CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

BRUNO SIQUEIRA CAMPOS MENDONÇA VILAR

Estudo sobre a utilização de ontologia na definição do modelo de domínio de um sistema hipermídia adaptativo

Palmas
2005

CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

BRUNO SIQUEIRA CAMPOS MENDONÇA VILAR

Estudo sobre a utilização de ontologia na definição do modelo de domínio de um sistema hipermídia adaptativo

"Relatório apresentado como requisito parcial da disciplina de Estágio Supervisionado em Sistemas de Informação do curso de Sistemas de Informação, orientado pela Profª. Parcilene Fernandes de Brito"

Palmas
2005

BRUNO SIQUEIRA CAMPOS MENDONÇA VILAR

Estudo sobre a utilização de ontologia na definição do modelo de domínio de um sistema hipermídia adaptativo

"Relatório apresentado como requisito parcial da disciplina de Estágio Supervisionado em Sistemas de Informação do curso de Sistemas de Informação, orientado pelo Profª. Parcilene Fernandes de Brito"

BANCA EXAMINADORA

_____

Profª. Parcilene Fernandes de Brito
Centro Universitário Luterano de Palmas

_____

Prof. Fabiano Fagundes
Centro Universitário Luterano de Palmas

_____

Profª. Thereza Patrícia Pereira Padilha
Centro Universitário Luterano de Palmas

Palmas
2005

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

AJAX = *Asynchronous JavaScript and XML*

API = *Application Programmer Interface*

CSV = *Comma Separeted Values*

DAML = *DARPA Agent Markup Language*

HA = *Hipermídia Adaptativa*

IDE = *Integrated Development Environment*

HP = *Hewlett-Packard Compan*

HTML = *Hyper Text Markup Language*

JSP = *Java Server Pages*

JSTL = *JSP Standard Tag Library*

OIL = *Ontology Inference Layer*

OWL = *Web Ontology Language*

RDF = *Resource Description Framework*

RDQL = *RDF Data Query Language*

SHA = *Sistemas de Hipermídia Adaptativa*

TAD = Tipo Abstrato de Dados

UML = *Unified Modeling Language*

URI = *Uniform Resouce Identifier*

URL = *Universal Resource Locator*

W3C = *World Wide Web Consortium*

XML = *eXtensible Markup Language*

RESUMO

As ontologias permitem a definição de domínios complexos, pois possuem características que visam desde a interoperabilidade, a contextualização de informação e a criação de universos inteligíveis, até a definição de axiomas. Tais características estão compreendidas entre os fatores relevantes na definição de domínios utilizados em Sistemas Adaptativos, que têm como propósito fornecer formas de adaptação da interface a determinados contextos, como fator de melhora da eficiência deste ambiente aos seus usuários. Assim, neste trabalho é realizado um estudo acerca das técnicas e referenciais teóricos envolvidos nos conceitos de Ontologias e Sistemas Hipermídia Adaptativos. Esta teoria é empregada para a utilização de recursos destinados à manipulação de ontologias, definidas pela linguagem OWL, como forma de verificar os recursos pertinentes a construção de sistemas que se adaptam aos seus usuários. O resultado desta aplicação é o desenvolvimento de um protótipo no qual o domíno de seu conteúdo é definido por uma ontologia, que representa parte do universo "ensino".

Palavras-chave: Hipermídia Adaptativa, ontologia e OWL

ABSTRACT

Ontologies allow the definition of complex domains, because they have characteristics as the interoperability, the contextualization of information and the creation of intelligible universes, until the definition of axioms. Such characteristics are understood between the relevant factors in the definition of domains used in Adaptive Systems, which have as intention to supply the forms of adaptation of the interface to determined contexts, as factor of improvement of the efficiency of this environment to its users. Thus, in this work it is realized a study concerning the techniques and theoretical referenciais in the concepts of ontologies and Hypermedia Adaptive Systems. This theory is used for the use of resources destined to the manipulation of ontologies, defined for OWL language, as form to verify the pertinent resources in the construction of systems that atapts to its users. The result of this application is the development of an prototype in which the domain of its content is defined by a ontolgy, that represents part of the universe "education".

Keywords: Adaptive Hypermedia, ontology and OWL

1 INTRODUÇÃO

No desenvolvimento de sistemas web para o ensino, a adaptação do ambiente a determinados contextos, a organização dos dados e a definição de caminhos para o usuário são fatores determinantes para o alcance da eficácia. Assim, tem-se no estudo de sistemas adaptativos hipermídia um fator relevante para a concretização de tal objetivo.

Existem várias técnicas que buscam fornecer a característica da adaptabilidade a sistemas hipermídia e um dos pontos básicos para a concretização de tal fato está numa coerente definição do domínio. A afirmação pode ser justificada pelo fato de que é preciso compreender a relevância e o papel de cada elemento dentro de um domínio para que seu emprego possa ser efetuado corretamente. Assim, quanto mais capacidade de extrair informações e fazer novas deduções o próprio domínio tiver, menos complexa será a navegação do usuário.

As ontologias permitem a definição de domínios complexos, pois possuem características que visam desde a interoperabilidade, a contextualização de informação e a criação de universos inteligíveis, até a definição de axiomas. Por universo inteligível é conferida idéia de que há, em um determinado domínio, a possibilidade de compreender seus elementos e, a partir disso, extrair informações. Isto é possível pela capacidade que as ontologias têm de representar as diferentes relações existentes entre os elementos que participam do domínio, além de permitir detalhar cada elemento através de suas características e atribuições.

Adiciona-se ao conjunto de fatores que atribuem às ontologias a capacidade de definir domínios um grande número de linguagens que foram desenvolvidas para tornar possível a definição de universos. As diferentes linguagens compreendem variações em sua composição que resultam em formas de representação com maior ou menor quantidade de recursos. Assim, tem-se nas ontologias um recurso propício para o desenvolvimento de

sistemas que promovem sua adaptação ao universo de utilização.

Dentro desse contexto, compreende-se que para criar sistemas hipermídia adaptativos na área do ensino é necessário tanto um entendimento da teoria que sustenta tais sistemas, como das técnicas que podem ser usadas para concretizar sua implementação. Desse modo, constitui um dos objetivos deste trabalho realizar o estudo de sistemas hipermídia adaptativos e de sua aplicação ao ensino, com o propósito de conhecer as atuais metodologias empregadas em sua construção. Acrescenta-se, também, o estudo de ontologias e a compreensão de suas características e aplicações. Esta tarefa é realizada com o propósito de empregar ontologias para melhorar a capacidade de adaptação de sistemas.

Após estabelecer a fundamentação teórica necessária ao trabalho com sistemas hipermídia adaptativos e ontologias, será descrito, no decorrer deste trabalho, o processo de aplicação do conhecimento na elaboração de um protótipo. Este protótipo deverá permitir a organização das informações através da representação de um domínio em OWL (*Ontology Web Language*), que consiste numa linguagem desenvolvida com o propósito de permitir o compartilhamento de informações que devem ser processadas por aplicações, e que não devem ser limitadas à compreensão por pessoas (MCGUINNESS e HARMELEN 2004). O intuito da realização do protótipo é manter um domínio descrito através de uma ontologia, que poderá ser consultada e gerenciada através de um *site*.

O trabalho foi estruturado através das seções: Revisão de Literatura, Materiais e Métodos, Resultados e Discussão e Conclusão. Na Revisão de Literatura são trabalhados os conceitos de Hipermídia Adaptativa (Seção 2.1), sua aplicação ao ensino (Subseção 2.1.1) e, então, na Seção 2.2, são relacionadas as definições de ontologias e suas principais aplicações. Em Materiais e Métodos são descritas as ferramentas necessárias ao desenvolvimento do projeto, bem como as tecnologias empregadas. Na seção de Resultados e Discussão são relatados os passos para a construção do protótipo responsável pelo gerenciamento de ontologias sobre conteúdo educacionais. Finalmente, na Conclusão é realizada uma análise das dificuldades encontradas e dos benefícios obtidos com a definição de um protótipo de sistema que utiliza ontologias para a estruturação de seu conteúdo.

2 REVISÃO DE LITERATURA

Para que os objetivos do trabalho pudessem ser alcançados, foram realizados estudos que envolveram a área de Hipermídia Adaptativa, uma área específica de aplicação, voltada para o Ensino, e os conceitos e aplicações de ontologias. Os resultados destes estudos são apresentados nas seções subseqüentes, apresentadas como a revisão de literatura.

2.1 Hipermídia Adaptativa

Um Sistema de Hipermídia Adaptativa (SHA) consiste em um sistema de hipertexto ou hipermídia que reflete algumas das características do usuário no modelo de usuário (BRUSILOVSKY 2004). O modelo de usuário é aplicado, complementa o autor, para que os vários aspectos visíveis do sistema sejam direcionados ao seu perfil.

A compreensão das necessidades ou características do usuário, para que a interface possa dirigir sua apresentação ao seu melhor aproveitamento, contribui, sob diferentes aspectos, para uma maior eficiência dos sistemas hipermídia. Estes aspectos podem significar uma aparência que melhor agrade o usuário, a apresentação de um conteúdo que melhor se adeque aos seus conhecimentos, uma velocidade maior na realização das tarefas, dentre outros.

Ao estabelecer uma relação entre a definição de SHA e parte dos benefícios obtidos por sua utilização, é possível obter uma caracterização dos objetivos do modelo de hipermídia adaptativa. Na concepção de Palazzo (2000):

> O objetivo geral dos sistemas e modelos de HA é portanto prover seus usuários com informação atualizada, subjetivamente interessante, com a ilustração multimídia pertinente, num tamanho e profundidade adequados ao contexto e em correspondência direta com o modelo do usuário.

Por oferecimento de tamanho e profundidade adequados ao contexto é possível entender que há uma preocupação em trabalhar corretamente a apresentação do conteúdo de acordo com o conhecimento e a destreza que cada um dos indivídos, que utlizam o sistema hipermídia em questão, possui. Esta questão é posta em evidência ao tratar de SHA aplicados ao Ensino, nos quais mantém-se a preocupação com o conhecimento que cada aluno possui e, de acordo com isto, a especificidade, a abrangência, e a ordem com que os conteúdos devem ser ser apresentados, para que a base de seu conhecimento seja corretamente trabalhada. O conceito de SHA direcionado ao Ensino é trabalhado na Seção 2.1.1.

A habilidade de conhecer o usuário, determinar o conteúdo referente a um contexto e direcionar a apresentação de modo que a interface gerada seja destinada ao perfil de quem a utilize, requer uma estrutura que suporte todas essas características. WU et al. (2001) relatam os componentes de uma arquitetura de um sistema de hipermídia adaptativa com finalidade não específica, composta por:

- **Modelo de Domínio**: mantém as informações sobre o conteúdo do domínio, como o conteúdo em si, sua representação, a estrutura etc. Este modelo serve como base para que as informações do modelo do usuário possam ser confrontadas e, dessa forma, a adaptação seja realizada;
- **Modelo do Usuário**: permite que o perfil de cada usuário do sistema seja traçado, através do armazenamento de suas preferências, conhecimento, objetivos, histórico de navegação e outros aspectos que sejam pertinentes ao contexto e auxiliem no direcionamento da apresentação;
- **Modelo de Adaptação**: o sistema precisa ser capaz de desenvolver o processo de adaptação do conteúdo e da estrutura de *links* baseado nos modelos de domínio e do usuário. Este processo deve ser realizado por intermédio de um conjunto de regras de adaptação, que além de guiar o processo de geração da apresentação, mantém informações sobre como atualizar o modelo do usuário.

Com a caracterização dos principais componentes envolvidos na construção de SHA, é possível reconhecer a relevância de cada componente indivualmente, bem como sua função dentro do conjunto. Entretanto, a compreensão das etapas de utilização de cada componente e a ordem de sucessão dos eventos envolvidos precisa ser estudada para que um SHA possa ser delineado e outros sistemas possam ser reconhecidos como um SHA.

Assim como a arquitetura básica de um SHA podem ser decomposta em três

componentes, o processo de geração de uma interface adaptativa pode ser visto sob três diferentes aspectos, que incluem a coleta de informações sobre os usuários, a elaboração de um modelo de usuário e a aplicação de ambos para que seja gerada a interface de forma adaptativa. A noção destes três aspectos, vistos na forma de três processos básicos, é mostrado por Brusilovsky (1996) como um modelo clássico de iteração desenvolvido em SHA. Para Palazzo (2002), estes três processos, também constituídos como um modelo de iteração, são citados como requisitos, ou critérios a serem satisfeitos, para que um sistema seja caracterizado como adaptativo. Adiante, na Figura 1, é possível acompanhar um modelo de iteração em que são empregados os três processos.

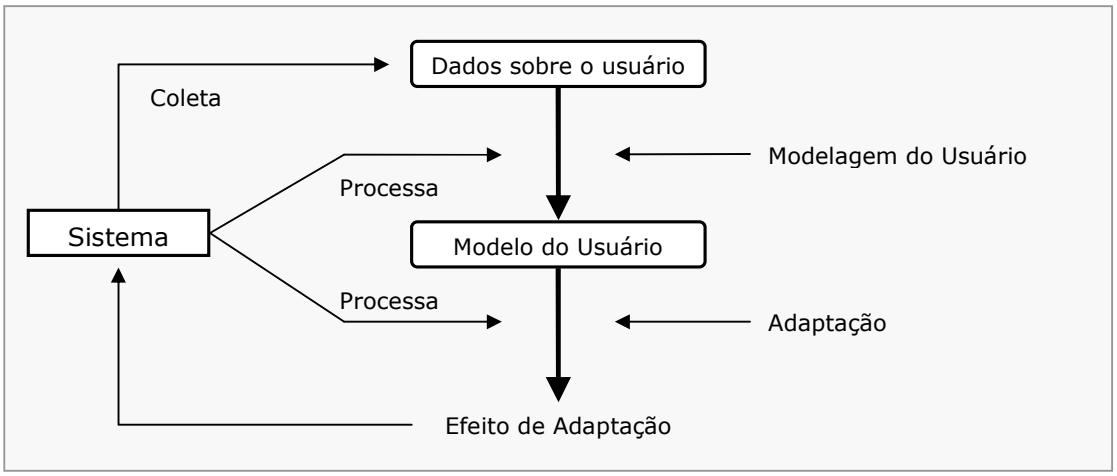

Figura 1. Modelo clássico de iteração "Modelagem do Usuário – Adaptação" em Sistemas Adaptativos (BRUSILOVSKY 1996).

Um dos modelos clássicos utilizados no desenvolvimento de sistemas adaptativos pode ser visto sob a concepção de três processos (Figura 1) (BRUSILOVSKY 1996): coleta de dados provenientes das interações com o usuário; processamento dos dados coletados de modo que o modelo do usuário seja construído ou atualizado; processamento das informações para que a interface seja gerada de acordo com o modelo do usuário.

A coleta das informações sobre o usuário pode ocorrer sob diferentes circunstâncias. Wu (2002) cita que a principal fonte de informações de um SHA é o acompanhamento, ou a monitoração, da navegação do usuário. Em seu trabalho, no entanto, é explicado que estas informações podem não ser suficientes para que todas as informações sejam recolhidas e que, para isto, pode ser necessário recolher informações diretamente, através de fomulários.

Em estágios iniciais de interação com o usuário, é possível observar todas as preferências definidas sobre a interface ou aplicar questionários que argumentem explicitamente as opções, preferências, ou que busquem a profundidade do conhecimento

existente. Em estágios posteriores de utilização do sistema, podem ser mantidos os procedimentos de coleta de informações pertinentes às adaptações que se pretende ter, como as obtidas diretamente pela definição do usuário, acompanhar o desenvolvimento e o percurso realizados como forma de mensurar o conhecimento adquirido (percebido pelo que é apresentado), dentre outras formas. Em geral estas formas de acompanhamento podem ser realizadas por meio de eventos.

Recolher informações sobre o usuário através dos eventos implica na observação das diferentes interações que podem ser realizadas por meio de Arquiteturas de Diálogo. Isto pode ser percebido ao verificar quais *links* são utilizados, os componentes da interface que sofrem algum tipo de interação com o usuário, como, por exemplo, a seleção de um *radio button* ou as ações de clicar, arrastar e soltar determinados elementos de uma página.

Ainda que tenham sido definidos os componentes e processos necessários aos sistemas adaptativos, encontrados nos trabalhos de Brusilovsky (1996, 2004), Palazzo (2000, 2002), Wu et al. (2001) e Wu (2002); são encontradas exigências a serem satisfeitas para que um sistema possa atuar de forma eficaz. Para Cannataro et al. (2001), a modelagem de conteúdos adaptáveis ao usuário e a apresentação de modo eficiente exigem uma construção que siga uma abordagem modularizada e escalável e, para tanto, é preciso:

- criar domínio de aplicação e modelos de adaptação que permitam encontrar os os elementos descritos de forma fácil;
- a modelagem do usuário precisa lidar com situações que vão além do acompanhamento das ações do usuário e da aplicação de questionários. Dentre outras preocupações que podem auxiliar a compreender melhor o perfil do usuário estão: a localização do usuário (país, sua região, etc.), que leva a alterações de cultura, comportamento, etc.; o equipamento utilizado para ter acesso ao sistema (desempenho, suporte a elementos multimídia, qualidade da imagem, etc.); questões relacionadas a velocidade de conexão, qualidade de serviço;
- a arquitetura de SHA precisa suportar o processo de adaptação de forma fácil e eficiente, além de ser flexível o suficiente para lidar com diferentes fontes que determinam as regras de adaptação.

Por fonte para o processo de adaptação pode-se entender a representação das regras, retrições e relacionamentos que serão aplicados aos conceitos para que os modelos do usuário e do domínio sejam corretamente processados e a interface possa ser gerada. Sua flexilidade pode ser vista pelo quão facilmente podem ser adicionadas novas regras,

restrições e relacionamentos, que pode ocorrer diretamente sobre sua fonte de descrição ou pela referência das novas fontes, como bases de dados, bases de conhecimento, documentos, etc.

O uso das técnicas trabalhadas tem como propósito delinear de forma mais completa os perfis dos usuários e realizar inferências para que o conteúdo seja adequadamente apresentado. Como resultado da aplicação destas técnicas são obtidos resultados que são empregados através da interface. A interface do usuário, que remete à parte específica dos sistemas adaptativos que é responsável pela apresentação, "é um artefato de software que melhora sua habilidade de interagir com o usuário através da construção de um modelo de usuário, o qual é baseado nas experiências obtidas com o mesmo" (LANGLEY 1999).

A forma pela qual as interfaces provêem as informações aos usuários pode variar de acordo com o modo com que são trabalhados os conteúdos e são definidos os três modelos (domínio, usuário e adaptação). Dentro da área de estudo de SHA é possível reconhecer uma abordagem comum do uso de técnicas de adaptação em que são mantidos dois grupos principais de suporte à adaptação. Para Wu et al. (2001), SHA desenvolvem a personalização de duas formas: suporte à navegação adaptativa e à apresentação adaptativa. Através da Figura 2, é possível observar a taxonomia dos diferentes métodos de adaptação.

