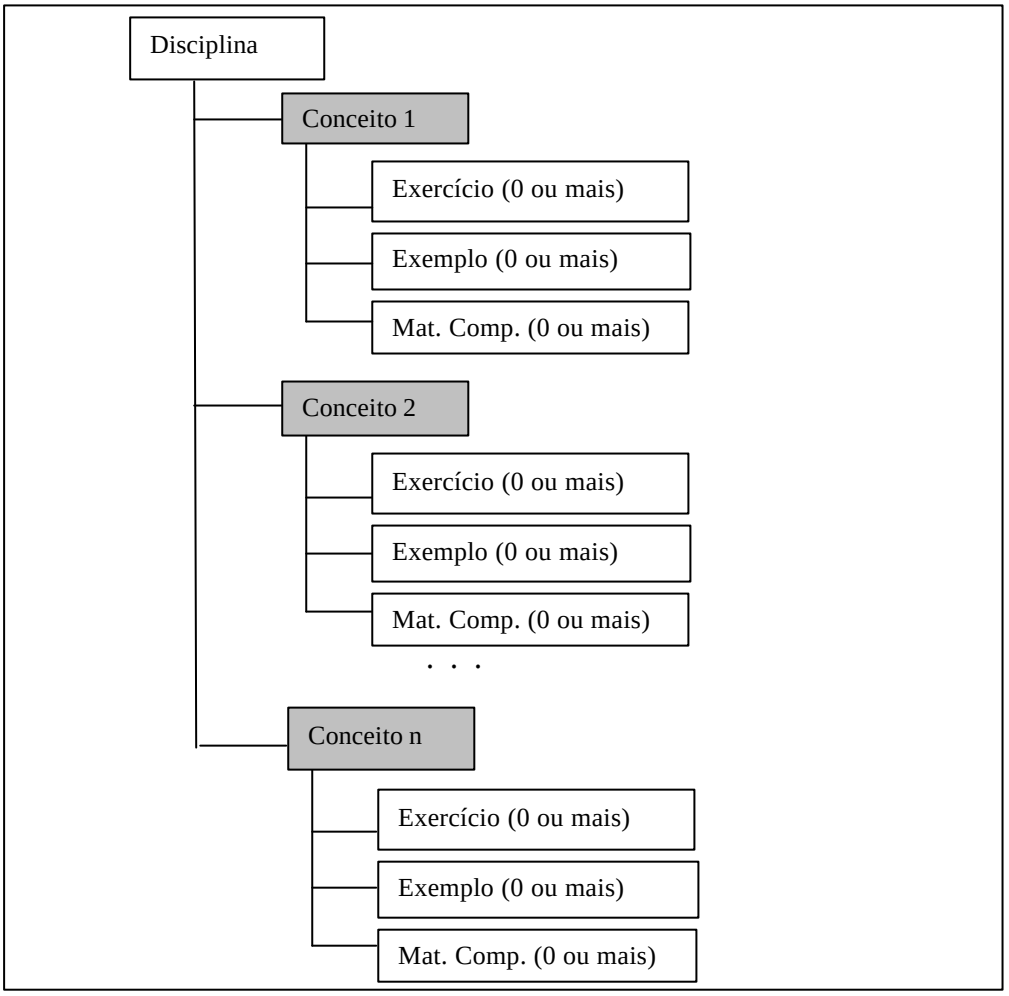

FIGURA 5.1 – Organização da estrutura de dados proveniente da autoria

Os conceitos possuem atributos que os caracterizam, nenhum destes é opcional, todos são obrigatórios. A tabela 5.1 mostra quais são estes atributos e que tipo de informações eles armazenam.

TABELA 5.1 – Elementos descritores dos conceitos

Elementos Descritores do Conceito	
Atributo	**Descrição**
Identificador Numconc	Cada conceito é identificado por um número que representa o seu nível na estrutura de conceito.
Nome	Descrição do nome do conceito de forma completa
Abreviação	Abreviação do nome.
Palavra-chave	Lista de palavras-chave relacionadas ao conceito.
Pré-requisito	Lista de pré-requisitos do conceito.

Além de armazenar dados descritores dos conceitos esta estrutura matricial contém dados sobre o conteúdo do conceito, exemplo, exercício e material complementar, quando estes três últimos constam.

Para cada conceito sempre deve ser anexado um arquivo HTML com o conteúdo instrucional. Ao incluir um exemplo, um exercício ou material complementar também deve ser associado a este um arquivo HTML.

Existem outras informações que são inseridas juntamente com o arquivo HTML, tais como: cursos que irão visualizar o conceito, informações sobre a complexidade de exercícios e exemplos e sobre o arquivo de conteúdo do conceito. Estes e outros elementos podem ser visualizados de maneira completa nas tabelas que seguem.

TABELA 5.2 – Elementos do conteúdo do conceito.

Descritores do Conteúdo do Conceito	
Atributo	**Descrição**
Nome do arquivo de conceito	Arquivo HTML associado ao conceito. Este arquivo sempre é requerido, pois o conceito é descrito através deste.
Nome dos arquivos associados	Arquivos contidos dentro do HTML de conceito. Estes devem ser declarados quando utilizados no HTML.
Identificador de ambiente tecnológico para arquivos associados	Número que corresponde a classificação dos arquivos associados ao HTML. Este número designará qual ambiente tecnológico poderá acessar os arquivos associados ao HTML. Utilizado durante a navegação do aluno e permite adaptar o conteúdo de acordo com o ambiente tecnológico.
Identificador do Curso	Armazena a lista de identificadores dos cursos que estão relacionados com o conceito em questão. Um conceito pode ter mais de um curso relacionado.
Status do Curso	Descreve para cada curso associado se o conceito está disponível ou não.

A tabela 5.2 lista os elementos do conteúdo do conceito. Entre eles pode se destacar o *nome do arquivo de conceito*, que é o arquivo principal de conteúdo do conteúdo instrucional, sem este não há como exibir o texto referente ao conceito. Este arquivo HTML pode conter outros arquivos, no AdaptWeb estes são chamados de *arquivos associados*, que podem ser figuras, animações, vídeos e outras mídias inseridas no arquivo HTML para melhor explanação do conteúdo. Cada um destes arquivos associado possui uma classificação, *identificador de ambiente tecnológico para arquivos associados*, que é utilizada na adaptação do ambiente tecnológico do aluno. Cada arquivo associado deve possuir um identificador que representa seu ambiente tecnológico. O aluno ao iniciar o processo de navegação no ambiente AdaptWeb, por meio de um questionário responde qual o seu ambiente tecnológico, que será armazenado no modelo do aluno. O ambiente tecnológico do aluno deve ser igual ao dos arquivos associados para que estes possam ser disponibilizados durante a navegação.

A disciplina pode ser formada apenas pelo conceito. Porém quando há a existência de exemplo, exercício ou material complementar, a entrada de dados do módulo de armazenamento em XML deve receber as informações referentes a eles. As tabelas 5.3, 5.4 e 5.5 mostram quais são os elementos descritores e de conteúdo de exemplos, exercícios e materiais complementares respectivamente.

TABELA 5.3 – Elementos descritores do exemplo e seu conteúdo.

Descritores do Exemplo e seu Conteúdo	
Atributo	**Descrição**
Identificador do exemplo	Identificador numérico que faz parte da descrição de um exemplo
Descrição do exemplo	Texto descrevendo o exemplo. É como um título que deve ser relacionado ao exemplo.
Arquivo do exemplo	Arquivo HTML associado ao exemplo.
Nível de complexidade do exemplo	Nível que caracteriza a complexidade do exemplo. Esta complexidade pode ser dada por um destes valores: Fácil, Médio, Complexo e Sem Classificação.
Identificador do curso do exemplo	Armazena a lista de identificadores dos cursos que estão relacionados com o exemplo em questão. Um exemplo pode ter mais de um curso relacionado.
Status do curso do exemplo	Descreve para cada curso associado se o exemplo está disponível ou não.
Lista dos Nomes do arquivo associado ao exemplo	Arquivos associados são figuras, animações, vídeos e outras mídias que estão inseridas no arquivo HTML relacionado ao exemplo. Estes devem ser denotados quando utilizados já no HTML. Pode haver mais de um arquivo associado.
Identificador de ambiente tecnológico para arquivos associados	Número que corresponde a classificação dos arquivos associados. Esta classificação é utilizada na adaptação do ambiente tecnológico do aluno. Cada arquivo associado de exemplo deve possuir um identificador de seu ambiente tecnológico.

TABELA 5.4 – Elementos descritores do exercício e seu conteúdo.

Descritores do Exercício e seu Conteúdo	
Atributo	**Descrição**
Identificador do exercício	Identificador numérico que faz parte da descrição de um exercício
Descrição do exercício	Texto descrevendo o exercício. É como um título que deve ser relacionado ao exercício.
Arquivo do exercício	Arquivo HTML associado ao exercício.
Nível de complexidade do exercício	Nível que caracteriza a complexidade do exercício. Esta complexidade pode ser dada por um destes valores: Fácil, Médio, Complexo e Sem Classificação.
Identificador do curso do exercício	Armazena a lista de identificadores dos cursos que estão relacionados com o exercício em questão. Um exercício pode ter mais de um curso relacionado.
Status do curso do exercício	Descreve para cada curso associado se o exercício está disponível ou não.
Nome do arquivo associado ao exercício	Arquivos associados são figuras, animações, vídeos e outras mídias que estão inseridas no arquivo HTML relacionado ao exercício. Estes devem ser denotados quando utilizados já no HTML. Pode haver mais de um arquivo associado.
Identificador de ambiente tecnológico para arquivos associados	Número que corresponde a classificação dos arquivos associados. Esta classificação é utilizada na adaptação do ambiente tecnológico do aluno. Cada arquivo associado de exercício deve possuir um identificador de ambiente tecnológico.

TABELA 5.5 – Elementos descritores do material complementar e seu conteúdo.

Descritores do Material Complementar e seu Conteúdo	
Atributo	**Descrição**
Identificador do material complementar	Identificador numérico que faz parte da descrição de um material complementar.
Descrição do material complementar	Texto descrevendo o material complementar. É como um título que deve ser relacionado ao material complementar.
Arquivo do material complementar	Arquivo HTML associado ao material complementar.
Identificador do curso do material complementar	Armazena o identificador correspondente ao curso que está relacionado com o material complementar em questão. Um exercício pode ter mais de um curso relacionado.
Status do curso do material complementar	Descreve para cada curso associado se o material complementar está disponível ou não.
Nome do arquivo associado ao material complementar	Arquivos associados são figuras, animações, vídeos e outras mídias que estão inseridas no arquivo HTML relacionado ao material complementar. Estes devem ser denotados quando utilizados já no HTML. Pode haver mais de um arquivo associado.
Identificador de ambiente tecnológico para arquivos associados	Número que corresponde a classificação dos arquivos associados. Esta classificação é utilizada na adaptação do ambiente tecnológico do aluno. Cada arquivo associado de material complementar deve possuir um identificador de ambiente tecnológico.

Nas tabelas 5.3, 5.4 e 5.5 é possível notar que exemplo, exercício e material complementar possuem uma estrutura semelhante. Todos contêm um identificador único, uma descrição, o arquivo HTML de conteúdo e as informações relativas aos cursos relacionados a estes.

Esta estrutura descrita está organizada de maneira seqüencial, ou seja, durante a inserção dos conceitos e suas informações não há na estrutura em memória uma organização hierárquica, os dados são armazenados na medida e na seqüência em que a autoria os fornece. Uma das funções do algoritmo gerador dos arquivos XML é transformar esta estrutura seqüencial em estrutura hierárquica para prover maior facilidade as demais etapas do ambiente.

O algoritmo gerador dos arquivos XML e os documentos XML de saída estão detalhados na seção 5.4 e 5.5 respectivamente.

5.4 Algoritmo gerador dos arquivos XML

Baseado nas DTD, que são descritas na seção 5.5 deste capítulo, foi implementado um algoritmo para geração dos arquivos XML que tem como entrada a estrutura de dados gerada na fase de autoria.

Para garantir que os documentos XML gerados por este algoritmo sejam confiáveis e retratem fielmente o conteúdo da estrutura de dados da fase de autoria

foram utilizados, além das definições formais dos tipos de documentos descritas nas DTD, um analisador semântico dos documentos XML e um verificador de erros da estrutura de dados em memória. A figura 5.2 ilustra o algoritmo gerador dos arquivos XML com todas as chamadas de suas funções.

```
Atribui matriz em memória para Variável conteudo_estrut;
Atribui tamanho da conteudo_estrut para Variável tamanho;
Chama função de verificação de erro da estrutura de dados;
Atribui retorno da função de verificação de erro para Variável controle_erro;
Se controle_erro = 0 então
Início
    Atribui zero para variável Contador;
    Enquanto contador < tamanho faça
    Início
        Chama função de geração dos arquivos XML de estrutura de conceito;
        Incrementa contador;
    Fim;
    Atribui zero para variável Contador;
    Enquanto contador < tamanho faça
    Início
        Chama função de geração dos arquivos XML elementos do conteúdo do conceito;
        Incrementa contador;
    Fim;
    Chama analisador dos arquivos XML de elementos do conteúdo do conceito;
    Chama analisador do arquivo XML de estrutura de conceito;
Fim
Senão
Início
    Não gera nenhum arquivo XML;
    Emite mensagem de erro;
Fim;
```

FIGURA 5.2 – Algoritmo gerador dos arquivos XML

Conforme citado anteriormente, a entrada de dados do módulo de armazenamento é dada na forma de uma estrutura matricial em memória. Como o algoritmo faz um percurso em toda a estrutura de dados é necessário o uso de uma variável que armazene o tamanho desta matriz, ou seja, a quantidade de linha, que corresponde à quantidade de conteúdos inseridos pelo autor.

A primeira função a ser executada é a de verificação de erro da matriz de entrada de dados. Esta função verifica se todos os dados da fase de autoria foram

preenchidos pelo autor, pois a maioria destes dados estão definidos na DTD como requeridos. O retorno da função é armazenado em uma variável de controle de erro, e ela se comporta da seguinte maneira: valor 0 significa que não foram encontrados erros na estrutura o valor 1 significa que foram encontrados erros.

Caso a função de verificação de erros da matriz de entrada de dados retorne valor igual a 1 não são gerados os arquivos XML e os erros detectados são listados para o autor.

As chamadas das duas funções que criam os arquivos XML propriamente ditos são feitas apenas se a variável de erro for igual a 0. Cada uma das funções devem ser executadas dentro de um laço de repetição que tem como objetivo percorrer a matriz de entrada. Nestes laços são criados os arquivos XML. O número de arquivos XML criados é sempre o tamanho da matriz acrescido de um. É criado um documento XML para a organizar a estrutura de conceito, como se fosse um índice destes, e um documento XML para o conteúdo de cada conceito existente.

