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

Um Modelo do Aluno Adaptativo para Sistemas na Web

por

ALESSANDRO BOEIRA DOS REIS

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

Profa. Rosa Maria Vicari
Orientadora

Porto Alegre, janeiro de 2001.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Reis, Alessandro Boeira dos
Um Modelo de Aluno Adaptativo para Sistemas na Web / Alessandro Boeira dos Reis. - Porto Alegre: UFRGS, 2001.

84 p:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2001. Orientador: Vicari, Rosa Maria.

1. Sistemas Tutores Inteligentes. 2. Modelo de Aluno. 3. Agentes. I. Vicari, Rosa Maria. II. Título.

“Mesmo as noites totalmente sem estrelas podem
anunciar a aurora de uma grande realização“.
Martin Luther King

Agradecimentos

Gostaria de fazer aqui um agradecimento a todos aqueles que de alguma forma tiveram uma contribuição nesta dissertação.

À minha orientadora Rosa, pela sua dedicação e cooperação de extrema importância.

Aos professores e colegas Adriana Soares Pereira, Francine Bica e Ricardo Silveira pela colaboração e disponibilidade em trocar idéias e me apoiar na realização deste trabalho.

A todos os amigos e colegas, pelo apoio, compreensão, descontração, carinho e amizade.

À minha família, que sempre esteve comigo me apoiando em todas as horas, obrigado pelo incentivo e carinho.

Sumário

Lista de Abreviaturas

API	Application Program Interface
ASP	Active Server Pages
BD	Banco de Dados
CALAT	Computer Aided Learning and Authoring environment for Tele-education
CGI	Common Gateway Interface
GIF	Graphics Interchange Format
EAD	Educação à Distância
ER	Entidade-Relacionamento
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IA	Inteligência Artificial
ITS	Sistemas Tutoriais Inteligentes
JDBC	Java Database Connectivity
JSP	Java Server Page
MPEG	Moving Picture Experts Group
MUD	Multiple Use Dimension
ODBC	Open Database Connectivity
RMI	Remote Method Invocation
SH	Sistemas Hipermídias
SQL	Structured Query Language
URL	Uniform Resource Locator
XML	Extensible Markup Language
WWW	Word Wide Web

Lista de Figuras

Lista de Tabelas

Resumo

Esta dissertação de mestrado está inserida no trabalho desenvolvido pelo grupo de pesquisa GIA/UFRGS e situa-se na área de Inteligência Artificial aplicada à Educação à Distância apresentando uma arquitetura distribuída no desenvolvimento de Ambientes de Ensino Inteligentes.

A *Web* facilita aos alunos encontrarem informações relevantes ao estudo que estão fazendo. Outra vantagem do ensino na *Web* é o indivíduo ser sabedor da existência da informação e de onde ela se localiza para que no momento adequado a acesse. Porém, o progresso que se poderá ter na área de informática aplicada à educação dependerá da adequação e qualidade do *software* educacional a ser utilizado. Por isso, temos que ter *software* de ensino inteligentes, que possam trazer esta qualidade e adequação conforme o aluno que está tentando adquirir um certo conhecimento.

O objetivo principal do presente trabalho se propõe a estudar as diferentes técnicas e mecanismos de se construir um modelo do aluno adaptativo na *Web*, aplicáveis em um cenário de educação à distância, a partir de um ambiente distribuído de ensino-aprendizagem inteligente baseado em uma arquitetura multiagentes que contempla o paradigma de ensino cooperativo (uma sociedade de agentes humanos e artificiais, que cooperam para alcançar um objetivo comum).

Além disso, o trabalho propõe o uso de estratégias de ensino segundo regras estipuladas pelo especialista. O agente responsável por este controle observa as modificações do comportamento do aprendiz durante a sua interação no ambiente, e seleciona uma estratégia de acordo com o desempenho do aprendiz.

Palavras-Chave: Ensino à Distância, Sistemas Tutorais Inteligentes (ITS), Modelo do Aluno.

TITLE: "A ADAPTATIVE STUDENT MODEL ON THE WEB SYSTEM"

Abstract

This master's degree dissertation was developed within the research group GIA/UFRGS, and is located in the field of Artificial Intelligence applied to distance learning, presenting a distributed architecture to develop intelligent teaching environments.

The Web make easier for students to find important information for the study hey are performing. Another advantage of learning on the Web is that one knows the information exist and where they are located in order to access it at the right moment. However, the advances in the area of learning applied computing will depend of the adaptation and quality of the educational software used. Thus, we need intelligent teaching software, which can bring us this quality and adaptation according to the student that is trying to acquire some knowledge.

The main goal of this work is to propose to study the different techniques and mechanisms for building an adaptive student model on the Web, that are applicable in a distance learning scenario, from an intelligent teaching-learning distributed environment based in a multi-agent architecture that contemplate the cooperative teaching paradigm (a society of human and artificial agent, which cooperate to reach a common goal).

In addition, this work proposes the use of teaching strategies according to rules given by the specialist. The agent responsible for this control observes the apprentice behavioral changes during his interaction with the environment and selects a strategy according to the apprentice performance.

Keywords: Distance Learning, Intelligent Tutorial Systems (ITS), Student Model

1 Introdução

A crescente utilização de inovações tecnológicas em escala planetária vem impondo transformações importantes em todos os setores sociais e produtivos das sociedades modernas. Redes de computadores interligados (Internet) interruptamente colocam à disposição de universidades e instituições de ensino um amplo espectro de serviços e aplicações que ainda não foram absorvidos e apreendidos na sua totalidade.

O novo paradigma tecnológico avança sobre duas questões fundamentais ao homem contemporâneo: o rompimento das dimensões tempo-espaço e a disponibilização de milhões de dados e informações através do uso de redes, que possibilitam a troca de informações e aprendizagem cooperativa, busca e recuperação em locais independentes de barreiras físicas, velocidade de propagação e disseminação do conhecimento existente em produção.

É sob os impactos destes novos paradigmas tecnológicos que se encontra o ensino. Escolas e educadores são os responsáveis pela inserção crítica destes novos paradigmas presentes hoje na sociedade e que contribuem para a determinação de um novo padrão de formação educacional e profissional.

O momento é de mudanças na educação e de constante avaliação destas mudanças. Novas práticas pedagógicas de ensino-aprendizagem demandam novas formas de relacionamento entre o educador-educando e novas medidas de avaliação na construção do conhecimento. A entrada de computadores nas escolas suscitou a adoção de novas posturas e práticas pedagógicas.

Um *software* educacional pode trazer também outras conseqüências pedagógicas desejáveis, tais como: individualização na aprendizagem, estímulo e motivação para o aluno, promoção da auto-estima no aluno e apresentação dos tópicos escolares de modo atrativo, criativo e integrado [GIR 95].

Uma das tecnologias que pode fazer uso destas afirmações é a *Web* (abreviatura para a WWW - *World Wide Web*), na qual os alunos podem navegar sobre as informações e facilmente adquirí-las em seu conhecimento.

Portanto, um ambiente de ensino na *Web*, denominado Educação à Distância (EAD), facilita aos alunos encontrarem estas informações relevantes ao estudo que está fazendo. Outra vantagem desta metodologia de ensino é o indivíduo ser sabedor da existência da informação e de onde ela se localiza para que no momento adequado a acesse.

O progresso que se poderá ter na área de informática aplicada à educação dependerá da adequação e qualidade do *software* educacional a ser utilizado. Por isso, temos que ter *software* de ensino inteligentes que possam trazer esta qualidade sobre o conhecimento a ser transmitido ao aluno e da mesma maneira possa adaptar-se a este mesmo aluno conforme o seu aprendizado.

Nos Sistemas Tutoriais Inteligentes (ITS), o ensino é apoiado sobre uma grande base de conhecimento a respeito do tema a ser ensinado, construído por um

especialista, a partir da qual o sistema interage com o aluno como um tutor computadorizado com o poder indutivo similar a um professor humano.

Assim, utilizando a *Web* e técnicas de Inteligência Artificial (IA), podemos fazer com que o *software* deixe de ser um mero "virador de páginas eletrônicas" e se torne um elemento mais ativo no processo de interação com o aluno. Entre outras palavras, não utilizar apenas um meio eletrônico para fazer o mesmo que no papel, sem algum ganho significativo ao nível de ensino-aprendizagem.

Recentemente foi incorporada a tecnologia de agentes na modelagem do ITS, e nos ambientes educacionais na Internet. Um agente é uma entidade autônoma e pode ser definido como um sistema capaz de perceber através de sensores e agir em um certo ambiente através de atuadores [RUS 95]. Existem vários tipos de agentes, entre eles estão os agentes pedagógicos. Estes agentes são denominados pedagógicos quando estão ligados a um ambiente onde existe uma sociedade de agentes que compõem um sistema de ensino-aprendizagem [PER 99].

Os agentes pedagógicos [GIR 98] podem ser considerados como:

- Tutores: destinados ao ensino dirigido ao aluno;
- Assistentes: colaboram com a aprendizagem do aluno;
- Agentes na *Web*: destinados a uma aplicação de ensino na Internet e,
- Agentes Mistos: que ensinam e aprendem.

Os agentes pedagógicos possuem um conjunto de regras que determinam os objetivos de ensino, e os planos para atingí-los. Estes planos são determinados pelo uso de estratégias de ensino [PER 99].

A proposta deste trabalho é a especificação de um modelo de aluno adaptativo para sistemas na *Web* que tem por objetivo propor a seleção de estratégias de ensino adaptadas às características individuais do aprendiz e utilizar ferramentas de rede que possam ser incorporados ao ensino individual do aluno.

Este trabalho está estruturado da seguinte forma:

- O capítulo 2 introduz características do ensino à distância;
- O capítulo 3 descreve os ITS tradicionais e na *Web*;
- O capítulo 4 trata como é modelado o aluno na *Web*;
- O capítulo 5 mostra os estudos de casos realizados;
- O capítulo 6 apresenta o modelo proposto no trabalho;
- O capítulo 7 descreve como foi realizado a implementação do modelo proposto;
- O capítulo 8 apresenta conclusões, limitações e trabalhos futuros.

2 Educação à Distância

Segundo [MOO 96] a definição mais citada de educação à distância é a criada por Desmond Keegan em 1980 que, baseando-se na definição do próprio Moore de 1972:

"O Ensino à Distância é o tipo de método de instrução em que as condutas docentes acontecem à parte das discentes, de tal maneira que a comunicação entre o professor e o aluno se possa realizar mediante textos impressos, por meios eletrônicos, mecânicos ou por outras técnicas". [NUN 92]

Neste contexto, seis (6) elementos são essenciais para uma definição clara de ensino à distância [MOO 96] :

1. Separação entre aluno e professor;
2. Influência de uma organização educacional, especialmente no planejamento e preparação dos materiais de aprendizado;
3. Uso de meios técnicos - mídia;
4. Providências para comunicação em duas vias;
5. Possibilidade de seminários (presenciais) ocasionais.
6. Participação na forma mais industrial de Educação.

A definição apresentada pela legislação brasileira contempla todas os itens necessários mencionados por Landim e Tripathi no seu artigo 1: "Educação a Distância é uma forma de ensino que possibilita a auto-aprendizagem, com a mediação de recursos didáticos sistematicamente organizados, apresentados em diferentes suportes de informação, utilizados isoladamente ou combinados, e veiculados pelos diversos meios de comunicação". (Diário Oficial da União decreto n.º. 2.494, de 10 de fevereiro de 1998)

Segundo estudo efetuado por [LAN 97] entre os principais conceitos de Ensino à Distância, apresenta-se as seguintes características com os percentuais de incidência de cada uma:

TABELA 2.1 - Característica conceituais da educação à distância

CARACTERÍSTICAS CONCEITUAIS DO ENSINO À DISTÂNCIA	Incidência em %
Separação professor-aluno	95
Meios técnicos	80
Organização (apoio-tutoria)	62
Aprendizagem independente	62
Comunicação bidirecional	35
Enfoque tecnológico	38
Comunicação massiva	30
Procedimentos industriais	15

Como não poderia deixar de ser, a separação professor-aluno se torna o principal fator e característica de um Ensino à Distância. De um certo modo, verificando-se as diferentes definições de Ensino à Distância, nota-se que cada uma corresponde a um contexto e/ou a uma instituição. A validade de cada uma depende do

quanto representem o significado de seu trabalho junto aos alunos e a comunidade onde atuam.

2.1 O Uso da Internet no Ensino

Neste século, a tecnologia tem se desenvolvido muito rapidamente. Um dos reflexos deste fato são as novas ferramentas que possibilitam avanços nas práticas de ensino e aprendizagem.

Na educação, sobre o ponto de vista dos educadores, a tecnologia educacional está focada sobre a Internet. Os recursos disponíveis na Internet, podem ser usados para apoiar as atividades de aprendizagem em grande escala, tais como acessar e compartilhar informações entre alunos e especialistas em qualquer parte do mundo. Assim, a Internet torna-se uma ferramenta livre de limitações físicas e restrições de tempo.

A Internet, segundo [MOR 97], é uma grande rede mundial de computadores que habilita interconexões entre conhecimentos e saberes antes compartimentados e diluídos. No cruzamento de interações eletrônicas, a Internet liga entre si milhões de usuários, num tráfego impressionante de raças, credos, idiomas e ideologias.

Para [LUC 97] a rede oferece conteúdos para currículos que não estariam disponíveis sem acesso à Internet, permitindo o contato com especialistas de diversas áreas de conhecimento, trazendo novos e velhos amigos e colegas para a sala de aula. Com o acesso à Internet, o usuário pode se tornar um precioso fornecedor de informações, além de se transformar em um usuário privilegiado de informações.

A rede Internet é um ambiente potencialmente rico e propício ao desenvolvimento de trabalho educacional cooperativo em redes. Esta tipologia de trabalho implica escolhas de propostas pedagógicas coerentes que se alicerçam nas percepções sociais, tecnológicas, ideológicas, políticas e econômicas de cada comunidade envolvida. Algumas das vantagens do Ensino na Internet são citadas abaixo:

- Respeita o processo de ensino-aprendizagem de cada aluno, respeitando sua velocidade de compreensão de novos conceitos usando para isto novos processos de assimilação, de acordo com sua disponibilidade pessoal, seu tempo e seus afazeres.
- Nenhuma ferramenta especial é exigida para começar a aprendizagem.
- Distribuição do conhecimento em larga escala (para o mundo inteiro).
- Relação custo e benefício, pois pela Internet não há custos de impressão e transporte, uso de cópias e muitas vezes grandes apostilas.
- As correções e atualizações são bem mais simples, pois são realizadas em um único site, sendo imediatamente disponibilizado a todos os usuários da Internet.
- São possíveis diversas técnicas de ensino, tais como texto, imagens, comunicação entre professores, professores e alunos, e entre alunos.

- O aluno tem mais facilidade em dar o seu feedback e condições de dirigir e comandar a maior parte de seu processo de aprendizagem (auto-aprendizagem).
- O aluno é sabedor da existência da informação e de onde ela se localiza para que no momento adequado a acesse.

A *Web* oferece a navegação através de hiperdocumentos pela Internet, como característica principal possui um protocolo cliente-servidor chamado *Hyper Text Transfer Protocol* (HTTP) e a linguagem *Hyper Text Markup Language* (HTML). O protocolo especifica como os programas vão se comunicar e a linguagem especifica um conjunto de primitivas para a visualização dos hiperdocumentos. Documentos HTML são portáveis, ou seja, são independentes da plataforma. A *Web* possui um esquema de nomeação uniforme para os recursos, as URLs (*Uniform Resource Locators*), que consistem de vários campos: nome_protocolo://endereço_servidor_internet/path, na qual o path é o caminho completo onde o documento a ser consultado está armazenado.

Conforme Figura 2.1, a configuração do Cliente-Servidor da *Web* depende do protocolo que está sendo utilizado, e o cliente *Web* converterá, se for necessário, a informação para HTML para depois exibi-la. A *Web* está crescendo exponencialmente e com isto, ela se torna uma das mais importantes formas de mídia para compartilhamento de informações a nível global.

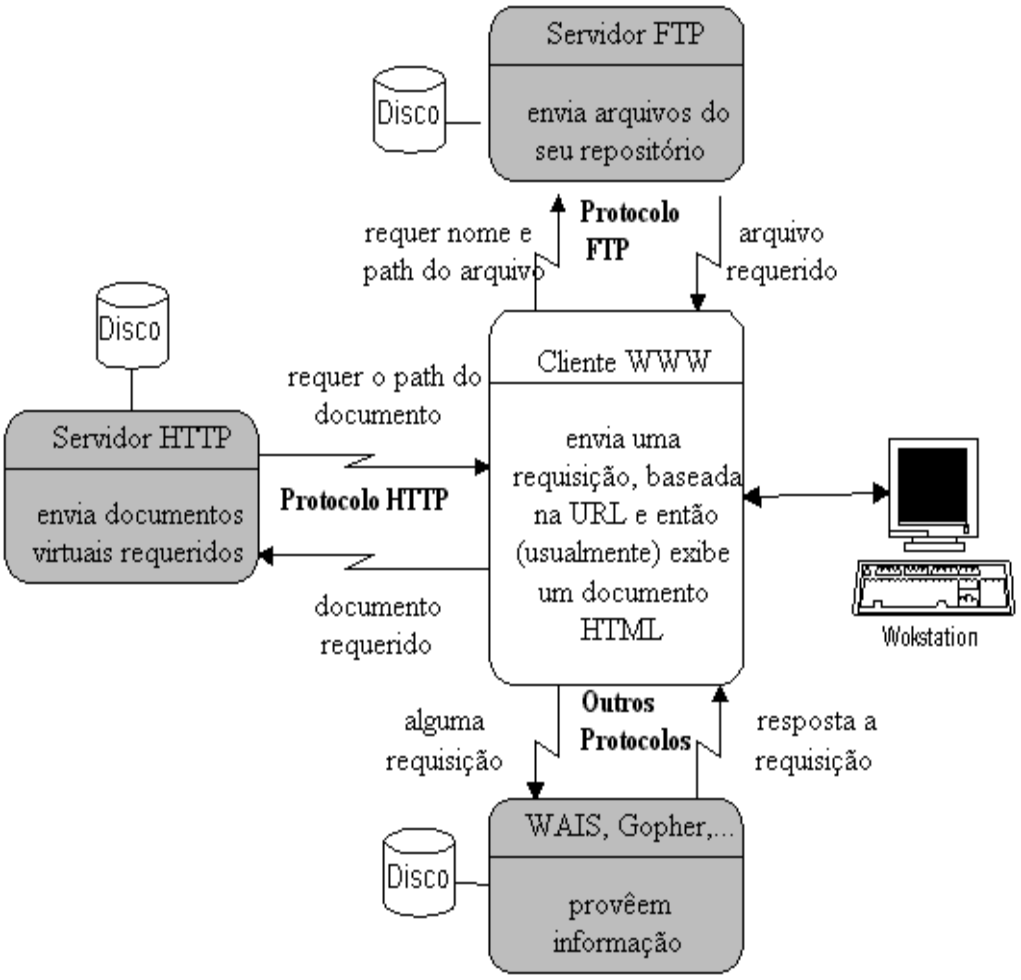

FIGURA 2.1 - Arquitetura *Web*

Os instrutores ou educadores podem fazer uso de ambientes de ensino (*courseware*) ao desenvolverem novos currículos ou mesmo novos *coursewares*. Para isso o único esforço necessário é fazer os *links* (atalhos para outros recursos) ou âncoras (marcações em uma determinada posição do documento) referindo-se aos recursos que deseja disponibilizar aos alunos [SIL 96].

3 Sistemas Tutoriais Inteligentes (ITS)

O objetivo fundamental dos ITS é proporcionar uma instrução adaptada ao aluno, tanto em conteúdo como na forma, superando desta maneira alguns dos problemas mais cruciais do *software* educativo na atualidade. Os ITS se comportariam de forma mais próxima a um professor.

As principais características de um ITS estão listadas abaixo:

- Tentam avaliar todas as respostas do aluno durante a interação.
- Método Socrático: pretendem proporcionar instrução através de diálogo com os alunos e detectar erros.
- Ambiente Interativo: trabalham um determinado assunto através da interação com o aluno utilizando a abordagem de diagnóstico desse aluno.
- Diálogo Bidirecional: utilizam um ambiente interativo com diálogo entre o aluno e o tutor, podendo haver alterações no comportamento do tutor face às respostas do aluno.
- Guia: alguns ITS funcionam como guias dos alunos e não se utilizam um diálogo e sim apresentam o conteúdo de outra forma.

Os ITS são programas de computador que se baseiam muito na IA para representar o conhecimento a fim de ter uma interação com o aluno, para que se possa analisar os padrões de erro, analisar o estilo e a capacidade de aprendizagem do aluno e oferecer instrução especial sobre o conceito em que o aluno está apresentando dificuldade.