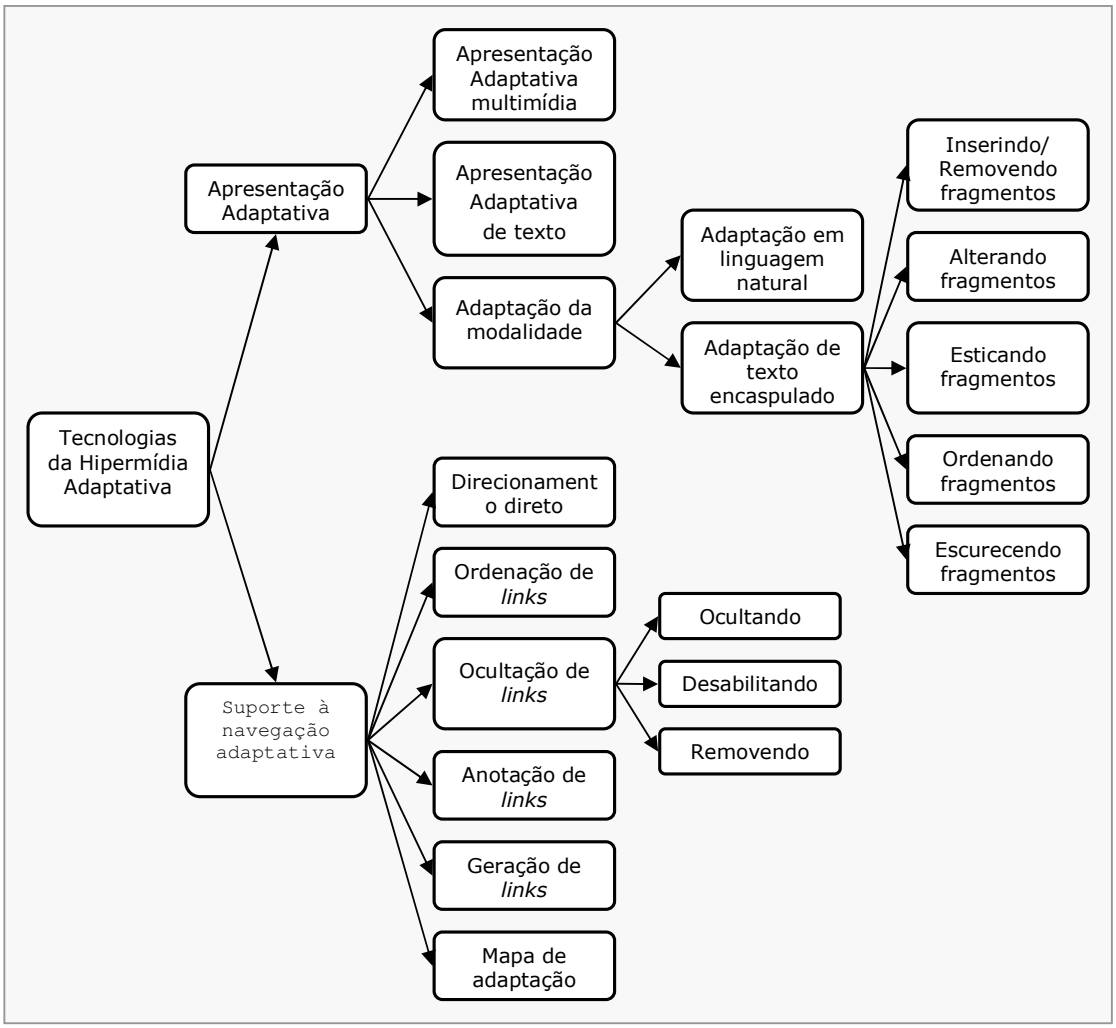

Figura 2. Taxonomia das tecnologias da HA (BRUSLOVISKY 2001)

Conforme apresentado em (BRUSILOVISKY 2001), dois grupos caracterizam a adaptação. Um dos grupos constitui a apresentação adaptativa, que altera o conteúdo dos textos apresentados de acordo com o perfil de cada usuário. Neste processo, trechos dos textos podem sofrer variações como seu ocultamento, a evidenciação, ou a modificação de seu conteúdo.

O segundo grupo apresenta a idéia de que a forma de acesso ao conteúdo é que é alterada. Isto implica na modificação da estrutura de *links*, que pode permitir ocultar ou apresentar uma sequência específica de *links*, dentre outras possibilidades, para dar ao usuário as opções de caminhos que lhes são apropriadas.

Em oposição à adoção de somente uma das formas citadas por Wu et al. (2001), e mesmo em outros trabalhos, como em Brusilovsky (1996), e no desenvolvimento do sistema educacional ALICE, de Kavčič et al. (2002), Calvi e Cristea (2002) relacionam os problemas envolvidos com essa abordagem. As autoras citam que, do ponto de vista do

responsável por desenvolver o conteúdo a ser apresentado de forma adaptativa, torna-se difícil utlizar exclusivamente uma estratégia de adaptação por *link* ou por conteúdo, já que para isto ambos precisam estar intrinsicamente relacionados.

O problema pode ser percebido ao verificar o procedimento empregado na elaboração de um material para ser apresentado através de um Sistema de Hipermídia Adaptativa. Ao adotar a navegação adaptativa como estratégia de adaptação é preciso que cada material acessado por uma URL (*Universal Resource Locator*) esteja claramente definido. Para que o material possa ser corretamente relacionado aos usuários de acordo com seu nível de conhecimento, é preciso que diferentes recursos sejam criados para segmentar o conteúdo de tal forma que existam *links* adequados a cada perfil de usuário, sem que algum conteúdo deixe de ser exibido quando necessário ou que conceitos inapropriados sejam exibidos por causa de outro conceito existente na mesma página.

Caso a estratégia de adaptação adotada consista na apresentação adaptativa, a variação dos endereços disponíveis aos usuários não dependerá do conteúdo, tornando o caminho a ser seguido mais conciso, porém, menos flexível. A diferença ocorre na quantidade de texto apresentado, em sua metodologia de apresentação ou nas mídias utilizadas, que podem enriquecer os detalhes ou variar a abstração do conteúdo.

2.1.1 Hipermídia Adaptativa aplicada ao Ensino

O desenvolvimento de Sistemas Educacionais de Hipermídia Adaptativa possui como objetivo (CRISTEA 2005):

> (...) melhorar ambientes educacionais online através do processo de personalização: o estudante precisa apenas receber o material que é apropriado a ele, de acordo com diferentes critérios como conhecimento, idade, sexo, perferências de aprendizado, estilos, estilos cognitivos etc.

Os recursos e estudos voltados para a HA são aproveitados para melhorar o processo de aprendizagem dos alunos através dos sistemas, sendo que para isto são criados critérios para o desenvolvimento do conteúdo. Para Dourado (2004) o papel dos sistemas de HA aplicados ao Ensino é permitir que os alunos realizem a consulta pelo conteúdo de um curso de acordo com suas metas, preferências e conhecimento, sem que seja necessária a pré-definição deste caminho por um professor.

Nesta aplicação de SHA o modelo do domínio deve, sobretudo, representar de forma estruturada o conteúdo que poderá ser passado para os alunos, para que seja possível, através das informações coletadas, conseguir reconhecer corretamente o que deve ser apresentado. Sob este aspecto, quanto mais minuciosa for a constituição do domínio,

apoiada por uma estruturação, formalismo e representação adequados, melhor poderão ser aplicados métodos de inferência para gerar uma seqüência de estudos eficiente, que, para o domínio de aplicação, incluem o caráter didático e a adequação do conteúdo ao aluno.

O modelo do usuário, de acordo com os diferentes trabalhos pesquisados que abrangem a área de SHA aplicados ao Ensino, dentre eles Dourado (2004), Langley (1999) e Brusilovsky (1996), passa a ser denominado como modelo do estudante, aprendiz ou aluno. Entre seus principais objetivos está a caracterização do conhecimento apresentado pelo aluno, devendo, para isto, manter-se constantemente atualizado sobre aquilo que é aprendido, por cada aluno, durante a utilização do sistema.

2.2 Ontologia

Dentro da área de Inteligência Artificial, uma ontologia pode ser vista como (GRUBBER 1999):

> (...) uma hierarquia estruturada de um conjunto de termos para descrever um domínio que possa ser usado como estrutura de uma base de conhecimento. Ela fornece os meios para descrever explicitamente a conceitualização do conhecimento representado em uma base de conhecimento.

Para Freitas apud Carneiro e Brito (2005),

> (...) uma ontologia não pode ser tratada apenas como uma hierarquia de conceitos, mas também como um conjunto de relações, restrições, axiomas, instâncias e vocabulário.

O principal papel das ontologias é, portanto, fornecer uma forma de representação de um domínio para que o conhecimento representado não se restrinja a quem a criou, além de permitir que o conteúdo mantenha-se reutilizável, permitindo aplicações de diferentes propósitos. Contudo, é preciso obter a noção da forma pela qual ontologias são compostas, como forma de compreender de que maneira obter a representação de um domínio.

Park e Hunting (2003) cita que uma ontologia inclui os seguintes componentes: Entidades e suas Propriedades, além de Funções, Processos, Restrições e Regras, que atuam sobre estas entidades e que compõem a representação do domínio. O autor completa que, em uma ontologia mais complexa, podem ocorrer também elementos denominados Axiomas. Adiante, os itens supracitados que estão compreendidos neste trabalho são melhor explicados:

- **Entidades ("coisas")**: são a representação das coisas que existem em um mundo, como pessoas, cursos, conceitos etc;

- **Relacionamentos entre estas entidades**: representam as diferentes formas de ligação que existem entre os objetos no mundo representado. Como formas de ligação, podem ser citadas: a orientação de um aluno por um professor, a relação de posse de um livro com suas folhas, a abordagem de uma disciplina sobre um determinado assunto, a especialização de um funcionário que consista em um gerente etc;
- **Propriedades (e valores de propridades) destas entidades**: atribuem um nome e um valor a uma determinada característica de uma entidade, representando que um conceito pode conter mais do que sua própria denominação ou seus relacionamentos, mas também podem manter um conjunto de qualidades ou predicados que auxiliem a melhor caracterizar as diferentes entidades. Estas propriedades podem ter, como exemplo: uma entidade disciplina que contém uma propriedade código e um valor para o mesmo; uma entidade pessoa pode ter a 'cor do cabelo' como uma propriedade e 'castanho' como valor desta propridade;
- **Restrições e regras sobre estas entidades**: são as imposições, ou formas de regular as entidades de um domínio sob diferentes aspectos, como impor que uma determinada entidade, para existir, deve ser formada por uma outra, ou apresentar uma determinada característica. Por exemplo, em um domínio de Iniciação Científica, para a existência de um artigo, é preciso que exista um professor orientador para o trabalho;
- **Axiomas**: "fornecem as informações factuais básicas a partir das quais podem ser derivadas conclusões úteis" (RUSSEL, S. J., NORVING 2004). As informações fornecidas formam uma afirmação que, dentro do domínio, é tida como uma verdade absoluta. Por exemplo, ao dizer que o membro de uma banca da disciplina de Estágio Supervisionado em Sistemas de Informação deve ser um professor e possuir uma graduação é um axioma dentro de um domínio de uma Instituição de Ensino Superior.

Ao elaborar uma forma de representação que habilite que algoritmos, empregados em computadores, possam trabalhar com conceitos, relacionamentos, restrições, axiomas, instâncias e vocabulários, é conferida a estes computadores a possibilidade de também explorar os conhecimentos que se tem deste domínio. Isto pode resultar em uma participação mais autônoma dos computadores em processos que exigiriam intermédio de pessoas para caracterizar o contexto de utilização ou na descoberta de características que podem ser utilizadas para melhor satisfazer os resultados de suas ações.

Os espaço de aplicações de ontologias é explorado por McGuinness (2003), que apresenta diferentes formas destas aplicações, fazendo uma separação entre ontologias constituídas de forma estruturalmente simples, caracterizadas como taxonomias, e de

ontologias com estruturas mais complexas. Adiante são apresentadas algumas das aplicações possíveis através de ontologias estruturalmente simples (MCGUINNESS, 2003):

- **Fornece um vocabulário controlado para um domínio**: permite que um conjunto de termos seja compartilhado entre usuários, desenvolvedores, aplicação, banco de dados etc.;
- **Organização e navegação**: a estrutura pode ser exibida no formato de uma árvore, em que seus itens podem ser expandidos ou contraídos, para facilitar a leitura, além de poder oferecer um acesso direto aos recursos por meio de URLs;
- **Definição de hierarquia**: a definição de uma hiearquia de forma que seus elementos sejam corretamente enquadrados pode ser utilizada como um padrão para consulta por diferentes aplicações;
- **Suporte à busca**: através de palavras-chave, é possível encontrar termos que estejam relacionados hierarquicamente e que, portanto, podem ser retornados por estarem relacionados de alguma forma que não seja apenas sintática;
- **Auxílio ao processo de eliminação de ambiguidade**: ao perceber que um termo ocorre em mais de um lugar dentro da hierarquia de uma taxonomia, é possível elevar este termo dentro dessa estrutura, que passa a ser mais concisa;

Os recursos oferecidos por uma ontologia estruturalmente simples sugerem apenas a definição de uma classe e suas subclasses, seguindo a estrutura de uma taxonomia. As aplicações empregam a hierarquia dos conceitos para formar apresentações em forma de árvore, buscar por conceitos semelhantes ou com maior ou menor generalização.

A construção de taxonomias é relevante inclusive na elaboração de uma ontologia definida com maior complexidade. Isto ocorre como um passo dentro de um processo de desenvolvimento de ontologias, que passa a ser relevante quando é preciso relacionar os conceitos que fazem parte de um domínio, assim como definir, hierarquicamente, a relação destes conceitos com os demais. Após delinear o escopo tratado do domínio, definindo suas classes e, opcionalmente, suas propriedades, a concepção de uma ontologia complexa pode se dar de modo mais natural, ao passo que o responsável por sua criação mantém uma visão melhor dos conceitos envolvidos e pode empregar uma análise mais detalhista para constituir a ontologia mais complexa. De forma análoga, a taxonomia está para uma ontologia estruturalmente complexa assim como um Modelo Conceitual (empregado na Engenharia de *Software* e representado pela UML) está para o Diagrama de Classes.

Ao passar a empregar ontologias estruturalmente complexas, extrapolando os

limites de uma taxonomia, os resultados obtidos podem levar a uma maior habilidade de automação de processos, conforme as aplicações adiante (MCGUINNESS, 2003):

- **Verificação da consistência**: se a linguagem da ontologia contém propriedades e restrições para as propriedades, é possível usá-las para a verificação dos valores das instâncias criadas, através de aplicativos;
- **Inferir informações**: ao coletar informações, é possível que diferentes sistemas identifiquem o que cada informação coletada representa, para que seja possível atribuir maior ou menor relevância, de acordo com o contexto de aplicação, e a empregue para promover alguma modificação ou melhoria;
- **Suporte à interoperabilidade**: ao especificar um domínio por meio de uma ontologia, todas as suas definições poderão ser aplicadas por diferentes pessoas e aplicações como forma de obter uma mesma visão do domínio representado;
- **Suporte à busca estruturada, comparativa e customizada**: ao definir uma busca por um determinado termo, é possível realizar uma busca sobre ontologias que permitirão buscar pelo termos presentes no mesmo contexto, ou semanticamente semelhantes, além de permitir uma apresentação das propriedades que sejam interessantes;
- **Explorar generalização/especialização de informações**: ao analisar um conjunto de informações provenientes de uma ontologia é possível aumentar ou diminuir o grau de abstração dos resultados apresentados, ao passo que são exibidos ou escondidos mais detalhes sobre os elementos ou ao navegar por meio de seus relacionamentos.

As possibilidades de aplicação de ontologias podem aproveitar conceitos, propriedades, restrições, axiomas e dirigir sua aplicação para a verificação de restrições de valores, buscas mais eficientes, abstração de conhecimento e descoberta de conceitos que não estão diretamente ligados. Estas aplicações é que criam a necessidade de que seja utilizada uma linguagem com recursos que permitam uma representação expressiva e bem estruturada de ontologias. Para tanto foram criadas ou estão em desenvolvimento linguagens para a representação de ontologias, como: OIL (OIL, 2000), DAML (DARPA, 2000), DAML+OIL (W3C, 2001) e a OWL (W3C, 2004).

Na próxima seção serão apresentados os recursos empregados durante o desenvolvimento do trabalho, dentre os quais estão a linguagem OWL e as ferramentas que foram úteis para o seu manuseio. A seção mantém os passos desenvolvidos, além das diferentes tecnologias, técnicas e ferramentas, aplicadas em diferentes circunstâncias ao longo dos estudos e desenvolvimento.

3 MATERIAIS E MÉTODOS

Neste capítulo será apresentada a metodologia empregada na elaboração do trabalho. O trabalho foi desenvolvido através da realização de pesquisa bibliográfica, utilização de equipamentos de *hardware*, complementada por um conjunto de *software* adequado. Estes itens, somados à orientação prestada, permitiram o desenvolvimento e a conclusão do trabalho.

3.1 Local e Período

O desenvolvimento do trabalho ocorreu no Centro Universitário Luterano de Palmas, no LABMÍDIA (Laboratório de Multimídia). O período de realização deu-se no 2º semestre do ano de 2005, como requisito para a disciplina de 'Estágio Supervisionado em Sistemas de Informação'.

3.2 Materiais

Os diferentes materias necessários ao desenvolvimento deste trabalho foram obtidos pelo Centro Universitário Luterano de Palmas, incluindo setores como a Biblioteca e o Complexo de Informática, no qual estão situados os recursos do curso de Sistemas de Informação. De modo complementar, utilizou-se a *Internet* como forma de pesquisa de fontes adicionais para formar o referencial teórico necessário.

3.2.1 Hardware

- Pentium IV, 2.4 Ghz e 512 Mb de RAM (Disponível no laboratório).

3.2.2 Software

Os recursos de *software* empregados no desenvolvimento do trabalho consistiram em: aplicativos para a leitura de documentos que serviram como referência bibliográfica, cujas extensões compreenderam: doc, pdf e ps; *container* JSP como suporte à tecnologia JSTL; Máquina Virutal para permitir a execução de aplicativos Java; IDE para a construção dos algorítmos na linguagem Java e de outras tecnologias como o HTML, JSP e JSTL; navegador para a realização de testes com as tecnologias *Web* empregadas, bem como para as consultas por materias na *Web*; uma IDE para o auxílio à construção de ontologias em OWL; sistema operacional que desse suporte a todos os itens citados nesta seção.

No desenvolvimento do trabalho foram utilizadas as ferramentas de *software*:

- Acrobat Reader 7.0.0.1;
- Ghostscript 8.1.4 e GSView 4.6;
- Apache Tomcat 5.5.9;
- Eclipse 3.1;
- Protégé;
- Internet Explorer 6.0;
- Máquina Virtual Java (j2sdk 1.4.2_08);
- Microsoft Office XP Professional;
- Microsoft Windows XP Professional.

3.2.3 Fontes Bibliográficas

A realização de pesquisas bibliográficas pôde ser realizada através de livros e periódicos disponíveis na Biblioteca do CEULP/ULBRA, além de diferentes materiais publicados na *Internet*. As diferentes fontes pesquisadas consistiram em:

- Livros;
- Dissertações de Doutorado;
- Dissertações de Mestrado;
- Trabalhos de Conclusão de Curso;
- Publicações Científicas;
- Artigos;

- Documentações de APIs;
- *Sites* diversos.