Após a finalização das funções de criação dos arquivos XML são chamados os dois analisadores, o analisador do documento XML de estrutura de conceito e o analisador dos documentos XML que armazenam o conteúdo do conceito.

Todas as implementações foram desenvolvidas utilizando linguagem PHP 4.2.1 que é uma linguagem *scripting*. O código PHP é interpretado e não compilado isto quer dizer que o PHP necessita sempre de um *parser* (analisador) para interpretar o seu código [PHP2001].

A diferença existente entre PHP e linguagens semelhantes a *Javascript* é que o código PHP é executado no servidor, sendo enviado para o cliente apenas o HTML puro. Desta maneira é possível interagir com banco de dados e outras aplicações existentes no servidor [PHP2001].

As particularidades e os aspectos de implementação das funções de geração dos arquivos XML e o analisador, tanto de estrutura de conceito como de elementos do conteúdo do conceito, bem como a função de verificação de erro da estrutura de dados são descritos nas próximas seções deste capítulo.

5.4.1 Função de verificação de erro da estrutura de dados

Os documentos XML gerados pelo módulo de armazenamento não podem possuir erros, pois servem de base para os módulos de Adaptação de conteúdo baseada no modelo do aluno [MAR2002] e Interface adaptativa [GAS2002].

Na definição das DTD foram especificados atributos obrigatórios no documento XML (*required*), portanto para que o processo de geração destes resulte em arquivos XML válidos [PIM2000] não pode ser admitido que a matriz em memória fornecida como entrada de dados tenha os campos equivalentes aos atributos definidos como requeridos na DTD vazios.

A função de verificação de erro da estrutura de dados percorre toda a matriz procurando por campos vazios. Os campos da matriz que são definidos como obrigatórios para o arquivo XML são:

- Número do conceito;
- Descrição do conceito;
- Abreviação do nome do conceito;
- Palavra chave relacionada ao conceito;

- Arquivo HTML com conteúdo do conceito;
- Número de identificação do curso relacionado ao conceito;

Se o arquivo HTML com o conteúdo do conceito possuir arquivos associados então o campo abaixo deve ser considerado obrigatório:

- Classificação do ambiente tecnológico do arquivo associado;

No caso da ocorrência de exercício, exemplo ou material complementar, devem ser tratados como obrigatórios os seguintes campos listados a seguir:

- Identificador de exercício;
- Descrição de exercício;
- Arquivo HTML com conteúdo do exercício;
- Complexidade de exercício;
- Identificador de exemplo;
- Descrição de exemplo;
- Arquivo HTML com conteúdo do exemplo;
- Complexidade de exemplo;
- Identificador de material complementar;
- Descrição de material complementar;
- Arquivo com conteúdo do material complementar;
- Nome de arquivos associados aos HTML de conceito, exemplo, exercício ou material complementar;

Além destes campos que estão armazenados na matriz em memória o módulo de autoria fornece mais uma variável requerida nos documentos XML, o código da disciplina, um identificador único para representar a disciplina que está sendo criada.

O valor de cada um destes campos é analisado e então comparado com uma *string* vazia. Quando a função se depara com um campo que tem valor vazio é emitida uma mensagem de erro para o autor. O autor pode ou não voltar a etapa de autoria para preencher estes dados, porém só é dada a seqüência ao processo de geração dos documentos XML se não forem encontrados novos campos vazios.

As atividades realizadas pela função de verificação de erro da estrutura de dados foram definidas como mostra a figura 5.3.

A função de verificação de erro da estrutura de dados é o módulo principal deste processo de validação. Nela são analisadas as variáveis que descrevem a estrutura básica do conceito. Estas variáveis são: código da disciplina, número do conceito, descrição, abreviação e palavras-chave relacionadas ao conceito.

Para a análise das variáveis relacionadas ao conteúdo do conceito, exemplo, exercício e material complementar foram desenvolvidas outras funções que são executadas a partir desta.

As funções de verificação de erro dos conteúdos de exemplo, exercício e material complementar possuem uma estrutura semelhante, pois seus conteúdos são organizados de maneira similar. Todas elas verificam se o identificador, a descrição, o arquivo HTML de conteúdo e os cursos relacionados são diferentes de vazio. Para

exemplo e exercício deve ser verificado ainda se a complexidade deste está devidamente preenchida.

O processo de verificação de erro do conteúdo dos conceitos difere dos citados acima, pois sua estrutura é mais simples. Nesta verificação só são validados o arquivo HTML com conteúdo e os cursos associados a este

```
Verifica se código da disciplina igual a vazio;
Se sim emite mensagem de erro e altera variável de controle;
Inicia laço de repetição percorrendo toda da matriz de conteúdo (linha a linha)
    Verifica se número do conceito da linha atual é igual a vazio;
    Se sim emite mensagem de erro e altera variável de controle;
    Verifica se descrição do conceito da linha atual é igual a vazio;
    Se sim emite mensagem de erro e altera variável de controle;
    Verifica se abreviação do conceito da linha atual é igual a vazio;
    Se sim emite mensagem de erro e altera variável de controle;
    Verifica se palavra chave do conceito da linha atual é igual a vazio;
    Se sim emite mensagem de erro e altera variável de controle;
    Chama função de verificação de erro no conteúdo do conceito;
    Chama função de verificação de erro no conteúdo do exemplo;
    Chama função de verificação de erro no conteúdo do exercício;
    Chama função de verificação de erro no conteúdo do material complementar;
    Se encontrou erro no conteúdo do conceito ou do exemplo ou do
    exercício ou do material complementar então altera variável de controle.
Fim do laço de repetição
Fim da função
```

FIGURA 5.3. – Função de verificação de erro da estrutura de dados em memória

A figura 5.3 descreve os passos realizados para a verificação de erros utilizando uma pseudo linguagem de projeto, porém todas estas funções foram implementadas em PHP [PHP2001].

5.4.2 Função de geração do arquivo XML de estrutura de conceito

A principal ação da função de geração do arquivo XML de estrutura de conceito é organizar os dados que estão linearmente armazenados na matriz de conteúdo em uma estrutura hierárquica, ou seja, em um grafo. A numeração dos conceitos é utilizada como guia para a construção deste grafo.

Sempre que o autor insere um novo conceito no processo de autoria é requerido que este tenha um número de identificação, esta numeração do conceito é utilizada para avaliar as relações existentes entre os conceitos e posterior criação dos nós do grafo.

O processo de criação da organização hierárquica no documento XML se dá através de um percurso por toda a matriz em memória. A fase de autoria já fornece a matriz com uma ordenação seqüencial e crescente baseada neste número de

identificação. É esta ordenação da matriz, juntamente com os identificadores, que serve para guiar o processo de criação do grafo de conceitos no arquivo XML.

A figura 5.4 ilustra as ações do algoritmo de geração do arquivo XML de estrutura de conceito. O percurso se inicia na primeira linha da matriz de conteúdo, assim o número de identificação desta linha, ou deste conceito, é o índice do primeiro elemento do grafo. Posteriormente se analisa o identificador da próxima linha e então é realizada uma comparação para verificar se este possui o mesmo nível do primeiro ou um nível inferior.

O nível do conceito é considerado inferior quando o próximo número é um valor menor do que o atual. E é considerado igual quando o próximo número é um valor maior do que o analisado.

Caso possua nível inferior (o valor é menor do que o anterior) então este é inserido no documento XML como um conceito filho do anterior. Se possuir valor maior então é inserido como irmão. Este processo é realizado até que se percorra a matriz em memória inteira.

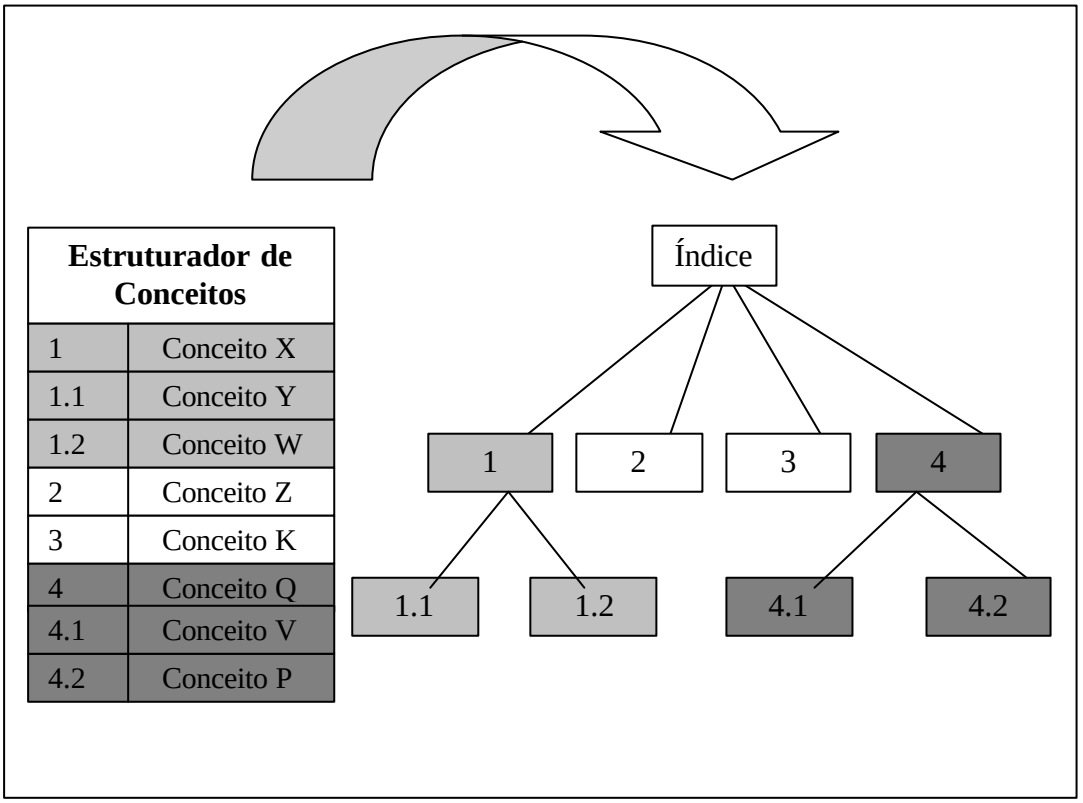

FIGURA 5.4 – Esquema de criação de grafo através da matriz em memória

Para cada conceito inserido na estrutura hierárquica do documento XML devem ser armazenados: o número do conceito, sua descrição, os pré-requisitos, a abreviação, o nome deste arquivo XML e a lista de palavras-chave, que são todos provenientes da estrutura de dados em memória. Devem ser armazenados também os cursos que podem visualizar este conceito e se este pode acessar os exercícios, exemplos e materiais complementares.

Um exemplo de instância do documento XML pode ser visto na seção 5.4 deste trabalho, que descreve as saídas de dados geradas pelos algoritmos aqui descritos.

5.4.3 Função de geração dos arquivos XML de elementos do conteúdo do conceito

A função de geração de arquivos XML de elementos do conteúdo do conceito também é desenvolvida a partir de um percurso em toda a estrutura de dados em memória.

Este arquivo deve armazenar os dados referentes ao conteúdo de conceito, exemplo, exercício e material complementar, como descrito nas DTD.

Para cada conceito cadastrado da matriz em memória deve ser criado um documento XML. O nome deste documento XML é dado pelo nome do arquivo HTML associado ao conceito, porém com extensão XML.

Após a criação do nome do arquivo este deve ser aberto para gravação. Para todos os arquivos de conceito devem ser obrigatoriamente armazenados o número de identificação do conceito e a disciplina a que este está relacionado.

Para cada conceito são chamadas funções para descrever o conteúdo do conceito propriamente dito, do exemplo, exercício e material complementar

Este algoritmo se comporta como segue na figura 5.5:

```
Inicia laço de repetição percorrendo toda da matriz de conteúdo (linha a linha)
Para cada linha lida realiza as seguintes ações:
   Criar nome do arquivo XML
   Abrir arquivo XML para gravação;
         Gravar número do conceito
         Gravar código da disciplina
         Chama função de conteúdo do conceito;
         Chama função de conteúdo do exemplo;
         Chama função de conteúdo do exercício;
         Chama função de conteúdo do material complementar;
   Fecha arquivo XML;
Fim do laço de repetição.
```

FIGURA 5.5 – Algoritmo de geração do arquivo XML de conteúdo do conceito

As funções de análise do conteúdo do conceito, exemplo, exercício e material complementar, apesar de serem distintas são muito similares. Todas elas buscam armazenar as informações básicas sobre estes elementos, tais como, identificador, descrição e arquivo principal HTML, bem como os cursos que possuem relacionamento. Também devem ser armazenados os arquivos associados ao HTML principal (figuras, vídeos e animações).

A estrutura básica pertinente a estas funções está descrita na figura 5.6:

```
Para cada linha da estrutura de dados:
        Verificar se o elemento possui conteúdo;
        Se o elemento possui conteúdo
        Início
                Armazenar o arquivo principal HTML relacionado a este elemento;
                Armazenar o conjunto de arquivos associados ao HTML (se
                houverem) com seu nome e respectivo código de ambiente
                tecnológico;
                Armazenar o conjunto  de cursos associados a este
                elemento, guardar o seu identificador e status;
                Passa para próxima linha da matriz;
        Fim
Finalizar percurso na matriz
```

FIGURA 5.6 – Estrutura básica para função de criação de conteúdo de conceito, exemplo, exercício ou material complementar.

Apenas para o conteúdo de conceito deve ser armazenado, juntamente com os dados descritos na figura 5.6 o nome do arquivo XML que está sendo criado.

5.4.4 Analisador semântico dos arquivos XML

O analisador foi desenvolvido com a finalidade de validar os documentos XML gerados pelo módulo de autoria. Como estes documentos devem ser válidos, o analisador tem como base do seu desenvolvimento não só os algoritmos de geração de arquivos XML mas também as DTD descritas na próxima seção.

A função de análise do documento XML ou *parser* é disparada após a criação destes documentos. Portanto primeiro ocorrem as verificações de erros na estrutura de dados, após estas verificações, se não houverem erros os arquivos XML são criados e devidamente analisados. O *parser* tanto deve contemplar o arquivo de estrutura de conceito como o arquivo de conteúdo do conceito. A figura 5.7 descreve a seqüências das ações e como o *parser* se comporta no processo de geração de arquivos XML.