O uso de técnicas de IA à construção de *software* educacionais se torna cada vez mais importante e seu estudo fará com que os mesmos tenham mais qualidade, destacando-se em particular:

- Os componentes de IA precisam ser integrados nos sistemas e estes estão longe do trivial no que concerne a implementação;
- A modelagem do domínio do conhecimento, particularmente no que se relaciona às habilidades trabalhadas proceduralmente e de forma acessível ao aluno são bem difíceis. Se um sistema é usado para provar um nível do problema, então isto pode ser visto como uma fonte de resolução do problema e os resultados podem ser usados pelo módulo de ensino;
- O uso de técnicas de IA ao que diz respeito aos aspetos pedagógicos estão longe de serem bem eficazes. Se os sistemas tiverem uma estrutura que os permitam construir uma estrutura com uma estratégia explícita de navegação selecionada a partir do aluno, os resultados serão mais eficazes;
- Tomar o conhecimento explícito e facilmente modificável para que o comportamento dos sistemas seja mais consistente e racional. Pode ser permitidas ao aluno, a interação com a base de conhecimento através da leitura, depuração e estendendo ou acrescentando mais informações.

Existem cinco componentes que podem ser considerados como componentes mínimos, formando uma arquitetura básica para os ITSs. Cada um destes componentes

constitui um módulo funcional distinto e são representados, respectivamente, como (ver Figura 3.1):

- **Módulo do Domínio**: contém o conhecimento que será ensinado;
- **Modelo do Aluno**: contém informações sobre o perfil do aluno que utilizou e/ou está utilizando o sistema;
- **Módulo de Estratégias de Ensino**: contém estratégias e táticas de ensino que devem ser adequadas aos alunos;
- **Módulo de Controle Tutorial**: responsável pelo gerenciamento de um ITS, realizando a seleção de estratégias de ensino;
- **Módulo de Interface**: responsável por administrar a interação do sistema com o aluno.

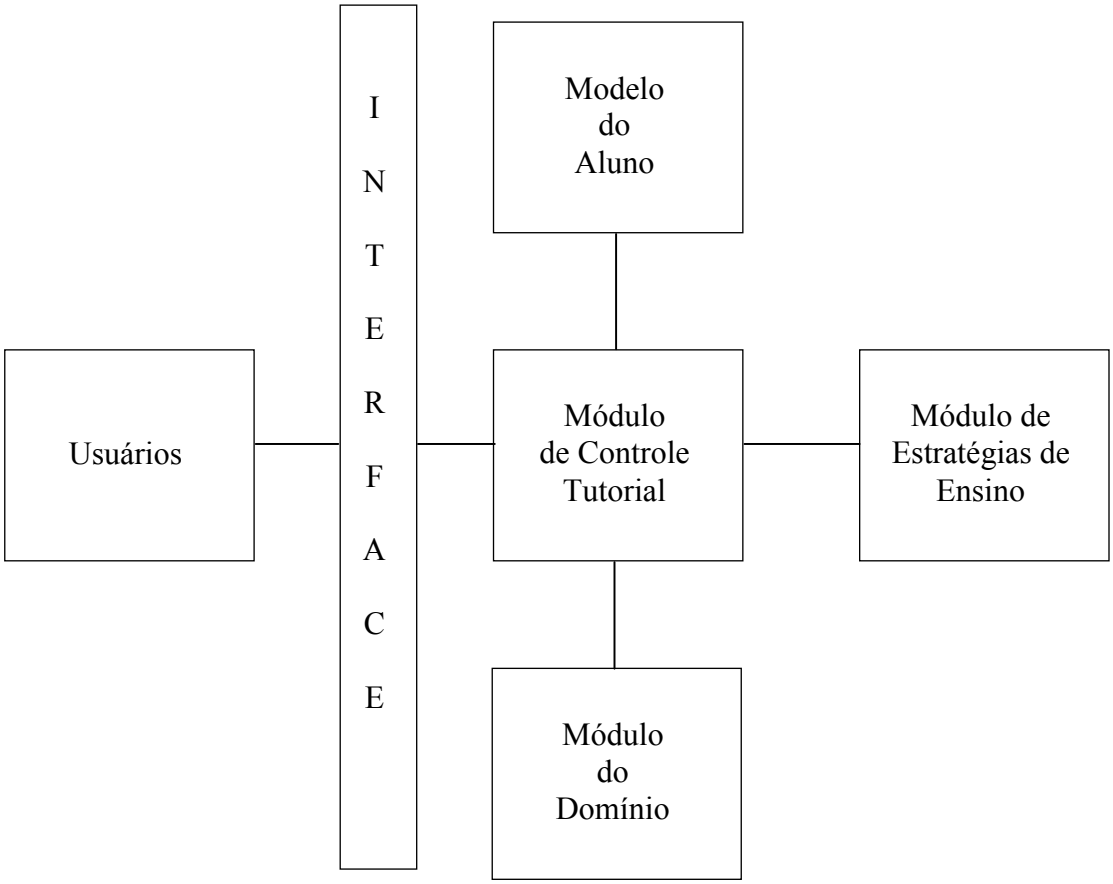

FIGURA 3.1 - Arquitetura de um Sistema Tutorial Inteligente (ITS)

3.1 Sistemas Tutoriais Inteligentes na *Web*

Propor uma prática de ensino na *Web*, não significa simplesmente dispor aos alunos um conteúdo de ensino que seja apenas acessado para estudo (livro texto). Temos que ter formas de estruturar tal ensino, para que realmente o aluno sinta-se como em sala de aula, usando todos os recursos que a *Web* possa permitir.

Para atingir este objetivo, umas das melhores formas, é de se adequar os ITS neste ambiente, principalmente para integrar na *Web* um mecanismo de adaptação individual que ativamente conduz o aluno pela *Web*, considerando o conhecimento do aluno, habilidade de compreensão, área de interesse, etc.

Os aplicativos educacionais baseados em conhecimento, como ITS, usualmente, são programas imensos e auto-suficientes que necessitam de uma plataforma especializada para serem executados.

A maioria dos ITS é projetada para suportar o desenvolvimento do conhecimento procedural do aluno por resolução de problemas. Tais sistemas geralmente não suportam a aquisição de conhecimento do domínio (conceitual).

Assim na *Web*, o desenvolvimento de um ITS tem que ser alterado. Como resultado, o sistema tem que ser auto-suficiente e completo, isto é, tem que suportar a aquisição tanto do conhecimento conceitual quanto do procedural, uma vez que o professor não está presente.

Os Sistemas Hipermídia (SH) fornecem uma ótima opção para suportar a aquisição de conhecimento conceitual. Neste sentido um ITS baseado em uma estrutura hipermídia fornece um modelo adequado para projetar ITS auto-suficientes sobre a *Web*.

Existem dois (2) extremos em ensino baseado em computadores [MUL 98]: restrição total (como um ITS) e sem restrições (como um sistema hipermídia). A maioria dos sistemas encaminha-se para restrição total. No seu ponto de vista, um sistema tutorial perfeito, deveria ser capaz de "deslizar" entre estes dois extremos de acordo com a necessidade do aluno e o estado de conhecimento, apresentando um ITS para um aluno novato ou um sistema hipermídia para um aluno avançado. Com base nesta afirmativa definem-se alguns aspectos importantes no desenvolvimento de um ITS na *Web* discutidos nos próximos tópicos.

3.1.1 ITS Genérico

Conforme sugerido acima, existe a necessidade do desenvolvimento de um sistema híbrido que tanto pode restringir o aluno a assegurar um ensino básico do domínio proposto, quanto permiti-lo a explorar o domínio em um gerenciamento de restrição menor uma vez alcançada um certo nível de experiência.

Para resolver este problema, a proposta de um ITS na *Web* é baseada em uma arquitetura multimídia de nodos (tópicos) e *links* com um módulo inteligente para ajudar na navegação e um modelo do aluno com o estado atual de experiência no domínio.

Para um aluno iniciante o sistema deveria oferecer um número de *links* limitados a partir de um conjunto de *links* disponível para um determinado nodo, relacionado aos movimentos anteriores do aluno na *Web*. Como o conhecimento do aluno é gradativamente aumentado, o controle pelo sistema é progressivamente abandonado, até o aluno ter o controle total do sistema. Assim os iniciantes estão liberados da complexidade associada com um sistema não familiar ou um tópico não familiar, enquanto a maioria dos alunos avançados estão liberados de restrições que não fazem mais sentidos, devido ao grau de conhecimento já adquirido.

Oferecendo poucos *links* que são fortemente relacionados ao nodo proposto, garante que os alunos novatos aprendam a estrutura de uma parte específica do domínio, ajudando-os a organizar o conhecimento armazenado pelo especialista no seu próprio conhecimento. Depois desta transferência de conhecimento ser passado por completo os alunos estão livres para navegar na *Web* e podem focar sobre o domínio atual de maneira mais aprofundada.

3.1.2 Identificação do Aluno na *Web*

Uma das maiores virtudes de um ITS é reconhecer e adaptar-se aos alunos (aprendizes) em um sistema. Na *Web*, até mesmo os sistemas educacionais, simplesmente expõe um assunto a ser aprendido, mas não levam em consideração se o aluno está conseguindo aprender e neste caso tentar adaptar-se de modo que este possa entender o conhecimento que está se tentando transmitir.

A *Web* simplesmente é utilizada na sua maioria para expor um texto passivo e estático na qual nenhum meio é fornecido para guiar ou ajudar os alunos a mover-se ou navegar no seu hiper-espaço. As principais necessidades são descritas abaixo em relação ao aprendizado individual:

- Identificação do aluno. O HTTP que é usado para a comunicação entre o servidor *Web* e o cliente, não fornecem nenhuma forma para identificar o aluno no lado cliente a partir do servidor. Além disso, os servidores *Web* trabalham de forma a responder uma requisição do aluno sem levar em consideração as requisições anteriores do lado do cliente. Para se adaptar ao estudo individual é necessária a identificação de quem está fazendo as requisições atuais e guardar as requisições anteriores (história).
- Controle de Janelas. A *Web* não fornece nenhum mecanismo para controlar as janelas abertas dos navegadores pelo aluno. Por exemplo, o aluno pode abrir uma janela para apresentar imagem ou texto e outra janela para os exercícios.
- Navegação. O aluno pode mover-se dentro de uma *Web* para onde ele desejar. Caso o aluno esteja perdido na navegação não existe nenhuma forma de guiar ou ajudar o aluno a recapturar seu estudo. Em um ITS é altamente desejável que o tutor restrinja algumas informações para o aluno.

Considerando o último aspecto de navegação, a grande questão defendida por especialistas da área, é que o cérebro humano pode fazer frente com um número limitado de tarefas ocasionando uma sobrecarga cognitiva, na qual a *Web* torna-se

confusa aos alunos pela riqueza de escolhas tornando-os perdidos em um labirinto de informações [MUL 98].

Por isso é aconselhável reduzir a amplitude de complexidades encontradas em um sistema hipermídia, até o aluno ter alcançado um nível que as complexidades não induzirão tal sobrecarga.

Os sistemas tutoriais tradicionais, geralmente são a antítese de sistemas de aprendizagem hipermídia, em que estes sistemas restringem os alunos e força-os a aprender métodos pré-determinados. Os ITS utilizam um modelo de conhecimento do aluno a fim de apresentar uma nova informação quando requerida pelo aluno, reforçar um assunto ou progredir no aprendizado e identificar conceitos errôneos.

Os ITS também ajudam os alunos a resolverem um determinado problema utilizando-se de várias formas de aprendizado para um domínio específico, pois os alunos podem apreender um determinado assunto de diferentes maneiras.

4 Modelo do Aluno na *Web*

Os modelos do aluno tradicionais são duramente criticados por sua modelagem ser apenas eficiente em domínios limitados, em que o modelo do aluno está baseado em hipóteses pré-definidas que giram em torno de regras também pré-definidas. Isto quer dizer que as dependências do domínio não podem descrever ou predizer toda a variedade do comportamento humano.

Originalmente, a *Web* foi usada para recuperar informações de qualquer parte do mundo. Entretanto, muito rapidamente, tornou-se claro que a *Web* teria que permitir uma maior interatividade. Com o aumento da utilização desta interatividade, surgiram vários sistemas de aprendizagem que introduziram os alunos em novos conhecimentos.

O número de *coursewares* está explodindo cada vez mais, trazendo novas capacidades de melhoramento para os *browsers* (*software* de navegação na *Web*). Porém até agora a maioria destes sistemas tem sido implementado em estágio experimental. Tais sistemas fornecem um suporte limitado aos alunos que não estão familiarizados com os novos domínios. Além do mais, existem poucos sistemas que se adaptam aos alunos individualmente como um sistema tutorial faz.

Em um ITS, duas características principais são destacadas na modelagem do aluno: ordenação curricular e a interação no suporte na resolução de problemas com o aluno [WEB 97]. A maioria dos sistemas de aprendizagem inteligentes são usados em salas de aula e por isso não necessariamente incluem todas estas características inteligentes. Muitos sistemas concentram-se em somente analisar os exercícios provendo suporte total na resolução de problemas durante a realização destes exercícios ou tarefas.

Entretanto, sistemas de aprendizagem baseados na *Web*, podem ser usados fora das salas de aula. Em tal situação, nenhum professor está diretamente disponível para ajudar durante o aprendizado de um aluno e adaptar o número e a natureza de novos conceitos apresentados para o aluno em seu atual nível de conhecimento. Portanto os sistemas de aprendizagem têm que representar o papel do professor o máximo possível.

4.1 Modelos do Aluno Propostos na *Web*

Segundo [WEB 97], um ITS tem que construir um modelo individual do aluno, para que cada aluno tenha o seu currículo escolar (tópicos do curso) de acordo com o seu conhecimento, ajudando o aluno na navegação do curso e apoiando na resolução dos problemas encontrados na realização dos exercícios ou tarefas.

Do ponto de vista de [ZUK 99] um modelo do aluno para prever as próximas necessidades (requisições) do aluno, pode ser criado através de uma adaptação global da maioria dos alunos acessando determinado curso e não levar em consideração apenas a modelagem individual.

Dentre estes pensamentos, os próximos tópicos apresentarão algumas formas de se construir um modelo do aluno que consiga atender principalmente às necessidades do aluno em adquirir um aprendizado de forma eficaz e interativa com o sistema.

4.1.1 Ordenação Curricular e Guia Adaptativo

A ordenação curricular descreve a ordem na qual uma nova unidade de conhecimento será aprendida e que táticas de ensino correspondente (por exemplo, apresentação de exemplos e demonstrações, questões, fornecer exercícios e testes, resolver exercícios) serão apresentadas para um aluno particular. Em livros escolares, a ordenação é pré-definida pelo autor do livro. O mesmo ocorre na maioria dos textos encontrados na *Web* na qual a ordenação é realizada pelo autor do texto ou o desenvolvedor do sistema, isto é, o autor fornece um caminho de aprendizagem otimizada assumindo um nível médio dos alunos que significa uma boa estratégia na escrita dos livros.

No caso de livros eletrônicos, a situação é totalmente diferente. Os livros eletrônicos são apresentados na forma de um hipertexto permitindo uma navegação aleatória sobre o texto. A fim de não se perder neste hiper-espaço, algum mecanismo de guia fornecido pelo sistema será de grande importância. Os *browsers* na *Web* somente podem guardar os *links* visitados, mas não são capazes de fornecer qualquer dica de quais páginas seria útil para uma próxima visita. Outra situação aparece quando o aluno não é um iniciante no domínio, mas já possui algum conhecimento dos tópicos a serem ensinados. Neste caso, é um desperdício o aluno ler todas as páginas da ordenação curricular, assim como resolver exercícios, passar por testes e etc., uma vez que este já está familiarizado com parte do domínio.

Em ambos os casos, a informação contida em um modelo do aluno individual pode ser usada para o sistema adaptar a apresentação das páginas para o aluno específico. Um tipo simples de modelo do aluno como um modelo de *overlay* pode ser útil para representar toda a seqüência curricular individualizada e um guia adaptativo no hipertexto. Nesta forma mais simples, o modelo do aluno contém informação se um tópico da base de conhecimento está aprendido completamente ou o estado (nível) de conhecimento do aluno sobre o domínio.

Um modelo do aluno mais elaborado pode diferenciar entre estágios de conhecimento mais detalhado. É possível distinguir entre páginas, descrevendo novos conhecimentos que foram apenas visitados ou páginas que os alunos executaram alguns testes e resoluções de problemas com sucesso. Adicionalmente, dependendo dos resultados dos testes e exercícios, o sistema pode sugerir que algum conhecimento (pré-requisito) deve ser revisto pelo aluno.

Porém, manter um modelo individual para cada aluno na *Web* e observar e diagnosticar o estado de conhecimento de cada um é muito mais complicado do que os ITS implementados em outro ambiente, uma vez que os *browsers* não tem controle da sessão do aluno. Atualmente poucos sistemas implementam uma ordenação curricular e um guia adaptativo em ITS na *Web*.

4.1.2 Ajuda Individual e Suporte na Resolução de Problemas

Os sistemas de ensino baseados em conhecimento apóiam os alunos enquanto eles estão trabalhando sobre os exercícios e durante a resolução de um dado problema. Existem duas técnicas utilizadas nestes casos.

De um lado, inúmeros sistemas fornecem diagnósticos inteligentes de soluções completas para os exercícios e resoluções de problemas. No outro lado, sistemas baseados em modelos passo a passo, fornecem uma interação contínua na resolução de problemas, apoiando os alunos durante a realização dos exercícios.

Hoje em dia, a maioria dos tutores mais avançados está utilizando o modelo passo a passo, enquanto os tutores para programação (ensino de uma linguagem de computador) são baseados no modelo ELM [WEB 97] apresentado no capítulo 5.

Nos tutores baseados em no modelo passo a passo, o sistema observa o aluno durante a resolução de um problema e aconselha quando o caminho da solução resultará em um erro. Este tipo de instrução é satisfatório em sistemas tutoriais sobre sistemas executados em plataforma local. Até agora observar a resolução de problemas passo a passo não pode ser executado em sistemas tutoriais na *Web*, principalmente por causa do tempo de espera causado pela comunicação do servidor com o cliente. No futuro, este problema será resolvido, criando-se agentes inteligentes incorporados no *browser* no aluno.

Os sistemas baseados em ELM utilizam um modelo do aluno episódico (casual). Um modelo do aluno episódico é satisfatório para diagnosticar soluções completas ou incompletas de um dado problema e fornecer ajuda individualizada. Esta abordagem é executada somente por requisições e não pelo processo de resolver um problema diretamente. Esta abordagem satisfaz a necessidade de comunicação cliente/servidor empregada em ensino baseado na *Web*.

4.1.3 Modelo do Aluno baseado em Filtros

A Internet tornou-se a grande rota na procura de informações. O aluno se depara com uma grande quantidade de informações disponíveis na rede. A filtragem de informação torna-se cada vez mais relevante.

Porém o uso de filtros nas informações oriundo na rede envolve alguns tipos de problemas:

- Critério efetivo e eficiente para filtrar as informações;
- Projetar uma interface amigável, inteligente e não perturbadora para levar o aluno às informações mais relevantes de acordo com o seu interesse.

A seguir são apresentadas algumas arquiteturas que filtram informações.

4.1.3.1 Arquitetura Geral

Uma arquitetura geral para desenvolvimento de um ITS que filtra informações é apresentada por [MAR 99]. A arquitetura é composta pelos seguintes módulos (ver Figura 4.1):

- O Modelo do Aluno representando as características e as informações necessárias de um aluno específico;
- Um componente de Modelagem de Aluno, capaz de construir um modelo do aluno dinamicamente através da interação entre o tutor e o aluno;
- Um componente de Recuperação Externa, que se comunica com o site de busca;
- O componente para filtrar informação, que seleciona um documento associado com o aluno, de acordo ao conteúdo do Modelo do Aluno;
- A Interface do Aluno que gerencia a interação.