3.3 Metodologia

O primeiro passo no desenvolvimento do trabalho foi a construção da base teórica acerca dos conceitos que o embasavam e das tecnologias que poderiam auxiliar no alcance dos objetivos estabelecidos. Inicialmente foram realizadas pesquisas sobre a área da Hipermídia Adaptativa com o intuito de reconhecer seus propósitos, entender os componentes utilizados nas implementações de sistemas desta área, compreender métodos e técnicas empregados, além da etapa atual da evolução desta área e seus principais problemas. Obtida a visão geral do tema, partiu-se para o estudo de uma área específica de atuação de SHA, sendo delimitada a área de sistemas educacionais. O estudo dessa área permitiu, novamente, identificar materiais, métodos, objetivos e problemas da área, com o intuito de auxiliar na correta fundamentação teórica para o desenvolvimento do trabalho.

Dado o propósito de aplicar ontologias para a definição de SHA, foram levantados os aspectos envolvidos no estudo de ontologias e sua construção. Este levantamento, realizado sobretudo pelas possibilidades de aplicação desta área, teve como principal ponto de análise sua adequação à area de Hipermídia Adaptativa, permitindo a compreensão dos principais pontos beneficiados com a combinação da área com a tecnologia.

Ao formar a base teórica necessária à compreensão das áreas e tecnologias supramencionadas, foram trabalhados os conhecimentos para que fosse possível criar a habilidade prática de manusear ontologias especificadas em OWL, aplicar consultas sobre as instâncias das ontologias e oferecer estes processos através de uma interface em hipermídia. A construção desta interface foi possível através de recursos da Plataforma Java, como a linguagem Java, o JSP (*Java Server Pages*) e o JSTL (*Java Standard Tag Library*). A primeira etapa do manuseio de ontologias e da criação de suas instâncias envolveu, sobretudo, a ferramenta Protégé. Entretanto, compreendidos os processos da criação por meio de uma interface gráfica, partiu-se para o manuseio destas ontologias através da API Jena 2.

Após concluir os estudos e obter habilidades separadas na criação, manuseio, consulta às ontologias, procurou-se integrar os recursos individuais na criação de um protótipo de um *site* cujo domínio tenha sido específicado através de uma ontologia e que a adição de novas informações, ou sua alteração, possa ser feita pelo próprio *site*.

Nas seções subsequentes são apresentadas as tecnologias mencionadas. As apresentações mantêm uma breve descrição sobre as tecnlogias e suas principais atuações dentro do trabalho.

3.3.1 OWL

Dentre os propósitos deste trabalho está o de montar um protótipo a partir de um domínio representado por uma ontologia. Diante disto, foi adotada a *Web Ontology Language* (OWL) para que o domínio fosse representado e suas instâncias pudessem ser criadas ou manipuladas. A OWL consiste numa linguagem de marcação semântica para a publicação e compartilhamento de ontologias na *Web*, cujo desenvolvimento ocorreu como uma extensão do vocabulário RDF, além de derivar da DAML+OIL *Web Ontology Language* (W3C, 2004).

A adoção desta linguagem permitiu que o domínio de conteúdos educacionais, construído por Carneiro e Brito (2005), fosse utilizado para a construção do protótipo. Esta construção envolveu o esquema da ontologia, contendo as classes, propriedades, restrições e axiomas referentes ao domínio de materiais com conteúdos educacionais. Através deste esquema foi possível gerar os formulários que permitem a inclusão de novas instâncias desta ontologia.

3.3.2 Java, JSP e JSTL

Tendo em vista a adoção da *API Jena 2 Ontology* para a manipulação de ontologias em OWL, foi adotada a linguagem Java (SUN 2003) como a base para a implementação dos algoritmos e contrução do protótipo. Acrescenta-se, ainda, que essa decisão foi sustentada pelas características da linguagem como a Orientação a Objetos e a possibilidade de ser empregada juntamente com outras tecnologias. Essa última característica permitiu que a aplicação fosse apresentada como um *site* na *Web*. Estas tecnologias consistem no JSP - *Java Server Pages* (SUN 2004) e JSTL - *Java Server Pages Standard Tag Library* (SUN 2005).

O JSP permite que classes e algoritmos implem entados pela linguagem Java sejam utilizados para o desenvolvimento de páginas em camadas distintas (habitualmente três camadas: lógica de negócios, acesso aos dados e apresentação). No presente trabalho esta tecnologia foi fundamental para que os algoritmos, desenvolvidos para o manuseio de

ontologias, fossem apresentados por uma interface na *Web*, que fosse independente do sistema operacional utilizado e que mantivesse o processamento dos algoritmos centralizado no servidor. Esta segunda característica é de fundamental importância, pois é necessário que o formato da apresentação não limite o acesso de usuários às informações.

Com o intuito de estruturar melhor as etapas de construção da interface, foi adotada a JSTL, que oferece um conjunto de *tags* que realizam diferentes ações com frequente ocorrência durante o desenvolvimento e que podem ser reutilizadas em diferentes páginas JSP. Estas funcionalidades, apresentadas em Sun (2005), consistem num conjunto de *tags* padronizadas que efetuam ações que normalmente seriam executadas por códigos inseridos na própria página, denomados *striptles*. Ao utilizar tais *tags* no lugar de *scriptles* são retiradas as codificações em Java, reforçando a idéia de separação entre a camada de apresentação e as demais. Isto ocorre ao analisar que o responsável pela interface de uma página pode não estar relacionado ao desenvolvimento da lógica de negócios, e desconhecer os recursos do Java. Além disso, o código de uma página desenvolvida através de JSP e JSTL é estruturalmente semelhante a documentos XML, desde que os elementos HTML sejam escritos dentro de suas suas convenções (*tags* corretamente aninhadas, apenas um elemento raiz etc.).

O uso do JSTL auxiliou, ainda, na construção das páginas JSP, por meio de elementos, como: laço de repetição, condicionais, acesso às classes sem o uso de código Java inserido, dentre outros. O uso desta tecnologia criou, também, o espaço para que sejam adicionados recursos que podem melhorar a aceitação e a utilização da interface de um modo geral, como oferecer internacionalização, usar tecnologias como AJAX, de modo simplificado, etc.

3.3.3 API Jena 2 Ontology

A API Jena, que consiste num *framework* para a construção de aplicativos para a Web Semântica, provê uma forma de manipular, através de algorítmos, documentos RDF, RDFS e OWL (HEWLETT-PACKARD 2005). Por sua capacidade de trabalhar com documentos OWL, além do fato de possuir diferentes formas de consultar suas instâncias, sua aplicação foi tomada como necessária dentro do trabalho proposto.

Os algoritmos criados para pesquisar as ontologias através de seus esquemas, assim como de suas instâncias, tiveram como base os recursos oferecidos pela API Jena, que dentre as formas de pesquisa, permite uma navegação por seus nós, de acordo com seus

relacionamentos, e a elaboração de consultas no formato da linguagem *SPARQ Query Language*, que consiste numa linguagem para consulta e em um protocolo (permite que as informações sejam trocadas através da *Web)* para o RDF (W3C 2005).

O formato da navegação dos nós respeita o formato de triplas RDF, no qual cada sentença é composta por um predicado, sujeito e recurso. A partir dessa composição, foi possível realizar percursos sobre as instâncias das classes, verificar suas propriedades e relacionamentos, e chegar às informações desejadas. Assim que estas informações são acessadas, é possível manipular seus elementos de modo a adicionar, alterar ou excluir seus valores.

As consultas realizadas pela *SPARQ Query Language* permitiram chegar diretamente aos nós desejados. Isso porque tais consultas utilizam os valores de um literal, sua propriedade, a URI (Uniform Resource Identifier) do recurso, e outras formas indiretas, como o relacionamento com outros conceitos ou buscas aproximadas. Assim que um elemento é retornado, no formato de uma sentença, é possível aplicar as mesmas manipulações citadas para o caso de uma navegação pelos nós.

Um ponto a ser ressaltado sobre a utilização da API Jena 2 é que no início das atividades do estágio, a API tinha como versão mais recente a 2.2, na qual eram trabalhadas as consultas pela linguagem RDQL. Antes da conclusão do presente trabalho a versão 2.3 da API foi lançada, passando a trabalhar com a *SPARQ Query Language*.

A estrutura das consultas da SPARQ QL apresenta diferenças com relação a estrutura da linguagem RDQL, como diferentes formas de referenciar a *namespace*, palavras reservadas distintas, utilizadas para denominar os diferentes recursos de suas consultas etc.. Além disso, são oferecidos recursos distintos que auxiliam na elaboração de consultas mais eficientes, como critério de ordenação, filtros mais complexos etc. Entretanto, ainda que existam diferenças em seus recursos e estrutura, as consultas realizadas na linguagem RDQL podem ser executadas pelo motor de busca que processa a SPARQ QL. Isto permitiu que as consultas que inicialmente foram construídas para serem executadas no motor de busca da RDQL pudessem ser mantidas e aplicadas na versão mais recente da API Jena, destinadas à execução da SPARQ QL.

4 RESULTADOS E DISCUSSÃO

A aplicação de recursos destinados à manipulação de ontologias e ao processamento de consultas que permitam usufruir parte dos benefícios de sua capacidade de representação de domínios constitui uma tarefa importante no processo de compreensão das técnicas e ferramentas úteis na construção de sistemas hipermídia com um domínio de aplicação bem contextualizado. Esta tarefa leva ao entendimento inicial dos recursos que podem auxiliar o desenvolvimento de um sistema hipermídia que adapte sua apresentação às características de seus usuários.

Com o propósito de estudar ferramentas e técnicas destinadas à aplicação de ontologias, foi desenvolvido um protótipo cujo conteúdo apresentado será mantido por um modelo de ontologia, que contém o escopo do domínio e a definição dos elementos dentro dele, e suas instâncias, que mantêm as informações que seguem o modelo. O desenvolvimento incluiu a elaboração de recursos para as ações de inserção e remoção de instâncias à ontologia. Além disso, foi trabalhada uma forma apresentação das instâncias da ontologia que mantenha a coerência com o propósito de sua aplicação. Dessa forma, nas subseções conseguintes serão relatadas os recursos criados para o desenvolvimento do protótipo e o resultado do desenvolvimento.

4.1 Estrutura do protótipo

O protótipo desenvolvido tem como característica a apresentação de informações através de páginas dinâmicas, que apresentam seu conteúdo de acordo com um conjunto de algoritmos. Os algoritmos empregam uma série de recursos oferecidos pela API Jena como forma de manusear e obter informações de uma ontologia, que são usadas para definir e limitar o escopo do conteúdo do sistema. Dessa forma, todo o conteúdo apresentado no

protótipo é mantido e guiado pelas definições de um domínio que, por consistir numa ontologia e ser representado por uma forma de representação recomendada pelo W3C, pode ser reconhecido, empregado e ampliado por diferentes sistemas .

O contexto de aplicação do sistema é o de ensino, do qual fazem parte conceitos como curso, disciplina e materiais didáticos. A elaboração do protótipo foi baseada numa ontologia cuja representação constituísse, pelo menos, parte do domínio "ensino" e que suas instâncias representassem os diferentes indivíduos presentes neste meio. Seu fluxo de trabalho normal é constituído pela interação do usuário com a interface, que gera o conjunto de rotinas para a criação, remoção ou consulta de conceitos presentes na ontologia. A base para a construção destas funcionalidades foi construída pela utilização dos recursos da API Jena. Na Figura 3 é apresentada a organização dos componentes.

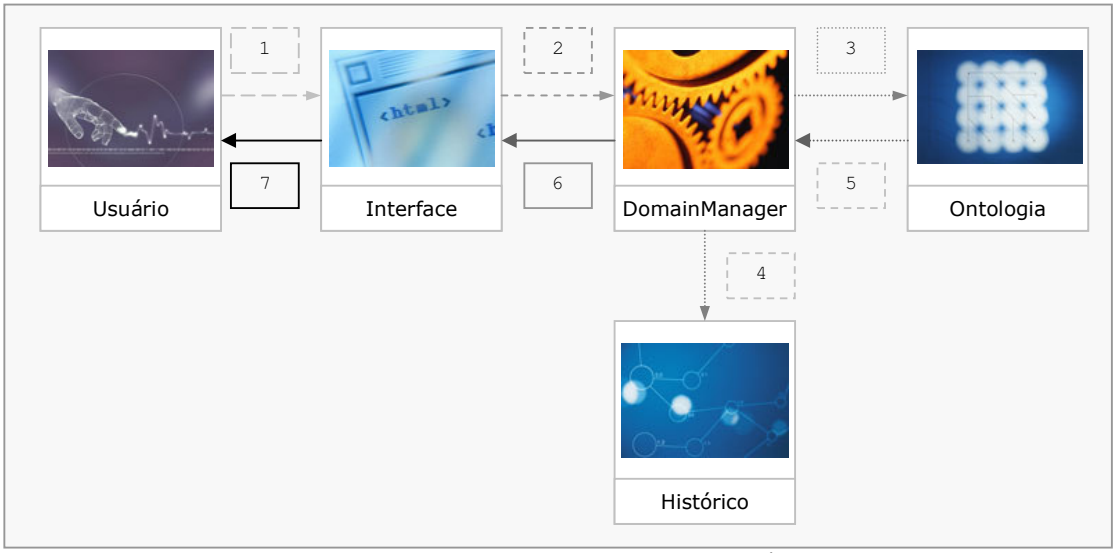

Figura 3. Fluxo de execução do protótipo.

Conforme é apresentado na Figura 3, assim que o usuário interage com a interface (1), especificando algum elemento para a busca, ou executa ações para criar ou remover uma instância, a interface utiliza o conjunto de classes implementadas em *DomainManagers* (2) para realizar as ações sobre a ontologia (3). Assim que estas ações são realizadas, o conteúdo da ontologia é novamente recuperado (5), a fim de atualizar as informações, e o conjunto de classes trabalha o resultado de modo a compor uma estrutura conveniente à passagem para a interface (6). Ao receber estas informações, a interface aplica o último conjunto de manipulações para apresentar o conteúdo recebido de uma forma compreensível ao usuário (7).

Caso sejam realizadas ações que alterem o conteúdo da ontologia, como a criação de uma instância ou a remoção de outros elementos, as classes pertencentes ao *DomainManagers* recebem um conjunto de eventos, demonstrando a natureza da ação

realizada (inserção ou remoção) e então as registram em um documento (4). Estes registros formam um histórico das ações executadas e podem ser utilizados com a finalidade de conhecer as transformações realizadas em um domínio.

Como pôde ser observado na Figura 3, os módulos envolvidos na construção do protótipo incluem:

- uma ontologia para a definição dos conceitos trabalhados no protótipo, assim como as diferentes regras, restrições e axiomas que regem estes conceitos;
- um documento de histórico que, de uma forma incipiente, captura as transformações realizadas na ontologia e que poderão servir para processos futuros de análise;
- um conjunto de classes, denominado *DomainsManagers*, que mantém implementações que utilizam os recursos da API Jena para manipular ontologias, capturar as ações realizadas sobre elas, além de realizar consultas que reflitam as solicitações recebidas pelo usuário;
- um conjunto de páginas que resulta na interface do protótipo, que serve de intermédio para que os usuários solicitem ou enviem informações, possibilitando que as classes implementadas possam executar os procedimentos necessários ao atendimento destes eventos.

A caracterização das classes dentro de um recurso *DomainsManagers* confere a idéia de que, para o domínio trabalhado, seja mantido um recurso específico para a sua administração, e que possam existir implementações que trabalhem com outros domínios dentro do protótipo. Ainda que o desenvolvimento do protótipo tenha como aplicação a área do "Ensino", buscou-se, também, oferecer recursos que atendam diferentes ontologias, sem direcionar a implementação à ontologia empregada inicialmente. Contudo, houve a necessidade de realizar implementações específicas, o que resultou em duas diferentes formas de abordagem.

Os recursos construídos para a consulta e o manuseio da ontologia foram modularizados e produzidos sob duas abordagens: a de permitir a manipulação e realização de consultas genéricas independentes da ontologia fornecida como definição do domínio e emprego das classes com aplicação genérica para que fossem realizadas consultas destinadas à aplicação "ensino", com o propósito de melhor contextualizar a apresentação aos usuários. A arquitetura desenvolvida para o protótipo é apresentada a seguir, na Figura 4

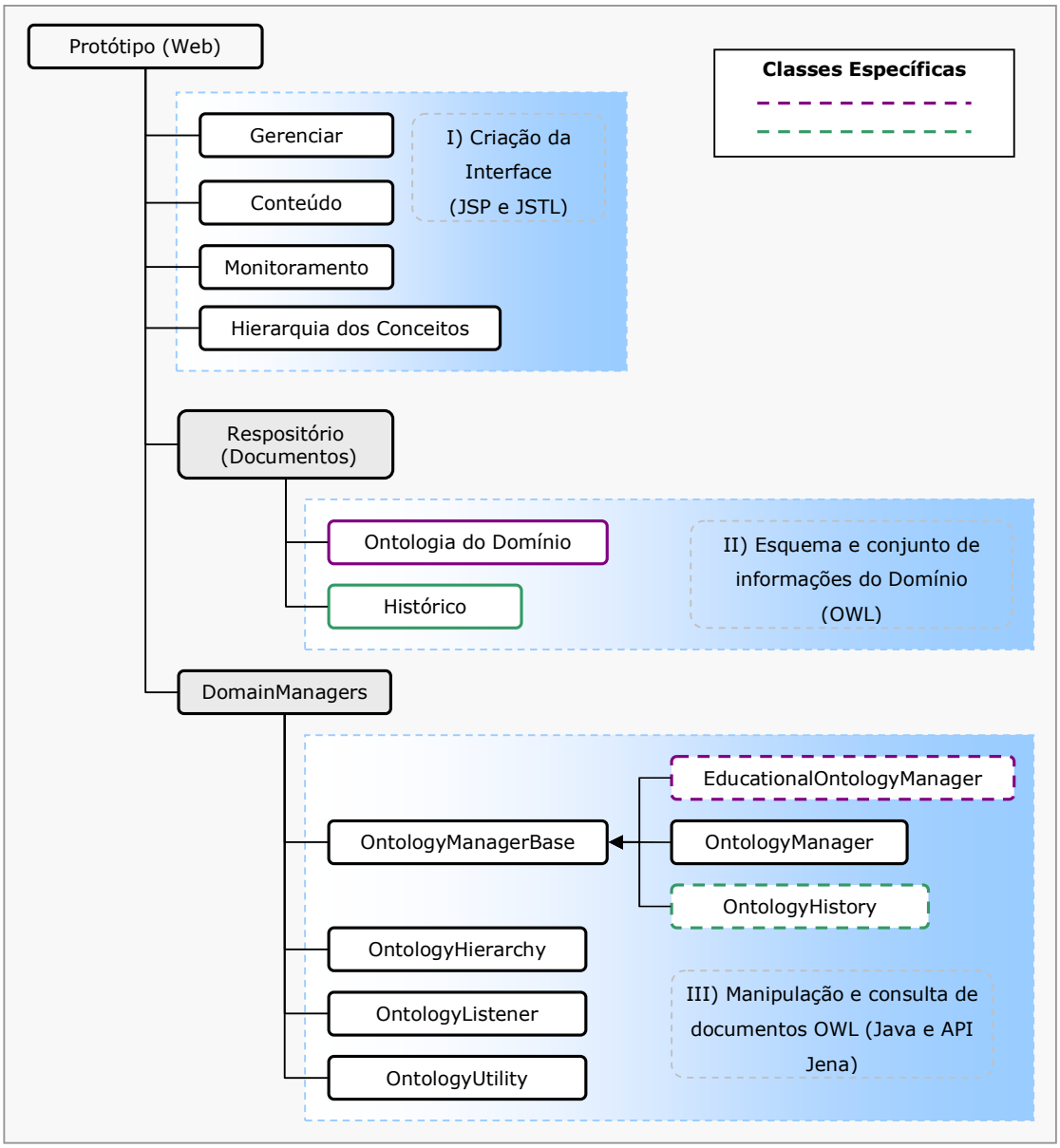

Figura 4. Arquitetura do protótipo.