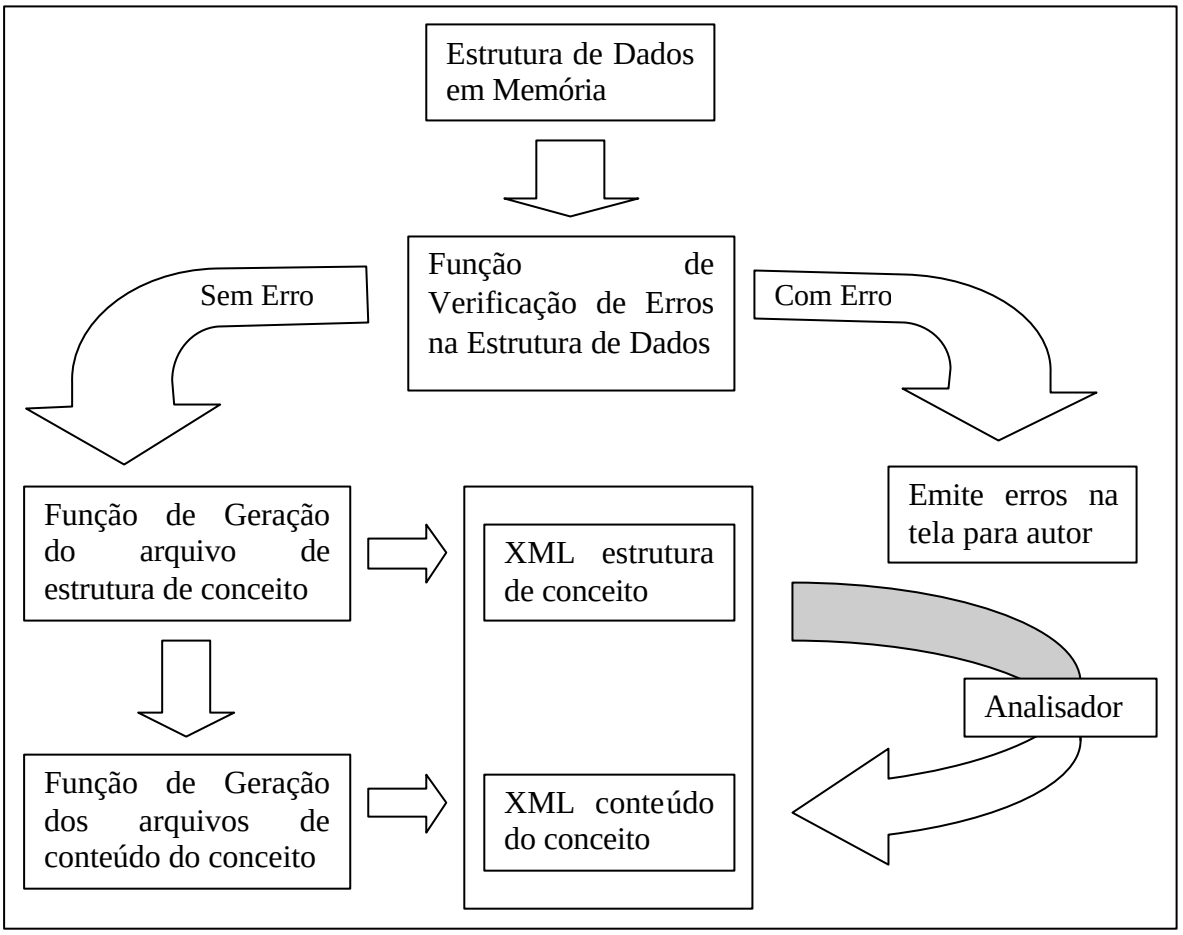

FIGURA 5.7 – Esquema do processo de geração de arquivos XML

Este analisador não se preocupa com erros sintáticos, já que o algoritmo deve gerar os arquivos XML sem este tipo de deficiência. Porém o analisador percorre todos os arquivos XML na sua extensão verificando se os elementos definidos na DTD estão na ordem correta, se os atributos classificados como requeridos estão devidamente preenchidos e se não há criação de nenhum fechamento de *tags* cruzadas.

O analisador foi desenvolvido utilizando a API DOM, esta foi utilizada como padrão de interface entre os documentos XML e os algoritmos do AdaptWeb que necessitam de uma representação hierárquica em memória, não só no módulo e armazenamento mas também nos módulos de Adaptação do conteúdo baseado no modelo do aluno [MAR2002] e Interface Adaptativa [GAS2002]

A API DOM é recomendada pela W3C [DOM98] pois é amplamente utilizada como interface por não depender de plataforma e nem de linguagem de programação. Esta provê formas de acesso para os programas e scripts atualizarem e consultarem dinamicamente os conteúdos, as estruturas e os estilos do documento XML [ROY2001]

O modelo DOM é uma forma de representar o documento XML em uma estrutura hierárquica [DOM98], como uma árvore ou um grafo. Nesta estrutura os nós são elementos, textos, informações e etc. e os elos são as ligações existente entre estes elementos. No caso do AdaptWeb o processador XML gera a estrutura de grafo na memória e a entrega ao analisador para este compará-la as DTD.

A navegação na estrutura criada pelo modelo DOM deve ser comparada as DTD, pois assim pode-se garantir que o documento XML é válido.

5.5 Saída de Dados do Armazenamento em XML

A saída de dados da presente fase é composta pelos arquivos XML. Tem-se um arquivo XML para a estrutura dos conceitos e um arquivo XML para cada conceito inserido pelo autor. Para criação destes arquivos XML foram definidas duas DTD (*Document Type Definition*) [W3C2000] que estão descritas a seguir. Na seção 5.4.2 são descritos os documentos XML gerados pelo algoritmo.

5.5.1 Apresentação das DTD

Os documentos XML gerados pelo módulo de armazenamento são classificados de acordo com [W3C2000] como documentos válidos, pois obedecem a todas as regras estabelecidas nas DTD e no cabeçalho do documento. As DTD utilizadas são externas, ou seja, não estão contidas no documento XML. As DTD retratam a definição formal dos tipos de documentos que são manipulados durante o processo de disponibilização do conteúdo instrucional.

Como citado anteriormente foram definidas duas DTD. Uma delas retrata a estrutura de conceitos definida pelo autor e a outra tem por finalidade descrever os conteúdos que são armazenados para cada conceito. A primeira DTD foi chamada de DTD da *estrutura de conceito*, e a segunda de DTD de *elementos do conteúdo do conceito*.

DTD da Estrutura de Conceito

Na fase de autoria todos os conceitos de uma disciplina devem conter suas devidas informações, tais como: descrição, número do conceito, pré-requisitos, palavras-chave relacionadas, entre outros elementos.

Para cada disciplina inserida na fase de autoria é criado um documento XML que tem como objetivo armazenar a estrutura hierárquica desta.

A DTD que retrata a estrutura do conceito é responsável pela definição da organização hierárquica do documento XML e está descrita a seguir na figura 5.8.

A raiz do documento XML definida nesta DTD é denominada material. Cada elemento material deve conter no mínimo um conceito, esta cardinalidade é definida através do operador +. A *tag* material possui o atributo disciplina, que deve conter o número de identificação da disciplina.

```
<!ELEMENT material (conceito+)>
<!ATTLIST material disciplina CDATA #REQUIRED>
<!ELEMENT conceito (prereq*,curso+,conceito*)>
<!ATTLIST conceito num          CDATA #REQUIRED
                   desc         CDATA #REQUIRED
                   abreviacao   CDATA #REQUIRED
                   arquivoxml   CDATA #REQUIRED
                   palchave     NMTOKENS #REQUIRED>

<!ELEMENT prereq EMPTY)>
<!ATTLIST prereq identprereq CDATA #REQUIRED>
<!ELEMENT curso (elementos*)>
<!ATTLIST curso identcurso CDATA #REQUIRED>
<!ELEMENT elementos (exemplo*, exercicio*, matcomp*)>
<!ELEMENT exemplo EMPTY>
<!ATTLIST exemplo possuiexemp CDATA #REQUIRED>
<!ELEMENT exercicio EMPTY>
<!ATTLIST exercicio possuiexerc CDATA #REQUIRED>
<!ELEMENT matcomp EMPTY>
<!ATTLIST matcomp possuimatcomp CDATA #REQUIRED>
```

FIGURA 5.8 – DTD da Estrutura de Conceito

O elemento *conceito* deve possuir um ou mais elementos curso, zero ou mais elementos *conceito* e zero ou mais elementos *prereq*. Esta organização caracteriza a montagem da estrutura hierárquica. Sempre o documento XML relativo à estrutura de conceito deve possuir no mínimo um conceito e este pode ou não possuir mais conceitos associados, como ilustra a Figura 5.9.

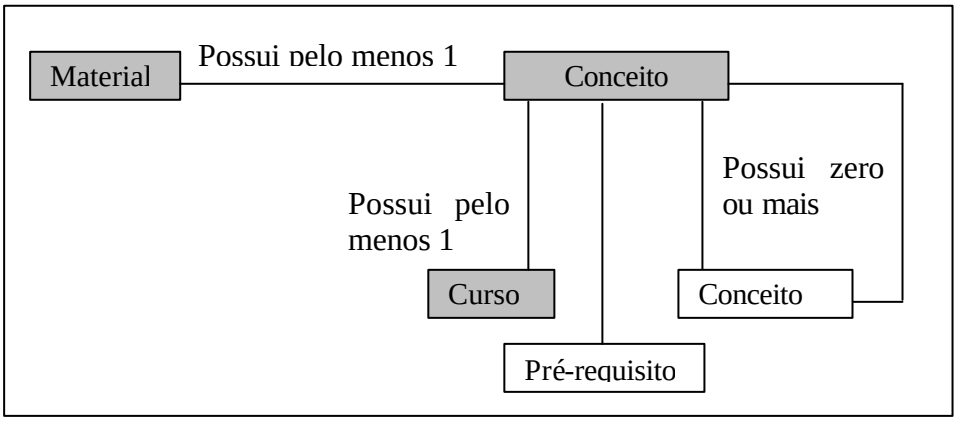

FIGURA 5.9 – Esquema de organização da estrutura hierárquica na DTD

Além deste relacionamento estabelecido na Definição de Tipo de Documento o elemento conceito possui os seguintes atributos listados na tabela 5.6:

TABELA 5.6. – Atributos do elemento conceito: DTD Estrutura do Conceito

Atributo	Descrição	Tipo	Ocorrência
num	Número do conceito.	CDATA	Obrigatório
desc	Descrição do conceito. Título	CDATA	Obrigatório
abreviacao	Abreviação do título do conceito	CDATA	Obrigatório
arquivoxml	Nome do arquivo XML relacionado a este conceito	CDATA	Obrigatório
palchave	Lista de palavras chaves relacionadas com o conceito	NMTOKENS	Obrigatório

Os atributos do elemento *conceito* são do tipo CDATA, pois são *strings.* Apenas o atributo *palchave* é do tipo NMTOKENS, pois este se caracteriza por ser uma lista de palavras separadas por espaços em branco [W3C2000].

O elemento *prereq* está relacionado com o elemento *conceito,* pois um conceito pode ter zero ou mais pré-requisitos. O elemento *conceito* também possui relação com o elemento *curso*. Para cada conceito da disciplina deve haver pelo menos um curso. Este permitirá que no módulo de adaptação do conteúdo baseado no modelo do aluno seja realizado um filtro para exibir apenas os conceitos associados ao curso a que o aluno pertence. Por exemplo, os alunos de um determinado curso só assistem as disciplinas que são relacionadas a este e só podem visualizar os conceitos que têm por definição o seu curso.

Cada curso tem como atributo um identificador único e não a sua descrição, este é requerido, pois é fator fundamental no processo de identificação do curso. Também deve ser armazenado se um curso, correspondente a um conceito, possui exemplo, exercício e material complementar vinculado a este. Caso o curso não possua exemplo, exercício e material complementar a *tag elementos* permanece vazia.

Estas informações devem estar contidas no documento XML, pois são utilizadas no processo de navegação do aluno. Assim durante a navegação do aluno é possível mensurar quais exercícios, exemplos ou material complementar estão relacionados com um determinado conceito e curso simultaneamente. Estes elementos que retratam a existência ou não de exercício, exemplo ou material complementar são dados como *implied,* pois não existe a obrigatoriedade de todo conceito ter os elementos citados acima.

DTD dos Elementos do Conteúdo do Conceito

Esta DTD tem por finalidade descrever os conteúdos que foram armazenados para cada conceito. Esses conteúdos educacionais foram divididos em conceito, exemplos, exercícios e materiais complementares.

Como pode ser observado na figura 5.10 a DTD que define os elementos que formam o conteúdo de um conceito possui como raiz a *tag textomaterial*. Para este estão definidos os seguintes elementos: conceito, exercício, exemplo e material complementar, nesta ordem. O elemento *conceito* deve existir exatamente uma vez dentro do documento XML, os demais citados podem existir zero ou mais vezes. O elemento raiz possui como atributo o código da disciplina e o número relacionado ao conceito, disciplina e num respectivamente, os dois são requeridos e são do tipo CDATA.

```
<!ELEMENT textomaterial (conceito+,exercicio*,exemplo*,matcomp*)>
<!ATTLIST textomaterial   disciplina CDATA #REQUIRED
                          num    CDATA #REQUIRED>
<!ELEMENT conceito (conteudo,curso+)>
<!ATTLIST conceito arquivo CDATA #REQUIRED>
<!ELEMENT exercicio (conteudo,curso+)>
<!ATTLIST exercicio idexerc      CDATA      #REQUIRED
                    descexerc    CDATA      #REQUIRED
                    arqexerc     CDATA      #REQUIRED
                    compexerc    CDATA      #REQUIRED>
<!ELEMENT exemplo (conteudo,curso+)>
<!ATTLIST exemplo idexemp      CDATA      #REQUIRED
                  descexemp    CDATA      #REQUIRED
                  arqexemp     CDATA      #REQUIRED
                  compexemp    CDATA      #REQUIRED>
<!ELEMENT matcomp (conteudo,curso+)>
<!ATTLIST matcomp idmatcomp    CDATA      #REQUIRED
                  descmatcomp  CDATA      #REQUIRED
                  arqmatcomp   CDATA      #REQUIRED>
<!ELEMENT conteudo (elementoconteudo*)>
<!ELEMENT elementoconteudo EMPTY>
<!ATTLIST elementoconteudo arqelem  CDATA    #REQUIRED
                           ambitec  CDATA    #REQUIRED>
<!ELEMENT curso EMPTY>
<!ATTLIST curso identcurso      CDATA      #REQUIRED>
```

FIGURA 5.10. – DTD Elementos do Conteúdo do Conceito

O primeiro elemento encontrado dentro da raiz é denominado *conceito*. Este retrata as informações pertinentes ao conteúdo instrucional do conceito, ou seja, a teoria relacionada ao conceito. O único atributo deste elemento é o nome do arquivo HTML que contém esta teoria.

Cada conceito possui um conteúdo e um ou mais cursos relacionado a este, que são representados apenas pelo seu identificador único. O conteúdo descreve os arquivos que são associados ao HTML que contém o conceito propriamente dito. Para estes arquivos devem ser armazenados seu nome e sua classificação com relação ao ambiente tecnológico do aluno. A classificação do ambiente tecnológico do aluno é utilizada no processo de apresentação do conteúdo. De acordo com o ambiente tecnológico do aluno estes arquivos associados ao HTML podem ou não ser exibidos.