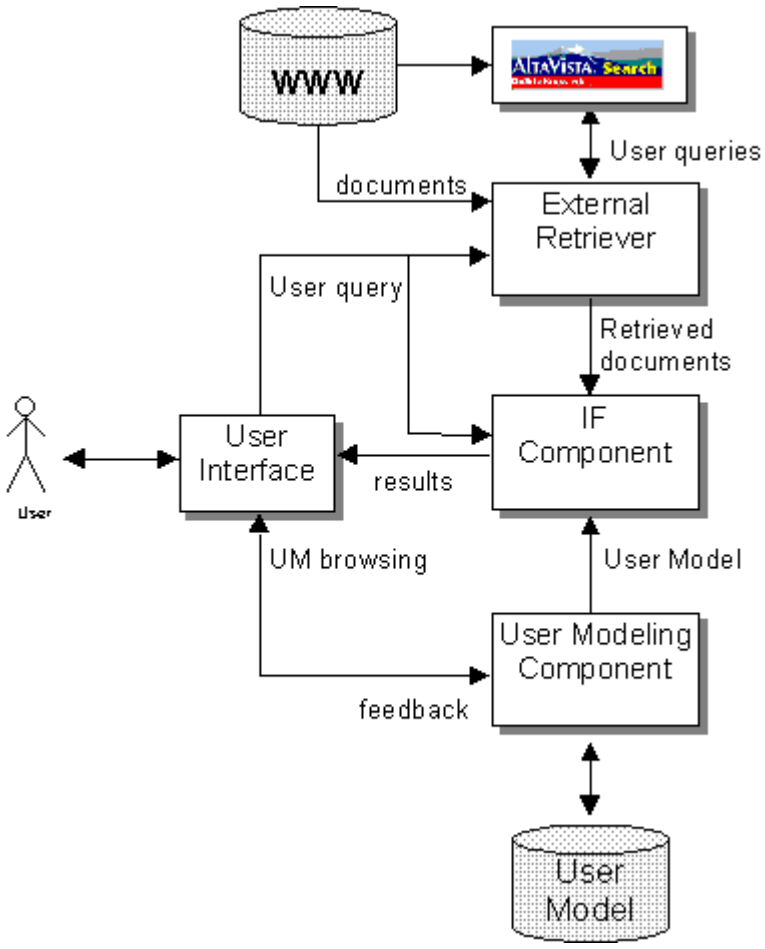

FIGURA 4.1 - Arquitetura geral de sistemas baseados em filtros

Quando o aluno interage com o sistema pela primeira vez o seu Modelo do Aluno necessita ser feito do zero (ponto de partida). A fim de construir um modelo rápido, uma entrevista é proposta ao aluno, expressando um escore de interesse para cada categoria do domínio, como mostrado na Figura 4.2. O aluno responde perguntas ao sistema que as envia para um sistema de busca na *Web* externo, obtendo documentos

que são filtrados e retornados ao aluno. No lado esquerdo o aluno define seu interesse no domínio. No lado direito são as perguntas elaboradas pelo sistema. Após, o sistema apresenta uma série de painéis, sendo que o primeiro painel é o filtro obtido na entrevista com o aluno. Posteriormente, no painel da esquerda é mostrada a lista de documentos recuperados pelo *site* de busca, e cada documento é detalhado no lado direito.

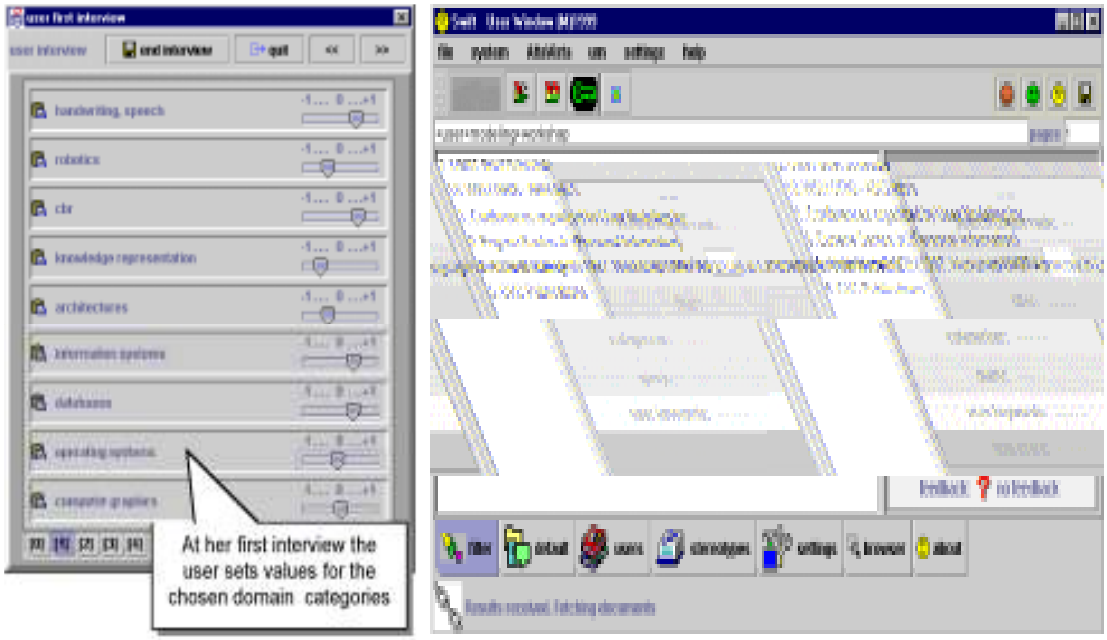

FIGURA 4.2 - Definição de interesse do aluno

Para o aluno usar o sistema de forma mais fácil, o sistema automaticamente classifica a lista de documentos para ajudá-lo a localizar os melhores documentos. O aluno procura os documentos necessários efetuando um duplo clique neles, e então o aluno pode expressar um simples *feedback* entre três valores diferentes a fim de facilitar a sobrecarga do aluno: muito bom, bom ou ruim. Nesta forma o sistema pode modificar seu modelo do aluno de acordo com as preferências do aluno.

4.1.3.2 Outras Arquiteturas baseadas em Filtros

Uma arquitetura tem sido desenvolvida [KAY 96] para a seleção e distribuição de objetos multimídia em uma rede de computadores em grande escala (Internet). A arquitetura (ver Figura 4.3) envolve verificar as informações dos objetos que são publicadas no centro da publicação e filtrar os objetos de acordo com o interesse do aluno. Cada objeto é acompanhado por um metadado que contém informação sobre o objeto, tais como palavras-chaves, nível, etc.

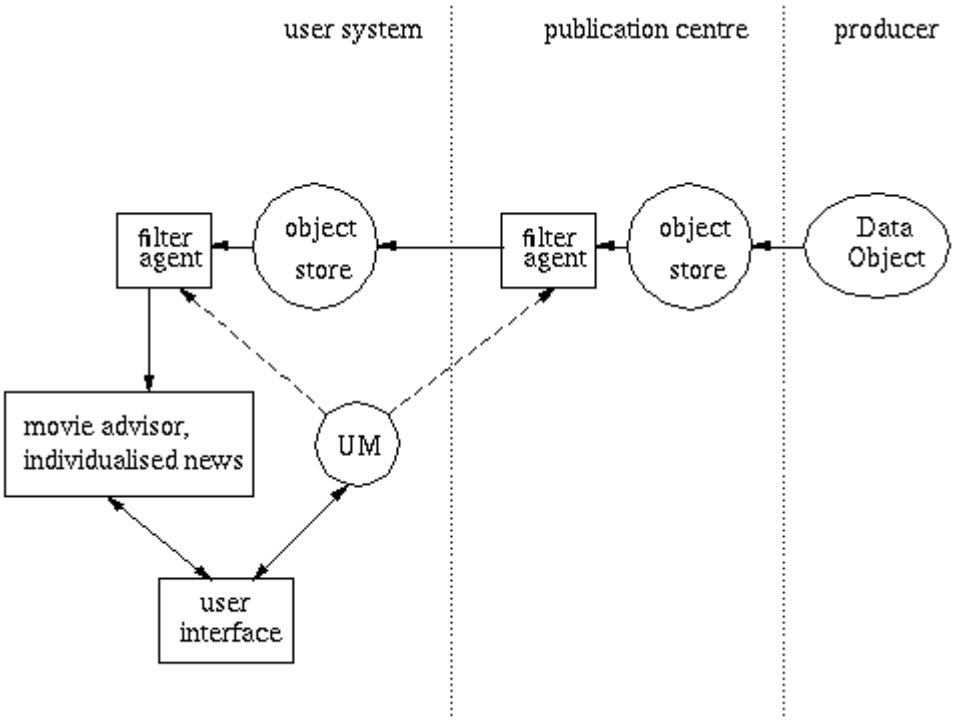

FIGURA 4.3 - Arquitetura de filtros baseado em objetos

O filtro é realizado usando as informações sobre o aluno a partir do modelo do aluno e examinando os metadados ou os dados atuais dos objetos. Este processo é executado por programas agentes que são construídos a partir de um modelo do aluno e registrados no centro da publicação. Os agentes examinam os objetos que são publicados e decidem se ignoram o objeto, mostra para o aluno ou mostra apenas os metadados do objeto. Esta arquitetura pode ser utilizada para distribuir páginas *Web*.

A *Web* é caracterizada como um sistema na qual os usuários utilizam a rede para recuperar objetos. Se um número grande de usuários desejar acessar uma página popular, o resultado da carga na rede e no servidor pode causar um grande problema. Isto pode ser parcialmente aliviado, por um sistema de caches, de maneira que o aluno, em uma determinada região, na primeira vez busque um objeto do servidor e posteriormente, nesta região, acesse o objeto localmente. Outro problema é que os alunos precisam usar *sites* de busca (ou efetuar a busca por conta própria) para encontrar novos objetos publicados conforme o seu interesse.

Nesta arquitetura, as páginas populares são transmitidas aos centros de publicações regionais, onde elas são examinadas por um agente e possivelmente, são encaminhadas para a máquina do usuário para visualização. Isto evita o problema de sobrecarga da rede, uma vez que o servidor na publicação original transmitirá o objeto para um número gerenciável de centros regionais e este processo pode ser repetido para enviar um objeto para subcentros. Também se pode ajudar ao aluno a encontrar tópicos novos e interessantes desde que os tópicos sejam notificados se foram aceitos pelos seus agentes de filtros pessoais.

A informação sobre as preferências do aluno em um domínio específico é extraída do modelo do aluno e incorporada em um pequeno agente. Este agente é

enviado para um centro de publicação usando um sistema de mensagens. Os objetos publicados recentemente (páginas Web, etc.) são passados para o agente para análise.

4.1.4 Prever requisições dos Alunos

Normalmente os alunos têm que esperar pela informação que foi requisitada na *Web*. A idéia agora é desenvolver um sistema que reduza o tempo de espera na requisição de um documento pelo aluno, pré-enviando documentos que o aluno pretende ver. Isto requer o desenvolvimento de um modelo do aluno que possa antecipar as requisições de um aluno na *Web*.

Outra questão, é que com o recente crescimento na *Web*, as fontes de informações *on-line* tem inspirado muitas pesquisas sobre agentes que ajudam os alunos a sugar as melhores informações de uma vasta quantidade disponível. Estes sistemas podem ser de um modo geral classificados em "sistemas recomendadores" [ZUK 99], que recomendam tópicos relevantes às informações de interesse do aluno e "sistemas de ação", que dão um passo mais adiante executando ações sobre o interesse do aluno. Exemplos de sistemas recomendadores são o *WebWatcher* (Joachims et al., 1997) e Letizia (Lieberman, 1995); exemplos de sistemas de ação são aqueles descritos em (Bestavros, 1996, Balabanovic, 1998). Ambos os tipos de sistemas requerem um modelo de predição que antecipa às preferências do aluno, incluindo documentos do interesse do aluno que este pode querer encontrar ou prever suas ações futuras. Estes modelos são obtidos aplicando-se técnicas de aprendizagem de máquina para identificar tais preferências ou futuras ações baseados nas preferências ou ações do (1) aluno por si mesmo (Davison e Hirsch, 1998, Joachims et al., 1997, Lieberman, 1995), (2) um grupo de alunos que tendem a similaridade (Alspector et al., 1997), ou (3) a população em geral (Bestavros, 1996, Albrecht et al., 1998).

Segundo [ZUK 99], analisar as informações obtidas em um servidor *Web* específico para prever as ações dos alunos rendem as seguintes características. (1) Pode-se observar somente um tipo de ação executada por um aluno, especialmente uma requisição de um documento na *Web* (e a ajuda neste ponto, é prever as próximas requisições dos alunos). (2) É extremamente difícil obter a clareza da representação do domínio. Tipicamente, existem números enormes de documentos localizados no servidor e muitos *links* entre eles (existem também *links* para localizações externas); a existência, localização e tamanho de documentos são assuntos para alterações contínuas, como os *links* entre os documentos. (3) Não existe nenhum objetivo claro que se aplica para todos os alunos - alguns alunos podem estar navegando, outros podem estar buscando uma informação específica; também podem existir muitas formas de alcançar um objetivo, desde que existem muitos caminhos para encontrar uma informação contida em um documento. (4) A seqüência das requisições efetuadas pelos alunos pode ser observada pelo servidor que fornece o documento, porém isto é somente um registro parcial dos movimentos do aluno através da Internet, pois nem todos os movimentos do aluno podem ser observados, uma vez que o aluno pode estar acessando diversos servidores. (5) Finalmente, a maioria dos *browsers Web* e servidores *proxy*, utilizam o mecanismo de cache dos documentos recebidos por um aluno. Assim, as requisições dos alunos de documentos anteriormente requisitados, não podem ser mais observadas por causa do *cache* que busca localmente e não mais do servidor.

4.2 Nível de Conhecimento do Aluno

O modelo do aluno deve construir o estado atual do aluno, armazenando os dados relacionados aos tópicos visitados, qualquer questão e resposta efetuada e a navegação do aluno sobre a *Web*. Este conhecimento é usado para decidir que nível de conhecimento o aluno tem adquirido e sugerir um ou mais *links* aos tópicos relacionados.

A incapacidade do aluno em subir de nível no conhecimento pode alertar ao sistema, que por sua vez pode tomar medidas de sugerir que o aluno volte a determinados *links* para rever seu conteúdo afim de concretizar este aprendizado, ou até mesmo propor diferentes *links* relacionado ao tópico. No ponto de vista do especialista, seguir passo a passo à estratégia adotada pelo aluno, pode determinar se a navegação está sendo efetuada corretamente ou determinar se o domínio está deficiente sobre algum aspecto.

Além disso, a informação de conhecimento do aluno pode ser utilizada para modificar o peso atual dos *links*. Por exemplo, no caso do Sistema Solar, se um aluno tem visitado seguidamente os nodos “Phobos” e “Titan”, o sistema pode entender que o aluno tem um grande interesse em Satélites, e aumentando os pesos dos *links* em Satélites, garante que o nível de conhecimento do aluno pode ser aumentado.

Este ponto é um grande alvo de discussão pedagógica, na questão em que o sistema seja capaz ou não de prover esta situação, ou seja, prevenir o aluno de aprofundar-se em uma única área do domínio ou guiá-lo para o conhecimento de todo o domínio. Também se deve levar em consideração a complexidade do domínio, na qual o sistema pode desejar guiar o aluno em uma área particular.

5 Estudo de Casos de ITS na *Web*

A crescente evolução de ITS na *Web* pode ser verificada pela existência de tutores já implementados, sendo que cada um contém uma particularidade especial, principalmente no que diz respeito à adaptação ao aluno.

Como a *Web* tornou-se um dos meios de ensino educacional mais importantes, é essencial que os autores de aplicativos educacionais desenvolvam seus sistemas cada vez mais adaptativos e inteligentes [BRU 97]. A *Web* é de fato um ambiente distribuído, sendo que existe a necessidade dos sistemas educacionais interagirem entre si.

Um sistema adaptativo é capaz de alterar o conteúdo ou aparência do sistema com base no compreendimento dinâmico de um aluno individual, adaptar o conteúdo ou apresentar certas características do aluno [EKL 98]. É importante ressaltar que adaptatividade não é tecnologia, mas um objetivo.

Muito recentemente, todos as aplicações educacionais na *Web* eram não-adaptativos. Hoje em dia, já existem alguns exemplos de Sistemas Tutoriais mais ou menos adaptativos baseados em *Web* como o CALAT, ELM-ART, PAT Online, Interbook e o CLEW. A parte principal desses sistemas é um modelo do aluno que mantém informações atualizadas do conhecimento e objetivos dos alunos. Estes sistemas serão apresentados no seguimento do trabalho juntamente com o AME-A e o Eletrotutor III, ambos desenvolvidos na Universidade Federal Rio Grande do Sul.

5.1 Calat

O sistema CALAT (*Computer Aided Learning and Authoring environment for Tele-education*) apresentado por [NAK 95] é um Sistema Tutorial Inteligente integrado em um ambiente multimídia distribuído suportado na *Web*.

A primeira implementação de CALAT é uma extensão da *Web* convencional tratada como cliente/servidor, incluindo um ITS no lado servidor e um visualizador multimídia no lado cliente (ver Figura 5.1). Nesta implementação a capacidade para adaptar ao aluno tornou-se possível pelo desenvolvimento de uma técnica de identificação do aluno. Uma quantidade enorme de recursos multimídia tais como sons, gráficos animados, bibliotecas eletrônicas e museus eletrônicos podem ser utilizados.

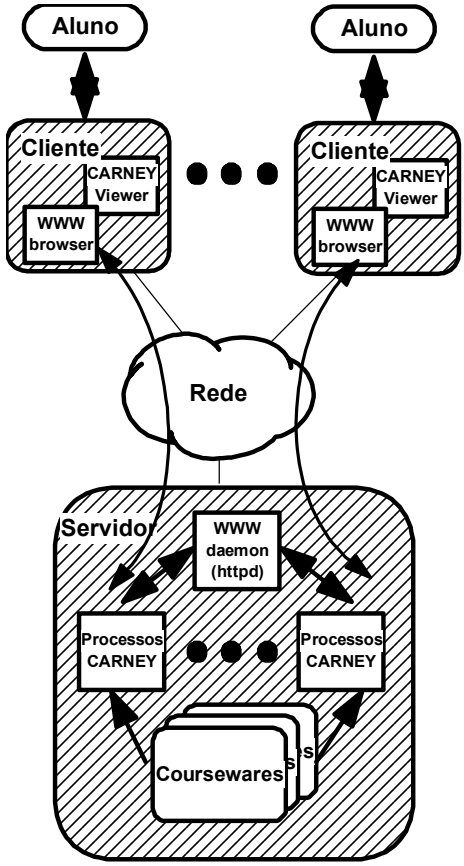

FIGURA 5.1 - Arquitetura CALAT

O sistema CALAT utiliza como base um ITS chamado CARNEY [FUK 95]. O CARNEY é um tutor inteligente que se caracteriza por ter um mecanismo de adaptação que torna possível modificar a estratégia de ensino conforme o estado de compreensão do aluno em relação ao assunto proposto. A arquitetura do CARNEY consiste de *coursewares*, informação sobre o aluno e sistemas tutoriais especialistas. Um *courseware* é dividido em cenários multimídia que correspondem a uma página de explanações ou exercícios e objetivos educacionais, que descrevem a estrutura lógica entre os cenários para estabelecer o objetivo final do courseware. As informações do aluno são atualizadas dinamicamente para refletir seu nível de compreensão. Assim aplicando as informações sobre o aluno ao sistema tutorial especialista, pode-se determinar os cenários mais apropriados para apresentação a fim do aluno ser guiado ao objetivo educacional.

A grande vantagem do CARNEY é que cada módulo é desenvolvido separadamente. Por causa desta estrutura modular, o autor do courseware não necessita nenhum conhecimento detalhado sobre o aluno, precisando apenas preparar os cenários e definir os objetivos educacionais.

5.1.1 Modelo do Aluno na *Web*

O método utilizado pelo sistema CALAT para identificação do aluno é de certo modo bem simples. Quando o aluno acessa o sistema pela primeira vez em cada sessão, é exigido que o aluno identifique-se pelo nome e selecione o *courseware* a ser estudado. Neste momento um processo CARNEY é inicializado contendo o nome e o courseware informado, e repassado uma nova tela HTML contendo esta informação.

Assim a cada requisição do aluno o servidor receberá o nome e o courseware, podendo registrar todos os passos efetuados pelo aluno.

Outro aspecto implementado em CALAT é o controle de janelas. O aluno pode requisitar várias janelas para visualizar imagens, enquanto na janela atual pode visualizar uma sentença. Na forma normal, os servidores *Web* apenas reenviam as páginas com as imagens para o cliente sem ter nenhum controle de quais janelas e o que está sendo visualizado no lado cliente. Em CALAT, é gerado um arquivo contendo um *script* (pedaços de código) de controle e dados a serem apresentados como resposta ao aluno como se fosse uma imagem GIF (*Graphics Interchange Format*) ou filmes MPEG (*Moving Picture Experts Group*). Assim, quando o *browser* recebe este *script* o servidor é requisitado novamente e um programa que controla as janelas, interpreta este scprit e apresenta a informação ao aluno.