Conforme apresentado na figura 4, para o desenvolvimento do protótipo foram incluídos alguns recursos, a saber: páginas dinânimas (Figura 4-I), classes para o acesso e o manuseio das informações (Figura 4-II), e documentos que, neste caso, são as ontologias (Figura 4-III). Como a hierarquia sugere, o conjunto de páginas (I) mantém acesso aos demais elementos de *Repositório* e *DomainManagers*, que mantêm seus recursos compartilhados.

Para permitir uma apresentação em formato de hipertexto, foi construído o conjunto de páginas (Figura 4-I), desenvolvidas com a tecnologia JSP e complementadas pela JSTL. Essas páginas coordenam o fluxo de execução e utilização geral do protótipo, atuando como a interface entre usuário e os recursos oferecidos. Assim que um usuário realiza uma

ação dentro da página, como clicar em um conceito para detalhá-lo, as páginas disparam eventos que invocam os métodos implementados nas classes de *DomainManagers*. Ao realizarem as ações, estas classes retornam um conjunto de dados processados e armazenados em tipos abstrados de dados como listas ou vetores. Dessa forma, cabe às páginas ler as informações destes TADs e então montar uma apresentação que seja compreensível ao usuário, como estruturas de tópicos, detalhamento dos nomes dos campos etc. De uma forma geral, o percurso através das informações dos TADs é realizada por meio de elementos da biblioteca de tags JSTL e o resultado final da construção é apresentado em HTML ou XHTML.

As classes pertencentes ao pacote *DomainManagers* (Figura 4 - III) implementam as tarefas de acesso aos documentos e oferecem as funcionalidades que permitem a criação de instâncias de ontologias, a busca pelos recursos de uma instância, a exclusão de elementos etc. Conforme citado anteriormente, foram definidas classes que permitem o trabalho com quaisquer esquemas de ontologias, que incluem as ações de criação de instâncias, acompanhamento das ações realizadas nos documentos e descoberta de recursos. Estas classes podem ser percebidas pelos elementos encontrados de *DomainManagers*, na Figura 4, que são: *OntologyManagerBase*, *OntologyHierachy*, *OntologyListener*, *OntologyUtility* e *OntologyManager*.

As classes de *DomainManagers* construídas para um domínio específico constituem as classes para a realização de tarefas específicas para um determinado modelo de ontologia. Assim, para cada documento OWL, que representa um contexto, tem-se uma classe adicional. A Figura 5 apresenta a estrutura destas classes assim como seus documentos.

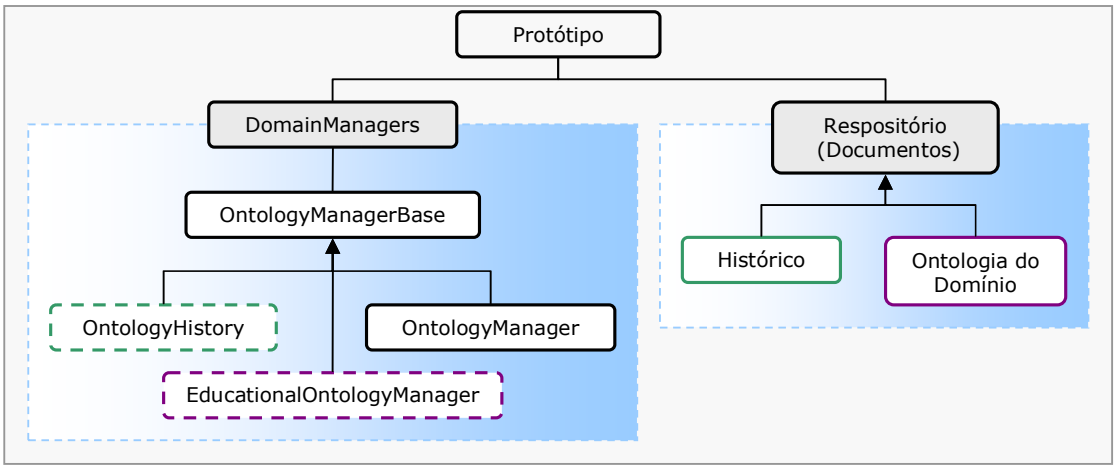

Figura 5. Hierarquia das classes e dos documentos gerenciados por estas classes.

As classes *EducationalOntologyManager* e *OntologyHistory* herdam os mesmos recursos da classe *OntologyManagerBase*, porém tratam, cada uma, de um domínio específico (Figura 5). Dentre os recursos herdados estão os métodos para a execução de consultas SPARQ QL e funções comuns de se trabalhar em ontologias, como a leitura de documentos e seu armazenamento em memória. A partir destes recursos, ambas as classes específicas (*OntologyHistory* e *EducationalOntologyManager*) implementam métodos que são úteis ao trabalho com as ontologias que gerenciam.

Os documentos constituídos em *Repositório* são as possíveis ontologias definidas para o domínio, que até o desenvolvimento do trabalho, constui a ontologia que representa parte do contexto de ensino e a destinada ao armazenamento do *Histórico*. Este segundo documento é empregado como uma forma de *Log* do protótipo, mantendo as alterações realizadas na ontologia.

O papel dos diversos componentes que compõem o protótipo, bem como a forma de implementação de suas ações, serão descritos nas seções subseqüentes. Contudo, na seção 4.2 será apresentada a ontologia que define o escopo do domínio do protótipo.

4.2 Ontologia para definição do domínio

A definição do domínio de aplicação do protótipo através de uma ontologia é um dos quesitos estabelecidos para a realização deste trabalho. Outro ponto para o desenvolvimento deste protótipo é o contexto de representação desta ontologia, que deve ser aplicado à área de ensino. Contudo, como a elaboração de uma ontologia não está compreendida no foco deste trabalho, empregou-se a ontologia definida no trabalho de Carneiro e Brito (2005) para delimitar o foco de apresentação do protótipo (Figura 6).

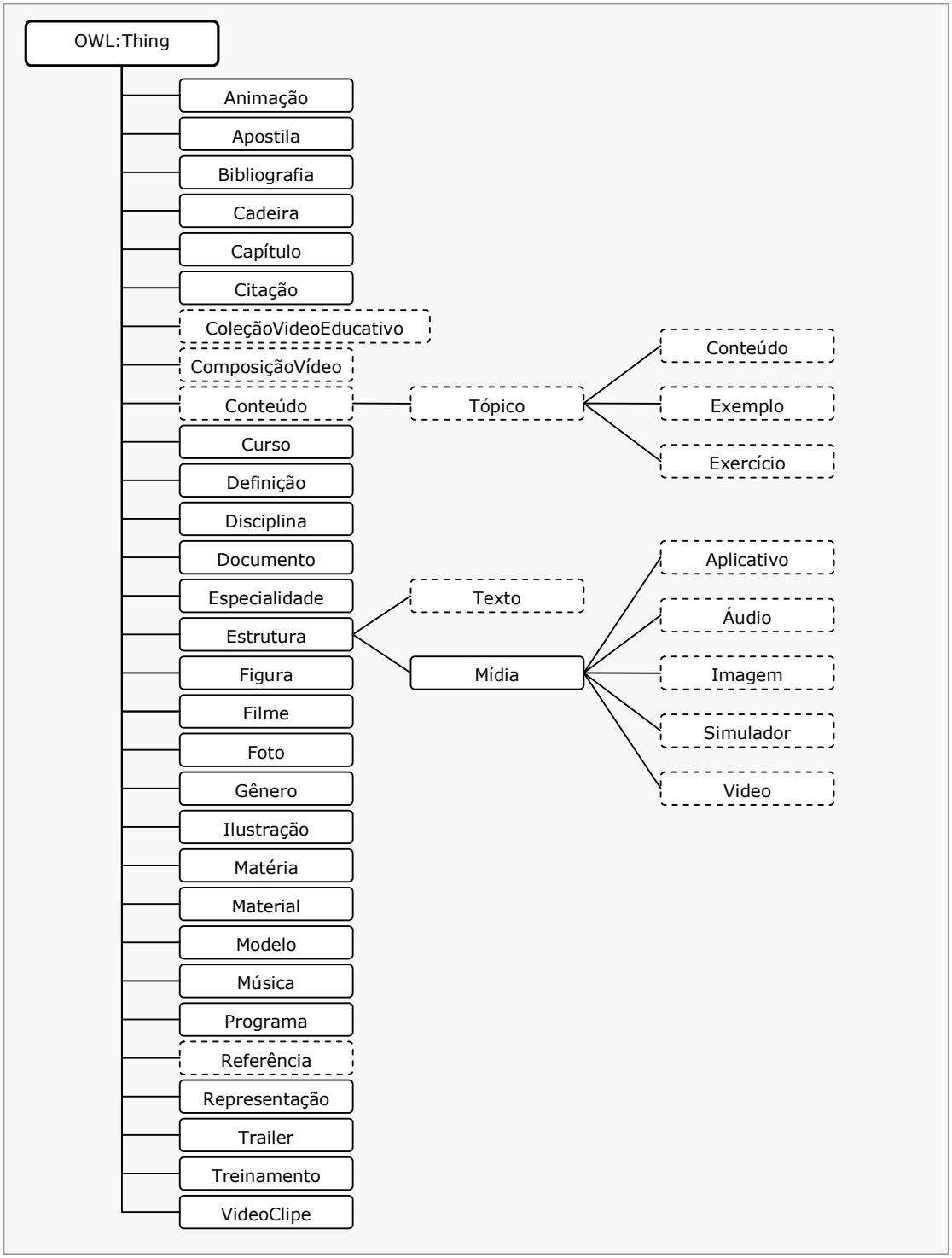

Figura 6. Hierarquia das classes (CARNEIRO e BRITO, 2005).

A ontologia definida por Carneiro e Brito (2005) compreende sobretudo os diferentes materiais que podem servir de conteúdo didático, porém, sem deixar de conter os principais conceitos do universo "ensino", como *Curso* e *Disciplina*, e os sub-elementos de um provável conteúdo, como as propriedades contidas pelos elementos (não especificadas

na imagem), como *Tópico*, *Referência* etc. As relações que transformam este conjunto de conceitos em uma hierarquia, resultando na idéia de elementos, subelementos e coleções, são criadas pelos atributos do tipo *ObjectProperty*, que mantém o domínio (*Domain*) de um objeto, no caso o item, e seu alcance (*Range*), o sub-item.

Um ponto importante a ser ressaltado na ontologia são os axiomas criados para que sejam obtidas alternativas à compreensão do domínio. Por exemplo, ao procurar por uma uma *matéria*, é possível procurar pelos conceitos de *matéria*, mas também pelos de *disciplina* e *cadeira*, que são termos que, de acordo com a ontologia, representam o mesmo conceito. Estes axiomas, que na Figura 6 aparecem com a linha tracejada, constituem "verdades" dentro do domínio. No exemplo citado a verdade indica que "uma disciplina é equivalente a uma matéria" e que "uma disciplina é equivalente a uma cadeira". Estas afirmações é que criam a possibilidade de obter novas informações, ainda que não estejam explicitamente demonstradas por relações definidas por *ObjectProperty*.

O primeiro exemplo citado de um axioma demonstra a equivalência entre termos. Um outro exemplo de axioma pode ser observado por *ColecaoVideoEducativo* que, de acordo com a ontologia especificada, traz a exigência de que, para ser uma coleção de vídeos educativos, é preciso que um vídeo possua, como valor da propriedade *possuiGênero*, um *Gênero* do tipo *educativo*.

4.3 Decomposição das funcionalidades do protótipo

Dada a definição de um domínio, conforme a ontologia apresentada na seção anterior, é preciso criar um conjunto de funcionalidades para que o protótipo seja capaz de gerenciar seu conteúdo, ou seja, criar, remover e consultar seus elementos. Estas funcionalidades são necessárias para que o protótipo seja completamente delineado por uma ontologia, sem que seja necessário empregar ferramentas adicionais, como o Protégé, para que as classes de um domínio sejam instanciadas, ou seja, tenham elementos de seu tipo criados.

Assim, foram criados métodos para: a importação de uma ontologia; a geração de um formulário para a criação de instâncias; a apresentação das instâncias de forma que seja possível navegar hierarquicamente por seus elementos; o monitoramento das ações realizadas sobre as instâncias da ontologia; a apresentação das funcionalidades através de uma interface hipermídia.

O conjunto das funcionalidades citadas é relatado nas próximas seções, incluindo os principais recursos necessários ao seu desenvovlimento.

4.3.1 Importação de uma ontologia

O primeiro passo no processo de importação de uma ontologia é a criação de um modelo e a leitura do documento OWL que contém sua definição. Dessa forma, foram criados dois métodos: um para trabalhar a leitura do modelo da ontologia e outro para armazená-lo em memória. Esta divisão diminui a quantidade de recursos utilizados na relação de leitura e atualização dos modelos, ao passo que permite identificar a necessidade de carregar novamente uma instância ou apenas manter a anteriormente lida. O primeiro destes métodos é apresentado a seguir, na Figura 7.

```
1.    protected static void abrir()
2.    {
3.        if(modelo == null || modelo.isEmpty())
4.        {
5.            spec = new OntModelSpec( OntModelSpec.OWL_MEM );
6.            modelo = ModelFactory.createOntologyModel(spec);
7.            ModelChangedListener listener = new OntologyListener();
8.            modelo.register(listener);
9.            carregar(false);
10.       }
11.   }
```

Figura 7. Método para a criação de um modelo de ontologia.

A criação de um modelo de ontologia, representado pela classe *OntModel*, é realizada pela classe criada para sua instanciação, constituída por *ModelFactory*. Contudo a criação de um modelo deve possuir uma especificação, definindo a natureza do documento OWL criado e o local de seu armazenamento. Dessa forma, na Linha 5, é instanciada a especificação de um modelo de ontologias OWL com armazenamento em memória e, adiante, na Linha 7, tal especificação é informada no momento da criação do modelo, através da passagem de parâmetro ao método.

Um ponto não obrigatório na definição de um modelo, porém, que foi trabalhado no protótipo, é o registro de uma classe para o acompanhamento dos eventos realizados sobre a ontologia. A classe utilizada para esta tarefa deve ser uma implementação da interface *ModelChangedListener*, que no caso foi implementada e instanciada por *OntologyListener*. A criação de *OntologyListener* e as funcionalidades implementadas serão tratadas na Seção 3.3.7 Após a instanciação da classe, sua associação ao modelo é realizada pelo método *registrer*, na Linha 8, e então é invocado o método para a leitura da ontologia e armazenamento de sua instância em memória (Linha 9). A Figura 8 apresenta o método responsável por esta leitura.

```
1.    protected static boolean carregar(boolean forcarRecarga)
2.    {
3.       try
4.       {
5.          if(modelo.isEmpty() || forcarRecarga )
6.          {
7.             modelo.read("file:" + caminho);
8.          }
9.       }
10.      catch(Exception ex)
11.      {
12.         return false;
13.      }
14.      return true;
15.   }
```

Figura 8. Método para a leitura de um ontologia e armazenamento em memória.

O método responsável pela leitura de um documento contendo a ontologia deve ter especificado o local do arquivo. Como a ontologia trabalhada é mantida fisicamente no mesmo meio de armazenamento que a aplicação, atribuiu-se o prefixo "*file:*" ao caminho especificado. Ainda que o método mantenha apenas uma linha de comando com a execução da ação, sua criação fez-se necessária para o controle de leitura do arquivo e o trabalho com possíveis exceções durante sua leitura. A opção de não ter que ler o arquivo toda vez que o modelo da ontologia for necessário a um dos métodos, confere a possibilidade de melhorar o desempenho durante múltiplas utilizações da ontologia pelo protótipo. Na Figura 9 é apresentada uma forma simples de utilização desses métodos para obter informações sobre o modelo carregado.

```
1.    public static ExtendedIterator getClasses()
2.    {
3.          abrir();
4.          return getModelo().listNamedClasses();
5.    }
```

Figura 9. Utilização do método *abrir*() e consulta dos recursos da ontologia.

Anteriormente às ações realizadas sobre o modelo da ontologia, o método *abrir* é chamado com o intuito de verificar a existência de uma instância da classe *OntModel* e verificar se o documento foi carregado. Na falta de um dos critérios, são realizados os procedimentos de leitura do documento OWL.

Ao término da execução dos métodos, é possível realizar diferentes ações sobre a ontologia, que é mantida em memória durante a execução do *site*. Ambos os métodos supracitados pertencem à classe abstrata *OntologyManagerBase* e são empregadas por suas subclasses para a leitura da ontologia. Na próxima seção será relatado o processo

necessário à descoberta dos recursos de uma classe.

4.3.2 Geração de um formulário para a criação de instâncias

O trabalho de gerar um formulário para que sejam criadas as instâncias de uma classe envolve, sobretudo, a descoberta das características dessa classe. Isto inclui a *URI* do recurso a ser instanciado, as propriedades existentes, os relacionamentos e restrições. O processo pode ser desenvolvido nas seguintes etapas:

- leitura da classe a ser instanciada;
- percurso através dos recursos da classe;
- reconhecimento das propriedades;
- definição das características do campo do formulário.

A tarefa de ler uma classe definida dentro de um domínio requer apenas a instância de um modelo e a URI da classe. A Figura 10 apresenta o processo.

```
1.    OntClass classe = getModelo().getOntClass(strElemento);
```

Figura 10. Leitura de uma classe.

A URI deve ser formada pela *namespace* do domínio da classe seguida de sua identificação, como: *http://www.ulbra-to.br/repositorio/ontologias/doc.owl#Curso*. Logo após manter uma referência à instância da classe, representada na API Jena por *OntClass*, é preciso realizar uma navegação por suas propriedades.

```
1.    for(ExtendedIterator i = classe.listDeclaredProperties(false);
2.    i.hasNext();)
3.    {
4.       OntProperty propriedade = (OntProperty) i.next();
5.       OntResource tipo = (OntResource) propriedade.getRange();
6.
7.       int cardinalidadeMinima =
propriedade.getCardinality(OWL.minCardinality);
8.       int cardinalidadeMaxima =
propriedade.getCardinality(OWL.maxCardinality);
9.       //...
10.   }
```

Figura 11. Iteração pelas propriedades de uma classe.