Exercício, exemplo e material complementar obedecem a uma estrutura semelhante a do conceito, pois também devem ter relacionados a estes o conteúdo e um ou mais cursos. O que os difere são os atributos de cada um.

Na descrição do elemento exercício são utilizados atributos que armazenam o identificador único do exercício, a descrição do exercício, o arquivo HTML relacionado a este e o nível de complexidade do exercício. Estes elementos são todos adquiridos através da estrutura de dados do módulo de autoria.

O elemento *exemplo*, na sua descrição possui, assim como exercício, um identificador único, uma descrição do exemplo, o arquivo HTML relacionado e o nível de complexidade do exemplo.

Material complementar difere dos dois elementos citados acima apenas por não possuir o atributo complexidade, porém este possui o atributo identificador do material complementar, descrição e arquivo relacionado ao material complementar que pode ser tanto um arquivo HTML como um programa ou outro tipo de arquivo que pressuponha *download* para utilização na máquina local. Como o próprio nome diz, os materiais complementares são utilizados para informações adicionais ao conceito, portanto não necessitam de classificação quanto a sua complexidade.

Todos os atributos listados nesta DTD são do tipo CDATA, armazenam *string* [W3C2000]. Apenas os atributos *arqelem* e *ambitec*, referentes à arquivos associados ao conteúdo, que respectivamente armazenam o nome do arquivo e sua classificação quanto ao ambiente tecnológico, são dados como não requeridos. Isto ocorre porque nem todos os arquivos HTML referentes à conceito, exercício, exemplo e material complementar possuem arquivos associados.

A estruturação dos relacionamentos existentes entre os elementos da DTD de elementos do conteúdo do conceito pode ser visualizada através da representação gráfica a seguir. Os elementos que constam na figura 5.11 em tom acinzentado devem ocorrer pelo apenas uma vez, portanto os elementos *textomaterial* e conceito constam uma única vez no documento XML, enquanto que o elemento conteúdo deve constar uma vez em cada conceito, exemplo, exercício e material complementar existente.

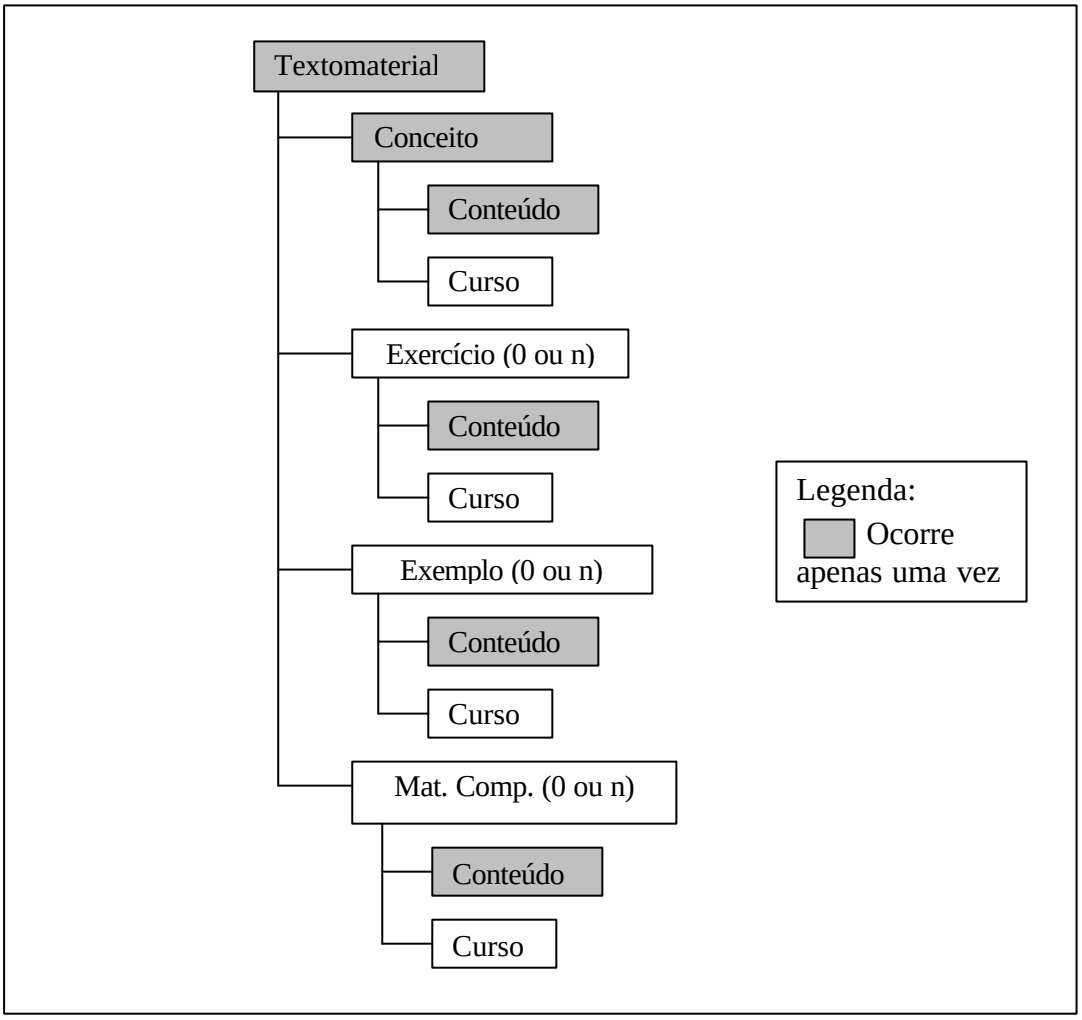

FIGURA 5.11 – Relacionamento dos Elementos de Conteúdo de Conceito

Assim como o ambiente AdaptWeb é totalmente orientado a conceito as DTD e por conseqüência os arquivos XML também são. Se a disciplina inserida no módulo de autoria possuir dez conceitos diferentes, então são criados dez arquivos XML distintos, um para cada conceito, com uma estrutura de elementos semelhante a da figura 5.11.

5.5.2 Apresentação dos documentos XML

Nas seções anteriores foram apresentados os algoritmos geradores dos documentos XML e as DTD que os descrevem, esta seção tem como objetivo detalhar os arquivos XML que foram gerados durante o módulo de armazenamento.

Durante a criação destes arquivos existem dois elementos que colaboram para que estes sejam gerados sem erros sintáticos e semânticos, são eles: as DTD e o *parser*. Além destes o algoritmo que foi implementado também está em conformidade com as DTD e com as regras sintáticas da linguagem XML.

Existem dois tipos de arquivos XML. Eles se diferem não só na estruturação, mas também na sua concepção, pois um deles é baseado na DTD de estrutura de conceito e o outro tipo é baseado na DTD de elementos do conteúdo do conceito. Os dois tipos de documentos XML gerados utilizam DTD, portanto são documentos que possuem a declaração *standalone* com valor *no*,

O primeiro tipo de arquivo XML, referente a DTD estrutura de conceito, é exemplificado através do código abaixo:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE material SYSTEM "dtd\Estrutura_Conceito.dtd">
<material disciplina="1">
    <conceito num=”1” desc=”Introdução” abreviação=”Introd”
    arquivoxml=”Introducao.html” palchave=”Computação, Introdução”>
            <curso identcurso=”1”>
                <elementos>
                    <exemplo possuiexemp="sim" />
                    <exercicio possuiexerc="sim" />
                    <matcomp possuimatcomp="sim" />
                </elementos>
            </curso>
            <curso identcurso="2">
                <elementos>
                    <exemplo possuiexemp="sim" />
                    <exercicio possuiexerc="sim" />
                    <matcomp possuimatcomp="sim" />
                </elementos>
            </curso>
            <curso identcurso="3">
                <elementos>
                    <exemplo possuiexemp="sim" />
                    <exercicio possuiexerc="sim" />
                    <matcomp possuimatcomp="sim" />
                </elementos>
             </curso>
        <conceito num="1.1" desc="Histórico" abreviacao="Hist"
        arquivoxml="Historico.html" palchave="Histórico">
        <prereq identprereq="1" />
            <curso identcurso="1">
                <elementos>
                    <exemplo possuiexemp="sim" />
                    <exercicio possuiexerc="sim" />
                    <matcomp possuimatcomp="sim" />
                </elementos>
            </curso>
            <curso identcurso="2">
                <elementos>
                    <exemplo possuiexemp="sim" />
                </elementos>
            </curso>
            <curso identcurso="3">
                <elementos>
                    <exemplo possuiexemp="sim" />
                </elementos>
            </curso>
        </conceito>
    </conceito>
</material>
```

As duas primeiras linhas do código do documento XML estrutura do conceito são chamadas de linhas de declaração do arquivo XML. Nelas foram declaradas a

versão da linguagem XML utilizada, o conjunto de caractere utilizado no documento XML, a existência de DTD, a definição da *tag* raiz e o local onde a DTD está armazenada.

A disciplina descrita por este documento XML possui código igual a 1, isto pode ser verificado na linha <material disciplina="1">. Esta é a *tag* raiz do documento XML. O documento XML possui duas *tags* conceito, que estão internas a *tag* material, portanto a disciplina de código 1 possui dois conceitos.

A organização do documento mostra que a *tag* conceito cujo atributo **num** é igual a 1.1 é interna a *tag* conceito que possui o atributo **num** é igual a 1, esta por sua vez é interna a *tag* material que é a raiz do documento XML.

Para cada *tag* conceito podem existir um ou mais cursos. Um curso é representado pelo seguinte trecho de código:

```
<curso identcurso="2">
    <elementos>
        <exemplo possuiexemp="sim" />
        <exercicio possuiexerc="sim" />
        <matcomp possuimatcomp="sim" />
    </elementos>
</curso>
```

Cada curso deve ter armazenado o seu identificador e se este possui exemplo, exercício ou material complementar. As *tags* exemplo, exercício e matcomp são ditas *tags* vazias (*empty*) e só são gravadas no documento XML se existirem exemplo, exercício ou material complementar para o curso em questão.

Como este documento XML possui dois conceitos então são gerados pelo algoritmo dois arquivos XML para o conteúdo de cada um dos conceitos. O arquivo XML que descreve o primeiro conceito do documento acima é dividido em dados sobre o conceito, exercícios, exemplos e material complementar. Esta divisão é bem explícita visto que a cada ocorrência do conceito, de um exercício, de um exemplo ou de um material complementar são criados pares de *tags* que englobam os elementos relacionados.

Assim, as informações referentes ao conceito são armazenadas em um par de *tags* <conceito> </conceito>. Em cada documento XML existe apenas um par deste, pois o conceito é único para cada arquivo XML. A ocorrência de exercício, exemplo e material complementar não é necessariamente unitária, portanto se houverem, por exemplo, três exercícios relacionados a um determinado conceito existirão três pares de *tags* <exercicio> </exercicio>. Esta organização também é válida para exemplo e material complementar.

Quando há a ocorrência de conceito, exemplo, exercício e material complementar sempre são armazenados nas respectivas *tags* os atributos descritores destes. O conceito possui apenas o atributo arquivo, e os demais possuem: identificador, descrição, arquivo relacionado e complexidade deste. Material complementar não possui atributo complexidade. Todos possuem *tags* que representam os cursos relacionados com o conteúdo, e um par de *tag* denominado conteúdo que armazena as informações sobre os elementos associados ao HTML principal.

O código do documento XML que armazena dados sobre o primeiro conceito relacionado no XML de estrutura de conceito discutido anteriormente é descrito a seguir.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Conceito.dtd">
<textomaterial disciplina="1" num="1">
    <conceito arquivo="Introdução.html">
        <conteudo>
            <elementoconteudo arqelem="t_can.jpg" ambitec="1" />
        </conteudo>
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
    </conceito>

    <exercicio idexerc="1" descexerc="primeiro exercicio"
    arqexerc="exercicio1.html" compexerc="facil">
        <conteudo>
            <elementoconteudo arqelem=" " ambitec="1" />
        </conteudo>
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
    </exercicio>

    <exemplo idexemp="1" descexemp="Primeiro Exemplo"
    arqexemp="EXEMPLO1.htm" compexemp="Fácil">
        <conteudo>
            <elementoconteudo arqelem="figura.gif" ambitec="1" />
        </conteudo>
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
    </exemplo>

    <matcomp idmatcomp="1" descmatcomp="material complementar 1"
    arqmatcomp="matcomp1.html">
        <conteudo>
            <elementoconteudo arqelem="figura_matcomp1.gif"
            ambitec="1" />
        </conteudo>
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
    </matcomp>
</textomaterial>
```

Como no documento XML anterior este também mostra a utilização de uma DTD, para este documento a DTD é Elementos do Conteúdo do Conceito, já descrita. No caso deste código o conceito de número 1, como é observado no elemento num da *tag* textomaterial, tem apenas um exercício, um exemplo e um material complementar. Para cada um destes elementos são listados seus atributos, seus elementos relacionados com o conteúdo do HTML principal e os elementos cursos relacionados.

6 Estudo de Caso

Com a finalidade de validar todo o processo de geração de conteúdo para ser apresentado no Ambiente AdaptWeb foi desenvolvido um estudo de caso aplicado em todos os módulos do ambiente.

A disciplina utilizada neste estudo de caso foi a disciplina de Computação Numérica e Algébrica. Devido a sua grande extensão de conteúdo foi escolhido o desenvolvimento de uma parte da disciplina que é denominada Sistemas de Equações Lineares Algébricas. Esta disciplina é ministrada pela Professora Dr. Maria Angélica O. C. Brunetto no Departamento de Ciência da Computação da Universidade Estadual de Londrina.

A área de Computação Científica, que envolve o projeto e análise de algoritmos Numéricos e algébricos, oferece oportunidades diversas de pesquisa na Computação e Matemática, além de dar suporte para soluções computacionais eficientes aplicáveis em diversas áreas, seja na indústria, nas engenharias, nas áreas biológicas e da saúde, e na própria computação.

Com o avanço de novas técnicas de computação simbólica e algébrica, novas soluções têm sido incorporadas nos tradicionais métodos numéricos. Isto requer uma atualização constante dos conteúdos instrucionais que são desenvolvidos nas disciplinas que abordam esta área de conhecimento.

Considerando que diversos cursos superiores da área de exatas têm em seu currículo uma disciplina de Métodos Numéricos, e que os interesses dos alunos são diferentes de acordo com a natureza do curso, observa-se a necessidade de uma abordagem diferenciada. Aliados a este fato têm-se diversos recursos que a tecnologia educacional vem oferecendo, podendo propiciar formas alternativas de aprendizagem.

O objetivo principal deste estudo de caso é a validação do Ambiente AdaptWeb, porém em conseqüência tem-se o desenvolvimento e a implantação de um curso via web sobre um conceito importante de Computação Numérica e Algébrica que possui três públicos distintos: alunos de Engenharia, alunos de Ciência da Computação e alunos de Matemática.