O módulo pedagógico contém o curso pré-programado através do módulo autor e onde o aluno escolhe, entre os diferentes estilos disponíveis, a forma como vai desenvolver seus estudos. Não existe uma adaptação das estratégias de ensino ao aluno durante o processo de aprendizagem em conseqüência de uma análise pedagógica realizada pelo ITS, mas sim uma escolha feita pelo aluno para uma das formas instrucionais disponíveis que estão pré-programadas.

5.2 ELM-ART

O ELM-ART [WEB 95] demonstra como a interatividade e adaptatividade podem ser implementadas em um ITS na *Web*. O tutor ensina como programar em LISP (linguagem de programação funcional baseada em listas).

O primeiro passo para a construção do tutor, foi traduzir os textos dos materiais impressos em arquivos HTML, dividindo-os em pequenas subseções e páginas com textos que são associadas com os conceitos a serem estudados. Estes conceitos estão relacionados uns com os outros, descrevendo seus pré-requisitos e seu conceito final, construindo uma rede conceitual.

Todas as interações do aluno são registradas em um modelo individual do aluno. Para cada página visitada, a unidade correspondente da rede conceitual é marcada. Quando é apresentado texto no *browser Web*, os *links* mostrados na seção e subseção da página e na página em geral (texto), são anotados como se fossem uma sinaleira de trânsito (cores), relacionando a informação a partir do modelo individual do aluno.

O ELM-ART fornece interatividade utilizando-se de exemplos e diagnósticos inteligentes nas soluções dos problemas. Os alunos podem entrar com soluções para um problema de programação em um editor no *browser* e então enviá-las para o servidor.

5.2.1 ELM-ART II - Guia Adaptativo

O ELM-ART II [WEB 97] representa o conhecimento em unidades a serem estudadas em termos de uma rede conceitual. Cada unidade é organizada hierarquicamente em lições, seções, subseções e páginas terminais (páginas que apresentam o texto final de cada conceito). Cada página terminal pode introduzir novos

conceitos ou oferecer problemas a serem resolvidos. Cada unidade é um objeto contendo *slots* (caixas) para o texto a ser apresentado com a página correspondente e para informar que pode ser usado para relacionar unidades e conceitos um com o outro. Os *slots* estáticos armazenam informações sobre os pré-requisitos, os conceitos relacionados, e as unidades de resultado (são conceitos que o sistema assume ser conhecido quando o aluno trabalhou naquela unidade de forma bem sucedida). As unidades para as páginas terminais têm um *slot* de teste que pode conter a descrição de um grupo de itens de testes para o aluno executar. Quando os itens de teste são resolvidos com sucesso, o sistema pode inferir que o aluno possua o conhecimento sobre os conceitos explicados na unidade.

Os *slots* dinâmicos são armazenados no modelo individual do aluno. Este modelo do aluno é atualizado automaticamente durante cada interação com o sistema. Para cada página visitada durante o curso, a unidade correspondente é marcada como visitada no modelo do aluno. Além disso, quando os testes ou um problema de programação são resolvidos, os conceitos do resultado desta unidade são marcados como conhecidos e um processo de inferência é iniciado. Neste processo, todos os conceitos que são pré-requisitos para esta unidade (e recursivamente todos os pré-requisitos para estas unidades) são marcados como inferidas. As informações contidas no *slot* dinâmico no modelo do aluno são usadas para anotar os *links* individualmente e guiar o aluno da melhor forma através do curso.

5.2.1.1 Anotação de Links

O ELM-ART II utiliza o mecanismo de uma sinaleira de trânsito (extensão) para anotar *links* visivelmente (ver Figura 5.2). No topo de cada página final (abaixo da linha dos botões de navegação), todos os *links* que fazem parte de uma mesma subseção são mostrados com os *links* anotados, de acordo com o seu estado. A bola verde, vermelha, amarela e laranja são usadas para anotar os *links* (adicionalmente, os textos dos *links* são descritos em estilos diferentes para ajudar os alunos daltônicos).

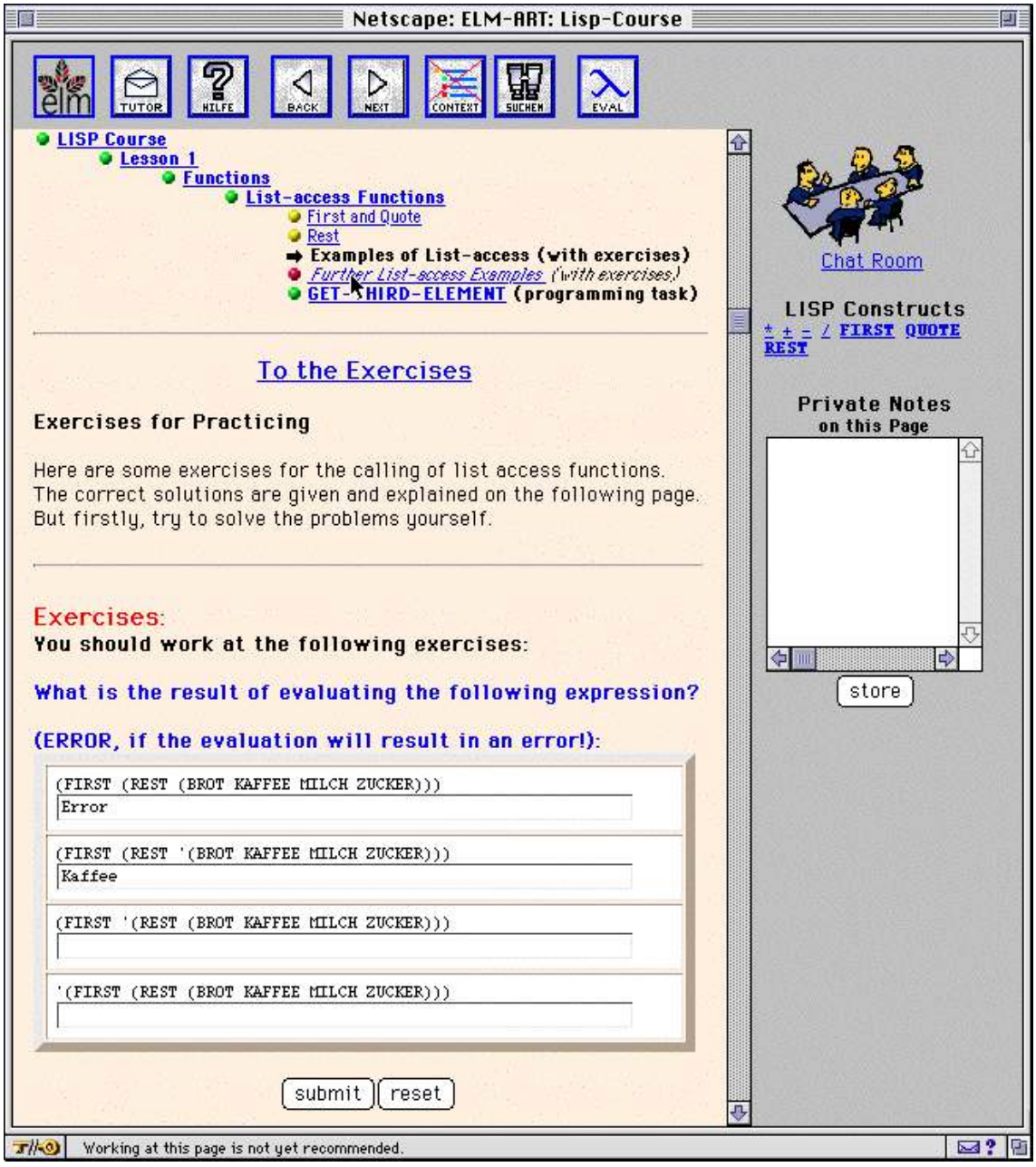

FIGURA 5.2 - Exemplo de uma página com anotação de *links*

O significado de cada bola (na frente do *link*) com a sua cor é mostrado abaixo:

- Uma bola verde indica que a página está pronta e é recomendada para ser visitada e os conceitos a serem ensinados nesta página já estão prontos para serem estudados.
- Uma bola vermelha indica que a correspondente página não está pronta para ser estudada. Neste caso, pelo menos um conceito que é pré-requisitado desta página, não foi aprendido pelo aluno, isto é, o sistema não pode inferir a partir de testes ou problemas de programação resolvidos com sucesso, o que determinaria que o aluno obteve o conhecimento deste conceito. Entretanto é permitido o aluno visitar a página e neste caso se for resolvido os testes e problemas de programação corretamente, o sistema infere todos os pré-requisitos necessários como conhecido.

- Uma bola amarela tem diferente significado dependendo do tipo de página que este *link* está apontando. No caso de uma página terminal, com um teste ou um problema, a bola amarela significa que o teste ou o problema foi resolvido corretamente. No caso de qualquer outra página terminal, a bola amarela indica que está página já foi visitada. No caso de uma lição, seção ou subseção, a bola amarela significa que todas as páginas subordinadas foram estudadas ou visitadas.
- Uma bola laranja tem diferentes significados também. No caso de uma página terminal, uma bola laranja significa que o sistema deduz a partir de outras páginas estudadas com sucesso, que o conteúdo desta página seria conhecido pelo aluno. No caso de uma lição, seção ou subseção, uma bola laranja significa que está página já tem sido visitada, mas nem todas as páginas subordinadas foram visitadas ou trabalhadas com sucesso.

Nos *browsers* que suportam *JavaScript*, o estado do *link* é explicado na linha de status no rodapé da janela, quando o cursor está localizado acima do *link* (Figura 5.2).

5.2.1.2 Seqüência Individual de Estudo (Ordenação Curricular)

Embora a anotação de *links* seja uma técnica poderosa para ajudar os alunos quando navegando através das páginas no curso, alguns alunos podem ficar confusos sobre qual o melhor passo a realizar para continuar o curso. Isto pode ocorrer quando o aluno move-se sobre o hiperespaço e perde sua orientação. Os alunos também podem desejar seguir um caminho ótimo através de uma seqüência de estudo, a fim de aprender de forma mais rápida quanto possível.

Para satisfazer estas necessidades, foi implantado um botão *NEXT* (Próximo) na barra de navegação permitindo que o aluno possa perguntar ao sistema qual a melhor página que deve ser visitada dependendo do conhecimento atual do aluno.

O algoritmo para selecionar a próxima página é: iniciando na página atual, a próxima página é selecionada se for anotada como "sugerida para ser visitada". Esta página pode ser uma página anterior, uma página com testes ou problemas que não foram resolvidos corretamente até o momento, desde que a página atual seja uma página terminal. O aluno completa o curso quando não foi encontrada uma próxima página, isto é, todos os pré-requisitos já foram visitados e não existe mais nenhum teste ou problema a ser resolvido (não foi feito, ou foi feito incorretamente).

5.3 PAT On-line

O tutor PAT On-line [RIT 97] está baseado no tutor PAT Álgebra I construído por Anderson, Hadley e Mark em 1995, que ajuda os alunos a compreender a equivalência matemática entre as diferentes representações, isto é, os alunos devem compreender a relação entre um problema, uma equação, um gráfico e uma tabela de valores. Um exercício típico no tutor é os alunos criarem uma planilha e um gráfico, que represente um problema e criem expressões algébricas, resolvendo as equações relacionadas com o problema.

O tutor PAT contém um sistema especialista capaz de resolver os problemas que são propostos para os alunos. Como os alunos resolvem um problema por passos

(por exemplo, preencher uma célula na planilha), o tutor considera se os passos realizados pelos alunos estão na direção correta da solução do problema. Caso estejam corretos, o tutor permanece silencioso e o aluno continua a resolução. Se as ações do aluno não são reconhecidas como um caminho para a solução, o tutor verifica se o passo é consistente com um conceito errado (*bug*). Nestes casos, o tutor é capaz de ajudar o aluno com a solução completa.

Cada uma das regras do sistema representam uma habilidade que o aluno necessita dominar, a fim de resolver o problema. O tutor assume um modelo cognitivo de aprendizagem por dois estados simples: não estudado e estudado. Cada habilidade é dominada ou não, e o tutor mantém, para cada aluno, a probabilidade que o aluno tenha dominado cada habilidade. Cada vez que o aluno encontra uma situação em que uma habilidade é necessária, existe alguma probabilidade, que é calculada pelo número de erros ocorridos pelo aluno nas habilidades que este tem dominado (algum "deslize") e a probabilidade que o aluno tenha acertado uma resposta sem ter dominado uma habilidade (o aluno imaginou corretamente).

5.3.1 Arquitetura do Sistema

No sistema PAT On-line, a ferramenta utilizada é um formulário HTML, e este formulário contém entrada de dados que são interpretados por um programa CGI (*Common Gateway Interface*). O programa CGI basicamente analisa os valores que o aluno informa no formulário e passa para o tradutor. O tradutor mapeia as ações dos alunos em mensagens para o tutor que retorna mensagens de saída que serão repassadas novamente para o tradutor, a fim que o programa CGI reconstrua um novo formulário baseado no *feedback* do tutor. O tutor contém todo o conhecimento do sistema especialista. Sua responsabilidade é fornecer ajuda, identificar erros e avaliar a performance do aluno sobre cada conhecimento. O servidor de registro do aluno mantém todas as informações sobre os alunos que tenham acessado o *site*.

A arquitetura do tutor PAT On-line é ilustrada na Figura 5.3.

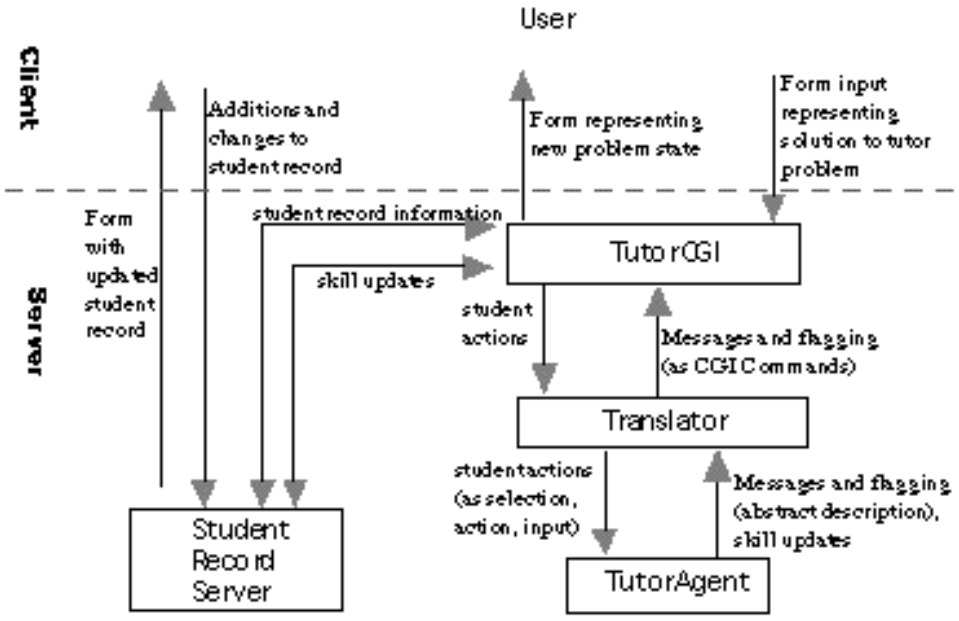

FIGURA 5.3 - Arquitetura do PAT On-line

5.3.2 A Interface do PAT On-line

O tutor PAT On-line é ilustrado na Figura 5.4. O uso de um formulário HTML, ao invés de planilha eletrônica (como o *Microsoft Excel* usado na primeira versão do sistema PAT), altera substancialmente as experiências do aluno. Na versão usada em sala de aula, os alunos recebem imediatamente o *feedback* em cada ação executada. Se um valor incorreto foi informado para uma célula, o sistema emite um som e apresenta o valor em um fonte diferente apontando o erro. Em formulários HTML, não é possível este tipo de interação. O servidor recebe as informações sobre as ações do aluno, somente quando o aluno submete aquela informação (através do uso de um botão "submit"). Para ajustar um modo de interação, o sistema permite ao aluno entrar com valores para muitas células antes de submeter. O tutor agente então avalia todos os valores e apresenta um *feedback* de cada um.

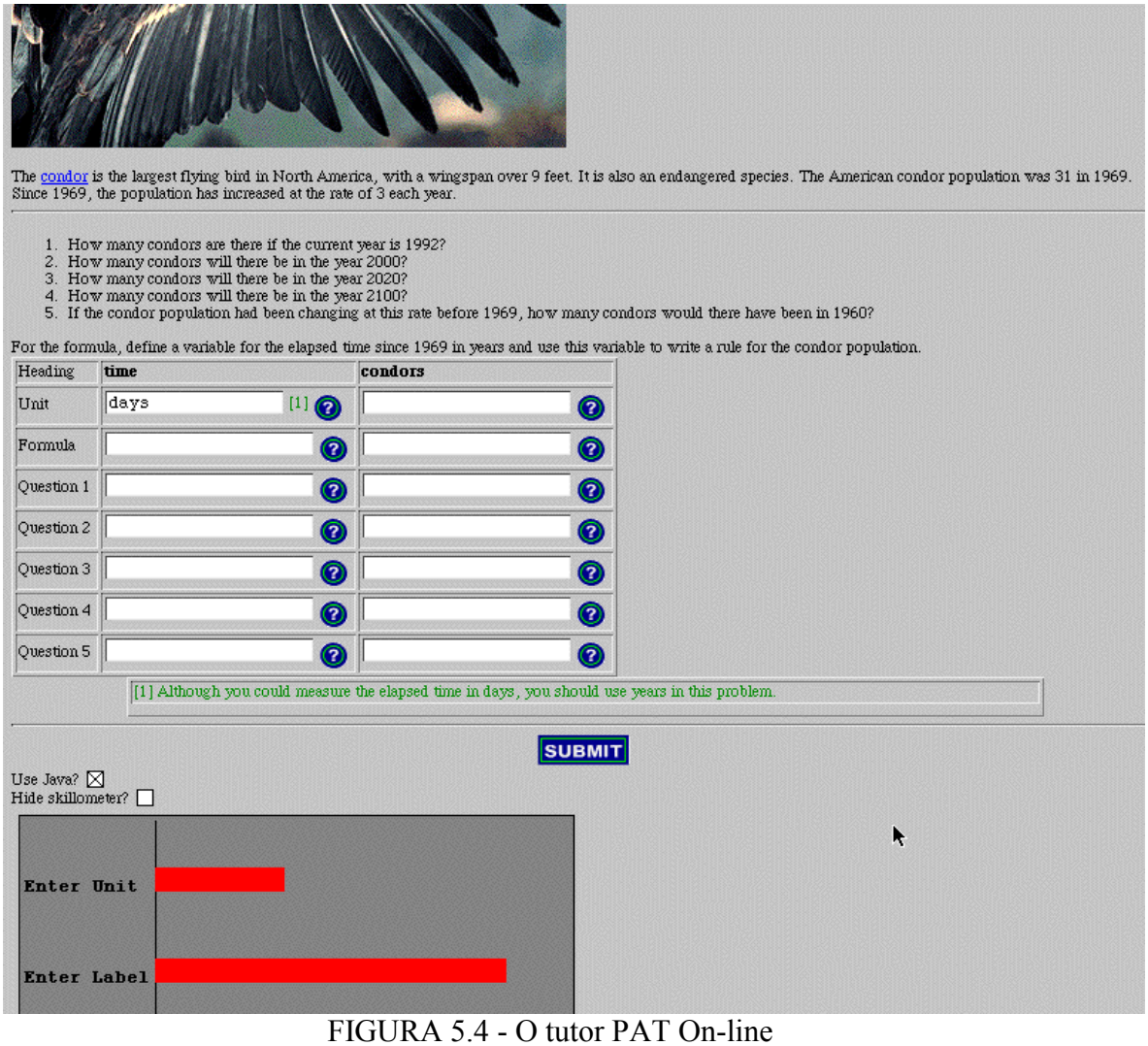

FIGURA 5.4 - O tutor PAT On-line

Na versão em sala de aula, o aluno pode requisitar ajuda em qualquer ação disponível. Para fazer isto, um item é selecionado (por exemplo, uma célula na planilha), e então a tecla de ajuda é acionada. O tutor identifica a célula selecionada, e usa aquela informação para guiar a ajuda. Quando um formulário HTML é submetido, nenhuma informação sobre a célula selecionada está disponível. Assim, para permitir o

aluno a requisitar ajuda sobre um item qualquer, foi somado um botão de ajuda para cada uma das células na planilha.