O procedimento de reconhecer as propriedades e restrições de uma classe requer a realização de um laço de repetição pelos elementos retornados em um iterator, obtido pela invocação do método *listDeclaredProperties* (Figura 11, Linha 1), pertencente à classe trabalhada. Dentro de uma iteração, a primeira tarefa a ser realizada é obter a instância da propriedade (Linha 4) para que em seguida seja obtido o tipo desta propriedade, que para o caso de ser um *DataTypeProperty*, pode conter valores como *Integer* e *String*, que devem

ser especificados para uma correta validação dos campos. Adiante, nas linhas 7 e 8, são obtidos os valores das cardinalidades desta propriedade, que também são relevantes no processo de geração de um formulário, restringindo as quantidades máxima e mínima de repetições de uma propriedade que a instância da classe poderá conter.

Todos os valores obtidos durante o código exemplificado na Figura 9 são úteis para ambos os tipos de propriedades (*DataTypeProperty* e *ObjectProperty*). Contudo, devem ser buscadas as propriedades específicas de cada tipo de propriedade e, dessa forma, é preciso começar com a identificação do tipo que a propriedade trabalhada possui. A principal influência da diferença entre o tipo das propriedades está no campo a ser gerado, que pode ser uma caixa de texto ou uma caixa de seleção, com diferentes opções disponíveis para que o usuário selecione. A forma de reconhecer o tipo de campo, bem como as propriedades específicas de cada um deles, é apresentada na Figura 12.

```
1.    if(propriedade.isDatatypeProperty())
2.    {
3.        //Campo de texto
4.        //Ações para a criação do campo
5.    }
6.    else
7.    {
8.       if( propriedade.getRange() != null)
9.       {
10.         //Caixa de seleção
11.         //Ações para a criação do campo
12.         OntClass dominio =
getModelo().getOntClass(propriedade.getRange().getURI().toString());
13.
14.         for (ExtendedIterator instancias = dominio.listInstances();
instancias.hasNext(); )
15.         {
16.            Individual instancia = (Individual) instancias.next();
17.            String nomeClasse = instancia.getLocalName().toString();
18.         }
19.       }
20.   }
```

Figura 12. Verificação do tipo de propriedade trabalhada.

O reconhecimento do tipo de uma propriedade pode ser realizado de forma simplificada, através do método *isDataTypeProperty* (Linha 1). Ao verificar que o campo consiste em uma *DataTypeProperty*, são realizadas as tarefas que auxiliarão a interface a criar o campo necessário, como: manter as informações de que o campo será uma caixa de texto, as cardinalidades máxima e mínima, e o tipo de valor que o campo poderá conter (*String*, *Integer* etc). Caso a propriedade seja reconhecida como um *ObjectProperty*, é determinado que o campo deverá ser uma caixa de seleção, também com cardinalidades mínima e máxima e, então, são relacionadas as instâncias da classe aceitas por esta

propriedade.

Como exemplo de geração de um campo para a propriedade é possível citar *Pré-Requisito* de uma disciplina. Este campo não possui uma cardinalidade definida, portanto é preciso dizer que ele é opcional e que, no entanto, pode ocorrer quantas vezes for preciso. Além disso, devem ser listadas dentro deste campo todas as instâncias da classe *Disciplina*, para que o usuário possa dizer quais delas constituem um pré-requisito para aquela que ele deseja adicionar.

Assim que as informações sobre uma propriedade são obtidas, é preciso definir uma forma de passar, para o recurso responsável pela criação da interface, todas as informações pertinentes à criação do campo. A forma encontrada de realizar tal tarefa foi a de agregar no nome do campo suas informações essenciais. Considerando as informações obtidas nas Figura 9 e 10, é demonstrada, na Figura 13, a forma de criação do nome de um campo.

```
1. nomeCampo =  "txt$" + tipo.getLocalName().toString()
    + "$" + propriedade.getLocalName().toString()
    + "$" + cardinalidadeMinima
    + "$" + cardinalidadeMaxima;
```

Figura 13. Agregação das informações sobre uma propriedade em uma *String*.

Como demonstrado na Figura 13, a primeira informação agregada ao nome de um campo é a natureza do campo a ser gerado, representado por "txt" como um indicativo de que esta propriedade deve ser representada por uma caixa de texto. Adiante, tem-se o caractere '$', que é responsável por delimitar as informações dos campos e facilitar a decomposição de todas as informações no nomento de sua geração. Logo após, é atribuído o nome da propriedade dentro do documento OWL, e então suas cardinalidades mínima e máxima.

A geração do nome do campo de uma propriedade *Ementa*, pertencente à classe *Disciplina*, tem como resultado, de acordo com a ontologia definida na Seção 4.2, o valor: "*txt$string$ementa$0$0*". Dessa forma, pode-se compreender que o campo deve ser um caixa de texto, que suporte valores de uma *String*, represente a propriedade *Ementa*, e que tenha as cardinalidades *0* e *0*, que resultam em uma não obrigatoriedade de ocorrerem em uma disciplina, porém, sem definir um número máximo de ocorrências.

Ainda que o nome do campo contenha as informações necessárias à sua criação, as informações atribuídas ao nome são armazenadas em uma estrutura de dados do tipo lista e então passada para a página JSP responsável pela exibição do formulário para o usuário. Para iterar sobre os elementos definidos para o formulário, foram utilizados os recursos da

JSTL, como laços de repetição e condicionais. A Figura 14 apresenta o trecho do código que gera o formulário em HTML.

```
1.    <c:forEach items='${itens}' var='item' varStatus="indice">
2.       <tr>
3.          <c:forEach items='${item}' var='subItem'
   varStatus="posicao">
4.             <c-rt:choose>
5.                <c-rt:when test="${ posicao.count mod 4 == 1 }">
6.                   <c:set var="tipoCampo" value="${subItem}"/>
7.                </c-rt:when>
8.
9.                <c-rt:when test="${ posicao.count mod 4 == 2 }">
10.                  <c:set var="valorCampo" value="${subItem}"/>
11.               </c-rt:when>
12.
13.               <c-rt:when test="${ posicao.count mod 4 == 3 }">
14.                  <th>
15.                     <span class="campo">
16.                        <c:out value='${subItem}'/>:
17.                     </span>
18.                  </th>
19.               </c-rt:when>
20.
21.               <c-rt:when test="${ posicao.count mod 4 == 0 }">
22.                  <!-- Especificação do campo - Figura 13 -->
23.               </c-rt:when>
24.            </c-rt:choose>
25.         </c:forEach>
26.      </tr>
27.   </c:forEach>
```

Figura 14. Iteração pelos elementos da lista por meio de uma tag JSTL.

Ao receber a lista com as informações de todas as propriedades, representada por '*itens'*, é realizado, para cada elemento, o processo de coleta das informações recebidas e então a criação da estrutura HTML que comportará os diferentes campos do formulário (Figura 14). Nas Linhas 1 e 3 pode ser observada a estrutura de repetição que passa por cada item e subitem, que respectivamente constituem as propriedades e suas características. Para cada posição dos subitens, são definidas variáveis que acessam seu conteúdo para uma posterior geração do campo. Para tanto, foi empregada a estrutura *when* da JSTL, que corresponde à estrutura de controle *switch,* do Java. Ao testar cada posição podem ser atribuídos os valores das variáveis ou realizar uma impressão direta em código HTML, conforme a Linha 16, inserida entre *tags* de uma célula de uma tabela do HTML. As demais variáveis vão utilizadas no trecho omitido na Linha 22, apresentado na Figura 15.

```
1.      <td>
2.        <c:if test="${ (tipoCampo eq 'text') }">
3.           <input type="text"
4.                  name="<c:out value='${subItem}'/>"
5.                  id="<c:out value='${subItem}'/>" />
6.
7.             <c:if test="${ (subItem != 'rdf:about') }">
8.                  <input type="button"
9.                         name="btn<c:out value='${subItem}'/>"
10.                        value="Adicionar"
11.                        onclick=”adicionarLinha(this)” />
12.            </c:if>
13.       </c:if>
14.       <c:if test="${ (tipoCampo eq 'textArea') }">
15.            <!-- -->
16.       </c:if>
17.       <c:if test="${ (tipoCampo eq 'list') }">
18.            <!-- -->
19.        </c:if>
20.     </td>
```

Figura 15. Criação do campo para uma propriedade.

A geração de cada campo mantém a estrutura de *tags* JSTL que, após verificar o tipo de cada campo, gera a arquitetura de diálogo correspondente, definindo os atributos *nome* e *id* de cada campo (Figura 15). Um ponto a ser ressaltado dentro da criação de um campo está na geração de um botão que permita adicionar ‘clones’ destes campos (Linha 7). Isto é necessário para que o usuário tenha a opção de registrar múltiplas ocorrências de uma mesma propriedade, desde que sua cardinalidade a suporte. Contudo, a validação do valor de uma cardinalidade, bem como dos valores digitados por um usuário, é realizada por *scripts* executados no navegador.

Após completar a composição de um formulário, a responsabilidade de recebimento dos dados é passada a outro método que, assim como a geração do formulário, pode ser utilizado para criar instâncias de qualquer ontologia, desde que seja informado seu modelo. A próxima seção trata do recebimento dos dados e da criação das instâncias.

4.3.3 Criação da instância de uma classe

O primeiro passo dentro do processo de criação de uma instância é a criação de um recurso, representado pela API Jena como *OntResource*. Entretanto, a criação deste recuso deve ser realizada pela instância do modelo que contém a classe que será instanciada e, para tanto, é preciso informar a URI que identificará esta instância. Na Figura 16 é demonstrado o processo de criação de um recurso.

```
1.    OntResource recurso = modelo.createOntResource(namespace +
   pagina.getParameter("rdf:about"));
2.    recurso.addRDFType(modelo.createResource(strElemento));
3.    modelo.enterCriticalSection(Lock.WRITE);
```

Figura 16. Criação de um recurso que represente uma classe.

Conforme apresenta a Figura 16, na primeira linha é criado um recurso cuja identificação tem o valor submetido pelo formulário criado na seção anterior. Na linha posterior, é definido o *RDFType* deste recurso, que estabelece que esta instância representará uma determinada classe, estipulada pela URI.

Um ponto importante no processo de criação de uma instância está na necessidade de manter a integridade de um documento OWL. Isto implica a necessidade de garantir que o documento não será escrito durante ou logo após a inserção de um recurso no documento, pois tal ação poderia fazer com que algumas informações fossem perdidas. Para tanto, na Linha 3 da Figura 16, é utilizado o método *enterCriticalSection*, "bloqueando" qualquer alteração de escrita que possa ocorrer enquanto toda a tarefa de criação de um recurso não estiver concluída.

```
 1. for(Enumeration i = pagina.getParameterNames();  i.hasMoreElements();) {
 3.    campo = i.nextElement().toString();
 4.    campos = campo.split("\\$");
 5.    valor = pagina.getParameter(campo);
 6.    if( campos[0].equals("txt") ){
 7.      recurso.addProperty (
               modelo.createDatatypeProperty (
               namespace + campos[2]),
               pagina.getParameter(campo).toString()
            ));
 8.     }
 9.    else if ( campos[0].equals("list") )
 10.    {
 11.       OntProperty referencia = modelo.createOntProperty(namespace +
campos[1]);
 12.       Resource alvo = modelo.getResource(namespace + valor);
 13.       recurso.addProperty(referencia, alvo);
 14.    }
 15.    else if ( campos[0].equals("textArea") ){
 16.       //(...)
 17.    }
 18. }
```

Figura 17. Atribuição das propriedades à instância.

Ao passar por cada campo recebido por parâmetro da uma submissão de um formulário gerado anteriormente, é possível reconhecer cada propriedade definida para a instância de uma classe. Para isto, é realizada a iteração pelos elementos recebidos (Linha 1, Figura 17) e, então, o nome do campo do formulário é recuperado (Linha 3). Considerando a estrutura criada para a composição de um nome, o primeiro passo para reconhecer cada parte dessa estrutura é a sua decomposição. Assim, na Linha 4, decompõe-se o nome do campo pelos valores delimitados pelo caracter '$'.

A informação sobre o tipo do campo é recuperada como forma de permitir o trabalho com a criação de cada propriedade (*ObjectProperty* e *DataTypeProperty*). Caso o tipo seja "*txt*", adiciona-se ao recurso uma propriedade do tipo *DataType,* atribuindo diretamente o valor digitado pelo usuário (Linha 8). Caso o tipo do campo constitua um "*list*", cria-se uma propriedade do tipo *Object* e, em seguinda, é localizado o elemento, correspondente a uma instância na ontologia, que o usuário selecionou no formulário. Isto cria um relacionamento entre a instância criada e a instância pré-existente, selecionada pelo usuário através da interface.

Assim que os dados enviados pelo formulário forem completamente percorridos e os recursos ou valores recuperados forem atribuídos ao recurso recém-criado, é necessário salvar as alterações do modelo no arquivo que o contém. A Figura 18 exibe este processo.

```
1. try
2. {
3.    modelo.write(new FileOutputStream(caminho),sintaxe);
4. }
5. finally
6. {
7.    modelo.leaveCriticalSection();
8. }
```

Figura 18. Escrita das alterações do modelo no arquivo.

O processo de atualização de um documento pode ser realizado de forma simplificada, conforme a Figura 18. Deve-se informar, ao modelo da ontologia, uma *Stream* que será responsável por informar o meio em as informações serão gravadas e a sintaxe que o documento apresentará (Linha 3). O segundo argumento, dentro do trabalho especificado, é definido com o valor "*RDF/XML-ABBREV*", que significa que o documento apresentará a sintaxe abreviava do modelo RDF escrito na linguagem XML.

Após salvar as alterações realizadas sobre o modelo, é preciso liberar o documento

que contém a ontologia para que outros processos possam realizar operações de escrita. Para tanto, na Linha 7 da Figura 18, é invocado o método *leaveCriticalSection*, da classe *OntModel,* que informa que a seção crítica foi deixada e que os demais processos podem transcorrer normalmente.

4.3.4 Leitura e interpretação das instâncias das classes

Após definir as formas de atribuição de novas instâncias ao domínio especificado na ontologia, foram construídos os métodos responsáveis pela leitura das instâncias de uma classe e sua interpretação, como forma de detalhar suas propriedades e relacionamentos. A forma de implementação destes métodos foi realizada de forma independente da ontologia fornecida, sendo que sua execução depende apenas de uma ontologia e uma URI que especifique uma instância dentro da mesma.

Dada a necessidade de fornecer uma instância para que suas características sejam apresentadas, em um primeiro momento foi definida uma alternativa à apresentação direta de uma instância, dando lugar à sua URI. Na Figura 19 pode ser observada a forma de recuperação de uma instância a partir de uma URI fornecida.

```
1. public static String getIndividuo(String uri)
2. {
3.     return representarIndividuo(
           getModelo().getIndividual(namespace + uri));
4. }
```

Figura 19. Recuperação de uma instância por uma URI.

Assim que uma URI é fornecida, é possível invocar o método *getIndividual*, pertencente à classe *OntModel*, para obter sua instância (Figura 19). A partir desta instância, são percorridas todas as suas características para que sua representação seja constituída, conforme apresenta a Figura 20.

```
1. protected static String representarIndividuo(Individual instancia)
2. {
3.    StringBuffer tabela = new StringBuffer();
4.    String type = "";

5.    for (ExtendedIterator i = instancia.listProperties();
i.hasNext();)
6.    {
7.       //Decomposição das propriedades - Figura 18
8.    }

9.    return tabela.toString();
10.}
```

Figura 20. Iteração pelos recursos de uma instância.

Como o processo de representação de um indivíduo fornece à interface apenas texto para ser exibido, sem uma estruturação mais complexa com tabelas ou elementos aninhados, as informações reconhecidas de um indivíduo são armazenadas em uma instância de *StringBuffer*, que se destina ao processamento de múltiplas operações com *String*. Após criar uma instância de *StringBuffer*, são obtidas as propriedades da instância trabalhada (Figura 20). Neste processo, cada valor é recuperado e decomposto no formato de triplas RDF, compostas por uma Sentença que possui um sujeito, um predicado e um objeto. A decomposição das propriedades em sentenças pode ser acompanhada pela Figura 21.

```
1.   Object objPropriedade = i.next();
2.   StatementImpl sentenca = (StatementImpl) objPropriedade;

3.   Property predicado = sentenca.getPredicate();
4.   Resource sujeito = sentenca.getSubject();
5.   Resource recurso = sentenca.getResource();

6.   String typeSentenca = getType(sentenca.asResource()).toString();

7.   tabela.append(predicado.getLocalName());

8.   if (recurso != null && !predicado.getLocalName().equals("type") )
9.   {
10.     //Adicicionar um link para que o elemento possa ser detalhado.
11.     tabela.append(predicado.getLocalName());
12.     tabela.append(": ");
13.     tabela.append("<a href=\"?EID=");
14.     tabela.append(recurso.getLocalName());
15.     tabela.append("\">");
16.  }

17.  if( recurso != null )
18.     tabela.append(recurso.getLocalName());
19.  else
20.  {
21.     try
22.     {
23.        tabela.append(sentenca.getLiteral().getValue().toString());
24.     }
25.     catch(Exception ex)   {     }
26.  }

27.  tabela.append("</a></p>");
```

Figura 21. Decomposição das propriedades de uma instância.

Como pode ser observado na Figura 21, cada sentença pode ser recuperada de um iterador e convertida para a classe correspondente na API Jena, como a *StatementImpl* (Linha 2). A partir desta classe são obtidos seu recurso, sujeito e predicado, que fornecem as informações necessárias ao detalhamento da instância da classe trabalhada.

O propósito da decomposição de uma instância, dentro dos objetivos do trabalho, é

o de mostrar todas as suas informações e, caso existam elementos relacionados, fornecer a possibilidade de detalhá-lo, como num processo de navegação hierárquica. Dessa forma, são recuperados os dados que permitem a representação de cada sentença, que é caracterizado por seu tipo, a sua identificação, que representa a URI, e o valor apresentado pela mesma, que também pode ser uma referência à outra classe, também representada por uma URI.

O tipo de uma sentença pode ser obtido pela propriedade RDF denominada *type*. Caso seja possível identificar o tipo de uma sentença, o identificador do seu predicado é atribuído como o nome que descreverá o valor, ou seu rótulo. O valor deste recurso, que constitui uma URI, é utilizado como *link* para uma possível exploração dos recursos deste elemento e mantém a idéia de que este elemento está aninhado à instância trabalhada.

Caso o recurso seja apenas uma propriedade simples, composta por um valor com *String*, *Integer, Boolean etc.*, não é necessário atribuir um *link* para o seu detalhamento, já que é suficiente mostrar o valor do *Literal* contido neste recurso. A compreensão desse fato pode ser obtida ao apresentar os recursos de uma disciplina. Ao exibir a descrição de uma disciplina, basta apresentar o nome dessa propriedade ("descricao") e então mostrar seu valor ("O objetivo desta disciplina é enfatizar os aspectos etc."). Um exemplo de situação que poderia exigir um *link* seria se esta disciplina tivesse uma propriedade "pré-requisito" referenciando uma outra disciplina. Dessa forma, é interessante que seja mostrado também o nome dessa propriedade ("préRequisito") e então fornecer um *link* para para o recurso. A Figura 22 apresenta as propriedades e os relacionamentos exibidos de um recurso selecionado.