6.1 Contextualização do Estudo de Caso de Computação Algébrica e Numérica

A disciplina de Computação Algébrica e Numérica possui vários conceitos no seu conteúdo programático. A título de teste, para este estudo de caso foi selecionado o módulo de Sistemas de Equações Lineares Algébricas, com seus sub-itens. Além destes foi inserido também o conceito de número 1, chamado Introdução. A tabela 6.1 descreve esta organização. A figura 6.1 é a representação hierárquica da tabela de conceitos.

No estudo de caso, para cada conceito foi relacionado um arquivo HTML com seu conteúdo teórico. Este arquivo HTML, os arquivos associados a ele e os respectivos conceitos seguem no Anexo 01.

TABELA 6.1. – Organização do Conceito Sistemas de Equações Lineares Algébricas

1. Contexto e Objetivos da Computação Algébrica e Numérica
2. Sistemas de Equações Lineares Algébricas
2.1 Introdução
2.2 Métodos Diretos
2.2.1 Método de Gauss
2.2.1.1 Algoritmo da Triangularização
2.2.1.2 Algoritmo da Retrossubstituição
2.2.2 Método de Gauss com pivotamento
2.2.2.1 Condicionamento de Matrizes
2.2.3 Método da Decomposição LU
2.2.4 Método de Cholesky
2.3 Métodos Iterativos
2.3.1 Método de Jacobi
2.3.2 Método de Gauss-Seidel

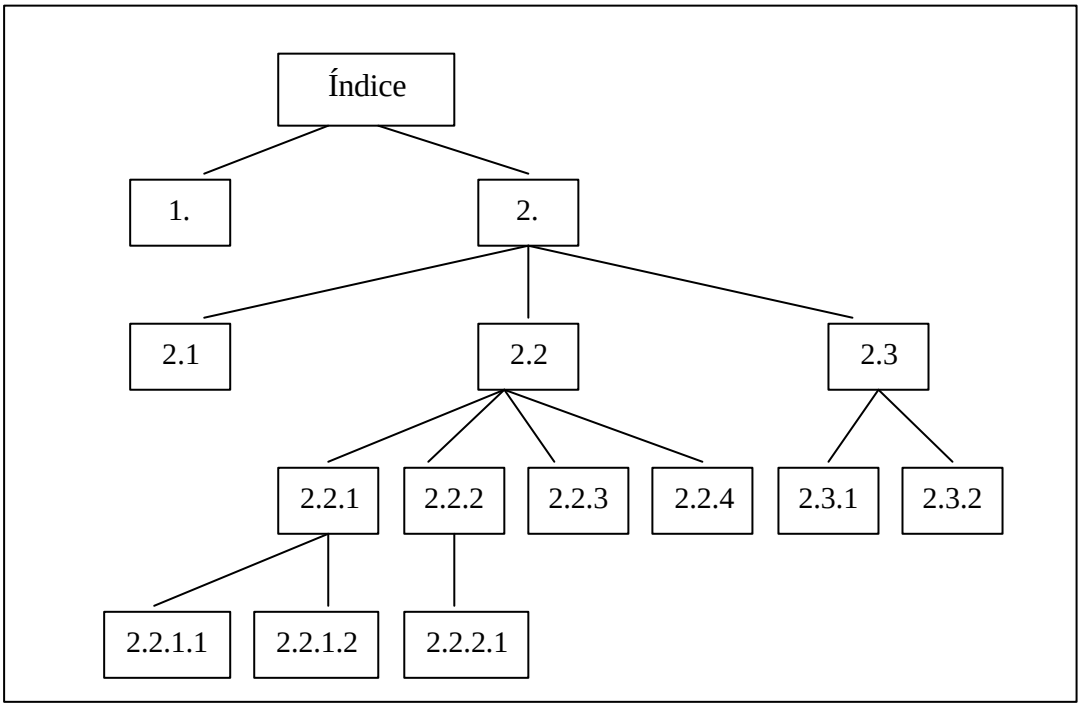

FIGURA 6.1 – Estrutura hierárquica baseada na tabela 6.1

Além de determinar os arquivos HTML e arquivos associados aos conceitos também devem ser especificados os cursos que podem visualizá-los. Estes cursos estão divididos de acordo com os três cursos de graduação que assistem à disciplina, são eles: Ciência da Computação, Engenharia Civil e Matemática. No estudo de casos os identificadores destes cursos são: 1, 2 e 3 respectivamente.

Os conceitos de número 2.2.2, 2.2.3, 2.2.4, 2.3.1 podem ser disponibilizados apenas para os alunos de Computação e Engenharia. Os alunos de Matemática não

possuem acesso a estes conceitos, por conseqüência não possuem acesso aos exemplos, exercícios, material complementar destes conceitos, se estes existirem. Os demais conceitos, listados na tabela 6.1, podem ser visualizados pelos três cursos.

Apenas o conceito Introdução, com identificador igual a 2.1, possui exercício associado. Exemplos são encontrados nos conceitos 2.2.1 e 2.2.2. Existem materiais complementares disponibilizados para os conceitos 2.2.1, 2.2.2, 2.2.2.1, 2.2.3, 2.2.4, 2.3.1 e 2.3.2.

Esta foi a estruturação definida no presente estudo de caso, e apartir destes dados foram gerados os arquivos XML discutidos na próxima seção deste capítulo.

6.2 Instância de documento XML utilizada no processo de armazenamento

Este estudo de caso gerou no total quinze documentos XML. Um documento XML corresponde a estrutura de conceito global, com a representação hierárquica dos quatorze conceitos abordados na disciplina. Os outros documentos XML são referentes ao conteúdo de cada um dos quatorze conceitos inseridos. A figura 6.2 mostra como foi organizado o primeiro conceito da disciplina de Computação Numérica e Algébrica no documento XML que representa a estrutura de conceito global.

O conceito denotado na figura 6.2 não possui exemplo, exercício e nem material complementar, possui apenas o conteúdo inerente à teoria, portanto o par de *tags* <elementos></elementos> está sempre vazio para todos os cursos que foram classificados como aptos para este conceito.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE material SYSTEM "dtd\Estrutura_Topico.dtd">
<material disciplina="1">
  <conceito num="1"  desc="Contexto e Objetivos da Computação Algébrica e
  Numérica"  abreviacao="Cont. Obj. C.A.N." arquivoxml="Contexto.html"
  palchave="Objetivos, Computação, Algébrica, Numérica">
    <curso identcurso="1">
        <elementos>  </elementos>
    </curso>
    <curso identcurso="2">
        <elementos>  </elementos>
    </curso>
    <curso identcurso="3">
        <elementos>  </elementos>
    </curso>
     </conceito>                    .
                                    .
                                    .
```

FIGURA 6.2 – Documento XML de Estrutura do Conceito

A ocorrência de exemplo, exercício ou de material complementar em um conceito é representada por um conjunto de *tags* mostrado na figura 6.3

```
                                                .
                                                .
                                                .
<conceito num="2.2.1" desctop="Método de Gauss"  abreviacao="Metod. Gauss"
arquivoxml="Sela_Gauss.html" palchave="Sistemas, Equações">
 <prereq identprereq="2"/>
  <curso identcurso="1">
      <elementos>
              <exemplo possuiexemp="sim"/>
               <matcomp possuimatcomp="sim"/>
      </elementos>
  </curso>
  <curso identcurso="2">
      <elementos>
              <exemplo possuiexemp="sim"/>
               <matcomp possuimatcomp="sim"/>
      </elementos>
  </curso>
  <curso identcurso="3">
      <elementos>
              <exemplo possuiexemp="sim"/>
               <matcomp possuimatcomp="sim"/>
      </elementos>
  </curso>

                                                .
                                                .
                                                .
```

FIGURA 6.3 – Código XML com ocorrência de exemplo e material complementar

Através deste documento XML com a estrutura geral de conceitos é possível montar o menu de conceitos que será exibido na tela para o aluno. A tarefa da montagem deste menu é de responsabilidade do módulo Interface Adaptativa do ambiente AdaptWeb [GAS2002].

A figura 6.4 mostra como foi organizado o documento XML que armazena o conteúdo do conceito de nome: Contexto e Objetivos da Computação Algébrica e Numérica, este é o primeiro conceito inserido na autoria. Este conceito não possui arquivos associados ao HTML de teoria, pois o par de *tag* <conteudo> está vazio. Os três cursos podem acessar este conceito.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Conceito.dtd">
<textomaterial disciplina="1" num="1">
      <conceito arquivo="Contexto.html">
            <conteudo>    </conteudo>
            <curso identcurso="1"/>
            <curso identcurso="2"/>
            <curso identcurso="3"/>
      </conceito>
</textomaterial>
```

FIGURA 6.4 – Documento XML de conteúdo do conceito de número 1

Quando há a ocorrência de exemplo, exercício ou material complementar em um conceito, existe a definição de quais cursos estão relacionados com este conceito, além disto são listadas no documento XML alguns atributos de identificação do exemplo, exercício ou material complementar.

Uma instância de documento XML que possui material complementar está descrita na figura 6.5. O conceito descrito é o de número 2.3.1, que descreve o conteúdo instrucional do Método de Jacobi. Este conceito não está disponível para o curso de Matemática, portanto os materiais complementares deste também não estão disponíveis para este curso. Os cursos que estão associados à *tag* conceito são os cursos de identificador 1 e 2, Computação e Engenharia. Apenas o arquivo HTML referente ao conteúdo instrucional do conceito possui arquivo associado. Este arquivo consta no documento XML da seguinte forma:

```
<conceito arquivo="Sela_Jacobi.html">
      <conteudo>
            <elementoconteudo arqelem="Sist_Linear27.gif" ambitec="1"/>
      </conteudo>
                                    .
                                    .
                                    .
</conceito>
```