O uso de um formulário em HTML permite somar formatação de textos e imagens. Isto tudo é manuseado pelo tradutor. Quando o tradutor recebe uma requisição para iniciar um problema, o tradutor procura um arquivo HTML com relação a este problema. O arquivo HTML contém todos os comandos comuns da linguagem, permitindo a inclusão de todos os tipos de imagens e formatação. O tradutor anexa comandos para este arquivo, construindo a visão atual da planilha. O arquivo HTML representando o problema contém linhas de comentários que indicam o início e o fim de segmentos críticos dos problemas (a introdução, cada uma das questões e a conclusão). Com esta informação, o tradutor é capaz de encontrar os segmentos corretos do texto a serem destacados e somar comandos HTML para executar este destaque antes de passar o arquivo ao aluno.

Outra característica da interface é a apresentação das mensagens de erro. Na versão do tutor, utilizando o *Excel*, uma mensagem de erro relacionada a uma resposta do aluno sobre uma determinada célula, seria localizada em uma "nota" para aquela célula, que aparece sobre um quadro vermelho. O aluno pode visualizar a nota selecionando a célula e acionando as notas. No PAT On-line, existem algumas considerações cruciais para exibir mensagens de erro. Primeiro, os alunos completam mais do que uma célula antes de obter o *feedback*, sendo que podem receber mais de uma mensagem de erro. Segundo, não existe um elemento em HTML que permita o tipo de interação utilizada no *Excel*. A solução neste caso foi apresentar as mensagens de erro como "notas ao pé da página". Quando uma célula tem uma mensagem de erro, um número é desenhado próximo a célula. Na parte inferior da página, este número aparece junto com o texto de erro.

5.3.3 Reconhecendo Múltiplos Alunos

O PAT On-line segue a mesma idéia adotada no sistema CALAT para reconhecer o aluno na *Web*. Quando o aluno acessa o sistema pela primeira vez, o tradutor dirige-se ao tutor agente para criar um objeto com o nome do aluno. O nome é repassado para os formulários HTML como um campo escondido (*hidden field*), de maneira que qualquer requisição oriunda do formulário pode ser identificada.

5.4 Interbook

O Interbook [BRU 98] é uma ferramenta para apresentação de livros textos adaptados ao aluno na *Web*, fornecendo aos autores a possibilidade de apresentar seu material educacional em uma interface eficiente e fácil de usar. O Interbook emprega tecnologias de adaptação ao aluno, suporte a navegação adaptativa e rastreamento do conhecimento.

A característica principal do Interbook é o seu modelo do aluno ser utilizado em uma arquitetura aberta, que é capaz de armazenar qualquer dado arbitrário sobre um aluno. O sistema armazena o estado atual do conhecimento do aluno.

O Interbook utiliza dois tipos de conhecimento: conhecimento sobre o domínio que está sendo ensinado (representado na forma de um modelo do domínio) e o

conhecimento sobre os alunos (representado na forma de modelos individuais do aluno). A forma mais simples do modelo do domínio é apenas um conjunto de conceitos do domínio.

5.4.1 Navegação Adaptativa

Visualmente o Interbook utiliza símbolos coloridos e diversas fontes para fornecer suporte a navegação adaptativa, através de anotação de *links* (similar ao *ELM-ART II*). Sempre que um *link* aparece sobre uma página (na tabela de conteúdo, no glossário ou em uma página normal), a fonte do texto e a cor do símbolo, informa ao aluno sobre o status do nodo (tópico de conhecimento) que está por trás deste *link*.

São utilizados quatro cores e três tipos de fontes. Um símbolo verde e o fonte em destaque (negrito), significam "pronto e preparado", isto é, o nodo está pronto para se estudado (todos os conceitos de pré-requisito para aquele nodo tem sido encontrado), mas ainda não foi estudado e contém alguns materiais novos. Um símbolo vermelho e o fonte em itálico avisam sobre um nodo não preparado para ser estudado (não contém conceitos de pré-requisitos que deveriam ser estudados), enquanto símbolos brancos significa "nada de novo", isto é, todos os conceitos apresentados sobre o nodo são conhecidos pelos aluno). A cor violeta é usada para marcar tópicos que não foram anotados por um autor. Uma marca é adicionada para os tópicos já visitados. Um exemplo de anotação adaptativa no Interbook é ilustrado na Figura 5.5.

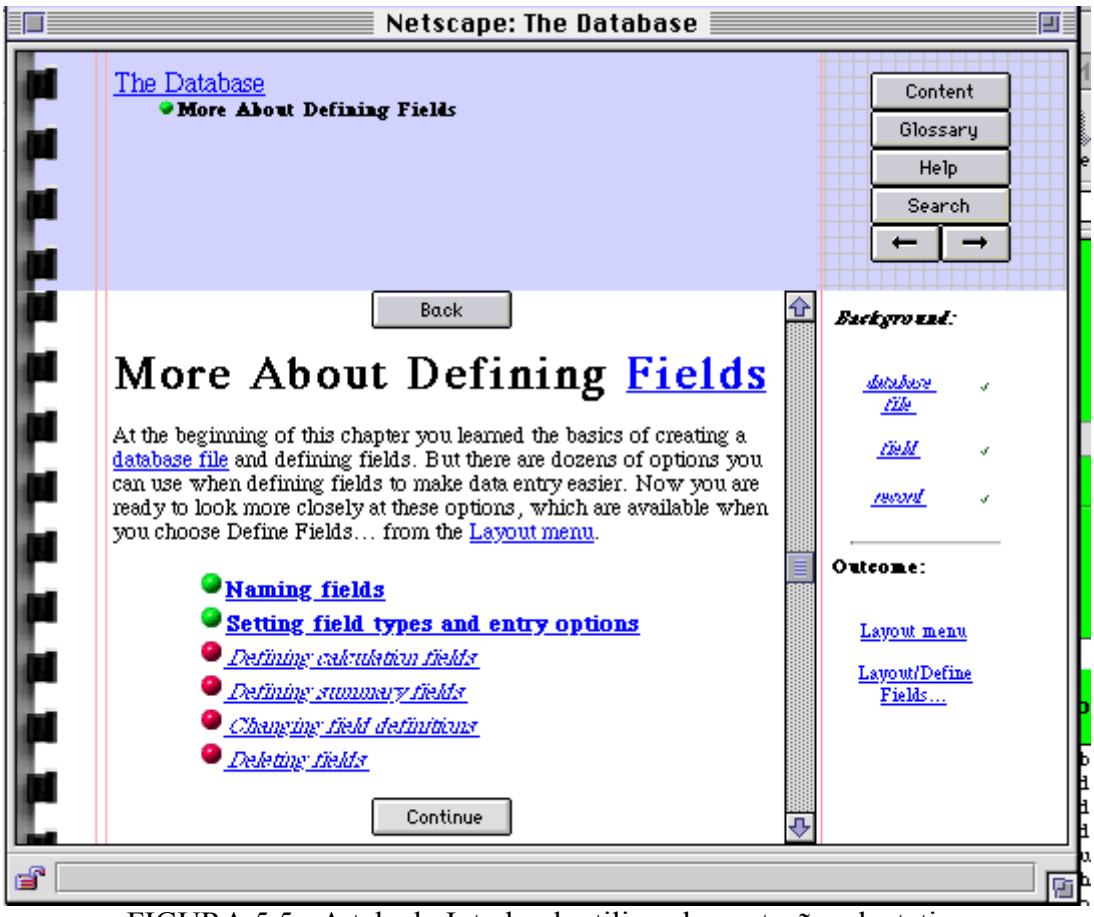

FIGURA 5.5 - A tela do Interbook utilizando anotação adaptativa

5.4.2 Modelo do Aluno

O modelo do aluno em Interbook é inicializado a partir de uma página de registro (formulário em HTML) gerada através de um modelo estereótipo, ou seja, modelado sobre as características básicas de um grupo de alunos. O modelo consiste de um arquivo individual em uma pasta chamada "alunos", que é atualizada conforme o aluno progride em relação ao material e a movimentação do aluno sobre o hiperespaço.

Para cada conceito no modelo do domínio, o modelo do aluno armazena algum valor que é uma estimativa do quanto o aluno aprendeu (nível de conhecimento) sobre este conceito. Este tipo de abordagem, visto como um modelo de *overlay*, pode ser atualizado facilmente, medindo o conhecimento do aluno sobre diferentes tópicos.

As ações dos alunos (páginas visitadas, problema resolvidos, respostas de questionários) são rastreadas e usadas para aumentar ou diminuir o nível de conhecimento do aluno. Outro componente do modelo do aluno são os objetivos de aprendizagem pelo aluno. Cada aluno pode ter um objetivo individual, ou seja, aprender apenas um conjunto de conceitos do domínio. Um guia adaptativo garante que o aluno alcance o primeiro objetivo em seqüência, depois o segundo e assim por diante, formando uma ordem individual de aprendizagem.

5.5 Ambiente de Aprendizado Cooperativo para *Web* (CLEW)

CLEW [RIB 98] é um ambiente para apoiar o aprendizado cooperativo, combinando o formato de apresentação de textos e gráficos da *Web* com conceitos de MUD (*Multiple Use Dimension*) e *workflow* para a criação e gerenciamento de cursos via Internet.

O construtivismo, as metáforas de aprendizado e a imersão formam a base educacional para guiar a construção dos cursos. O aspecto cooperativo é resultante da interação entre os participantes de um curso, permitindo o aprendizado entre as habilidades e qualificações entre os alunos.

Existem vários serviços disponíveis no CLEW, como os serviços administrativos, onde estão as funcionalidades que controlam os cursos. Os serviços didáticos fornecem aos professores os meios para apresentarem um curso, seu conteúdo e suas avaliações. Os serviços de comunicação facilitam a comunicação entre os participantes, e o serviço de cooperação auxilia a colaboração nas atividades dos alunos.

Neste ambiente, o professor coloca o material do seu curso com a ajuda do sistema. Não há uma seleção de estratégias de ensino para um determinado aluno, mas o curso ajuda o professor na construção do mesmo baseando-se em teorias construtivistas, como dito acima.

5.6 Ambiente Multiagente de Ensino-Aprendizagem (AME-A)

AME-A [DAM 99] é um ambiente multiagente de ensino-aprendizagem, no qual é realizado o estudo e o desenvolvimento de um sistema educacional interativo

para o ensino à distância. A proposta é o ensino genérico e adaptável às características psico-pedagógicas do aprendiz [PER 99].

Este ambiente utiliza a abordagem de sistemas multiagentes. Cada agente é responsável por suas tarefas e age continuamente no ambiente. O agente modela o aprendiz (aluno) conforme suas características psico-pedagógicas, motivação e nível de conhecimento, através de um questionário que é dado ao aprendiz no início do curso. Posteriormente é validado o modelo de aluno para a utilização de estratégias.

5.7 Eletrotutor III

O objetivo das diferentes implementações do Eletrotutor [SIL 92] [SIL 94] e [SIL 96] foi desenvolver um instrumento para verificar a eficácia do uso de diferentes abordagens de ambientes de ensino por computador na escola.

O Eletrotutor III [BIC 99] implementa um ambiente distribuído de ensino-aprendizagem inteligente (ITLE) baseado em uma arquitetura multiagente. A sociedade de agentes é composta por sete agentes, cada um deles possui uma função específica e como objetivo principal tem-se o aprendizado do aluno.

É de vital importância a coordenação do comportamento dos agentes e da maneira pela qual eles compartilham seus conhecimentos, objetivos, habilidades e seus planos para, em conjunto, realizar as ações necessárias para solucionar um problema. Para que diferentes agentes autônomos possam cooperar mutuamente a fim de atingirem seus objetivos é necessário que a sociedade possua organização (arquitetura) e comunicação.

A organização diz respeito à natureza e à função da sociedade e de seus elementos constituintes e a comunicação é o principal instrumento que os agentes utilizam para desenvolver a coordenação de suas ações [SIL 00].

5.7.1 Arquitetura do Eletrotutor III

A sociedade de agentes proposta na terceira versão do Eletrotutor contêm agentes autônomos que se comunicam uns com outros, na qual cada agente possui funções e objetivos dentro de sua especialidade. A Figura 5.6 representa a arquitetura do sistema, baseada em [SIL 97].

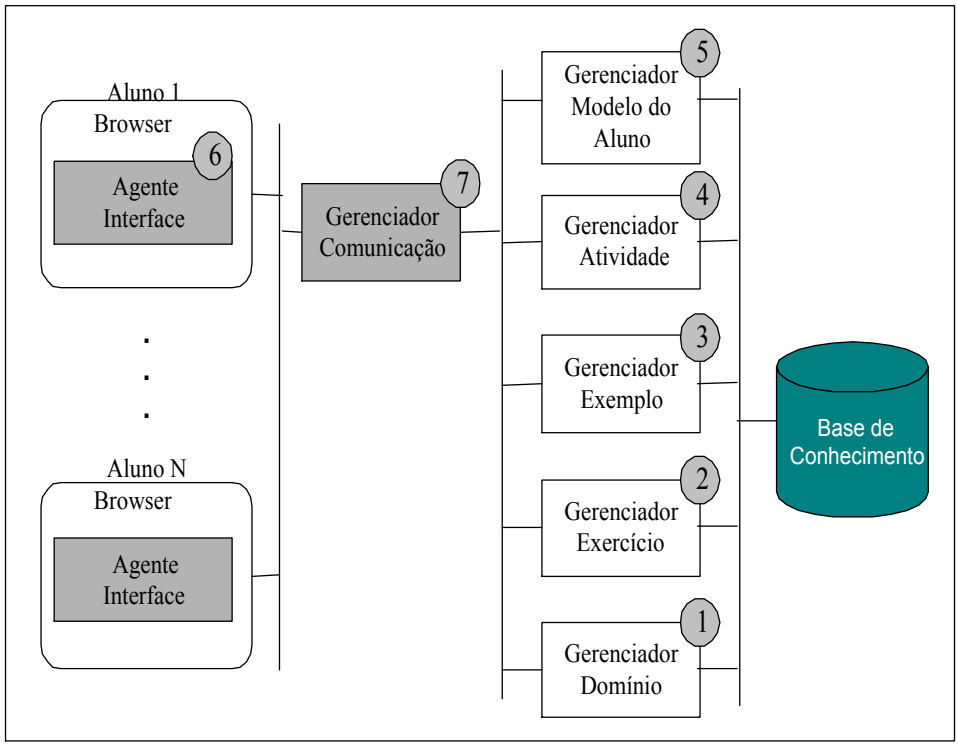

FIGURA 5.6 - Arquitetura do Eletrotutor III

O ambiente de aprendizado inteligente proposto, seguindo a Figura 5.6, contém um agente responsável pela recuperação do conhecimento do domínio sobre cada ponto a ser apresentado ao aluno, agentes responsáveis pela tarefa de propor exercícios e avaliação de respostas, exemplos ao aluno e atividades extras [BIC 99].

5.7.2 Metodologia de Ensino

O Eletrotutor quando acionado, pode ser utilizado de duas maneiras: o modo Tutorial e o modo Autônomo. No modo Autônomo, o aluno detém total controle sobre a sessão de estudo, podendo realizar qualquer lição, acompanhar qualquer exemplo ou fazer qualquer exercício, na seqüência que melhor lhe convier. Nesta modalidade, não é mantido qualquer registro dos dados do aluno ou das lições realizadas [BIC 99].

No modo Tutorial, o Eletrotutor assume o controle da sessão, definindo a seqüência mais adequada de lições, exemplos e exercícios. Para tanto, o tutor faz uso de um dos pontos mais importantes de um sistema de ensino inteligente auxiliado por computador: o diagnóstico cognitivo do aluno. Este diagnóstico é realizado através do registro de todas as ações do aluno no sistema. Assim as estratégias de ensino observam o histórico do aluno para tomar as próximas ações no sistema.

As estratégias de ensino consistem na seqüência de conteúdos, de exemplos e de exercícios que serão propostos ao aluno. A avaliação é realizada por uma série de até, no máximo, sete vezes o mesmo tipo de exercícios para as lições que contém exercícios.

6 Modelo Adaptativo Proposto

O modelo proposto neste trabalho implementa um ambiente distribuído de ensino-aprendizagem inteligente utilizando-se ao máximo a adaptatividade e dinamismo que um sistema pode obter. Com base nos diversos estudos de casos mostrados, principalmente o Eletrotutor III [BIC 99], tentou-se criar uma estrutura flexível que permite aos diversos especialistas montarem seu próprio conteúdo, estratégias, táticas, ferramentas de auxílio, etc., conforme o curso que estes propuseram a ensinar em um ambiente *Web*.

O funcionamento do sistema é apresentado nas próximas seções.

6.1 Modelo do Aluno

O modelo do aluno está baseado no método de *overlay*, ou seja, o sistema registra o que o aluno já aprendeu e compara com o domínio do curso para propor os próximos assuntos.

Através do acompanhamento de todos os passos do aluno no sistema, o tutor poderá adaptar-se ao aluno individualmente e apresentar alguns recursos definidos pelos especialistas a fim de reforçar seu aprendizado.

6.2 Estratégias de Ensino

As estratégias de ensino serão aplicadas de acordo com as táticas disponibilizadas pelo especialista. Conforme a ação do aluno o tutor pode mudar a forma de apresentação de uma determinada lição. Esta possibilidade determina-se pela tática atual que está sendo adotada pelo aluno e pela próxima tática que o especialista determina que deverá ser utilizada para esta ação ocorrida no sistema.

Esta funcionalidade se baseia no fato de diferentes alunos aprenderem de diferentes maneiras. O tutor pode verificar que um aluno específico não está aprendendo por uma determinada tática e alterar a forma de apresentação.

As estratégias também terão ligação com outros dois tópicos do sistema: avaliação e opções de recursos e ferramentas.

6.3 Avaliação

As avaliações servem para determinar o que o aluno aprendeu sobre o domínio. A proposta apresentada neste trabalho, é que para cada lição e tática (juntas) apresentada ao aluno, o especialista determine um método de avaliação. Isto faz com que o tutor não deixe para o final do curso uma avaliação geral, de modo que ficaria complicado determinar em que ponto do curso o aluno teve problemas no aprendizado.

Para efetuar tais avaliações é exigido que o especialista determine um valor quantitativo para cada lição e tática apresentada. Este valor é determinado como MÉDIA de avaliação. Para o aluno obter esta MÉDIA existem duas formas descritas abaixo:

1. Regras do Histórico: a idéia inicial deste trabalho era disponibilizar duas (2) regras específicas sobre o histórico do aluno. A primeira é determinada pelo acesso do aluno em uma lição. Por exemplo, se o aluno acessou todo o conteúdo (páginas) de uma determinada lição o aluno obtém determinado PESO de acesso. A segunda regra é pelo tempo utilizado pelo aluno para compreender esta lição, pois para um especialista seria impossível o aluno aprender em pouco espaço de tempo uma determinada lição. Porém esta segunda opção não será implementada neste trabalho, pois ainda fica inviável determinar se o aluno está concentrado sobre um conteúdo ou navegando sobre outras páginas na *Web*, ou até mesmo afastado do computador. Por exemplo, o sistema computa a hora de entrada do aluno em um determinado conteúdo (9:00) e computa a hora de saída do conteúdo (9:30). O problema é verificar se dentro desta meia-hora o aluno estava estudando o conteúdo ou simplesmente entrou no conteúdo e saiu da frente do computador retornando meia-hora depois avançando no conteúdo.
2. Página de avaliação: este método de avaliação faz com que o especialista crie uma página HTML contendo alguma forma objetiva de quantificar as respostas informadas pelo aluno. Por exemplo, o aluno está utilizando a tática de domínio (teoria) para uma determinada lição e ao final será apresentado um questionário para este responder. A avaliação será realizada por uma prova programada pelo especialista que cadastrará várias questões, sendo que cada uma com um nível de dificuldade e o tutor irá dinamicamente montar a prova para o aluno.

Além de determinar algum método de avaliação, o especialista deverá gerar estratégias que determinem as ações caso o aluno seja aprovado ou reprovado em determinada lição.

6.4 Recursos e ferramentas disponíveis

O especialista pode determinar para um determinado conteúdo a ser apresentado, que recursos irá disponibilizar ao aluno. Estes recursos serão apresentados na forma de *link*, na qual terá como objetivo três tipos de ações:

1. Mudança de estratégia: o especialista pode criar recursos como, por exemplo, um aluno não estar entendendo a forma de como está sendo apresentada determinada lição. Neste caso pode-se criar um *link* "não entendi" e o tutor ao verificar esta chamada por parte do aluno, acionar uma estratégia relacionada a esta ação e mudar a maneira de apresentar esta lição.
2. Ferramentas locais: estas ferramentas apesar de serem apresentadas diretamente no *browser*, não terão o conceito de rede, em virtude de não haver comunicação externa com outros pontos. Estas ferramentas são:
   - Ajuda: explica ao aluno a funcionalidade da interface, do tutor, etc.
   - Dicas: dicas para resolver um determinado exercício, aprender mais sobre um determinado conteúdo, etc.
   - Calculadora: utilizada para a resolução de exercícios.
3. Ferramentas na rede: o modelo proposto irá disponibilizar determinadas ferramentas de rede para que o aluno possa ter mais recursos para tirar suas

dúvidas, obter mais conhecimento sobre determinado tópico, etc. Estas ferramentas são de *chat*, fórum e busca. Porém o sistema é flexível em aceitar outras ferramentas, pois a chamada destas é feita por uma URL (*Uniform Resource Locator*) normal (*link*). Inclusive qualquer ferramenta acessada, não importando se esta ferramenta seja do modelo proposto ou uma outra qualquer, será registrada no modelo do aluno. Uma sugestão é o especialista criar uma página de *links* relacionados com uma lição que está sendo apresentado ao aluno.