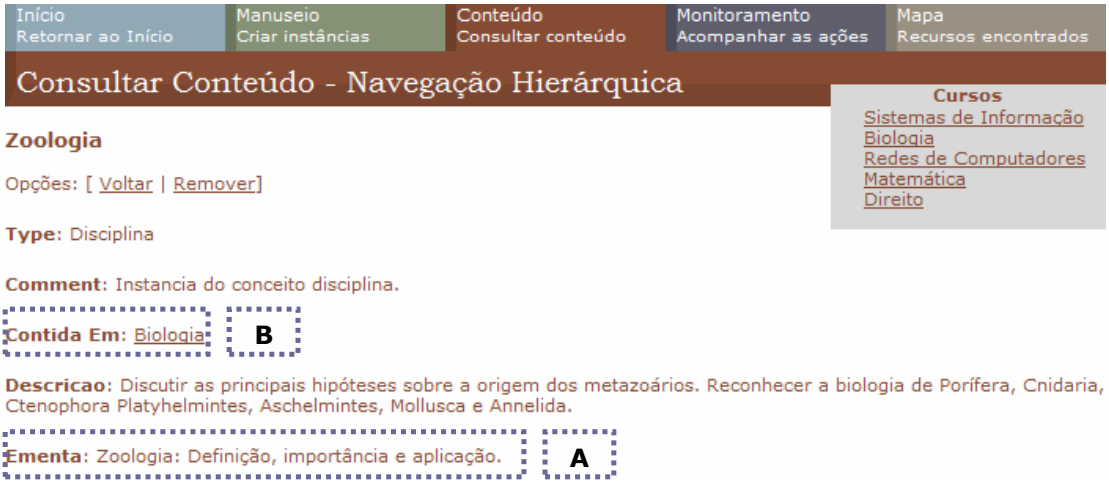

Figura 22 Exploração dos recursos de uma disciplina.

Como pode ser observado, os recursos associados diretamente pela disciplina *Zoologia*, previamente selecionada para ser explorada, são apresentados em seqüência

através de uma interface hipermídia. Como exemplo de uma propriedade simples, pode ser observada a ementa, que inclui os termos "Zoologia: Definição, importância e aplicação" (Figura 22 – A). Como exemplo de uma relação que parte do elemento exibido, pode ser citada a propriedade "*contidaEm*", cujo valor a associa ao curso de Biologia, fornecendo um *link* para explorá-lo (Figura 22 – B).

A idéia de fornecer uma forma de acessar o recurso é que confere a navegabilidade através do conteúdo definido pela ontologia. Através do método especificado a capacidade de realizar uma navegação é possível apenas partindo das propriedades e relacionamentos que pertencem ao mesmo objeto. Assim, são exibidas as propriedades de uma instância, como nome, descrição, valor etc., e as relações que partem do próprio conceito, como "possuiElemento", "pertenceA" etc. Esta característica impede que a navegação seja realizada pelos elementos que se relacionam ao recurso trabalho, já que estes não estão representados dentro do próprio objeto.

Para permitir a navegação de um conceito com maior abstração para um conceito com menor abstração, ou vice-versa, é preciso permitir que os elementos "contidos" relacionem os elementos que o "contêm" e a relação inversa. No caso de uma disciplina, é preciso dizer quais conteúdos ela contêm. Da mesma forma, em um conteúdo, é interessante apresentar em quais disciplinas ele está contido, para dar ao usuário a opção de realizar diferentes percursos para chegar ao mesmo conceito, desde que eles existam.

Ao procurar conferir ao protótipo a navegação nos dois sentidos de abstração (maior ou menor), foi exigida a implementação de um método que buscasse as referências de outras instâncias àquela trabalhada no momento. Na Figura 23 é exibida a forma de consulta realizada.

```
1.  String queryString =
     " SELECT ?Elemento, ?range, ?domain " +
     " WHERE  (?Elemento, ?pred, ?valor ) " +
     "(?pred, <http://www.w3.org/2000/01/rdf-schema#range>, ?range )"+
     "(?pred, <http://www.w3.org/2000/01/rdf-schema#domain>, ?domain )
";

2. QueryResults results = consultar(queryString);

3. Hashtable ht = new Hashtable();
4. Hashtable controle = new Hashtable();

5. for (Iterator iter = results; iter.hasNext(); explosao = "")
6. {
7.    ResultBinding res = (ResultBinding)iter.next();
8.    Resource elemento = (Resource)res.get("Elemento");

9.    Resource l = (Resource)res.get("range");
10.   Resource li = (Resource)res.get("domain");

11.   if( estaContido(elemento, strElemento) && (
!controle.containsKey(elemento.getLocalName() ) ) )
12.   {
13.   // Adiciona uma referência do elemento à HashTable (Figura 21)
14.   }
15.}
```

Figura 23. Consulta por elementos que tenham uma relação com a instância trabalhada.

A consulta por elementos foi realizada inicialmente por uma consulta na linguagem RDQL (Figura 23). Na consulta, são procurados os sujeitos que mantêm como predicado uma relação, ou seja, mantenham um *Range* e um *Domain*. Para cada elemento encontrado, são recuperados os valores de *Elemento*, *Range* e *Domain* e, então, é verificado se o valor de *Range* representa a URI da instância trabalhada, bem como se o elemento já não foi registrado anteriormente para aquela relação. Caso as condições sejam satisfeitas, o valor de *Domain* é guardado como forma de dizer que existe um elemento que está relacionado à instância trabalhada.

A adição de um elemento à lista que mantem os recursos que referenciam a instância é realizada por meio de uma *HashTable*, que representa a implementação do Java para o tipo abstrato de dados Tabela Hash. Por meio deste recurso, os valores são adicionados à tabela e agrupados pelo tipo de relação de cada elemento que faz referência à instância. A Figura 24 apresenta a forma de armazenamento destes valores.

```
1.  controle.put(elemento.getLocalName(), "0");

2.  String tipo = getType(elemento);
3.  ArrayList elementos;

4.  String = "<li><a href=\"./?CID=" + elemento.getLocalName() + \">"
+ elemento.getLocalName() + "</a></li>\n";

5.  if(!ht.containsKey(tipo))
6.  {
7.     elementos = new ArrayList();
8.  }
9.  else
10. {
11.    elementos = (ArrayList) ht.get(tipo);
12. }

13. elementos.add(link);
14. ht.put(tipo, elementos);
```

Figura 24. Classificação dos valores pela relação com a instância.

Conforme é apresentado na Figura 24, cada elemento tem sua URI adicionada como uma chave da *HashTable* de controle de ocorrência e então é recuperada a informação sobre a propriedade *RDFType* do recurso recuperado em *elemento*. Na Linha 4 (Figura 24) é composta a estrutura de *link* para o elemento encontrado, mantendo a possibilidade de realizar uma navegação e então é verificada a existência do tipo da propriedade na *HashTable*. Caso não exista, uma nova lista é criada para adição dos elementos que apresentarem o mesmo tipo. No caso de já existir, a lista anteriormente criada, armazenada pela chave com valor igual ao seu tipo, é recuperada para o novo *link* seja adicionado.

A organização destes *links* estabelece a possibilidade de melhor visualizar as referências aos elementos que, sem este tratamento, poderiam aparecer apenas como um *link* para um recurso, sem demonstrar o tipo de relação que há entre este recurso e a instância detalhada. A recuperação dos elementos da *HashTable* para o envio das informações à página responsável pela apresentação é mostrada na Figura 25.

```
1.    ArrayList elementos;
2.    String chave;
3.    for(Enumeration e = ht.keys(); e.hasMoreElements();)
4.    {
5.       chave = e.nextElement().toString();
6.       retorno.append("<br/><br/>");
7.       retorno.append(chave);
8.       elementos = (ArrayList) ht.get(chave);
9.       for(int indice = 0; indice < elementos.size(); indice++)
10.      {
11.         retorno.append(elementos.get(indice));
12.      }
13.    }
```

Figura 25. Recuperação e organização das propriedades por uma *HashTable*.

Ao percorrer todos as chaves da Tabela *Hash*, são recuperados os valores dos nomes das propriedades que permitem acessar o conjunto de itens relacionados a ela (Figura 25). Assim, para a apresentação dos valores, primeiro é inserida a chave dentro da *String* de retorno, contextualizando os itens, e então cada elemento é adicionado, montando uma apresentação aninhada. Na Figura 26 é apresentado o resultado da exploração dos recursos que referenciam a instância explorada.

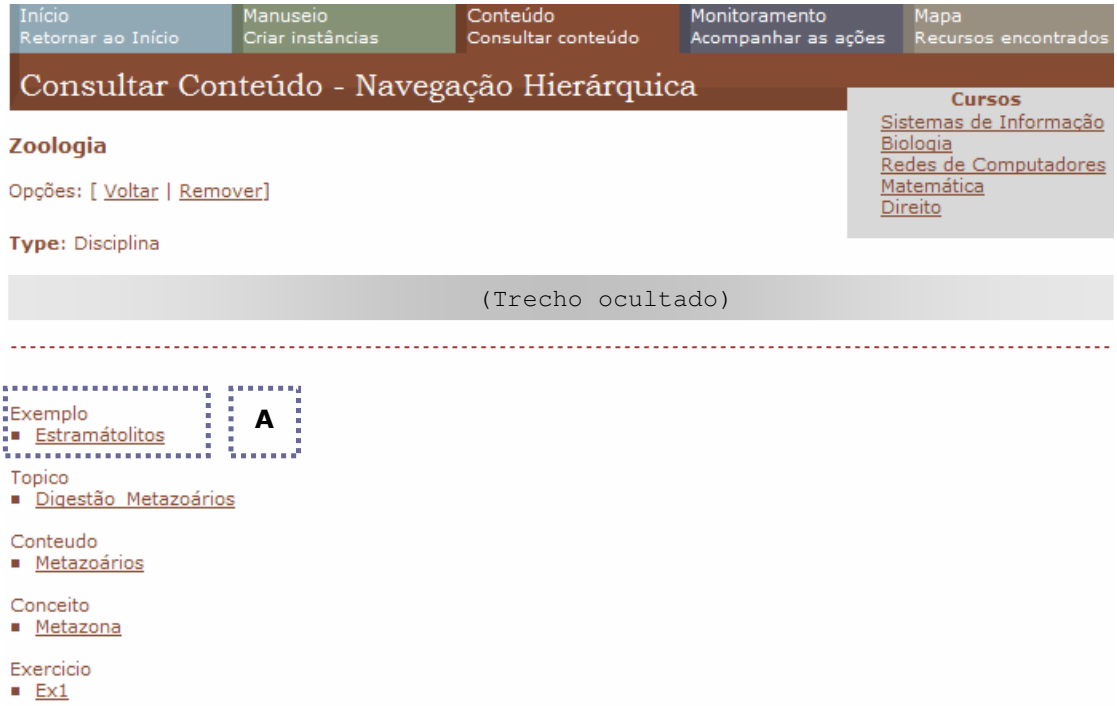

Figura 26. Listagem dos elementos que refererenciam a instância explorada.

Como pode ser acompanhado pela Figura 26, após apresentar a lista de propriedades diretamente associadas pela disciplina, é realizada a apresentação dos itens que se relacionam à disciplina. Para o caso apresentado na Figura 26, tem-se um conceito *Exemplo* que está relacionado à disciplina explorada (A). Dessa forma, é apresentado o nome do conceito envolvido e as instâncias deste conceito que contêm uma relação com disciplina.

4.3.5 Eliminação de uma instância

Dadas as possiblidades de consultar e inserir conteúdo na ontologia utilizada para a definição do domínio, foi desenvolvida a funcionalidade que permite ao usuário remover as instâncias desejadas. A forma de remoção de um recurso é demonstrada na Figura 27.

```
1. public static boolean remover(String uri)
2. {
3.    OntModel modelo = getModelo();
4.    boolean excluido = false;
5.    try
6.    {
7.        try
8.        {
9.            modelo.enterCriticalSection(Lock.WRITE);
10.             Individual instancia = modelo.getIndividual(namespace +
uri);
11.            instancia.remove();
12.            salvarModelo();
13.            excluido = true;
14.        }
15.        catch (Exception ex)
16.        {
17.            excluido = false;
18.        }
19.    }
20.    finally
21.    {
22.        modelo.leaveCriticalSection();
23.    }
24.    return excluido;
25. }
```

Figura 27. Remoção de uma instância da ontologia.

O método especificado para realizar a tarefa de remoção tem como requisito apenas a passagem de um parâmetro que indique a URI do elemento a ser excluído (Figura 27, Linha 1). Após ter a certeza de que o modelo está carregado em memória, especifica a entrada em uma área de seção crítica e carrega o recurso a ser apagado através de sua URI. A ação de remover o recurso pode ser realizada apenas com a chamada do método *remove*, a partir da instância que constitui uma classe *Individual*. Assim que o método é excluído, o elemento deixa de existir dentro do modelo armazenado em memória e, dessa forma, faz-se necessária a chamada do método *salvar*, que aplica as alterações ao arquivo que contém o modelo. Com a finalização do processo, é especificada a saída da região crítica.

4.3.6 Monitoramento das ações sobre a ontologia

Dentro da área de Hipermídia Adaptativa, capturar as ações dos usuários é um processo fundamental para a compreensão de suas ações e que auxiliam a trabalhar na construção de seu perfil. Com o intuito de realizar um acompanhamento inicial das ações realizadas sobre

a ontologia trabalhada, foi construída uma classe responsável pelo acompanhamento dos recursos inseridos ou excluídos no modelo. A classe, denominada *OntologyListener*, consiste na implementação da interface *ModelChangedListener*, fornecida pela API Jena. Na Figura 28 são exibidos os métodos definidos pela interface *ModelChangedListener*.

```
1.  public class OntologyListener implements ModelChangedListener
2.  {
3.     public void addedStatement( Statement s ) {}
4.     public void addedStatements( Statement [] statements ){}
5.     public void addedStatements( List statements ){}
6.     public void addedStatements( StmtIterator statements ){}
7.     public void addedStatements( Model m ){}
8.     public void removedStatement( Statement s ){}
9.     public void removedStatements( Statement [] statements ) {}
10.    public void removedStatements( List statements ){}
11.    public void removedStatements( StmtIterator statements ){}
12.    public void removedStatements( Model m ) {}
13.    public void notifyEvent(Model m, java.lang.Object event) {}
14. }
```

Figura 28. Conjunto de métodos definidos pela interface *ModelChangedListener*.

O conjunto de métodos disponíveis para implementação pela interface *ModelChangedListener* (Figura 28) incluem 3 operações básicas: a adição e a remoção de sentenças e a contextualização sobre os eventos disparados, mostrando o modelo que sofreu a alteração e as informações sobre o evento, se é o término ou o início. A principal diferença entre as diferentes sobrecargas de métodos existentes está na quantidade de elementos que sofreram uma ação. Ao capturar as informações de uma inserção com múltiplas sentenças, é observada a ação de carregar um modelo especificado em um arquivo para a memória. Para as ações realmente executadas por um usuário, são empregados os métodos que utilizam apenas uma sentença como parâmetro.

Ao reconhecer a forma de trabalho dos métodos implementados pela interface, foi elaborado um modelo OWL para o armazenamento das informações obtidas por ações de usuários no protótipo. Dessa forma, chegou-se a elaboração de um modelo contendo a seguinte especificação:

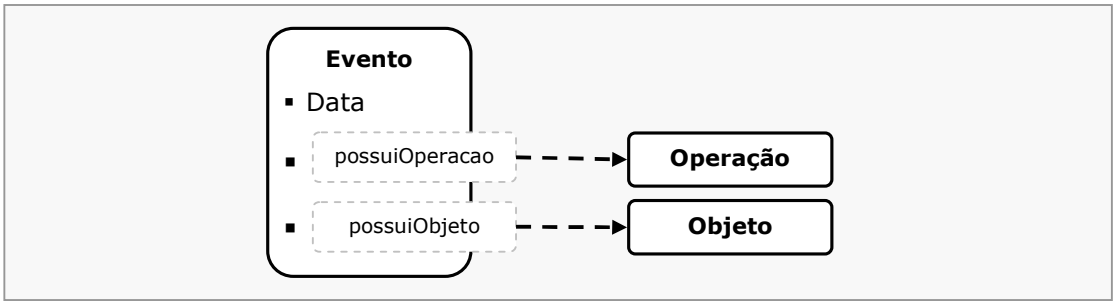

Figura 29. Modelo de Ontologia criado para manter o histórico de eventos.

Como uma forma ainda incipiente de manter os histórico das informações, para cada ação realizada na ontologia é criada uma instância do conceito Evento, definindo uma data, a operação relacionada e então o objeto alvo da modificação (Figura 29). Na operação, as instâncias Adição e Remoção já foram criadas e são referenciadas no nomento da instanciação de um evento. O objeto mantém uma referência ao objeto manipulado, o que permite que os conceitos das ontologias trabalhadas tenham seu histórico de manipulações armazenado.

As ações de criação das instâncias no documento de histórico é realizada pela classe *OntologyHistory*, que utiliza os recursos implementados na superclasse *OntologyManagerBase.* Assim, o processo de inserção dos elementos pode ser realizado assim que uma ação é disparada dentro de *OntologyListener.* A Figura 30 apresenta o modelo de chamada dos métodos de inserção no histórico.

```
1. public void addedStatement( Statement s )
2. {
3.    OntologyHistory.adicionarElemento("Adicionar", getHoraAtual(),s);
4. }

5. public void removedStatement( Statement s )
6. {
7.    OntologyHistory.adicionarElemento("Remover", getHoraAtual(), s);
8.}
```

Figura 30. Chamada do métodos de inserção de um evento.

Assim que um dos métodos de inserção e remoção da classe *OntologyListener* são chamados, é realizada a inserção do evento no histórico, identificando a operação realizada, a hora de realização e o elemento que sofreu a ação (Figura 30). O resultado da inserção em histórico pode ser acompanhado na Figura 31.

```
1. <Evento rdf:about="http://www.ulbra-
to.br/ontologias/historico.owl#Evento28/10/2005 - 11:6:25:531">
2.   <operacao rdf:resource="http://www.ulbra-
to.br/ontologias/historico.owl#Remover"/>
3.    <objeto rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
4.    <hora>28/10/2005 - 11:6:25:531</hora>
5.    <objeto rdf:resource="http://www.owl-
ontologies.com/unnamed.owl#Introducao_Teoria_EstatÃ-stica"/>
6. </Evento>
```

Figura 31. *Evento* registrado em *Histórico*.

Para cada evento disparado, é realizada a inserção de uma instância de conceito em *Histórico*. Os eventos apresentam como identificação o prefixo "Evento" seguido da hora em que ele foi inserido (Figura 31, Linha 1). Conforme especificado em seu modelo, o evento mantém uma referência à instância que representa a operação realizada, que neste caso referencia a instância correspondente a *Remover* (Linha 2). Na propriedade seguinte é encontrada a hora em que a ação é realizada, contendo a especificação da data e da hora específica em que ocorreram.

4.3.7 Interface

Após implementar os recursos destinados à manipulação das ontologias e suas diferentes formas de consultar elementos, foi elaborada a interface do protótipo, baseando a divisão de suas funcionalidades em quatro grupos distintos de página, além da inicial, que não apresenta uma funcionalidade em uso. A Figura 32 apresenta o *layout* da interface, assim como as divisões criadas para os grupos de funcionalidades.