No código acima existe apenas um arquivo associado ao conteúdo do arquivo Sela_Jacobi.html. Este arquivo, Sist_Linear27.gif, possui o atributo *ambitec*, código do ambiente tecnológico igual a 1. Este atributo é utilizado no módulo de Adaptação do Conteúdo baseado no modelo do aluno no processo de filtro de arquivos associados. Estes arquivos serão exibidos ao aluno se o ambiente tecnológico armazenado no documento XML for igual ou inferior ao ambiente tecnológico do aluno no momento da navegação.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Topico.dtd">
<textomaterial disciplina="1" num="2.3.1">
    <conceito arquivo="Sela_Jacobi.html">
        <conteudo>
            <elementoconteudo arqelem="S ist_Linear27.gif" ambitec="1"/>
        </conteudo>
         <curso identcurso="1"/>
         <curso identcurso="2"/>
    </conceito>
    <matcomp idmatcomp="1" descmatcomp="Programas em PHP - Jacobi"
    arqmatcomp="Jacobi_pt.html" >
        <conteudo>    </conteudo>
         <curso identcurso="1"/>
         <curso identcurso="2"/>
    </matcomp>
    <matcomp idmatcomp="2" descmatcomp=" Programas em PHP - Jacobi"
    arqmatcomp="Jacobi_pt.php" >
        <conteudo>    </conteudo>
        <curso identcurso="1"/>
        <curso identcurso="2"/>
    </matcomp>
    <matcomp idmatcomp="3" descmatcomp=" Programas em PHP - Jacobi"
    arqmatcomp="Jacobi2_pt.php" >
        <conteudo>    </conteudo>
        <curso identcurso="1"/>
        <curso identcurso="2"/>
    </matcomp>
</textomaterial>
```

FIGURA 6.5 – Documento XML de conteúdo do conceito de número 2.3.1

Neste estudo de casos foram criados 14 arquivos que estão em Anexo (Anexos 2 a 11). Cada um deles possui suas particularidades, pois cada um descreve um conceito levando em consideração seu arquivo HTML, com ou sem arquivos associados, seus exercícios, exemplos e materiais complementares com os respectivos cursos.

7 Conclusão

O trabalho desenvolvido utilizou a linguagem XML como mecanismo para armazenamento de conteúdo instrucional do AdaptWeb, um ambiente hipermídia adaptativo, orientado a conceito que possui adaptabilidade na navegação e na apresentação. Os documentos XML apresentam todo o conteúdo do curso, bem como informações que serão utilizadas pelos demais módulos do ambiente para a aplicação dos conceitos de adaptabilidade.

Com o uso de XML o conteúdo instrucional foi mantido separado da apresentação, tornando assim mais fácil a aplicação de técnicas de adaptabilidade tanto para a navegação como para a apresentação [W3C2000]. Não são inseridos conteúdos em HTML dentro dos arquivos XML. O autor apenas anexa os arquivos HTML de conteúdo durante a utilização da ferramenta de autoria e este não necessita conhecer a linguagem XML. Assim todo o processo de estruturação de conteúdo, apresentação e navegação tornam-se transparente para o autor.

O ambiente AdaptWeb é composto por quatro módulos distintos: (1) Autoria, (2) Armazenamento de dados em XML, (3) Adaptação do conteúdo baseado no modelo do aluno e (4) Interface adaptativa, que podem ser visualizados através da Figura 4.1. O módulo de Armazenamento de conteúdo instrucional em documentos XML, objeto deste trabalho, realiza a conversão de dados proveniente da fase de autoria em arquivos XML, fornecendo-os para os próximos módulos do ambiente. Estes, por meio dos documentos XML apresentam o conteúdo instrucional ao aluno adaptado de acordo com o seu perfil e com os pré-requisitos.

Com a finalidade de tornar o uso de XML efetivo no ambiente AdaptWeb foram desenvolvidas as seguintes atividades:

- Definição dos mecanismos de integração entre o módulo de Autoria e o módulo de Armazenamento de Dados em XML.
- Definição dos mecanismos de integração entre o módulo de Armazenamento de Dados em XML e os módulos de Adaptação de conteúdo baseado no modelo do aluno e Interface Adaptativa.
- Desenvolvimento das DTD [W3C2000]. Foram desenvolvidas duas DTD, uma para definição do documento XML de estrutura do conceito e outra para definição do documento XML de elementos do conteúdo do conceito. Ambos arquivos baseiam-se nos conceitos da disciplina, visto que o ambiente AdaptWeb é orientado à conceito.
- Implementação do algoritmo de detecção de erro na estrutura de dados em memória proveniente da fase de autoria. Este algoritmo verifica se todos os campos da estrutura de dados foram devidamente preenchidos durante a fase de autoria. Se houver algum erro neste preenchimento este deve ser sinalizado ao usuário e os arquivos XML não são gerados.
- Implementação do algoritmo de conversão da estrutura de dados para os arquivos XML. Este algoritmo tem como entrada uma estrutura matricial linear e realiza a conversão e organização dos conceitos de forma hierárquica no documento XML de estrutura do conceito. Além deste arquivo de estrutura do conceito, para cada conceito inserido na fase de autoria é gerado um arquivo XML com seus elementos e características.

- Implementação do *parser* (analisador) dos arquivos XML para validação semântica dos arquivos XML, utilizando como base para programação a API (*Application Programming Interface)* DOM (*Document Object Model*) [DOM98]. A API DOM trata a informação armazenada no documento XML como um modelo de objetos hierárquicos, criando uma árvore de nós baseada na estrutura do documento XML. Todos os arquivos XML gerados passam por este *parser* para serem validados de acordo com as DTD.
- Aplicação do Estudo de Casos para validação dos algoritmos desenvolvidos.

Os arquivos XML foram criados como elementos genéricos, permitindo assim a disponibilização de qualquer tipo de disciplina. A recomendação XML admite que sejam especificadas regras sintáticas que, impostas a documentos XML, definem uma linguagem com a qual os documentos devem estar em conformidade, isto e feito através das DTD [PIM2001].

Assim, os algoritmos criados, com base nas DTD desenvolvidas, forneceram as diretivas para a geração de material em XML, a partir de materiais diversos em HTML, inseridos pelo autor na fase de autoria. Estes documentos XML foram utilizados como base para o desenvolvimento das técnicas de adaptabilidade de navegação e apresentação nos demais módulos do ambiente.

Foi implementado um algoritmo que gera um documento XML para armazenamento e organização da estrutura de conceitos de uma disciplina, exibindo os relacionamentos existentes entre os conceitos, pré-requisitos e arquivos HTML associados a estes. Os outros arquivos XML gerados dizem respeito às informações sobre o conteúdo instrucional de um determinado conceito. Estes documentos contêm as informações sobre os conteúdos instrucionais de um conceito, inclusive informações sobre o arquivo HTML de conteúdo.

O arquivo XML de estrutura dos conceitos é utilizado para prover a adaptabilidade de navegação e apresentação. Através da análise deste documento XML é possível estabelecer qual conceito pode ou não ser visitado por um aluno. O uso de XML facilita este processo de filtragem dos conceitos, pois podem ser analisados os conceitos e seus pré-requisitos e quais conteúdos estão disponíveis para cada perfil de usuário.

Os arquivos XML que armazenam as informações sobre os conceitos são utilizados para adaptabilidade do conteúdo, pois as mídias existentes nos conteúdos instrucionais podem ser filtradas de acordo com o ambiente tecnológico em que o aluno se encontra. O filtro é aplicado no arquivo HTML de conteúdo inserido pelo autor.

No módulo de armazenamento de conteúdo instrucional não foi utilizado banco de dados. Todo o processo de armazenamento e organização do conteúdo instrucional se deu através de arquivos XML.

Para o propósito deste trabalho, XML substituiu o uso de banco de dados, pois a necessidade do ambiente em questão não era apenas o armazenamento dos dados, mas também a possibilidade de autodescrição do conteúdo instrucional armazenado, isto pode ser conseguido através da definição de *tags* com rótulos relevantes ao contexto tratado e também através da organização da marcação dos documentos XML. Além disto XML é portável e facilita o processo de armazenamento dos dados de forma hierárquica, assim como os bancos de dados, XML também proporciona

armazenamento, esquemas lógicos, linguagens de consulta e programação por meio de interfaces.

7.1 Trabalhos Futuros

O desenvolvimento deste trabalho mostrou que o uso da linguagem XML em ambientes hipermídia adaptativos em geral, e também no ambiente AdaptWeb, é de grande valia, pois o fato de XML ser uma linguagem extensível permite que os domínios educacionais, por mais distintos que sejam, possam ser armazenados de maneira a facilitar os aspectos de adaptabilidade.

Neste trabalho as DTD desenvolvidas foram fixas. Existiam apenas dois modos de organização do conteúdo instrucional em documentos XML. Propõe-se como trabalho futuro o desenvolvimento de um ambiente onde o autor possa, na fase de autoria, criar a estrutura de sua disciplina de maneira livre. Com esta permissão, a criação das DTD e de todo processo de armazenamento de dados em documento XML será dinâmico não só no que tange os conteúdos, mas também no que se refere a forma como estes conteúdos devem ser organizados.

Anexo 1 Organização dos Arquivos do Estudo de Caso

Tabelas de organização dos arquivos utilizados no Estudo de Casos. São listados o conceito, arquivo HTML principal e arquivos associados ao HTML existente.

Número e nome do Conceito	Arquivo Principal (HTML)	Arquivos Associados
1. Contexto e Objetivos da Computação Algébrica e Numérica	Introdução.html	Nenhum
2. Sistemas de Equações Lineares Algébricas	Sistemas_de_Equaçoes Lineares_Algébricas.html	Nenhum
2.1 Introdução	Sela_introd.html	Down1.gif
2.2 Métodos Diretos	Métodos_diretos.html	Nenhum
2.2.1 Método de Gauss	Sela_gauss.html	Sist_linear28.gif a Sist_linear34.gif T_can.jpg
2.2.1.1 Algoritmo da Triangularização	Sela_triang.html	Nenhum
2.2.1.2 Algoritmo da Retrossubstituição	Sela_retro.html	Nenhum
2.2.2 Método de Gauss com pivotamento	Sela_gaussp.html	Sist_linear01.gif a Sist_linear05.gif
2.2.2.1 Condicionamento de Matrizes	Condicionamento_matrizes .html	Sist_linear12.gif
2.2.3 Método da Decomposição LU	Sela_LU.html	Sist_linear13.gif a Sist_linear 19.gif
2.2.4 Método de Cholesky	Sela_Cho lesky.html	Sist_linear20.gif a Sist_linear26.gif
2.3 Métodos Iterativos	Métodos_iterativos.html	Nenhum
2.3.1 Método de Jacobi	Sela_Jacobi.html	Sist_linear27.gif
2.3.2 Método de Gauss-Seidel	Sela_gaussseidel.html	T_can.jpg

TABELA 1 – Organização dos conceitos e seus arquivos

Número e nome do Exercício	Arquivo Principal (HTML)	Arquivos Associados
2.1 Exercícios Introdutórios	Ex1facil.html	Nenhum
2.1 Exercícios Introdutórios	Ex2facil.html	Nenhum
2.1 Exercícios Introdutórios	Ex3medio.html	Nenhum
2.1 Exercícios Introdutórios	Ex4medio.html	Nenhum
2.1 Exercícios Introdutórios	Ex5complexto.html	Nenhum
2.1 Exercícios Introdutórios	Ex6.html	Nenhum
2.1 Exercícios Introdutórios	Ex7facil.html	Nenhum
2.1 Exercícios Introdutórios	Ex8facil.html	Nenhum

TABELA 2 – Organização dos exercícios e seus arquivos

Número e nome do Exemplo	Arquivo Principal (HTML)	Arquivos Associados
2.2.1 Exemplo do método de gauss	Exp_gauss.html	Sist_linear35.gif a Sist_linear40.gif
2.2.2 Exemplo do método de Gauss com pivotamento	Exp_Gauss_pivot.html	Sist_linear6.gif a Sist_linear10.gif

TABELA 3 – Organização dos exemplos e seus arquivos

Número e nome do Material Complementar	Arquivo Principal (HTML)	Arquivos Associados
2.2.1 Bibliotecas do Maple	BibM_gauss.html	Gausselim01.gif Gausselim02.gif Backsub01.gif Backsub02.gif
2.2.1 Recursos do Maple para programar o método de Gauss (Maple Versão 4)	Teoria_gauss.mws (para download)	Nenhum
2.2.1 Recursos do Maple para programar o método de Gauss	Teoria_Gauss.html	Teoria_Gauss1.gif a Teoria_Gauss17.gif
2.2.1 Programas usando Maple	Elim_gauss.html	Nenhum
2.2.1 Programas usando Maple (download)	Elim_gauss.mws	Nenhum
2.2.1 Programas usando Maple (filme)	Elimina_Gauss.avi	Nenhum
2.2.1 Programas Maple (filme para download)	Elimina_Gauss.exe	Nenhum
2.2.2 Programas usando Maple em html	Gauss_pivot.html	ProgM_Sela2221.gif a ProgM_Sela22223.gif
2.2.2 Programas em maple para download	Gauss_pivot.mws	Nenhum
2.2.2 Programas em PHP para demonstração	Gauss_pt.html Gauss_pt.php Gauss2_pt.php	Nenhum
2.2.3 Programas em Maple para decomp. LU	Ludecomp.html	Sist_Linear49.bmp a Sist_Linear54.bmp
2.2.3 Programas em Maple - para download	Ludecomp.mws	Nenhum
2.2.3 Programas em PHP para demonstração	Ludecomp_pt.html Ludecomp_t.php Ludecomp2_pt.php	Nenhum
2.2.4 Bibliotecas do Maple – Cholesky		Nenhum
2.2.4 Programas em Maple html		Nenhum
2.2.4 programas em Maple (download)		Nenhum
2.2.4 Programas em PHP	Cholesky_pt.html Cholesky_pt.php Cholesky2_pt.php	Nenhum

2.3.1 Jacobi - Programas PHP	Jacobi_pt.html Jacobi_pt.php Jacobi2_pt.php	Nenhum
2.3.2 Gauss-Seidel Programas Php	Gauss-seidel_pt.html Gauss-seidel_pt.php Gauss-seidel2_pt.php	Nenhum

TABELA 4 – Organização dos Materiais Complementares e seus arquivos

Anexo 2 Documento XML Estrutura dos Conceitos

Documento XML Estrutura dos Conceitos gerado no Estudo de Casos

Documento XML de descrição da Estrutura dos Conceitos gerados no estudo de casos

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE material SYSTEM "dtd\Estrutura_Topico.dtd">
<material disciplina="1">
  <topico numtop="1" desctop="Contexto e Objetivos da Computação Algébrica e
      Numérica" abreviacao="Cont. Obj. C.A.N." arquivoxml="Contexto.html"
      palchave="Objetivos, Computação, Algébrica, Numérica">
    <curso identcurso="1"> <elementos /> </curso>
    <curso identcurso="2"> <elementos /> </curso>
    <curso identcurso="3"> <elementos /> </curso>
  </topico>
  <topico numtop="2" desctop="Sistemas de Equações Lineares Algébricas"
      abreviacao="Sist. Lin. de Eq. Alg"
      arquivoxml="Sistemas_de_Equacoes_Lineares_Algebrica.html"
      palchave="Sistemas, Equações">
    <prereq identprereq="1" />
    <curso identcurso="1"> <elementos /> </curso>
    <curso identcurso="2"> <elementos /> </curso>
    <curso identcurso="3"> <elementos /> </curso>
    <topico numtop="2.1" desctop="Introducao" abreviacao="Intro"
        arquivoxml="Introducao.html" palchave="Sistemas, Equações,
        Introdução">
      <prereq identprereq="1" />
      <curso identcurso="1">
        <elementos>
          <exercicio possuiexerc="sim" />
        </elementos>
      </curso>
      <curso identcurso="2">
        <elementos>
          <exercicio possuiexerc="sim" />
        </elementos>
      </curso>
      <curso identcurso="3">
        <elementos>
          <exercicio possuiexerc="sim" />
        </elementos>
      </curso>
    </topico>
    <topico numtop="2.2" desctop="Métodos Diretos" abreviacao="Metod.
        Dir." arquivoxml="Metodos_Diretos.html" palchave="Método,
        Direto">
      <curso identcurso="1"> <elementos /> </curso>
      <curso identcurso="2"> <elementos /> </curso>
      <curso identcurso="3"> <elementos /> </curso>
      <topico numtop="2.2.1" desctop="Método de Gauss"
          abreviacao="Metod. Gauss" arquivoxml="Sela_Gauss.html"
          palchave="Sistemas, Equações">
        <prereq identprereq="2" />
        <curso identcurso="1">
```

```
      <elementos>
          <exemplo possuiexemp="sim" />
          <matcomp possuimatcomp="sim" />
      </elementos>
    </curso>
    <curso identcurso="2">
      <elementos>
          <exemplo possuiexemp="sim" />
          <matcomp possuimatcomp="sim" />
      </elementos>
    </curso>
    <curso identcurso="3">
      <elementos>
          <exemplo possuiexemp="sim" />
          <matcomp possuimatcomp="sim" />
      </elementos>
    </curso>
    <topico numtop="2.2.1.1" desctop="Algoritmo
        Triangularizacao" abreviacao="Alg. Triang."
        arquivoxml="Sela_Triang.html"
        palchave="Triangularização">
       <prereq identprereq="2" />
      <curso identcurso="1"> <elementos /> </curso>
      <curso identcurso="2"> <elementos /> </curso>
      <curso identcurso="3"> <elementos /> </curso>
    </topico>
    <topico numtop="2.2.1.2" desctop="Algoritmo
        Retrosubstituição" abreviacao="Alg. Retrosub."
        arquivoxml="SEla_Retro.html"
        palchave="Retrosubstituição">
       <prereq identprereq="2.2.1.1" />
      <curso identcurso="1"> <elementos /> </curso>
      <curso identcurso="2"> <elementos /> </curso>
      <curso identcurso="3"> <elementos /> </curso>
    </topico>
  </topico>
  <topico numtop="2.2.2" desctop="Método de Gauss com
      Pivotamento" abreviacao="Gauss Pivot."
      arquivoxml="Sela_Gaussp.html" palchave="Gauss,
      Pivotamento">
     <prereq identprereq="2.2.1" />
    <curso identcurso="1">
      <elementos>
          <exemplo possuiexemp="sim" />
          <matcomp possuimatcomp="sim" />
      </elementos>
    </curso>
    <curso identcurso="2">
      <elementos>
          <exemplo possuiexemp="sim" />
          <matcomp possuimatcomp="sim" />
      </elementos>
    </curso>
    <topico numtop="2.2.2.1" desctop="Condicionamento de
        Matrizes" abreviacao="Cond. Mat."
```

```
            arquivoxml="Condicioamento_Matrizes.html"
            palchave="Condicionamento, Matriz">
          <curso identcurso="1"> <elementos /> </curso>
          <curso identcurso="2"> <elementos /> </curso>
        </topico>
      </topico>
      <topico numtop="2.2.3" desctop="Método da Decomposicao LU"
          abreviacao="Decomp. LU" arquivoxml="Sela_LU.html"
          palchave="Decomposicao">
        <prereq identprereq="2" />
        <curso identcurso="1">
          <elementos>
              <matcomp possuimatcomp="sim" />
          </elementos>
        </curso>
        <curso identcurso="2">
          <elementos>
              <matcomp possuimatcomp="sim" />
          </elementos>
        </curso>
      </topico>
      <topico numtop="2.2.4" desctop="Método de Cholesky"
          abreviacao="Metod. Chol." arquivoxml="Sela_Cholesky.html"
          palchave="Cholesky">
        <prereq identprereq="2.2.1" />
        <prereq identprereq="2.2.2" />
        <prereq identprereq="2.2.3" />
        <curso identcurso="1">
          <elementos>
              <matcomp possuimatcomp="sim" />
          </elementos>
        </curso>
        <curso identcurso="2">
          <elementos>
              <matcomp possuimatcomp="sim" />
          </elementos>
        </curso>
      </topico>
    </topico>
    <topico numtop="2.3" desctop="Métodos Iterativos" abreviacao="Metod.
        Iterat." arquivoxml="Metodos_Iterativos.html"
        palchave="Iterativos">
      <prereq identprereq="2" />
      <prereq identprereq="2.1" />
      <prereq identprereq="2.3" />
      <curso identcurso="1"> <elementos /> </curso>
      <curso identcurso="2"> <elementos /> </curso>
      <curso identcurso="3"> <elementos /> </curso>
      <topico numtop="2.3.1" desctop="Método de Jacobi"
          abreviacao="Metod. Jac." arquivoxml="Sela_Jacobi.html"
          palchave="Jacobi">
        <prereq identprereq="2" />
        <curso identcurso="1">
          <elementos>
              <matcomp possuimatcomp="sim" />
```

```
                </elementos>
              </curso>
              <curso identcurso="2">
                <elementos>
                    <matcomp possuimatcomp="sim" />
                </elementos>
              </curso>
            </topico>
            <topico numtop="2.3.2" desctop="Método de Gauss-Seidel"
                abreviacao="Metod. GS" arquivoxml="Sela_GaussSeidel.html"
                palchave="Gauss-Seidel">
              <prereq identprereq="2.2" />
              <prereq identprereq="2.3" />
              <curso identcurso="1">
                <elementos>
                    <matcomp possuimatcomp="sim" />
                </elementos>
              </curso>
              <curso identcurso="2">
                <elementos>
                    <matcomp possuimatcomp="sim" />
                </elementos>
              </curso>
              <curso identcurso="3">
                <elementos>
                    <matcomp possuimatcomp="sim" />
                </elementos>
              </curso>
            </topico>
          </topico>
        </topico>
    </material>
```

Anexo 3 Documento XML dos Conceitos de Contexto e Definição

Documentos XML de descrição dos conceitos denominados Contexto e Objetivos da Computação Algébrica e Numérica e Sistemas de Equações Lineares Algébricas.