6.5 Banco de Dados

A primeira parte da proposta visa modelar o banco de dados para que o especialista descreva a forma de como irá conduzir o curso, registrando todos os passos em uma base de dados. Assim o sistema evita ao máximo o desenvolvimento estático dentro do programa (condições *if/else*) utilizando o banco de dados como guia de execução.

6.5.1 Modelo Entidade-Relacionamento (ER)

Para armazenar todas as informações necessárias para a execução do sistema, desde a apresentação do conteúdo e o armazenamento dos dados do aluno, foi necessário criar um modelo de Entidade e Relacionamento (ER) mostrado na figuras abaixo. Este modelo é composto por tabelas ditas básicas e compostas descritas respectivamente nos próximos itens. Este modelo comporta vários cursos, portanto, poderá ser utilizado para várias propostas/implementações de ambientes de ensino/aprendizado que comportem esta modelagem. A modelagem foi dividida em duas partes para melhor compreensão: o conteúdo (ver Figura 6.1) e a parte de avaliação (ver Figura 6.2). Após é feito o detalhamento de cada tabela e seus relacionamentos.

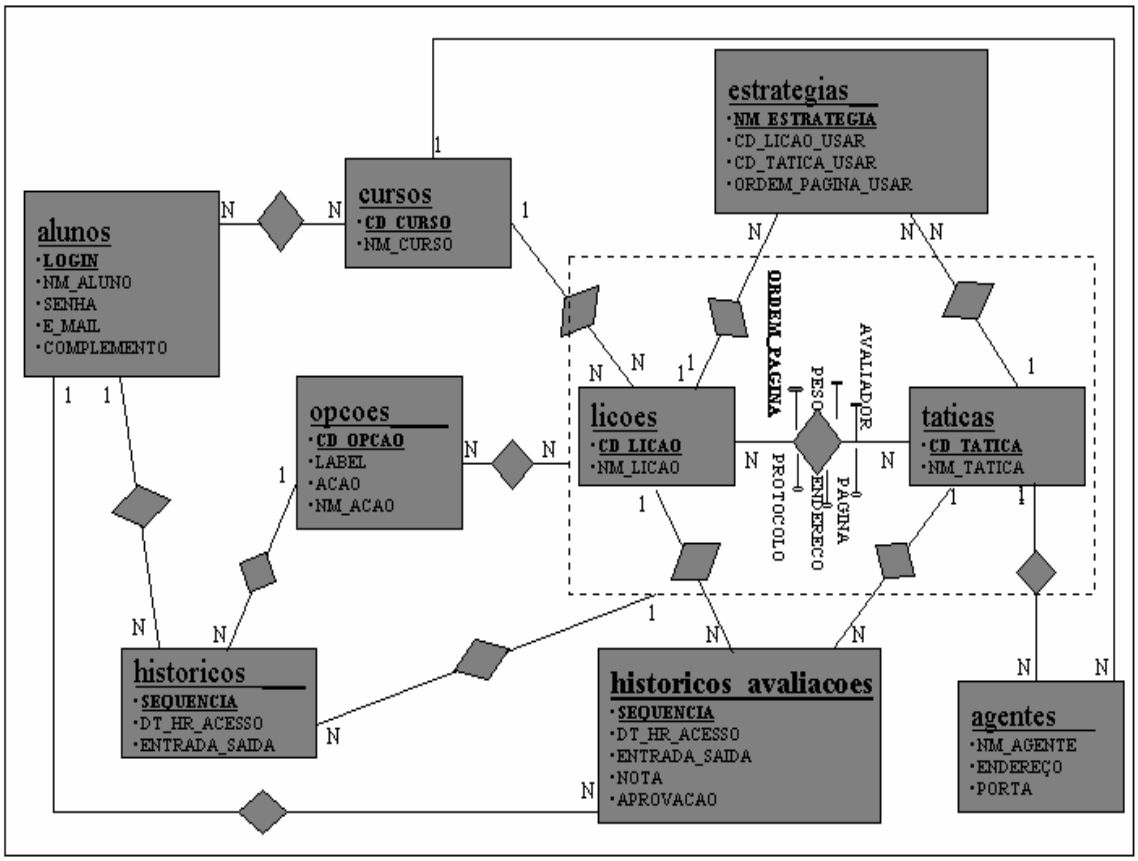

FIGURA 6.1 - Modelo ER relacionado com o conteúdo a ser ensinado

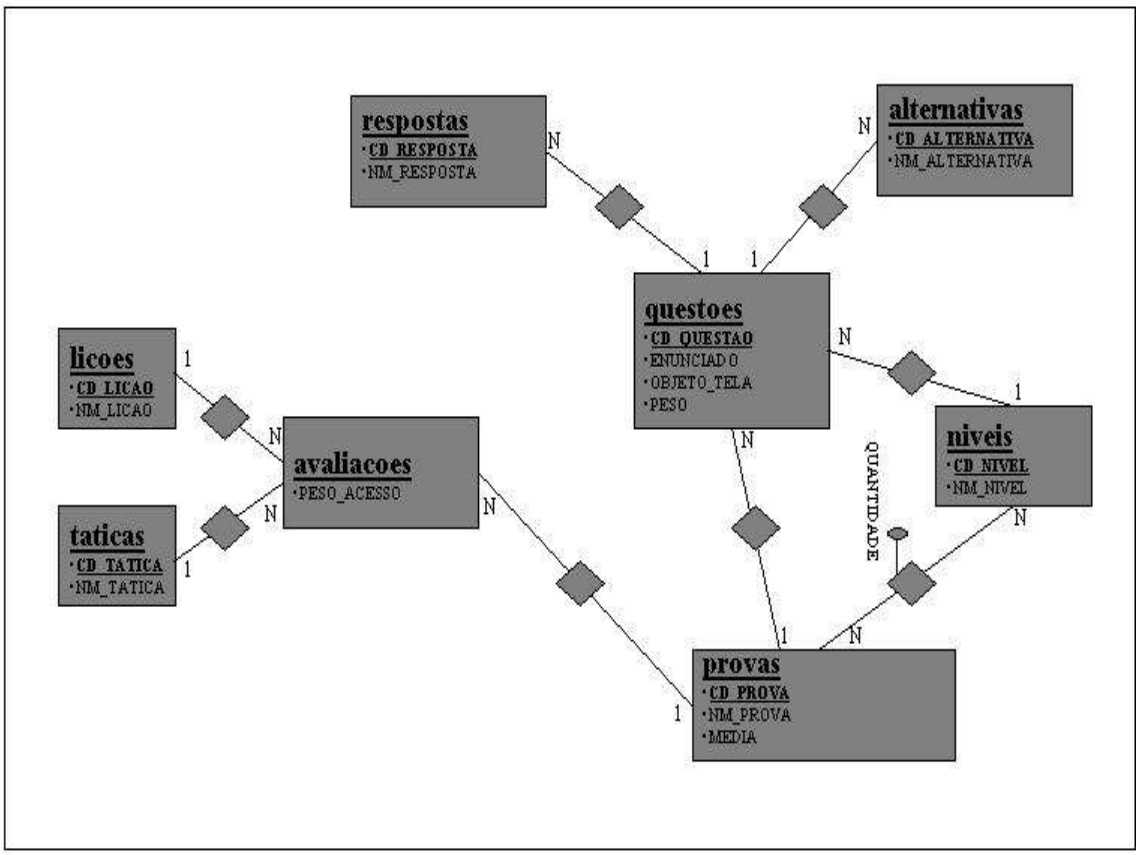

FIGURA 6.2 - Modelo ER relacionado com a avaliação do aluno

6.5.1.1 Tabela “cursos”

A tabela de “cursos” representa os cursos que podem ser utilizados no ITS proposto.

TABELA 6.1 - Tabela “cursos”

CAMPOS	**DESCRIÇÃO**
CD_CURSO	Código do curso (chave primária).
NM_CURSO	Nome do curso.

6.5.1.2 Tabela “alunos”

A tabela de “alunos” é utilizada para cadastrar os dados do aluno.

TABELA 6.2 - Tabela “alunos”

CAMPOS	**DESCRIÇÃO**
LOGIN	Identificador do aluno no tutor (chave primária).
NM_ALUNO	Nome do aluno.
SENHA	Senha do aluno.
E_MAIL	E-mail do aluno para contato.
COMPLEMENTO	Informações complementares como endereço, formação, conhecimentos, etc.

6.5.1.3 Tabela “cursos_alunos”

A tabela de “cursos_alunos” representa quais os alunos que fazem parte de um determinado curso.

TABELA 6.3 - Tabela “cursos_alunos”

CAMPOS	**DESCRIÇÃO**
#CD_CURSO	Código do curso (chave primária e estrangeira).
#LOGIN	Identificador do aluno (chave primária e estrangeira).

6.5.1.4 Tabela “licoes”

A tabela de “licoes” representa quais as lições que fazem parte de um determinado curso.

TABELA 6.4 - Tabela “licoes”

CAMPOS	**DESCRIÇÃO**
#CD_CURSO	Código do curso (chave primária e estrangeira).
CD_LICAO	Código da lição (chave primária).
NM_LICAO	Nome da lição.

6.5.1.5 Tabela “taticas”

A tabela de “taticas" representa as diferentes formas de se apresentar uma lição.

TABELA 6.5 - Tabela “taticas”

CAMPOS	DESCRIÇÃO
CD_TATICA	Código da tática (chave primária).
NM_TATICA	Nome da tática.

6.5.1.6 Tabela “agentes”

A tabela de “agentes” é gerada a partir de uma tática qualquer.

TABELA 6.6 - Tabela “agentes”

CAMPOS	DESCRIÇÃO
#CD_CURSO	Código do curso a qual o agente faz parte (chave primária e estrangeira).
#CD_TATICA	Código da tática (chave primária e estrangeira).
NM_AGENTE	Nome do Agente. Este nome será importante, pois será o nome gerado para a classe que o agente corresponderá na hora da implementação.
ENDERECO	Endereço onde o agente será carregado.
PORTA	Número da porta que rodará o agente.

6.5.1.7 Tabela “licoes_taticas”

A tabela “licoes_taticas" representa as diferentes táticas de ensino utilizadas por cada lição e o conteúdo que deve ser apresentado ao aluno. Este conteúdo é apresentado em uma página HTML, criada pelo especialista.

TABELA 6.7 - Tabela “lições_taticas”

CAMPOS	DESCRIÇÃO
#CD_CURSO	Código do curso (chave primária e estrangeira).
#CD_LICAO	Código da lição (chave primária e estrangeira).
#CD_TATICA	Código da tática (chave primária e estrangeira).
ORDEM_PAGINA	Indica a ordem da página para determinada lição e tática. Esta ordem pode ser utilizada, por exemplo, para indicar o próximo conteúdo a ser apresentado quando o aluno acionar a opção de avançar no aprendizado (chave primária).
PROTOCOLO	Conjunto de três (3) opções: • *http://* - indica acesso direto a uma URL • *https://* - indica acesso direto a uma URL por *site* seguro • *file://* - indica acesso local a disco, ou seja, por questão de segurança o especialista não deseja que o aluno acesse o conteúdo por URL (caso este descubra o seu caminho) e apenas o sistema consiga carregar a página (utilizando a tecnologia de *servlets*).
ENDERECO	Local onde a página estará localizada no servidor *Web* ou no disco local.
PAGINA	Nome da página que contém o conteúdo a ser apresentado de acordo com uma lição e tática.

6.5.1.8 Tabela de “opcoes”

A tabela de “opcoes” permite que para cada lição associada a uma tática, possa-se aumentar as opções de tela para o aluno. Estas opções representam *links* para determinados recursos e/ou ferramentas locais ou na Internet.

Esta tabela é de extrema importância, pois o especialista pode determinar novos rumos de ensino a um determinado aluno. Por exemplo, o especialista pode criar a opção “não entendi” ou “está difícil” que faz com que o ITS utilize uma nova estratégia de ensino caso esta opção seja acionada pelo aluno indicando que este não está compreendendo a forma de como está sendo apresentado um determinado conteúdo.

Outra alternativa para o especialista é a criação de *links* que acionam ferramentas locais ou na rede via URL. Neste trabalho, o ITS suporta algumas opções como: *chats*, fórum de discussão, busca e calculadora. Porém o sistema fica aberto a chamada de outras ferramentas na rede inclusive registrando no histórico do aluno.

TABELA 6.8 - Tabela “opcoes”

CAMPOS	**DESCRIÇÃO**
CD_OPCAO	Código da opção (chave primária).
LABEL	Descrição da opção que fica visível para o aluno.
ACAO	Nome da ação. Caso seja uma estratégia é o nome da estratégia ou caso seja uma URL o seu próprio caminho.
NM_ACAO	Conjunto de duas (2) opções: • ESTRATEGIA – indica que uma nova estratégia deve ser acionada; • URL – indica a chamada de um recurso na Internet.

6.5.1.9 Tabela de “licoes_taticas_opcoes”

A tabela “licoes_taticas_opcoes” representa as opções que podem fazer parte de um determinado conteúdo.

TABELA 6.9 -Tabela “lições_táticas_opcoes”

CAMPOS	**DESCRIÇÃO**
#CD_CURSO	Código do curso (chave primária e estrangeira).
#CD_LICAO	Código da lição (chave primária e estrangeira).
#CD_TATICA	Código da tática (chave primária e estrangeira).
#CD_OPCAO	Código da opção (chave primária e estrangeira).

6.5.1.10 Tabela de “estrategias”

A tabela de “estrategias” é o ponto chave para determinar as ações que o tutor irá tomar (como ensinar o curso) para cada aluno individualizado.

As estratégias também podem ser vistas como regras. O tutor verifica qual a tática e lição atualmente utilizada pelo aluno (através da tabela de históricos) e conforme for o caso, altera para uma nova lição, para uma nova tática ou para um outro conteúdo (sempre iniciando na primeira página), dependendo sempre da criatividade e interesse do especialista.

Existem três (3) estratégias conhecidas internamente no tutor adotadas para qualquer curso:

1. Início: inicialmente o tutor apresenta quais as táticas existentes para apresentar a primeira lição a fim do aluno selecionar uma (a que mais lhe agrada) iniciando o modo tutorial do sistema.
2. Avançar/Próximo: apresenta o próximo conteúdo conforme a ordem da lição e conteúdo.
3. Retroceder/Anterior: apresenta o conteúdo anterior conforme a ordem da lição e conteúdo.

TABELA 6.10 - Tabela "estrategias"

CAMPOS	**DESCRIÇÃO**
#CD_CURSO	Código do curso (chave primária e estrangeira).
#CD_LICAO	Código da lição (chave primária e estrangeira).
#CD_TATICA	Código da tática (chave primária e estrangeira).
NM_ESTRATEGIA	Nome da estratégia. Este campo é importante, pois relata algum acontecimento ocorrido no ensino, como "não entendi", "está difícil", "reprovado", etc. (chave primária).
#CD_LICAO_USAR	Indica qual a lição a ser apresentada conforme a estratégia a ser adotada. Pode ser a mesma lição ou inclusive alterar para uma próxima. (referência estrangeira)
#CD_TATICA_USAR	Indica qual a tática a ser adotada conforme a estratégia a ser adotada. Pode ser a tática atual como pode haver alteração na forma de apresentar determinado conteúdo. (referência estrangeira).

6.5.1.11 Tabela de "historicos_avaliacoes"

A tabela de "historicos_avaliacoes" representa o resultado da avaliação do aluno, para a lição e tática que está sendo apresentado ao aluno marcando o tempo de entrada e saída da avaliação.

A tabela pode ser vista como histórico do aluno, porém para evitar desperdício de campos, foi criada esta nova tabela (a maioria do conteúdo acessado pelo aluno não será uma avaliação).

TABELA 6.11 - Tabela "históricos_avaliacoes"

CAMPOS	**DESCRIÇÃO**
#SEQUENCIA	Como o aluno pode ser avaliado mais de uma vez (se for reprovado) tem-se esta seqüência.
#CD_CURSO	Código do curso (referência estrangeira).
#LOGIN	Identificador do aluno (referência estrangeira).
#CD_LICAO	Código da lição (referência estrangeira).
#CD_TATICA	Código da tática (referência estrangeira).
DT_HR_ACESSO	Em que momento de tempo o aluno começou e terminou de efetuar a avaliação.

ENTRADA_SAIDA	Indica se a data a ser registrada é quando o aluno está iniciando uma avaliação (E) ou terminando de uma avaliação (S), onde E significa entrada e S saída.
NOTA	Indica a nota final para o aluno
APROVACAO	Indica se o aluno foi aprovado (A) ou reprovado (R)

6.5.1.12 Tabela de "historicos"

A tabela de "historicos" representa todo o acesso efetuado pelo aluno em relação ao conteúdo do curso e as opções acessadas.

TABELA 6.12 - Tabela "históricos"

CAMPOS	**DESCRIÇÃO**
#SEQUENCIA	Como o aluno pode acessar mais de uma vez um conteúdo tem-se esta seqüência. (chave primária)
#CD_CURSO	Código do curso (referência estrangeira).
#LOGIN	Identificador do aluno (referência estrangeira).
#CD_LICAO	Código da lição (referência estrangeira).
#CD_TATICA	Código da tática (referência estrangeira).
#CD_OPCAO	Código da opção (referência estrangeira).
#ORDEM_PAGINA	Indica qual a página que o aluno acessou em determinada lição e tática. (referência estrangeira).
DT_HR_ACESSO	Em que momento de tempo o aluno entrou ou saiu de um conteúdo.
ENTRADA_SAIDA	Indica se a data a ser registrada é quando o aluno está entrando em um conteúdo (E) ou saindo de um conteúdo (S).

6.5.1.13 Tabela de "avaliacoes"

A tabela de "avaliacoes" representa a avaliação que será realizada para uma lição e tática. O especialista pode decidir não aplicar uma prova, apenas aprovar o aluno se este acessou todas as páginas sobre uma tática em uma lição. Para fazer isto basta deixar o campo CD_PROVA como nulo e informar o PESO_ACESSO para esta avaliação.

TABELA 6.13 - Tabela "avaliacoes"

CAMPOS	**DESCRIÇÃO**
#CD_CURSO	Código do curso (chave primária e estrangeira).
#CD_LICAO	Código da lição (chave primária e estrangeira).
#CD_TATICA	Código da tática (chave primária e estrangeira).
#CD_PROVA	Código da prova que pode ser nulo (referência estrangeira).
PESO_ACESSO	Indica quanto que o aluno recebe na sua média por ter acessado todo o conteúdo de uma lição e tática.

6.5.1.14 Tabela de “provas”

A tabela de “provas” representa uma prova que será relacionada com uma lição e uma tática.

TABELA 6.14 - Tabela “provas”

CAMPOS	**DESCRIÇÃO**
CD_PROVA	Código da prova (chave primária).
NM_PROVA	Nome da prova.
MEDIA	Média adotada para esta prova. Interessante usar um valor entre 0-10.

6.5.1.15 Tabela de “niveis”

A tabela de “niveis” representa o nível de dificuldade que poderá ser aplicado a uma questão e posteriormente informar quantas questões de determinado nível a prova vai conter.

TABELA 6.15 - Tabela “niveis”

CAMPOS	**DESCRIÇÃO**
CD_NIVEL	Código do nível (chave primária).
NM_NIVEL	Nome do nível, como BAIXO, MÉDIO, ALTO, etc.

6.5.1.16 Tabela de “provas_niveis”

A tabela de “provas_niveis” representa como será formada a prova por níveis. O especialista poderá mesclar a prova com diferentes níveis de dificuldade.

TABELA 6.16 - Tabela “provas_niveis”

CAMPOS	**DESCRIÇÃO**
#CD_PROVA	Código da prova (chave primária e estrangeira).
#CD_NIVEL	Código do nível (chave primária e estrangeira).
QUANTIDADE	Indica quantas questões deste nível será utilizado na prova.