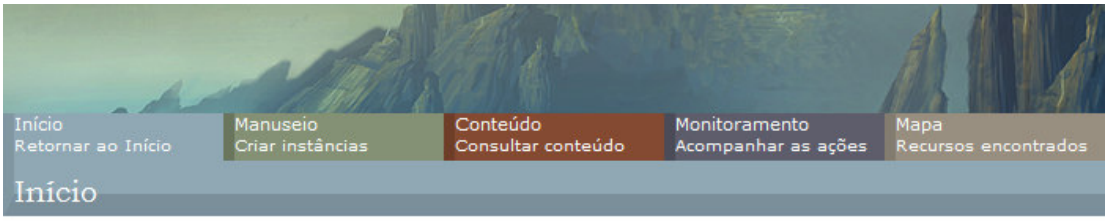

Figura 32. Apresentação da interface com as divisões por grupos de funcionalidades.

Conforme apresenta a Figura 32, os grupos de páginas definidos consistem em: *gerenciar*, destinado às inserções de conteúdos nas ontologias; *conteúdo*, que permite a navegação pelos conceitos da ontologia, dado o ponto de partida da seleção de um dos cursos; *monitorar*, que apresenta o histórico dos conceitos manipulados; e *mapa*, que apresenta os conceitos trabalhados na ontologia e permite uma forma alternativa de chegar

às instâncias destes conceitos.

1º Grupo de Funcionalidades – Conteúdo

O grupo de funcionalidades existente em *Conteúdo* fornece a possibilidade de navegar pelo conteúdo oferecido pelo protótipo a partir do seu conceito mais geral, que constiste no "curso". Assim que a página de apresentação é acessada, são exibidos os cursos registrados na ontologia do domínio, que constituem um *link* para a exibição de suas características e dos conceitos relacionados (Figura 33).

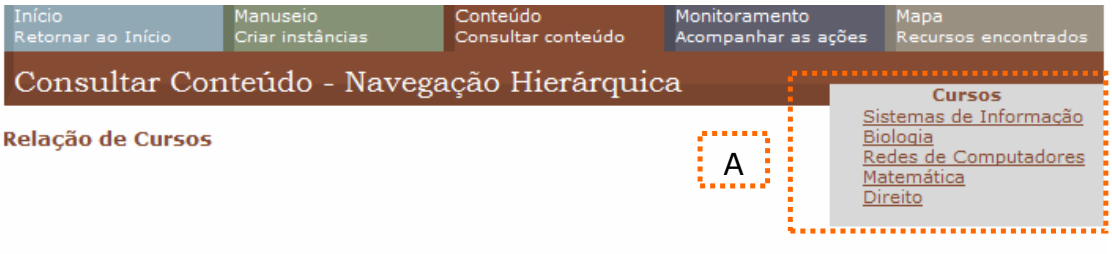

Figura 33. Primeiro passo para o processo de navegação pelos conceitos.

Conforme apresenta a Figura 33, no momento em que a página de conteúdo é acessada, é oferecido como ponto de partida a lista dos cursos presentes na ontologia do domíno (A). Isto permite que um dos cursos apresentados seja clicado e que, então, os conceitos aninhados ao curso sejam explorados.

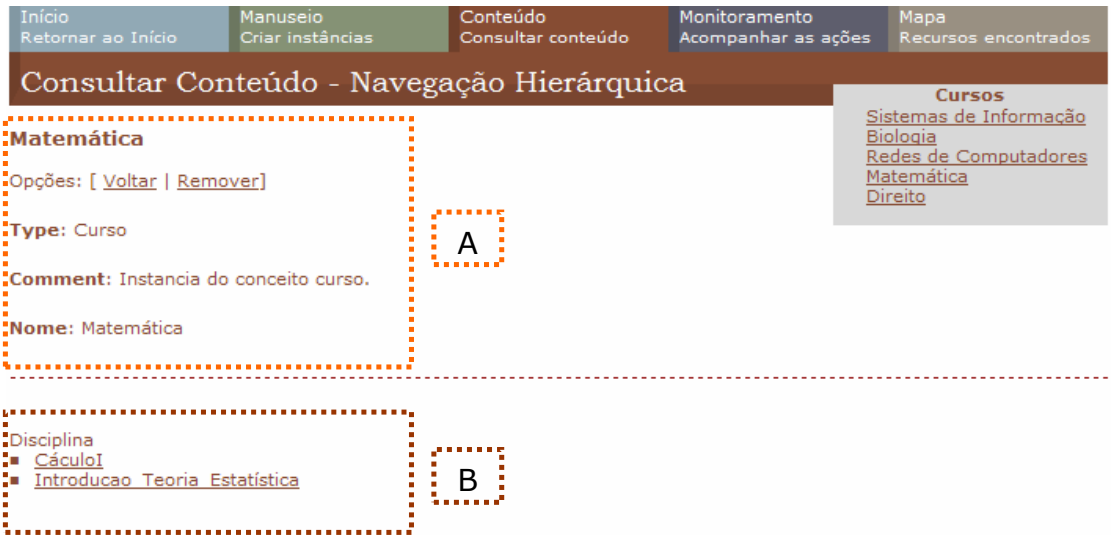

Figura 34. Exploração do conteúdo de um curso.

Ao solicitar o detalhamento de um curso, é exibido o conjunto de propriedades e relacionamentos diretamente especificado pela instância (Figura 34 – A) e logo abaixo o conjunto de relacionamentos que partem de outros conceitos para o curso (Figura 34 – B).

Os primeiros recursos listados são apresentados de acordo com a ordem de leitura de seus elementos pela API Jena, assim, não há uma ordem especificada para a apresentação das propriedades dos recursos exibidos (A). Ainda que também não apresentem uma ordem de ocorrência em sua listagem, os recursos definidos em "B" aparecem de forma contextualizada, contudo, sem respeitar a alguma característica de ordem alfabética.

Ao recurso que é explorado na tela de apresentação, é dada a possibilidade de exclusão, conforme o item "*Remover*", presente em "*Opções"* (Figura 34 - A). Esta opção foi fornecida para que seja possível realizar a ação de deleção de um item dentro do protótipo, como forma de explorar os recursos de manipulação de uma ontologia. Contudo, em um sistema que não esteja em fase de "protótipo" esta opção requer um tratamento diferenciado, como a exigência de que o usuário tenha um nível de permissão capaz de modificar o conteúdo definido.

Para um recurso apresentado que constitua mais do que um valor, ou seja, que possua, também, uma ligação entre um curso e um outro elemento, é dada a possibilidade de explorar seu conteúdo. Dessa forma, ao clicar no recurso *CálculoI,* por exemplo, a página é atualizada para que as características dessa disciplina sejam descritas. A Figura 35 exibe o resultado da apresentação dos recursos da disciplina.

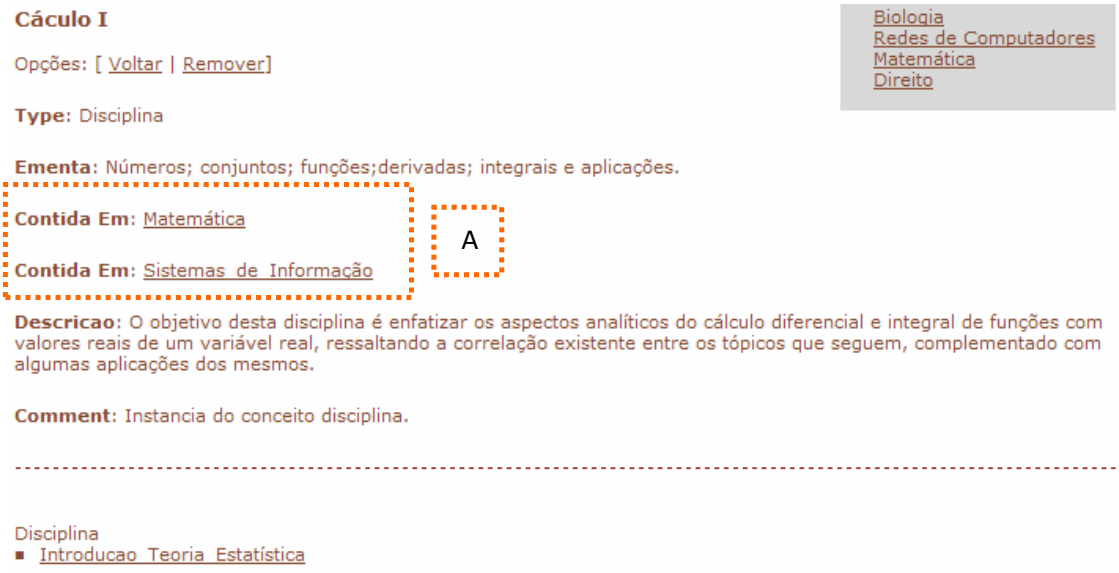

Figura 35. Exploração dos recursos de uma disciplina.

Após solicitar a exploração dos recursos de uma disciplina, a estrutura da página é mantida, porém, são listados os recursos contidos ou relacionados à disciplina selecionada (Figura 35). Um ponto que deve ser observado é a possibilidade de continuar o processo de navegação com o aumento do nível de detalhamento dos recursos ou, ainda, retroceder esta

ação, partindo do conceito mais específico para o mais geral. No caso exemplificado da Figura 35 (A), chegou-se a disciplina *CálculoI* através da seleção do curso de *Matemática* e então a seleção da disciplina mencionada. Ao perceber que a disciplina também está contida no curso de *Sistemas de Informação*, é possível acessar o curso como forma de analisar as características existentes no mesmo. Isto permite que num processo de procura por materiais relacionados a uma disciplina, por exemplo, seja possível encontrar o conteúdo buscado em alternativas diferentes ao curso inicialmente esperado, como em cursos que estejam presentes na área de Ciências Exatas.

2º Grupo de Funcionalidades – Gerenciar

O grupo de funcionalidades referentes ao contexto "Gerenciar" permite que sejam criadas as instâncias de todos os conceitos presentes na ontologia, além de relacioná-las às demais instâncias cujas relações estão previstas. A geração do formulário é o resultado do processo de descoberta dos recursos e criação dos campos apresentados na Seção 4.3.2, que permite gerar um formulário e posteriormente criar a instância para qualquer ontologia definida para o protótipo. A Figura 36 apresenta o formulário de criação das instâncias.

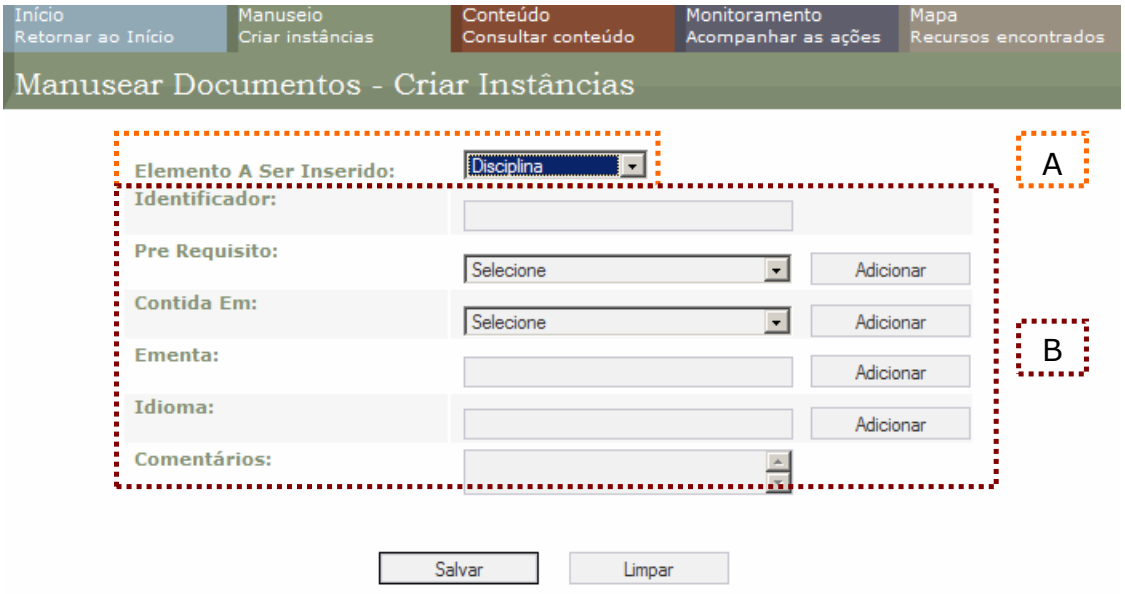

Figura 36. Tela de criação das instâncias dos conceitos presentes na ontologia.

O formulário de criação de uma instância tem como primeiro campo uma caixa de seleção para o conceito a ser instanciado (Figura 36 - A). Dessa forma, são listados todos os conceitos disponíveis no domínio, para que um deles seja selecionado e então o

formulário correspondente seja gerado. Ao selecionar *Disciplina* como o conceito a ser instanciado, é gerado o formulário especificado na Figura 36 – B. Como conseqüência são apresentados os campos de texto *Indentificador*, *Pré-Requisitos*, *Ementa*, *Idioma* e *Comentários;* além dos campos de seleção *Pré-Requisito* e *Contida Em*.

Uma funcionalidade implementada através de *scripts* contidos na página responsável pela apresentação é a multiplicação dos campos de acordo com as cardinalidades máxima e mínima apresentadas. Isto exigiu que, além da verificação da cardinalidade, fosse dada a possibilidade de atribuir múltiplas vezes uma determinada propriedade, além de permitir ao usuário remover a propriedade 'duplicada' caso seja necessário. A forma de realização das múltiplas atribuições de uma propriedade é apresentada na Figura 37.

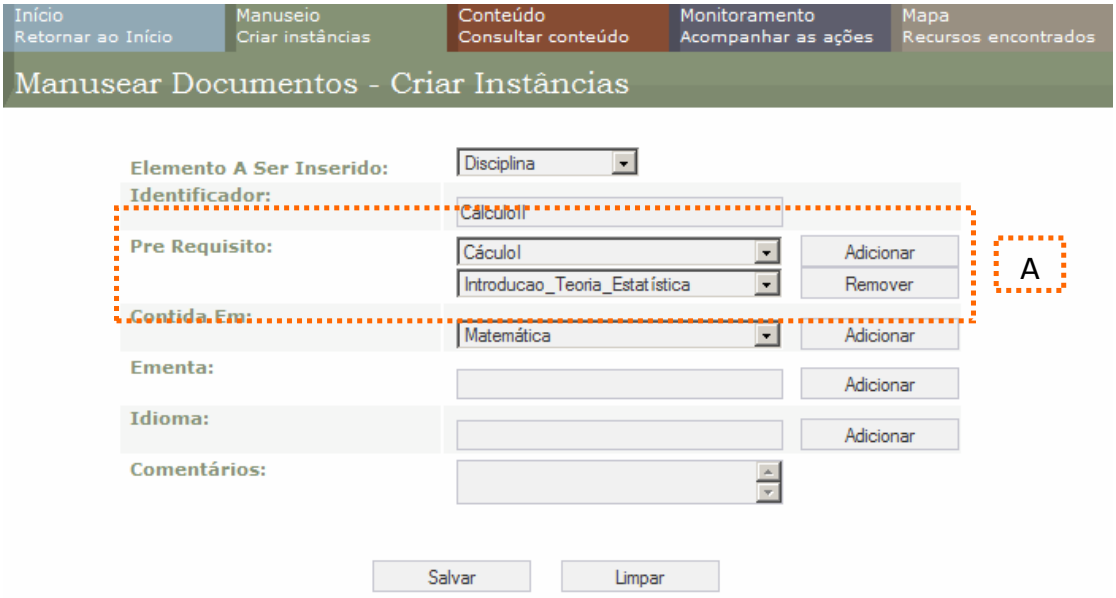

Figura 37. Duplicação dos campos de um formulário.

Ao criar uma instância como a do conceito Disciplina, é possível que sejam atribuídas mais de uma mesma propriedade, com valores distintos, para a sua instância. Assim, para cada campo que tenha uma cardinalidade superior a 1 (uma) ocorrência, é habilitado o botão de Adicionar novo campo para esta propriedade (Figura 37 – A). No momento em que o botão é clicado, o *script* criado para a duplicação dos campos verifica a propriedade relacionada e então adiciona uma nova versão deste campo, assim como um botão que permita removê-lo posteriormente.

A possibilidade de duplicação dos campos não resultou em qualquer diferença no processo de captura do formulário e criação das instâncias, que foi realizado de forma genérica. Contudo, isto foi possível mantendo a nomenclatura dos campos no nomento de

sua ‘clonagem’, porém, com a adição de um valor que não interferisse na decomposição do nome e análise de seus valores.

3º Grupo de Funcionalidades – Ações de Monitoramento

As funcionalidades que permitem realizar o monitoramento do Protótipo representam, em um caráter experimental, uma forma de visualizar as ações realizadas sobre o modelo da ontologia. O recurso foi trabalhado para fornecer uma visualização simples dos eventos ocorridos, sem que fossem apriomoradas as formas de exibição dos eventos ou que os recursos envolvidos nas ações fossem completamente armazenados em um modelo com função de “limbo”. A Figura 38 permite observar a página de apresentação dos eventos.

Figura 38. Exibição dos últimos eventos ocorridos.

A página para o acompanhamento dos últimos eventos ocorridos aplica a consulta por recursos para exibir a lista das últimas ações realizadas sobre a ontologia (Figura 38). O resultado da busca exibe a identificação do evento, a classe que ele representa (*Evento*), e as informações sobre a ação, como o tipo de ação (*Adicionar*/*Remover*), a hora de realização e o objeto afetado.

Cabe ressaltar que a adoção de uma forma de monitoramento com o registro das ações em uma ontologia deve ser condicionada a uma análise de desempenho e ganho de

expressividade nas consultas. Dessa forma, devem ser analisadas alternativas, como manter o histórico em um arquivo de texto, como no formado CSV (*Comma Separeted Values*), ou em formatos alternativos como XML, OWL ou, ainda, em uma base de dados. Contudo, o uso de um documento OWL pode ser visto como a forma de adicionar um conjunto expressivo de informações sobre as ações do usuário dentro do domínio, assim sua utilização deve ser analisada já considerando essa característica.

4º Grupo de Funcionalidades – Mapa do Protótipo

O grupo de funcionalidades referentes ao "Mapa" do Protótipo constitui uma forma alternativa de acesso ao seu conteúdo. Ao invés de fornecer um caminho com início do conceito mais geral, considerado como o *Curso*, apresenta ao usuário a lista de conceitos existentes no domínio do protótipo, que pode facilitar o processo de descoberta de recursos e o reconhecimento do domínio definido. O resultado da criação do "Mapa" é apresentado na Figura 39.