Documento XML de descrição do conceito Contexto e Objetivos da Computação Algébrica e Numérica

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Topico.dtd">
<textomaterial disciplina="1" numtop="1">
   <conceito arquivo="Contexto.html">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
   </conceito>
</textomaterial>
```

Documento XML de descrição do conceito Sistemas de Equações Lineares Algébricas

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Topico.dtd">
<textomaterial disciplina="1" numtop="2">
   <conceito arquivo="Sistemas _de_Equacoes_Lineares_Algebrica.html">
      <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
    </conceito>
</textomaterial>
```

Anexo 4 Documento XML do Conceito Introdução

Documento XML de descrição do conceito denominado Introdução.

Documento XML de descrição do conceito Introdução

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Topico.dtd">
<textomaterial disciplina="1" numtop="2.1">
  <conceito arquivo="Introducao.html">
    <conteudo />
      <curso identcurso="1" />
      <curso identcurso="2" />
      <curso identcurso="3" />
  </conceito>
  <exercicio idexerc="1" descexerc="Exercicio Introdutorio 1"
      arqexerc="Ex1facil.html" compexerc="Fácil">
    <conteudo />
      <curso identcurso="1" />
      <curso identcurso="2" />
      <curso identcurso="3" />
  </exercicio>
  <exercicio idexerc="2" descexerc="Exemplo Introdutório 2"
      arqexerc="Ex2Facil.html" compexerc="Fácil">
      <conteudo />
      <curso identcurso="1" />
      <curso identcurso="2" />
      <curso identcurso="3" />
  </exercicio>
  <exercicio idexerc="3" descexerc="Exemplo Introdutório 3"
      arqexerc="Ex3Medio.html" compexerc="Medio">
      <conteudo />
      <curso identcurso="1" />
      <curso identcurso="2" />
      <curso identcurso="3" />
  </exercicio>
  <exercicio idexerc="4" descexerc="Exemplo Introdutório 4"
      arqexerc="Ex4Medio.html" compexerc="Medio">
      <conteudo />
      <curso identcurso="1" />
      <curso identcurso="2" />
      <curso identcurso="3" />
  </exercicio>
  <exercicio idexerc="5" descexerc="Exemplo Introdutório 5"
      arqexerc="Ex5Complexo.html" compexerc="Complexo">
      <conteudo />
      <curso identcurso="1" />
```

```
        <curso identcurso="2" />
        <curso identcurso="3" />
    </exercicio>
    <exercicio idexerc="6" descexerc="Exemplo Introdutório 6"
        arqexerc="Ex6.html" compexerc="Nenhum">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
    </exercicio>
    <exercicio idexerc="7" descexerc="Exemplo Introdutório 7"
        arqexerc="Ex7Facil.html" compexerc="Fácil">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
    </exercicio>
    <exercicio idexerc="8" descexerc="Exemplo Introdutório 8"
        arqexerc="Ex8Facil.html" compexerc="Fácil">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
    </exercicio>
</textomaterial>
```

Anexo 5 Documentos XML dos Conceitos Métodos Diretos e Método de Gauss

Documentos XML de descrição dos conceitos denominados Métodos Diretos e Método de Gauss.

Documento XML de descrição do conceito Métodos Diretos

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Topico.dtd">
<textomaterial disciplina="1" numtop="2.2">
   <conceito arquivo="Metodos_Diretos.html">
       <conteudo />
       <curso identcurso="1" />
       <curso identcurso="2" />
       <curso identcurso="3" />
    </conceito>
 </textomaterial>
```

Documento XML de descrição do conceito Método de Gauss

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Topico.dtd">
<textomaterial disciplina="1" numtop="2.2.1">
   <conceito arquivo="Sela_Gauss.html">
      <conteudo>
          <elementoconteudo arqelem="Sist_Linear28.gif" ambitec="1" />
          <elementoconteudo arqelem="Sist_Line ar29.gif" ambitec="1" />
          <elementoconteudo arqelem="Sist_Linear30.gif" ambitec="1" />
          <elementoconteudo arqelem="Sist_Linear31.gif" ambitec="1" />
          <elementoconteudo arqelem="Sist_Linear32.gif" ambitec="1" />
          <elementoconteudo arqelem="Sist_Line ar34.gif" ambitec="1" />
       </conteudo>
       <curso identcurso="1" />
       <curso identcurso="2" />
       <curso identcurso="3" />
   </conceito>
   <exemplo idexemp="1" descexemp="Exemplo do Método de Gauss"
        arqexemp="Exp_gauss.html" compexemp="Fácil">
      <conteudo>
          <elementoconteudo arqelem="Sist_Linear35.gif" ambitec="1" />
          <elementoconteudo arqelem="Sist_Linear36.gif" ambitec="1" />
          <elementoconteudo arqelem="Sist_Linear37.gif" ambitec="1" />
          <elementoconteudo arqelem="Sist_Linear38.gif" ambitec="1" />
          <elementoconteudo arqelem="Sist_Linear39.gif" ambitec="1" />
```

```
        <elementoconteudo arqelem="Sist_Linear40.gif" ambitec="1" />
    </conteudo>
    <curso identcurso="1" />
    <curso identcurso="2" />
    <curso identcurso="3" />
</exemplo>
<matcomp idmatcomp="1" descmatcomp="Biblioteca do Maple "
    arqmatcomp="Bib_MGauss.html">
    <conteudo>
        <elementoconteudo arqelem="Gausselim01.gif" ambitec="1" />
        <elementoconteudo arqelem="Gausselim02.gif" ambitec="1" />
        <elementoconteudo arqelem="Backsub01.gif" ambitec="1" />
        <elementoconteudo arqelem="Backsub02.gif" ambitec="1" />
    </conteudo>
    <curso identcurso="1" />
    <curso identcurso="2" />
    <curso identcurso="3" />
</matcomp>
<matcomp idmatcomp="2" descmatcomp="Recursos do Maple para programar
    o Método de Gauss" arqmatcomp="Teoria_Gauss.mws ">
    <conteudo />
    <curso identcurso="1" />
    <curso identcurso="2" />
    <curso identcurso="3" />
</matcomp>
<matcomp idmatcomp="3" descmatcomp="Recursos do Maple para programar
    o Método de Gauss" arqmatcomp="Teoria_Gauss.html">
    <conteudo />
    <curso identcurso="1" />
    <curso identcurso="2" />
    <curso identcurso="3" />
</matcomp>
<matcomp idmatcomp="4" descmatcomp="Programas usando Maple "
    arqmatcomp="Elim_Gauss.html">
    <conteudo />
    <curso identcurso="1" />
    <curso identcurso="2" />
    <curso identcurso="3" />
</matcomp>
<matcomp idmatcomp="5" descmatcomp="Programas usando Maple (Para
    download)" arqmatcomp="Elim_Gauss.mws ">
    <conteudo />
    <curso identcurso="1" />
    <curso identcurso="2" />
    <curso identcurso="3" />
```

```
    </matcomp>
    <matcomp idmatcomp="6" descmatcomp="Programas usando Maple (Filme)"
        arqmatcomp="Elim_Gauss.avi">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
    </matcomp>
    <matcomp idmatcomp="7" descmatcomp="Programas usando Maple (Filme
        para download)" arqmatcomp="Elim_Gauss.exe">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
    </matcomp>
    <matcomp idmatcomp="8" descmatcomp="Programas em PHP"
        arqmatcomp="ProgPHP_Gauss.php">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
    </matcomp>
</textomaterial>
```

Anexo 6 Documentos XML dos Conceitos Algoritmo da Triangularização e Retrosubstituição

Documentos XML de descrição dos conceitos denominados Algoritmo da Triangularização e Algoritmo da Retrosubstituição.

Documento XML de descrição do conceito Algortimo da Triangularização

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Topico.dtd">
<textomaterial disciplina="1" numtop="2.2.1.1">
   <conceito arquivo="Sela_Triang.html">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
   </conceito>
</textomaterial>
```

Documento XML de descrição do conceito Algortimo da Retrosubstituição

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Topico.dtd">
<textomaterial disciplina="1" numtop="2.2.1.2">
   <conceito arquivo="SEla_Retro.html">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
   </conceito>
</textomaterial>
```

Anexo 7 Documento XML do Conceito Método de Gauss com Pivotamento

Documento XML de descrição do conceito denominado Método de Gauss com Pivotamento.

Documento XML de descrição do conceito Método de Gauss com Pivotamento

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Topico.dtd">
<textomaterial disciplina="1" numtop="2.2.2">
  <conceito arquivo="Sela_Gaussp.html">
    <conteudo>
      <elementoconteudo arqelem="Sist_Linear01.gif" ambitec="1" />
      <elementoconteudo arqelem="Sist_Linear02.gif" ambitec="1" />
      <elementoconteudo arqelem="Sist_Linear03.gif" ambitec="1" />
      <elementoconteudo arqelem="Sist_Linear04.gif" ambitec="1" />
      <elementoconteudo arqelem="Sist_Linear05.gif" ambitec="1" />
      <elementoconteudo arqelem="Sist_Linear06.gif" ambitec="1" />
      <elementoconteudo arqelem="Sist_Linear07.gif" ambitec="1" />
      <elementoconteudo arqelem="Sist_Linear08.gif" ambitec="1" />
      <elementoconteudo arqelem="Sist_Linear09.gif" ambitec="1" />
      <elementoconteudo arqelem="Sist_Linear10.gif" ambitec="1" />
      <elementoconteudo arqelem="Sist_Linear11.gif" ambitec="1" />
      <elementoconteudo arqelem="Sist_Linear12.gif" ambitec="1" />
    </conteudo>
    <curso identcurso="1" />
    <curso identcurso="2" />
  </conceito>
  <exemplo idexemp="1" descexemp="Exemplo do Método de Gauss"
      arqexemp="Exp_Gauss_Pivot.html" compexemp="Fácil">
    <conteudo />
    <curso identcurso="1" />
    <curso identcurso="2" />
  </exemplo>
  <matcomp idmatcomp="1" descmatcomp="Programas do Maple em html"
      arqmatcomp="Gauss_Pivot.html">
    <conteudo />
    <curso identcurso="1" />
    <curso identcurso="2" />
  </matcomp>
  <matcomp idmatcomp="2" descmatcomp="Programas em Maple para
      download" arqmatcomp="Gauss_Pivot.mws">
    <conteudo />
    <curso identcurso="1" />
```

```
        <curso identcurso="2" />
    </matcomp>
    <matcomp idmatcomp="3" descmatcomp="Programas php para demonstração"
        arqmatcomp="Gauss_pt.html">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
    </matcomp>
    <matcomp idmatcomp="4" descmatcomp="Programas php para demonstração"
        arqmatcomp="Gauss_pt.php">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
    </matcomp>
    <matcomp idmatcomp="5" descmatcomp="Programas php para demonstração"
        arqmatcomp="Gauss2_pt.php">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
    </matcomp>
</textomaterial>
```

Anexo 8 Documentos XML dos Conceitos Método de Condicionamento de Matrizes e da Decomposição LU

Documentos XML de descrição dos conceitos denominados Método de Condicionamento de Matrizes e Método da Decomposição LU.

Documento XML de descrição do conceito Método de Condicionamento de Matrizes

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Topico.dtd">
<textomaterial disciplina="1" numtop="2.2.2.1">
   <conceito arquivo="Condicioamento_Matrizes.html">
      <conteudo />
      <curso identcurso="1" />
      <curso identcurso="2" />
   </conceito>
</textomaterial>
```

Documento XML de descrição do conceito Método da Decomposição LU

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Topico.dtd">
<textomaterial disciplina="1" numtop="2.2.3">
   <conceito arquivo="Sela_LU.html">
      <conteudo>
         <elementoconteudo arqelem="Sist_Linear13.gif" ambitec="" />
         <elementoconteudo arqelem="Sist_Linear14.gif" ambitec="" />
         <elementoconteudo arqelem="Sist_Linear15.gif" ambitec="" />
         <elementoconteudo arqelem="Sist_Linear16.gif" ambitec="" />
         <elementoconteudo arqelem="Sist_Linear17.gif" ambitec="" />
         <elementoconteudo arqelem="Sist_Linear18.gif" ambitec="" />
         <elementoconteudo arqelem="Sist_Linear19.gif" ambitec="" />
      </conteudo>
      <curso identcurso="1" />
      <curso identcurso="2" />
   </conceito>
   <matcomp idmatcomp="1" descmatcomp="Biblioteca do Maple para
      decomposicao LU" arqmatcomp="BibliLudecomp.html">
      <conteudo />
      <curso identcurso="1" />
      <curso identcurso="2" />
   </matcomp>
   <matcomp idmatcomp="2" descmatcomp="Programas em Maple para
      Decomposicao LU" arqmatcomp="Ludecomp.html">
      <conteudo />
```

```
        <curso identcurso="1" />
        <curso identcurso="2" />
    </matcomp>
    <matcomp idmatcomp="3" descmatcomp="Programas em Maple para
        download" arqmatcomp="Ludecomp.mws ">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
    </matcomp>
    <matcomp idmatcomp="4" descmatcomp="Programas em PHP para
        demonstracao" arqmatcomp="Ludecomp_pt.html">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
    </matcomp>
    <matcomp idmatcomp="5" descmatcomp="Programas em PHP para
        demonstracao" arqmatcomp="Ludecomp_p.php">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
    </matcomp>
    <matcomp idmatcomp="6" descmatcomp="Programas em PHP para
        demonstracao" arqmatcomp="Ludecomp2_pt.php">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
    </matcomp>
</textomaterial>
```

Anexo 9 Documento XML do Conceito Método de Cholesky

Documento XML de descrição do conceito denominado Método de Cholesky.