6.5.1.17 Tabela de “questoes”

A tabela de “questoes” representa várias questões cadastradas para uma prova. Posteriormente o sistema irá buscar as questões aleatoriamente conforme a configuração da prova (provas_níveis). É recomendável que o especialista cadastre um número de questões maior que a configuração da prova. Por exemplo, se uma prova exige duas (2) questões de nível baixo e duas (2) questões de nível alto, seria interessante o especialista cadastrar o dobro (quatro (4)) ou mais questões de cada nível para esta prova, assim a probabilidade do tutor montar provas diferentes será bem maior.

TABELA 6.17 - Tabela “questoes”

CAMPOS	DESCRIÇÃO
#CD_PROVA	Código da prova (chave primária e estrangeira).
CD_QUESTAO	Código da questão, ou seja, a ordem da questão (chave primária).
#CD_NIVEL	Código que indica qual o nível da questão.
ENUNCIADO	Enunciado da questão.
OBJETO_TELA	Indica qual o objeto HTML que será apresentado para esta questão. Existe três (3) possibilidades: • radio: botões de uma escolha. • edit: campo de edição. • select: caixa de opções.
PESO	Indica o peso da questão.

6.5.1.18 Tabela de “alternativas”

A tabela de “alternativas” representa quais as alternativas para uma questão (se houver a necessidade, pois no caso de ser setado a questão com o tipo de OBJETO_TELA igual a edit, não existirá a necessidade de alternativas).

TABELA 6.18 - Tabela “alternativas”

CAMPOS	DESCRIÇÃO
#CD_PROVA	Código da prova (chave primária e estrangeira).
#CD_QUESTAO	Código da questão (chave primária e estrangeira).
CD_ALTERNATIVA	Código da alternativa, ou seja, a ordem em que a alternativa irá aparecer (chave primária).
NM_ALTERNATIVA	Texto que aparecerá para o aluno.

6.5.1.19 Tabela de “respostas”

A tabela de “respostas” representa a(s) resposta(s) para uma questão. O especialista pode especificar várias respostas corretas para uma questão.

TABELA 6.19 - Tabela “respostas”

CAMPOS	DESCRIÇÃO
#CD_PROVA	Código da prova (chave primária e estrangeira).
#CD_QUESTAO	Código da questão (chave primária e estrangeira).
CD_RESPOSTA	Código da resposta (chave primária).
NM_RESPOSTA	Resultado correto da questão.

6.6 Ferramenta Administrativa

Para a inserção de todo o processo tutorial pelo especialista foi desenvolvida uma ferramenta administrativa que disponibiliza as seguintes opções:

- inserir o curso;
- inserir os alunos;
- informar quais alunos fazem parte de determinado curso;

- inserir as lições;
- inserir as táticas;
- inserir os agentes;
- inserir as opções;
- informar o conteúdo para cada tática e lição;
  - informar as opções para o conteúdo;
  - informar o método de avaliação;
    - inserir a prova;
    - inserir os níveis de dificuldade;
    - informar o nível da prova;
    - inserir as questões para a prova;
    - inserir as alternativas para a prova;
    - inserir a(s) resposta(s) para a prova;
- informar as estratégias de ensino;

6.6.1 Relatório do Modelo do Aluno

O sistema disponibiliza ao especialista um relatório de todos os passos efetuados por um determinado aluno no seu aprendizado. Este relatório é gerado através do histórico de cada aluno.

6.7 Interface adaptável

O modelo proposto não tem como base à implantação de interfaces. A única preocupação é com o espaço (local) onde o tutor irá apresentar o conteúdo e os recursos disponíveis. O layout das páginas e opções gerais vai ser determinado por quem está utilizando o modelo. Este responsável deverá apenas efetuar uma chamada (seja por um botão, *link*, etc.) ao modo tutorial proposto.

6.8 Estrutura de Agentes

A sociedade de agentes proposta neste trabalho contém agentes autônomos que se comunicam uns com outros, sendo que cada agente possui funções e objetivos dentro de sua especialidade. A Figura 6.3 representa a arquitetura elaborada neste trabalho, baseada em [SIL 97].

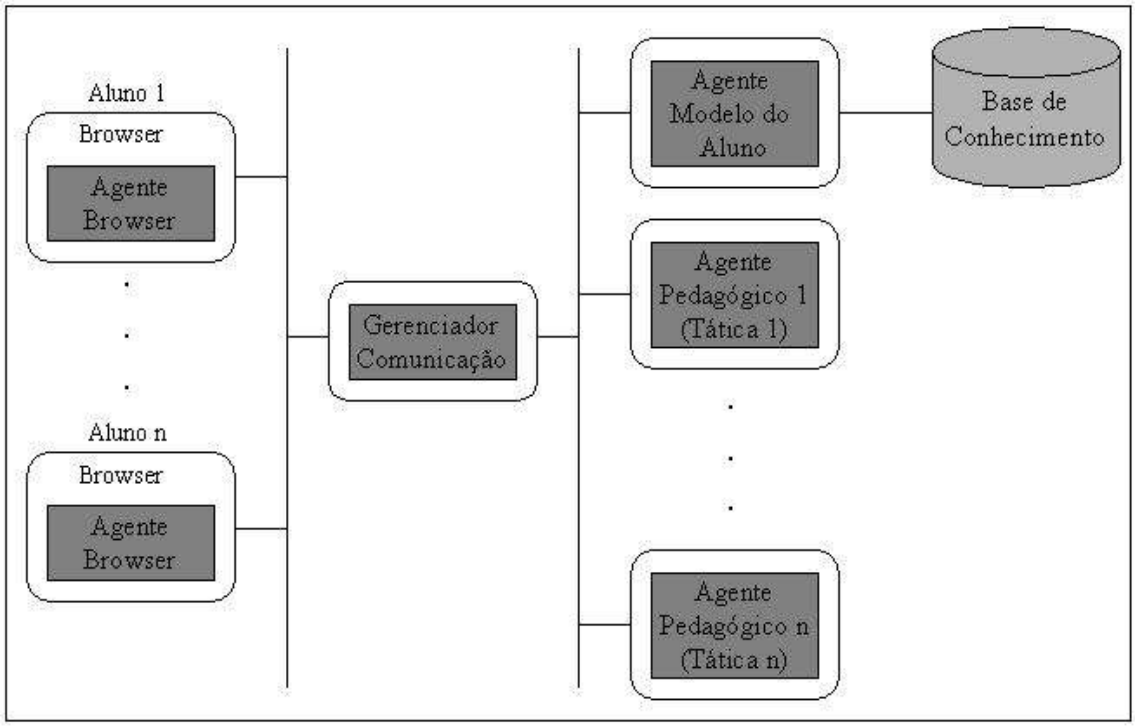

FIGURA 6.3 - Arquitetura do Sistema Tutorial proposto

O ambiente de aprendizado inteligente proposto, seguindo a Figura 6.3 contém um agente responsável pelo controle geral do sistema de multiagentes, denominado Modelo do Aluno. A comunicação entre os agentes é realizada pelo agente Gerenciador de Comunicação. Os outros agentes, chamados de agentes pedagógicos, são responsáveis pelas tarefas relacionadas com sua tática de ensino, na qual cada um pode ter suas tarefas específicas de acordo com o seu propósito. Porém estes agentes, já contem tarefas padrões, uma vez que toda a tática está relacionada a um conteúdo, sendo assim, funções como mostrar o conteúdo atual, avançar no conteúdo, mostrar conteúdo anterior, etc., já serão incorporados às tarefas do agente.

As características dos agentes do ambiente são definidas nas seções a seguir:

6.8.1 Agente Navegador

O agente navegador possui como função o controle do *browser* no ambiente do aluno. As mensagens são recebidas e enviadas a todos os agentes através do agente Gerenciador de Comunicação, ou seja, o agente apenas repassa as atividades do aluno na interface do sistema e recupera as informações a serem retransmitidas aos alunos.

6.8.2 Agente Gerenciador de Comunicação

O agente Gerenciador de Comunicação possui a função de comunicar-se com os outros agentes e encaminha a eles alguma ação acionada pelo aluno no tutor. As mensagens são recebidas e enviadas a todos os agentes através do mecanismo de comunicação Java/RMI (*Remote Method Invocation*).

6.8.3 Agente Modelo do Aluno

Este agente é responsável por construir e manter uma base de conhecimento que modele o estado cognitivo dos alunos que estejam conectados ou que tenham estado conectados ao sistema.

Quando a sessão tutorial inicia, o agente recebe informações sobre a identificação do aluno e recupera o Modelo do Aluno correspondente, assim como o agente corrente que estava comunicando-se com o aluno. Primeiramente o agente valida o *login* do aluno em um determinado curso. Se ao recuperar o Modelo do Aluno o agente constatar que nenhum agente pedagógico foi iniciado, é apresentado um conjunto de táticas (agentes) ao aluno para que este possa escolher como deverá ser o início de seu estudo. Após esta primeira fase, o agente selecionado passar a ser o responsável por todas as novas requisições efetuadas. No caso da avaliação, o agente também pode encaminhar ao aluno novamente o conjunto de táticas, pois pode ocorrer da próxima lição não conter a tática (agente) que estava sendo usada.

Todas as ações de acesso aos dados do aluno são realizadas pelo agente Modelo do Aluno, assim quando houver necessidade de um agente pedagógico atualizar o histórico do aluno, este agente repassa ao agente Modelo do Aluno os dados para este atualizar, assim como qualquer outra modificação no estado de ensino do aluno. As principais funções do agente Modelo do Aluno são:

- setar o estado atual do aluno;
- gerar todo o histórico do aluno;
- gerar o relatório de todos os passos do aluno;
- gerar o resultado das avaliações;
- selecionar as estratégias de ensino;
- verificar último acesso;
- verificar as táticas disponíveis para uma determinada lição.

6.8.4 Agentes Pedagógicos

Os agentes pedagógicos são gerados a partir de uma tática previamente definida pelo especialista do curso. Suas tarefas serão definidas conforme a necessidade do agente, porém como o tutor baseia-se na apresentação de conteúdo (apresentação de páginas HTML), algumas tarefas já são previamente definidas para todos os agentes pedagógicos. A seguir são descritas estas tarefas (estas tarefas também podem ser definidas como estratégias de ensino):

- Mostrar conteúdo atual: o agente comunica-se com o Modelo do Aluno para recuperar na base de conhecimento qual o conteúdo atual que está sendo apresentado ao aluno por este agente e encaminhar ao *browser* do aluno.
- Avançar: o agente comunica-se com o Modelo do Aluno para recuperar na base de conhecimento qual o próximo conteúdo que foi apresentado ao aluno com base no conteúdo atual mostrado pelo agente e encaminhar ao *browser* do aluno.
- Voltar: o agente comunica-se com o Modelo do Aluno para recuperar na base de conhecimento qual o conteúdo anterior que foi apresentado ao aluno

com base no conteúdo atual mostrado pelo agente e encaminhar ao *browser* do aluno.

- Opções: significa que o agente pode ter relação com opções a serem apresentadas ao aluno, porém não é obrigatório que este agente contenha opções, mas a tarefa é implementada para recuperar quais as opções este agente pode oferecer ao aluno, bem como invocar a ação que uma opção pode proporcionar.
- Atualizar histórico: a cada tarefa implementada, deve se registrar no Modelo de Aluno as ações que este está realizando sobre o tutor. Por exemplo, no caso em que é mostrado o próximo conteúdo, o agente insere o histórico do aluno com a data e hora de saída de conteúdo atual e insere o histórico do aluno com a data e hora de entrada de um próximo conteúdo.
- Comunicação: o agente implementa uma função que localiza onde o Gerenciador de Comunicação se encontra para repassar informações.
- Avaliação: o agente deve ter um mecanismo de avaliação ao aluno para a tática que o agente representa. Esta tarefa de avaliar não segue um padrão para todos os agentes, mas deve obrigatoriamente ser implementada.

7 Implementação do Modelo Adaptativo Proposto

A implementação aplicada a este trabalho, segue a estrutura definida no modelo proposto no capítulo anterior. A base do desenvolvimento utiliza a linguagem Java com o objetivo de manter a independência entre os diversos sistemas operacionais.

O protótipo para validação do modelo foi construído utilizando-se a estrutura de conteúdos e interface do Eletrotutor III [BIC 99].

A implementação foi dividida em três partes que serão descritas a seguir: ferramenta administrativa, modo tutorial e ferramentas visuais de auxílio ao aluno. Porém antes serão apresentadas as principais tecnologias utilizadas no desenvolvimento do sistema.

7.1 Tecnologias Utilizadas

A linguagem Java cada vez mais tem se destacado pela sua constante atualização e diversificação de produtos principalmente para a Internet. Sendo assim, a implementação (codificação) na parte servidor é toda ela desenvolvida em Java. Na parte cliente foi utilizado código HTML e *Javascript*. A seguir são mostrados alguns recursos da linguagem Java que foram utilizados e serão referenciados nas implementações juntamente com a comunicação entre os agentes.

7.1.1 *Servlet*

A plataforma Java é conhecida como uma tecnologia que vai até o inusitado, usada por diversos dispositivos como computadores, televisores, torradeiras, telefones e s*martcards*.

Java pode ter funcionalidade também nos servidores *Web* trazendo uma série de recursos e possibilidades. Uma das soluções Java para os servidores são os *servlets*, que tem o intuito de substituir, com vantagens, as atuais aplicações CGI. As aplicações CGI são aqueles programas que rodam nos servidores e tem a capacidade de receber dados vindos da Internet, processar estes dados e enviar uma resposta ao cliente. Na maioria das vezes que se preenche um formulário *on-line* pela Internet estes dados são processados por um programa localizado no servidor e as respostas retornadas ao remetente. Toda vez que o servidor *Web* recebe uma requisição CGI ele carrega um novo programa, executa-o, envia a resposta para o cliente e é finalizado.

No início da Internet um *site* que recebia 100 visitas por dia era um *site* bastante visitado, mas hoje alguns *sites* recebem mais de 50 mil acessos por dia e todo este processo de carregar, executar e finalizar um CGI resulta num significativo overhead do servidor. Os *servlets* são persistentes, independentes de plataforma e com eles podem ser usados todos os recursos disponíveis para as aplicações Java incluindo RMI, JDBC (*Java Database Connectivity*) e interações com *applets* Java.

Os *servlets* oferecem algumas vantagens sobre as aplicações baseadas em CGI:

- **Independência de plataforma**: Os *servlets* podem rodar em qualquer plataforma sem serem reescritos ou até mesmo compilados novamente. Os

Scripts PERL (linguagem de programação estável e multiplataforma), por serem interpretados, também possuem esta funcionalidade, mas são muito lentos. As aplicações CGI que necessitam de alta performance são escritas em C.

- **Performance**: Um novo programa CGI é carregado para cada requisição ao servidor. Isto quer dizer que se você tiver 10 requisições simultâneas, você tem 10 programas iguais na memória. Os *servlets* são carregados apenas uma vez e para cada nova requisição o *servlet* gera um novo *thread*. O método *init*() do *servlet*, assim como nas *applets*, ocorre apenas na primeira vez que a classe é carregada. É geralmente no método *init*() que, por exemplo, estabelecemos uma conexão ao Banco de Dados (BD). Cada uma das *threads* geradas pode usar a mesma conexão aberta no método *init*(). Este tipo de tratamento aumenta em muito o tempo de performance do *servlet*, já que se faz a conexão ao BD apenas uma vez e todos as outras requisições usam esta conexão.
- **Extensibilidade**: Com *serlvets* você pode criar aplicações muito mais modulares, tirar todo o proveito da orientação a objetos e utilizar o grande número de APIs (*Application Program Interface*) disponíveis pela *Javasoft* e terceiros.

7.1.2 JSP – *Java Server Pages*

Pode-se observar uma grande mudança nos *sites* da Internet. Já há algum tempo todos os grandes têm tentado interagir e descobrir o gosto de seus visitantes, guardando as suas preferências e características. Isto pode ser visto na própria Amazon.com, por exemplo, que a partir da busca de um livro apresenta outras publicações similares dentro do mesmo tema, focando assim a sua propaganda.

A tecnologia que permite este tipo de recurso é chamada de geração de páginas dinâmicas. Consiste em *scripts* que são inseridos nas páginas HTML e processados pelo servidor *Web* antes de serem enviadas para os usuários, separando desta forma a geração da interface da geração do conteúdo. Isto permite ao projetista mudar toda a interface do *site* sem precisar se preocupar com o conteúdo, já que este será gerado dinamicamente.

Entre os representantes mais populares desta tecnologia temos as páginas ASP (*Active Server Pages*) da Microsoft; PHP (br.php.net), que surgiu para fortalecer a geração de páginas em servidores Unix mas já está disponível para Linux e WindowsNT; o JScript Server Side da Netscape; e o ColdFusion.

Além de todas estas soluções citadas acima, mais uma começa a ser explorada, o *Java Server Pages* (JSP), que já disponibilizou a versão 1.1 de sua especificação. Assim como as páginas ASP ou o *PHP*, as páginas JSP contêm pequenas porções de código (*Scriptlets*) junto ao código HTML que são processadas pelo servidor e depois disto enviadas para o cliente. Existem algumas vantagens desta nova tecnologia frente às outras, mas vão ser discutidas aqui principalmente as diferenças entre JSP e ASP.

O ponto fraco do ASP é sem dúvida sua dependência do sistema operacional e do servidor *Web*. Ou seja, ele está casado com o Windows NT Server e o IIS *(Internet Information Services)*. E as soluções que permitem que estas páginas sejam usadas nos servidores *Unix* ainda não foram aprovadas pela maioria dos projetistas e desenvolvedores, devido principalmente à baixa performance. Outro ponto é que a

tecnologia ASP tem seu padrão fechado, ou seja, somente a Microsoft tem total conhecimento de como as coisas funcionam internamente.

Já as páginas JSP podem ser processadas por diversos servidores *Web*, como o Apache, NetscapeTM e IIS, e sob várias plataformas, como o Solaris, Linux, Mac OS, Windows NT, Windows 98 e outras. Uma outra vantagem é o padrão aberto para a indústria, contando com o apoio de empresas como Oracle, Netscape e IBM. Isto significa que mais empresas poderão fornecer servidores *Web* compatíveis com JSP e outros adendos para facilitar o trabalho dos desenvolvedores.

Para desenvolver a lógica em páginas ASP é possível o uso das linguagens VBScript, enquanto páginas JSP podem ser desenvolvidas com *scriptlets* escritos na linguagem Java ou JavaScript, ou usar *tags* XML (*Extended Markup Language*) que serão processadas pelo JSP *engine*.

A Javasoft, divisão da Sun que controla a linguagem Java, tem prometido a adoção de outras linguagens para a criação das páginas, o que não é um processo muito complicado. Esta diferença é importante, pois uma equipe que tem total conhecimento em Visual Basic e já possui um ambiente baseado em Microsoft deve pensar muito bem antes de uma mudança como esta.

Em contrapartida, se a equipe já possui algum conhecimento em Java e se existe a possibilidade de adoção de novas plataformas (Linux) ou novos servidores *Web*, então é uma alternativa interressante.

Uma das vantagens mais marcantes do JSP é a possibilidade de integração com todas as APIs Java já existentes (java.net, java.rmi, java.util, java.sql, java.bean, java.text, java.etc) e principalmente a utilização de componentes Java *Beans* e *Enterprise* Java *Beans*. A tecnologia de componentes tem sido adotada cada vez mais pela comunidade de desenvolvimento de *software* e tem respondido com presteza às necessidades. Todos os componentes de *software* que você já criou para suas aplicações Java podem ser utilizados pelas páginas JSP. Além disso, outros recursos como criação de *tags* personalizadas também é possível, o que permite maior flexibilidade, podendo até criar *tags* que sejam específicas para os seu componentes Java *Beans*.

E finalmente a promessa de aumento de performance aparece como mais um ponto positivo para a adoção desta tecnologia, pois uma página JSP é convertida em um *servlet* Java antes de ser executada e fica na memória do servidor já compilada. Se você tiver em seu servidor o compilador dinâmico *HotSpot*, melhor ainda, pois ele gera código nativo em tempo de execução, o que permite que a página seja executada tão rápido como aplicações nativas escritas em C (pelo menos é a promessa da Sun).

7.1.3 RMI

O RMI permite programar objetos distribuídos usando Java. Este objetos podem ser novos objetos ou simplesmente APIs já existentes encapsuladas em Java. O RMI estende o modelo: escreva uma vez, rode em qualquer lugar.

Neste modelo de objetos distribuídos, um objeto remoto é um objeto cujos métodos podem ser invocados de outra máquina virtual java, potencialmente em outro

host. São objetos locais aqueles que não são conhecidos fora da máquina virtual onde foram criados e cujos métodos não podem ser invocados remotamente.