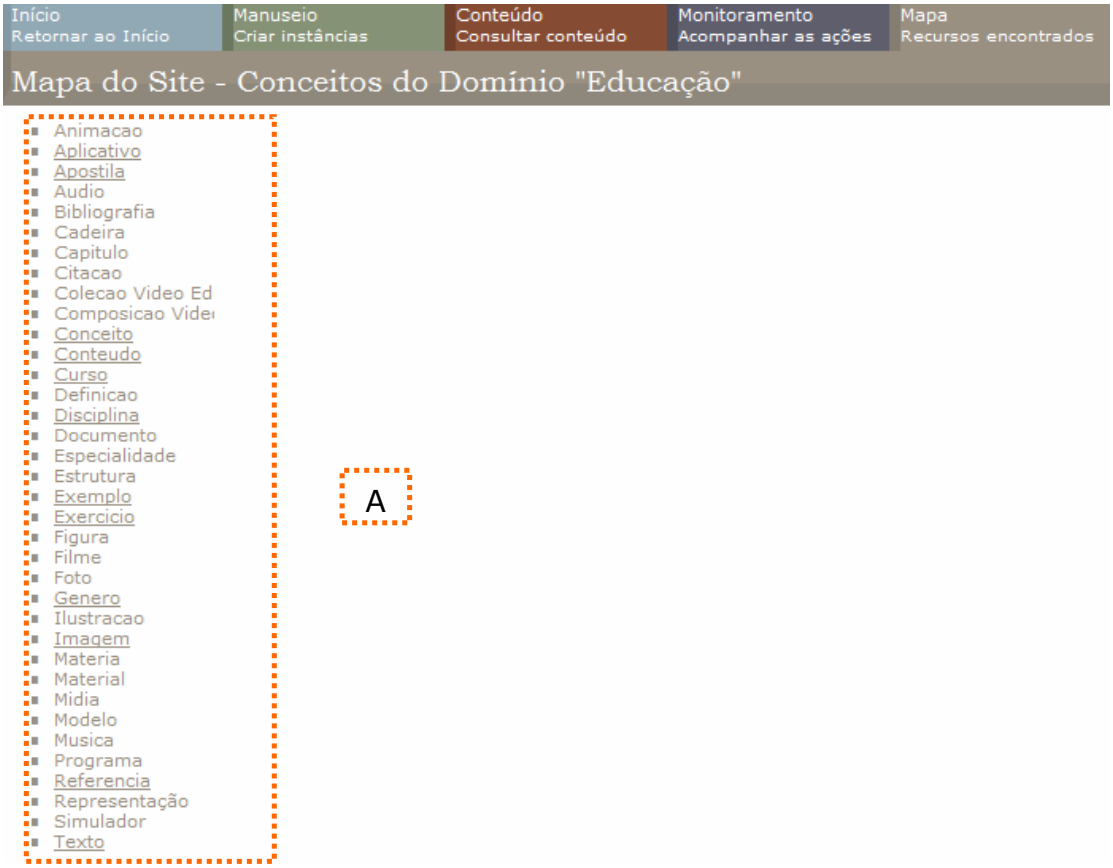

Figura 39. Lista de conceitos do protótipo como representação do Mapa do Protótipo.

O Mapa do protótipo realiza uma listagem de todos os conceitos presentes na

ontologia do domínio, representada parcialmente na Figura 39. Além de demonstrar o conceito é permitido ao usuário clicar sobre um dos conceitos e então conferir a lista de instâncias existentes para este conceito. O resultado desta ação pode ser acompanhado na Figura 40.

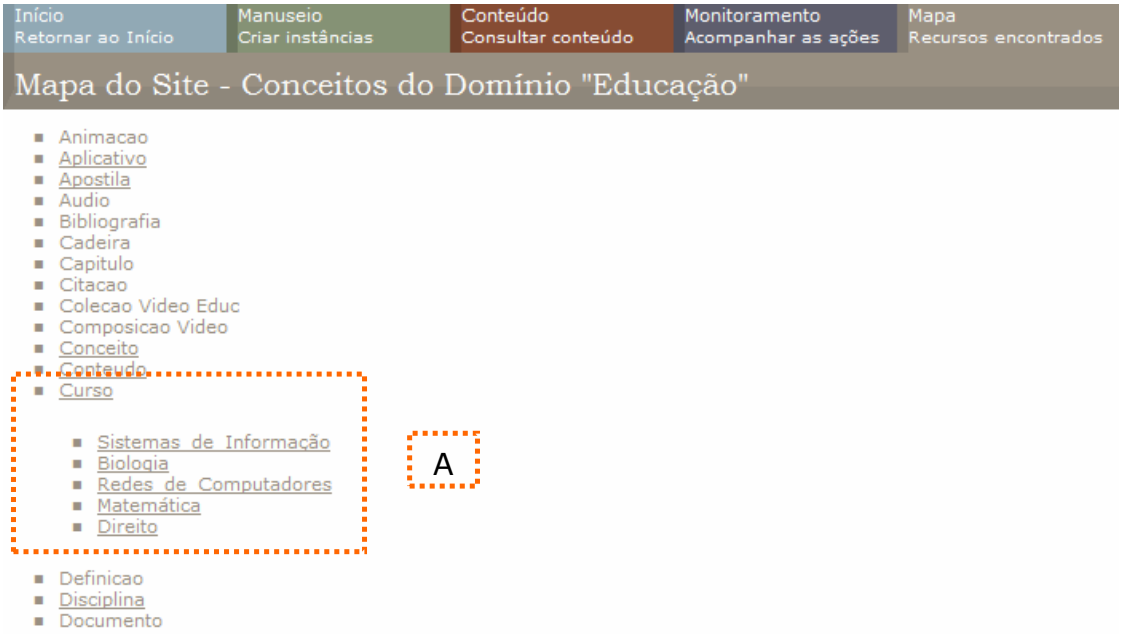

Figura 40. Expansão dos sub-items do conceito Curso dentro do Mapa do Protótipo.

Assim que um dos conceitos é clicado, a lista de instâncias deste conceito é apresentada em uma estrutura hierárquica (Figura 40 – A). No caso de um curso, por exemplo, ao clicar sobre seu nome são exibidos os itens "Sistemas de Informação", "Biologia" e "Matemática", que consistem nas instâncias de *Curso* registradas na ontologia do domínio do protótipo. Após esta ação é possível retrair a lista e expandir outros conceitos, ou clicar sobre um dos sub-itens e obter os detalhes de suas instâncias, conforme a apresentação gerada no grupo de funcionalidades de Conteúdo.

Com o término do desenvolvimento do protótipo, puderam ser estabelecidos recursos que vão desde a criação das instâncias até a consulta de suas informações e ao acompanhamento das ações realizadas sobre a ontologia. Estes recursos conferem ao protótipo a autonomia na inserção de seus conteúdos e na sua apresentação, que permitem sua utilização como um *website* funcional. Na próxima seção serão apresentadas as considerações finais sobre o trabalho.

5 CONCLUSÃO

O trabalho teve como objetivo montar um protótipo de um sistema hipermídia no qual a representação e o escopo do domínio estivessem representados por uma ontologia. Correlacionado à construção deste protótipo, foram estudados os aspectos relevantes na construção de SHA, sobretudo em sua aplicação ao ensino, e sobre ontologias, como forma de representar um contexto. Considerando os objetivos definidos, o propósito geral do trabalho consistiu na verificação dos recursos empregados no manuseio de ontologias que podem ser úteis na construção de um sistema de hipermídia adaptativo.

A construção do protótipo envolveu os requisitos funcionais de registro de novas instâncias de ontologias. Assim, a partir de um modelo aplicado ao ensino, foi possível permitir a remoção de conceitos, além de fornecer uma forma de busca conveniente à interação através da Internet e da hierarquia dos conteúdos apresentados pelo domínio. Para a construção destes recursos foram utilizados a API Jena, que dá suporte ao uso de consultas pelas linguagens RDQL e SPARQ QL, ambas úteis na elaboração de consultas que auxiliam a descoberta de recursos de ontologias. Como linguagem de implementação foi utilizada a linguagem Java, que forneceu a possibilidade de montar a arquitetura de classes do protótipo seguindo conceitos de orientação à objetos e, posteriormente, o emprego destes objetos através de páginas especificadas sob a tecnologia JSP. O uso do JSP, por sua vez, permitiu a criação de interfaces para a apresentação do conteúdo aos usuários e a solicitação de métodos às classes implementadas. Contudo, o uso do JSP foi apoiado pela coleção de *tags* JSTL, que facilitou o processo de organização do conteúdo dentro das páginas JSP.

Com o propósito de manter o domínio do protótipo através de uma ontologia, foi utilizado o modelo criado por Carneiro e Brito (2005). O modelo tem como foco a área do ensino e, dentro desta, especifica as diferentes relações que podem ser construídas para a definição de conteúdos educacionais. Tais relações envolveram desde relacionamentos

simples, estabelecidos por propriedades, como relações complexas, construídas a partir da definição de axiomas, que habilitaram a elaboração de resultados de buscas que, inicialmente, poderiam ser obtidos apenas com sua definição explícita na consulta.

O desenvolvimento do sistema foi realizado com a abordagem de construção das funcionalidades com a separação entre classes destinadas à ontologia fornecida por Carneiro e Brito (2005) e aquelas que poderiam ser empregadas para quaisquer ontologias em OWL. Esta forma de trabalho resultou na construção de classes genéricas mais complexas, que agregaram a grande maioria das funcionalidades e que obtiveram sucesso na aplicação de diferentes ontologias. As classes criadas com propósito específico apresentaram uma complexidade inferior, dada a utilização dos recursos fornecidos pelas classes genéricas. Assim, obteve-se como resultado a possibilidade de empregar outras ontologias e, para a sua aplicação, fornecer a implementação de uma classe que sirva para informar o ponto de partida das consultas. No caso da ontologia aplicada ao "ensino", a classe *EducationalOntologyManager* oferece, na falta de definição de um elemento, as instâncias do conceito Curso.

As funcionalidades de pesquisa, inserção e remoção foram implementadas de modo independente do domínio fornecido e, dessa forma, pode ser percebida a possibilidade de manter o escopo de um *website* apenas pelo gerenciamento de ontologias. Uma forma reconhecida de permitir uma implementação inteiramente independente da ontologia aplicada pode ser a definição de uma ontologia que represente as características da página, como a URI empregada para a importação dos elementos, o caminho das ontologias disponíveis, os elementos que servem como base para as consultas iniciais etc.

Um ponto importante na utilização das ontologias é a possibilidade de manter a integridade do modelo mesmo no caso do uso concorrente do protótipo e da realização de alterações por diferentes usuários. Isto foi possível pela utilização do recurso que permite especificar uma região crítica de acesso ao modelo para a escrita ou a leitura.

Ao considerar a necessidade que sistemas hipermídia adaptativos têm de acompanhar as ações de seus usuários dentro do domínio, foi realizado um acompanhamento inicial das atividades executadas sobre a ontologia do domínio do protótipo. Esta funcionalidade emprega uma segunda ontologia para relacionar a operação realizada, o objeto envolvido e a hora de realização, o que pode permitir uma análise das transformações realizadas no modelo.

O resultado do desenvolvimento da navegação pelo protótipo pode ser visto de forma satisfatória, considerando que foi permitido, aos usuários, uma navegação pelos

conceitos mais gerais, como curso e disciplina, e então acessar o conjunto de informações contidas nestes conceitos. Assim, é possível aumentar a granularidade dos conceitos exibidos e partir para elementos cada vez mais específicos. Este formato de navegação hierárquico foi montado com a possibilidade de que seja realizado o sentido inverso da abstração dos conceitos visualizados, permitindo, por exemplo, que de um tópico seja alcançada a disciplina que o contém e, desta disciplina, o curso no qual ela está inserida.

O trabalho realizado no protótipo pode ser visto como os primeiros passos na elaboração dos 3 componentes de um sistema hipermídia adaptativo: Modelo do Usuário, Modelo do Domínio e Modelo de Adaptação. Estes três componentes, relacionados em (WU et al. 2001), e citados no processo de adaptação de Brusilovsky (1996) e Palazzo (2002), puderam ser trabalhados de alguma forma, ainda que incipiente, durante a elaboração do protótipo.

O componente mais visível utilizado é o Modelo do Domínio, que consiste no uso da ontologia para a apresentação do conteúdo apresentado. Através deste recurso, é possível inserir novos componentes, com uma estruturação expressiva e capaz de identificar recursos que, ao serem relacionados, formam um material com maior expressividade de conteúdo.

O Modelo do Usuário pode ser reconhecido no processo inicial de acompanhamento das informações trabalhadas na ontologia. Contudo, para a correta atribuição do termo "Modelo do Usuário", é preciso ampliar o universo descrito sobre o usuário e, para tanto, é possível trabalhar na definição de uma ontologia que represente as diferentes características, necessidades e realizações dos usuários, ou especificamente dos estudantes, que utilizarão o sistema.

Após tratar dos Modelos do Domíno e do Usuário, resta estabelecer a forma de sujeitar as informações do primeiro modelo às características registradas no segundo. Isto implica a criação de um Modelo de Adaptação, que representa a forma como as características do usuário irão influenciar na geração da interface. Para tanto, dentro dos recursos estudados, podem ser utilizados conjuntos de regras aplicados às consultas ou motores de inferência, disponíveis para aplicação sobre os modelos armazenados em memória pela API Jena.

O resultado obtido com a elaboração deste trabalho é a compreensão de que o uso de ontologias e seus diferentes recursos de consulta, inferência e criação de instâncias é adequado ao trabalho com Sistemas Hipermídia Adaptativos. Dessa forma, podem ser citados como trabalhos futuros a construção de um sistema que mantenha domínio,

características dos usuários e regras de adaptação definidos por ontologias. Contudo, é preciso realizar um estudo sobre formas de realização de inferências para que, assim, sejam obtidos resultados expressivos no entendimento das necessidades dos usuários e da criação de formas “inteligentes” de auxiliá-los.

6 REFERÊNCIAS BIBLIOGRÁFICAS

(BRUSILOVSKY, 1996) BRUSILOVSKY, P. "Methods and techniques of adaptive hypermedia". User Modeling and User-Adapted Interaction, vol. 6, n. 2-3, 1996. pg. 87-129.

(BRUSILOVSKY, 2001) BRUSILOVSKY, P. "Adaptive Hypermedia". User Modeling and User-Adapted Interaction, v11, 2001. pg. 87-110.

(BRUSILOVSKY, 2004) BRUSILOVSKY, P. "Adaptive navigation support: From adaptive hypermedia to the adaptive Web and beyond". **PsychNology** Journal, 2004 vol. 2, n. 1, pg. 7-23.

(CALVI e CRISTEA, 2002) CALVI L. & CRISTEA A., "Towards Generic Adaptive Systems: Analysis of a Case Study", AH 2002, Adaptive Hypermedia & Adaptive Web-Based Systems**, Lecture Notes in Computer Science** 2347, Springer, 79-89

(CANNATARO ET. AL, 2001) CANNATARO, M., CUZZOCREA, A., PUGLIESE, A. (2002). "XAHM: an adaptive hypermedia model based on XML". In **Proceedings of the 14th international Conference on Software Engineering and Knowledge Engineering** (Ischia, Italy, July 15 - 19, 2002). SEKE '02, vol. 27. ACM Press, New York, NY, 627-634.

(CARNEIRO e BRITO, 2005) CARNEIRO, Raquel Elias; BRITO, Parcilene Fernandes de. "Definição de uma Ontologia em OWL para Representação de Conteúdos Educacionais". *VII ENCONTRO DE ESTUDANTES DE INFORMÁTICA DO ESTADO DO TOCANTINS*, 2005, Palmas. **Anais** p. 111-120. Palmas: 2005.

(CRISTEA, 2005) CRISTEA, A. "Authoring of Adaptive Hypermedia". **Educational Technology & Society**, 8 (3), 2005. pg. 6-8.

(DARPA, 2000) DARPA "The DARPA Agent Markup Language Homepage". Disponível em: <http://www.daml.org/>, Novembro.

(DOURADO, 2004) DOURADO, da F. J. R. "Um Modelo Para Sistemas Tutores Inteligentes Adaptativos". 2004. 107p. Dissertação (mestrado em Ciência da Computação) - Universidade de Brasília Brasília.

(GRUBBER, 1999) GRUBBER, T. "***What is an Ontology?"***. Disponível em: <http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>, Novembro.

(HEWLETT-PACKARD, 2005) Hewlett-Packard (2005) Hewlett-Packard Development Company LP "Jena – A Semantic Web Framework for Java". Disponível em < http://jena.sourceforge.net/>, Novembro.

(KAVČIČ ET AL, 2002) KAVČIČ, A., PRIVOŠNIK M., MAROLT M., DIVJAK S. "Educational Hypermedia System ALICE: An Evaluation of Adaptive Features". Faculty of Computer and Information Science,University of Ljubljana, SLOVENIA.

(LANGLEY, 1999) LANGLEY, P. "User modeling in adaptive interfaces". In **Proceedings of the Seventh International Conference on User Modeling** (Banff, Canada). J. Kay, Ed. Springer-Verlag New York, Secaucus, NJ, 357-370.

(MCGUINNESS, 2003) MCGUINNESS, Deborah L. "Ontologies Come of Age". In Dieter Fensel, J im Hendler, Henry Lieberman, and Wolfgang Wahlster, editors. Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press, 2002.

(MCGUINNESS e HARMELEN 2004) MCGUINNESS, D. & HARMELEN, F. "OWL Web Ontology Language Overview" Disponível em: <http://www.w3.org/TR/2004/REC-owl-features-20040210/>, Novembro.

(OIL, 2000) **OIL** "The Ontology Inference Layer OIL" Disponível em: <http://www.ontoknowledge.org/oil/TR/oil.long.html>, Novembro.

(PALAZZO, 2000) PALAZZO, L. A. M. "Modelos Proativos para Hipermídia Adaptativa", Dissertação de Doutorado – UFRS (2000).

(PALAZZO, 2002) PALAZZO, L. A. M. "Sistemas de Hipermídia Adaptativa : Texto-base em formato html do minicurso apresentado em 18/19 de julho de 2002" na **XXI Jornada de Atualização em Informática** (JAI 2002). Florianópolis, SC, 2002, 38 pg.

(PARK e HUNTING, 2003) PARK, J. & S. Hunting, "XML topic maps: Creating and using topic maps for the web". Boston, MA: Addison-Wesley, 2003.

(RUSSEL e NORVING, 2004) RUSSEL, S. J. & NORVING, P. "Inteligência Artificial: tradução da 2ª ed. publicada por Prentice Hall". Rio de Janeiro : Elsevier, 2004.

(SUN, 2003) Sun Microsystems "JavaTM 2 SDK, Standard Edition Documentation". Disponível em: <http://java.sun.com/j2se/1.4.2/docs/api/>

(SUN, 2004) Sun Microsystems "JavaServer Pages Technology - Documentation" Disponível em: <http://java.sun.com/products/jsp/docs.html>

(SUN, 2005) Sun Microsystems "JavaServer Pages Standard Tag Library" Disponível em: <http://java.sun.com/products/jsp/jstl/reference/docs/index.html>

(W3C, 2001) **DAML+OIL** "DAML+OIL (March 2001) Reference Description" Disponível em: <http://www.w3.org/TR/daml+oil-reference >, Novembro

(W3C, 2004) **OWL** "OWL Web Ontology Language Reference" Disponível em: <http://www.w3.org/TR/owl-ref/>, Novembro

(W3C, 2005) **SPARQ QL** “SPARQL Query Language for RDF”. Disponível em: <http://www.w3.org/TR/rdf-sparql-query/>, Novembro.

(WU, 2001) WU, H., de Kort, E., e De Bra, P. 2001. Design issues for general-purpose adaptive hypermedia systems. In **Proceedings of the Twelfth ACM Conference on Hypertext and Hypermedia** (Århus, none, Denmark, August 14 - 18, 2001). HYPERTEXT '01. ACM Press, New York, NY, 141-150.

(WU, 2002) WU, H. "A Reference Architecture for Adaptive Hypermedia Applications", Eindhoven: Technische Universiteit Eindhoven, 2002. Tese de Doutorado.