Documento XML de descrição do conceito Método de Cholesky

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Topico.dtd">
<textomaterial disciplina="1" numtop="2.2.4">
   <conceito arquivo="Sela_Cholesky.html">
      <conteudo>
         <elementoconteudo arqelem="Sist_Linear20.gif" ambitec="" />
         <elementoconteudo arqelem="Sist_Linear21.gif" ambitec="" />
         <elementoconteudo arqelem="Sist_Linear22.gif" ambitec="" />
         <elementoconteudo arqelem="Sist_Linear23.gif" ambitec="" />
         <elementoconteudo arqelem="Sist_Linear24.gif" ambitec="" />
         <elementoconteudo arqelem="Sist_Linear25.gif" ambitec="" />
         <elementoconteudo arqelem="Sist_Linear26.gif" ambitec="" />
      </conteudo>
      <curso identcurso="1" />
      <curso identcurso="2" />
   </conceito>
   <matcomp idmatcomp="1" descmatcomp="Biblioteca do Maple - cholesky"
      arqmatcomp="BibMapCholesky.html">
      <conteudo />
      <curso identcurso="1" />
      <curso identcurso="2" />
   </matcomp>
   <matcomp idmatcomp="2" descmatcomp="Programas em Maple html"
      arqmatcomp="ProgMap_Cholesky.html">
      <conteudo />
      <curso identcurso="1" />
      <curso identcurso="2" />
   </matcomp>
   <matcomp idmatcomp="3" descmatcomp="Programas em Maple download"
      arqmatcomp="ProgMap_Cholesky_Down.exe">
      <conteudo />
      <curso identcurso="1" />
      <curso identcurso="2" />
   </matcomp>
   <matcomp idmatcomp="4" descmatcomp="Programas em PHP"
      arqmatcomp="Cholesky_pt.html">
      <conteudo />
      <curso identcurso="1" />
```

```
        <curso identcurso="2" />
    </matcomp>
    <matcomp idmatcomp="5" descmatcomp="Programas em PHP"
        arqmatcomp="Cholesky_pt.php">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
    </matcomp>
    <matcomp idmatcomp="6" descmatcomp="Programas em PHP"
        arqmatcomp="Cholesky2_pt.php">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
    </matcomp>
</textomaterial>
```

Anexo 10 Documentos XML dos Conceitos Método Iterativo e de Jacobi

Documentos XML de descrição dos conceitos denominados Médoto Iterativo e Método de Jacobi

Documento XML de descrição do conceito Método Iterativo

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Topico.dtd">
<textomaterial disciplina="1" numtop="2.3">
   <conceito arquivo="Metodos_Iterativos.html">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
   </conceito>
</textomaterial>
```

Documento XML de descrição do conceito Método de Jacobi

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Topico.dtd">
<textomaterial disciplina="1" numtop="2.3.1">
   <conceito arquivo="Sela_Jacobi.html">
      <conteudo>
         <elementoconteudo arqelem="Sist_Linear27.gif" ambitec="1" />
        </conteudo>
        <curso identcurso="1" />
        <curso identcurso="2" />
   </conceito>
   <matcomp idmatcomp="1" descmatcomp="Programas em PHP - Jacobi"
        arqmatcomp="Jacobi_pt.html">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
   </matcomp>
   <matcomp idmatcomp="2" descmatcomp="Programas em PHP - Jacobi"
        arqmatcomp="Jacobi_pt.php">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
   </matcomp>
   <matcomp idmatcomp="3" descmatcomp="Programas em PHP - Jacobi"
        arqmatcomp="Jacobi2_pt.php">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
   </matcomp>
 </textomaterial>
```

Anexo 11 Documento XML do Conceito Método de Gauss-Seidel

Documento XML de descrição do conceito denominado Médoto de Gauss-Seidel

Documento XML de descrição do conceito Método de Gauss-Seidel

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE textomaterial SYSTEM "dtd\Elementos_Topico.dtd">
<textomaterial disciplina="1" numtop="2.3.2">
   <conceito arquivo="Sela_GaussSeidel.html">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
   </conceito>
   <matcomp idmatcomp="1" descmatcomp="Programas em PHP - Gauss-Seidel"
        arqmatcomp="Gauss-Seidel_pt.html">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
   </matcomp>
   <matcomp idmatcomp="2" descmatcomp="Programas em PHP - Gauss-Seidel"
        arqmatcomp="Gauss-Seidel_pt.php">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
   </matcomp>
   <matcomp idmatcomp="3" descmatcomp="Programas em PHP - Gauss-Seidel"
        arqmatcomp="Gauss-Seidel2_pt.php">
        <conteudo />
        <curso identcurso="1" />
        <curso identcurso="2" />
        <curso identcurso="3" />
   </matcomp>
</textomaterial>
```

Referências

[ABI99] ABITEBOUL, Serge. On views and XML. **ACM SIGMOD Digital Symposium Collection**, [S.l.], v. 2, n. 1, 2000. Trabalho apresentado no ACM SIGACT – SIGMOD – SIGART Symposium on Principles of database systems, PODS, 8., 1999.

[AMA2002] AMARAL, Marília A.; FREITAS, Veronice de; MARÇAL, Viviane P.; GASPARINI, Isabela; PROENÇA Jr., Mario Lemes ; BRUNETTO, Maria Angélica C.; PIMENTA, Marcelo S.; RIBEIRO, Cora H. F. Pinto; LIMA, José Valdeni de, OLIVEIRA, José M de. AdaptWeb: an Adaptive Web-based Courseware**.** In: INTERNATIONAL CONFERENCE OF INFORMATION AND COMMUNICATION TECHNOLOGIES IN EDUCATION, 2002. **Proceedings...** [S.l.: s.n.], 2002.v.1, p. 131-134.

[BON2000] BONGIGLI, Maria Elena; CASADEI, Giorgio; SALOMONI, Paola. **Adaptive Intelligent Hypermedia using XML.** New York: ACM Special Interest Group on Applied Computing Publisher, 2000. Disponível em: <http://www.acm.org/conferences/sac/sac00/Proceed/FinalPapers/WW-06/>. Acesso em: 10 maio 2002.

[BOS98] BOSWORTH, A. **A proposal for na XSL Query Language. Position Paper for the W3C query language Workshop.** 1998. Disponível em: <http://www.w3.org/tands/ql/ql98/pp/microsoft.html>. Acesso em: 07 ago. 2001.

[BRA2002] DE BRA, Paul. **AHA! meets AHAM.** Disponível em: <wwwis.win.tue.nl/~debra/ah2002.pdf>. Acesso em: 15 jul. 2002.

[BRA2002a] DE BRA, Paul. **AHA! Adaptive Hypermedia for All.** Disponível em: <wwwis.win.tue.nl/~debra/sane2002.pdf>. Acesso em: 15 jul. 2002.

[BRA2001] DE BRA, Paul; RUITER, J.P. **AHA! Adaptive Hypermedia for All.** In: WEBNET CONFERENCE, 2001. **Proceedings…** [S.l.:s.n.], 2001. p. 262-268.

[BRA98] DE BRA, Paul; CALVI, L. AHA: a Generic Adaptive Hypermedia System. In: WORKSHOP ON ADAPTIVE HYPERTEXT AND HYPERMEDIA, 1998. **Proceedings …** Disponível em: <http://wwwis.win.tue.nl/ah98/DeBra .html>. Acesso em: 05 abr. 2002.

[BRU96] BRUSILOVSKY, Peter. Methods and techniques of adaptive hypermedia. **User Modeling and User Adapted Interaction**, [S.l], v.6, n. 2-3, p. 87-129, 1996. Special Issue on adaptive hypertext and hypermedia.

[BRU96a] BRUSILOVSKY, Peter. **Adaptive Hypermedia:** an Attempt to Analyze and Generalize. 1996. Disponível em: <http://www2.sis.pitt.edu/~peterb/papers/MHVR96.pdf>. Acesso em: 27 jul. 2001.

[BRU98] BRUSILOVSKY, Peter. **Web-based education for all:** a tool for development adaptive courseware. 1998. Disponível em: <http://www7.scu.edu.au/programme/fullpapers/1893/com1893.htm>. Acesso em: 27 jul. 2001.

[BRU99] BRUSILOVSKY, Peter. Adaptive Hypermedia: from Systems to Framework. **ACM Computing Surveys**, New York, v. 31, n. 4, Dec. 1999.

[BRU99a] BRUSILOVSKY, Peter. **Adaptive and Intelligent Technologies for Web-based Education**. In: *Künstliche Intelligenz*, Special Issue on Intelligent System and Teleteaching, 19-25 /4, 1999. Disponível em: <http://www.contrib.andrew.cmu.edu/~plb/papers/KI-review.html>. Acesso em: 05 out. 2001.

[BRU2001] BRUSILOVSKY, Peter. Adaptive Educational Hypermedia. In: INTERNATIONAL PEG CONFERENCE, PEG, 10. , 2001, Tampere, Finland. **Proceedings…** Disponível em: <http://www2.sis.pitt.edu /~peterb/papers/PEG01.html>. Acesso em: 09 jun. 2002.

[BRY97] BRYAN, Martin. **An Introduction to the Extensible Markup Language.** The SGML Centre, 1997. Disponível em: <http://www.personal.u-net.com/~sgml/xmlintro.htm>. Acesso em: 28 jul. 2001.

[CAM96] CAMPOS, Fernanda; CAMPOS, Gilda; ROCHA, Ana Regina. Dez etapas para o desenvolvimento de software educacional do tipo hipermídia. In: CONGRESSO IBERO-AMERICANO DE INFORMÁTICA EDUCATIVA, 1996, Brasília. **Actas...** Disponível em: <http:// upf.tche.br/~carolina/ pos/etapas.html>. Acesso em: 29 out. 2002.

[DOM98] DOCUMENT OBJECT MODEL (DOM). **Level 1 Specification Version 1.0.** October 1, 1998. Disponível em: <http://www.w3.org/TR/ 1998/REC-DOM-Level-1-19981001/>. Acesso em: 09 jun. 2002.

[FRE2003] FREITAS, Veronice. **Autoria Adaptativa de Hipermídia Educacional.** 2003. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[GAS2003] GASPARINI, Isabela. **Interface Adaptativa no ambiente AdaptWeb.** 2003. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[GIR99] GIRAFFA,L.M.M. **Uma Arquitetura de tutor utilizando estados mentais.** 1999. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[ISO86] ISO. **ISO 8879:** Information Processing – Text and Office Systems – Standard Generalized Markyp Language (SGML). [S.l.], 1986.

[HEN2000] HENZE, Nicola. **Adaptive Hyperbooks:** Adaptation for Project-Based Learning Resources. 2000. Disponível em: <www.kbs.uni-hannover.de /~henze/diss.pdf>. Acesso em: 03 jul. 2002.

[KOB2001] KOBSA, Alfred. **Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships.** 2001. Disponível em: <http://citeseer.nj.nec.com/kobsa01 personalized.html>. Acesso em: 08 jun. 2002.

[LAN98] LANDER, Richard. **Introduction to XML.** 1998. Disponível em: <http://pdbeam.uwaterloo.ca/~lander/xml/intro_xml.html>. Acesso em: 10 ago. 2001.

[LIG99] LIGHT, Richard. **Iniciando em XML.** São Paulo: Makron Books,1999.

[MAC98] MACE, Scott et al. What's Wrong with HTML. In: Weaving a Better Web, March 1998. **Proceedings...** Disponível em: <http://www.byte.com/art/9803 /sec5/art3.htm>. Acesso em: 15 jun. 2002.

[MAR2003] MARÇAL, Viviane. **Adaptação de Conteúdo baseada no modelo do aluno em um Ambiente de Ensino Adaptativo**. 2003. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[MUR98] MURRAY, T.; CONDIT, C.; HAUGSJAA, E. MetaLinks: A preliminary framework for concept-based adaptive hypermedia. In: INTERNATIONAL CONFERENCE ON INTELLIGENT TUTORING SYSTEMS, ITS, 4., 1998. San Antonio, TX. **Proceedings...** Berlin: Springer-Verlag, 1998. Disponível em: <http://www.aml.cs.umass.edu/~stern/webits/itsworkshop/murray.html>. Acessado em: 27 jun. 2002.

[PHP2001] TUTORIAL da Linguagem PHP. Disponível em: <http://phpbrasil.com/tutoriais/tutorial.php?id=4>. Acesso em: 07 ago. 2001.

[PIM2000] PIMENTEL, Maria da Graça Campos; TEIXEIRA,César Augusto Camillo; SANTACHÈ, André. **XML**: Explorando suas aplicações na WEB. 2000. Disponível em: <http://www.icmc.sc.usp.br/~mgp/jai2000>. Acesso em: 04 dez.2001.

[PIM2001] PIMENTEL, Maria da Graça C; TEIXEIRA, César Augusto Camilo. A Prática XML em Aplicações WEB. In: SBMIDIA, 2001. **Anais...** Florianópolis: UFSC, 2001.

[ROY2001] ROY, Jaideep; RAMANUJAN, Anupama. XML Schema Language Taking XML to the Next Level. **Proceeding of the IEEE**, New York, v.1, p. 37-40, Apr. 2001.

[SEL2001] SELIGMAN, Len. XML's Impact on Databases and Data Sharing. **Proceeding of the IEEE**, New York, v. 34, n. 6, p. 59-67, June 2001.

[TEI2001] TEIXEIRA, Eduardo C.; VIEIRA, Marina T.P. (2001) **XML:** Criação de Documentos e Integração com Banco de Dados. São Carlos: UFSCar, 2001. Relatório Técnico.

[W3C2000] EXTENSIBLE Markup Language - XML. W3C Recommendation. 06 October 2000. Disponível em: <http://www.w3.org/TR/REC-mls/>. Acesso em: 09 abr. 2002.

[W3C2001] XML Schema Part 0: Primer. W3C Recommendation. 2 May 2001. Disponível em: <http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/>. Acesso em: 25 maio 2002.

[WU2001] WU, H.; DE BRA, P. Sufficient Conditions for Well-Behaved Adaptive Hypermedia Systems. In: WEB INTELLIGENCE: RESEARCH AND DEVELOPMENT, ASIA-PACIFIC CONFERENCE, WI, 1., 2001, Maebashi City, Japan. **Proceedings...** Berlin: Springer-Verlag, 2001. (Lectures Notes in Computer Science, 2198).