O sistema RMI consiste de três camadas: *Stubs/Skeletons, Remote reference* e *Tranport.* A fronteira entre cada camada é definida por uma interface e um protocolo específico. Cada camada é independente das restantes podendo ser substituída por outra implementação sem afetar o sistema.

Para conseguir a transmissão transparente de objetos de um espaço de endereçamento para outro é usada a técnica de Serialização de Objetos. Os objetos passados entre espaços de endereçamento sejam como argumentos de métodos remotos ou como valores de retorno, têm de implementar a interface *Serializable*. Os objetos locais são passados por cópia. Os objetos remotos são passados por referência que é na realidade uma referência para um *stub*.

7.1.4 JDBC

A tecnologia JDBC é uma API para acessar base de dados por meios de comandos SQL (*Structured Query Language*). O JDBC vem resolver uma das fraquezas da linguagem Java, que era não possuir meios, através de uma API, de se conectar a fontes de dados externas. Por exemplo, antes não havia meios de se conectar a banco de dados locais ou remotos diretamente a partir dos *applets* Java.

Assim como o ODBC (*Open Database Connectivity*), o JDBC fornece aos desenvolvedores Java uma API comum para a maioria dos bancos de dados. O padrão JDBC define um conjunto de classes Java que executam operações padrão em um BD, como administração de conexões, comandos SQL, conjunto de resultados e metadados. O JDBC utiliza um *Drive Manager* que suporta múltiplos *drivers* de DB simultaneamente.

Algumas Características são relacionadas abaixo:

- **Segurança com *Stored Procedures*** - tira o direito de modificar diretamente os dados e permite ao usuário somente permissão de execução. Isto significa que a única interação possível com o BD é através dos procedimentos armazenados que você tenha escrito, lhe dando o controle sobre tudo que o usuário pode fazer.
- **Facilidade de Manutenção** - o fato de o Java ser Orientado a Objetos faz com que seja possível desenvolver componentes que podem ser incorporados em aplicações de BD; sua hierarquização facilita muito no sentido de manter seus componentes organizados. O utilitário JavaDoc também simplifica o desenvolvimento de um código automaticamente documentado.
- **Prototipação rápida** - a natureza Orientada a Objetos também facilita o desenvolvimento de componentes reutilizáveis e o aproveitamento de componentes de outros desenvolvedores uma vez que eles definiram seu conjunto de ferramentas, protótipos podem ser desenvolvidos rapidamente.
- **API consistente** – o JDBC possibilita desenvolver sistemas Java independentes de BD desenvolvidos por ela. Entre a variedade API de BD disponíveis para Java e de *drivers* que utilizam JDBC, é possível desenvolver aplicações e *applets* que conversam com muitos servidores de BD.

7.1.5 A Comunicação entre os agentes

Com a finalidade de definir uma forma de cooperação entre os agentes, foi elaborado um protocolo de troca de mensagens. Tal comunicação entre os agentes ocorre através de uma definição de mensagem baseada em KQML [BIC 99], buscando assim, construir uma arquitetura de agentes robusta e padronizada o quanto for possível e que permita a reutilização de códigos para os diferentes tipos de agentes.

As mensagens são definidas em duas partes:

- Cabeçalho da mensagem: segue o padrão KQML e;
- Conteúdo da mensagem: pode assumir variados formatos, dependendo da situação na qual o agente se encontra.

7.1.5.1 Mensagens

Neste tutor a troca de mensagens entre os agentes é feita pelo método ponto-a-ponto. Este método é apropriado à arquitetura proposta, já que os agentes e seus endereços estão cadastrados na base de conhecimento.

A implementação da mensagem [BIC 99], ilustrada na Figura 7.1, é composta de dois módulos: um formato geral que possui especificação KQML (classe KQML) e um formato específico que compõem a tarefa que o agente deve executar. A especificação da mensagem foi elaborada desta maneira, pois RMI permite a passagem de objetos criados pelo implementador.

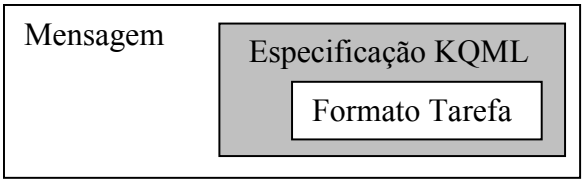

FIGURA 7.1 - Módulos da Mensagem

7.1.5.2 Implementação KQML

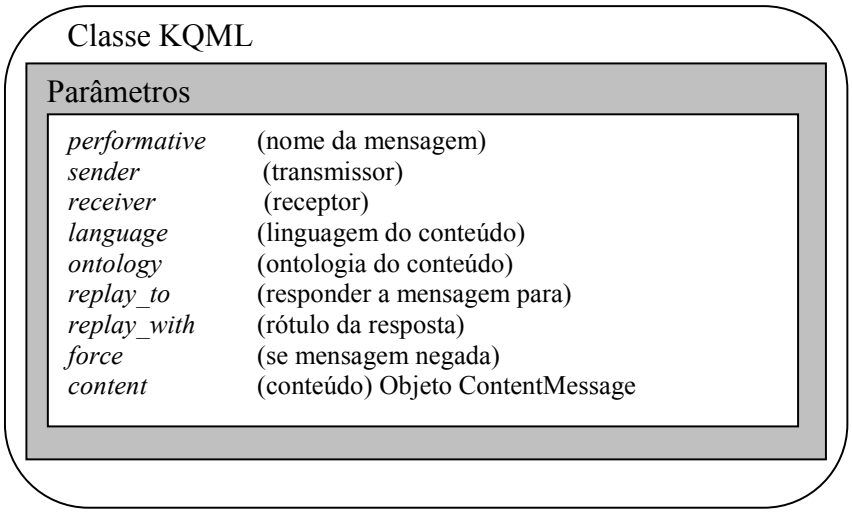

FIGURA 7.2 - Classe KQML

A Figura 7.2 identifica a estrutura da classe que implementa o KQML com os seus atributos. Dentre estes atributos três deles são fundamentais na implementação: *sender* (agente que está enviando), *receiver* (agente que vai receber a requisição) e *ontology* (nome de um método (funcionalidade) a ser executado por um agente).

7.1.5.3 Conteúdo das Mensagens

Seguindo o formato de mensagem KQML, é necessário especificar o parâmetro *content*, para isso, foi implementada a classe Content, como mostra a Figura 7.3, que possui todos os parâmetros necessários para compor uma tarefa. Tais parâmetros são utilizados conforme o agente e o tipo de mensagem que este deseja enviar.

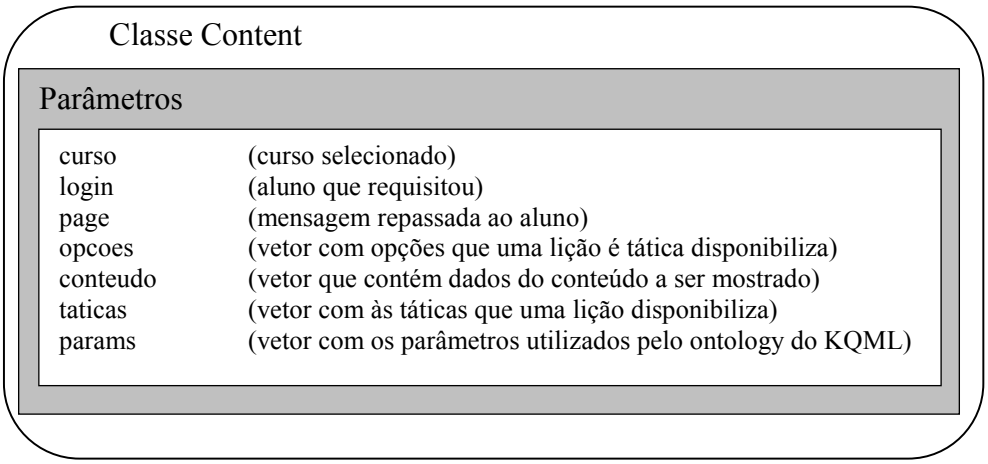

FIGURA 7.3 - Classe Content

7.2 Ferramenta Administrativa

A ferramenta administrativa foi o primeiro módulo desenvolvido em virtude do especialista ter facilidades em dispor um curso com suas lições, estratégias, táticas, conteúdo, etc., para que posteriormente o modo tutorial pudesse ser testado.

Todas as tabelas geradas na base de dados possuem um *link* na página principal, na qual o especialista pode posteriormente incluir, editar, remover e consultar os registros de uma tabela específica.

A Figura 7.4 ilustra a entrada na ferramenta administrativa com todas as tabelas da base de dados e uma introdução de funcionamento. Esta introdução é importante, pois nela está contido o código de curso *default*, pois a maioria das tabelas fazem referência ao código do curso, e para evitar que o especialista informe sempre este código a ferramenta parte do pressuposto de um código de curso default que pode ser alterado na página inicial.

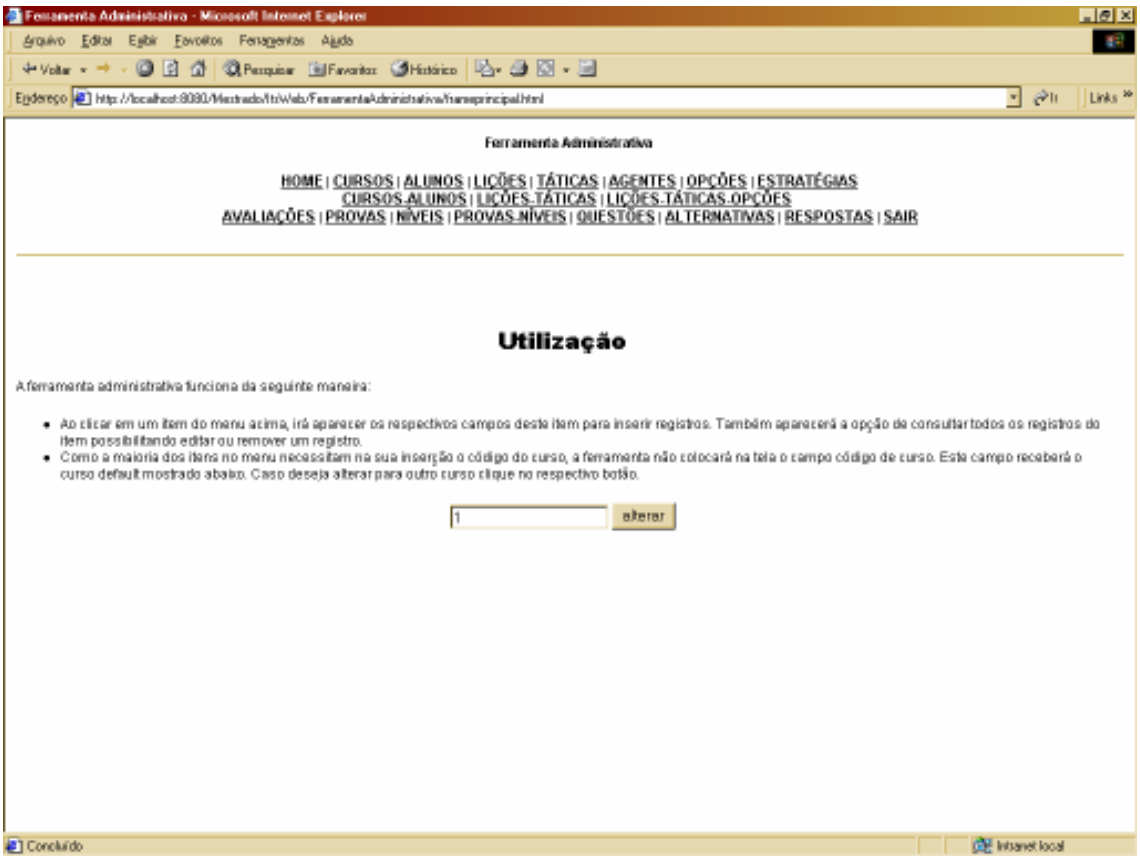

FIGURA 7.4 - Tela inicial da Ferramenta Administrativa

Ao selecionar uma tabela é apresentada uma descrição desta tabela, juntamente com os campos e uma descrição individual de cada campo. Neste momento, o especialista pode inserir um novo registro (ver Figura 7.5) ou consultar todos os registros da tabela. Neste caso uma nova página é gerada e o especialista pode editar ou remover um registro específico (ver Figura 7.6).

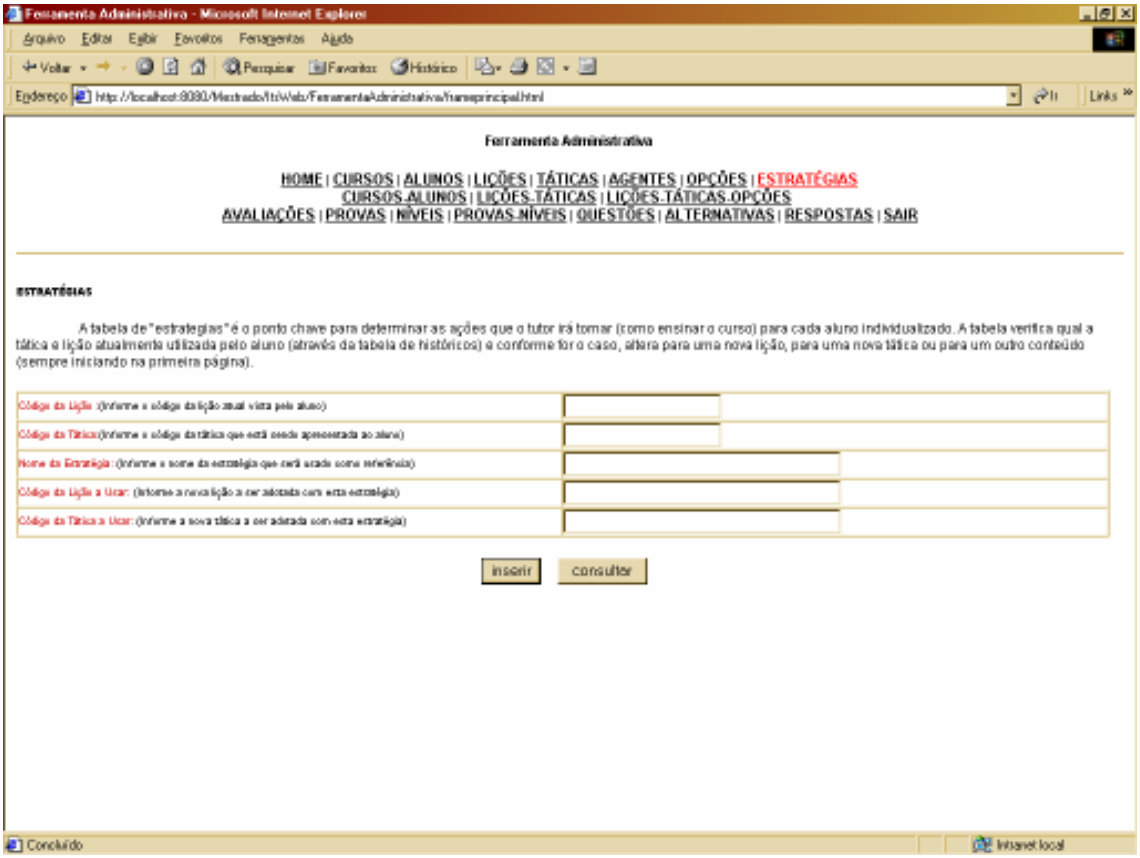

FIGURA 7.5 - Inserir registro na tabela de estratégias

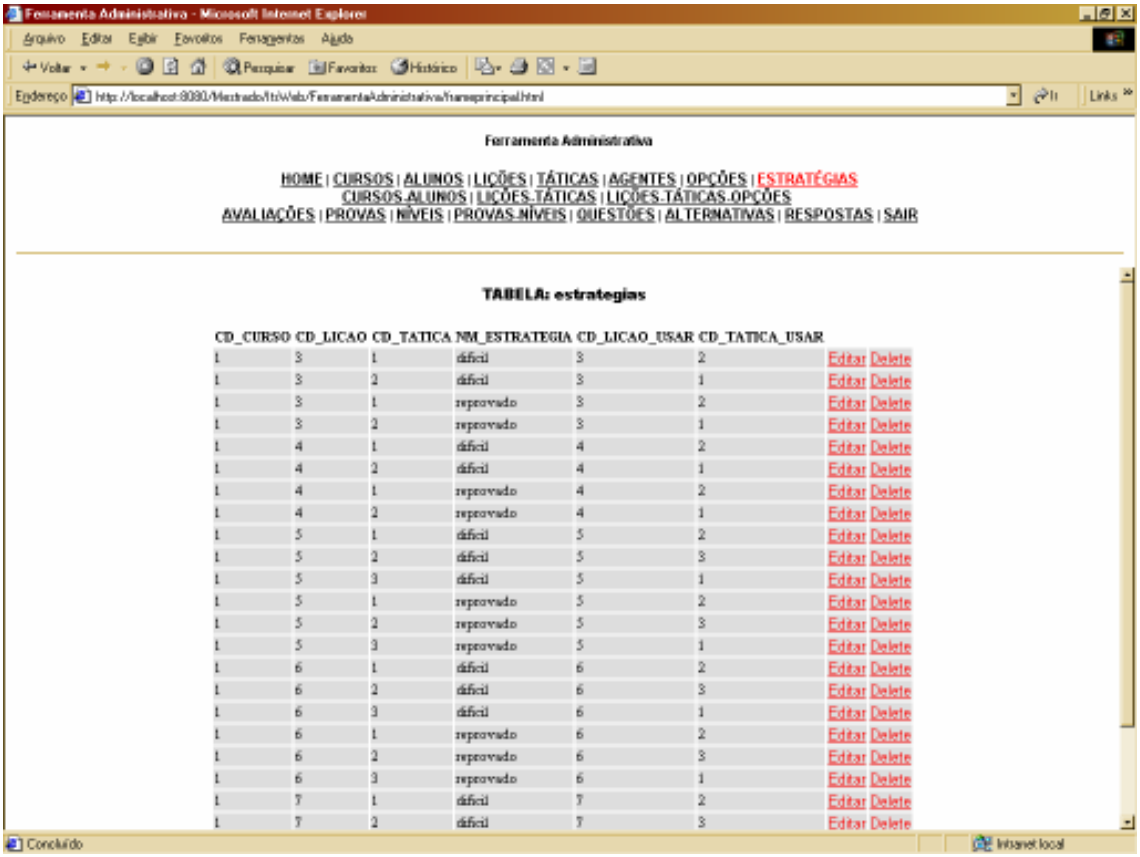

FIGURA 7.6 - Consulta de registros da tabela estratégias

7.2.1 Relatório do Modelo do Aluno

O especialista também dispõe de uma ferramenta que possibilita verificar todo o histórico do aluno. A implementação utiliza uma chamada a *servlet* para buscar no modelo do aluno primeiramente todos os alunos cadastrados em um determinado curso e após selecionar o aluno, mostra todo o histórico do aluno ordenado por data.

7.3 Modo Tutorial

O modo tutorial é a implementação principal do trabalho. Nesta fase a implementação visa apresentar o conteúdo do curso ao aluno provendo todo o mecanismo de adaptação, a aplicação das estratégias estipuladas pelo especialista e a avaliação do aluno sobre cada lição e tática apresentada.

7.3.1 Interface

A interface do tutor não foi planejada por não ser o objetivo deste trabalho, porém o tutor foi implementado de maneira que a camada de interface fique totalmente independente da aplicação. O exemplo usado é ilustrado neste trabalho. A interface utilizada é a mesma do Eletrotutor III (ver Figura 7.7). A integração da interface com o modo tutorial é referenciada por uma chamada de um *servlet*: http://www.host.com.br:porta/Mestrado/servlet/PkModoTutorial.Interface.C_Servlet?FPAGE=valor, na qual é dividido nas seguintes partes:

- **servidor** e **porta**: este dois parâmetros devem ser conhecidos por quem quer integrar o modo tutorial.
- **FPAGE**: este parâmetro pode assumir quatro (4) premissas básicas. A primeira é PLOGIN que efetua uma chamada para o tutor verificar o *login* do usuário; a segunda e terceira é PAVANCAR e PVOLTAR significando uma ação do aluno sobre o conteúdo e a quarta é PSAIR que informa o tutor o abandono do aluno no modo tutorial.

FIGURA 7.7 - Tela inicial do Eletrotutor

A restrição maior da interface é que esta deve ser criada em uma estrutura de *frames*, em que o *frame* principal deve ser nomeado para "itsWebabr" e todas as chamadas (exceto PSAIR que chama a tela inicial na página inteira do *browser*) devem ter como *target* "itsWebabr". Assim o modo tutorial é apresentado neste *frame*.

A estrutura do modo tutorial no *frame* "itsWebabr" é um outro *frame* e está ilustrado na Figura 7.8. O lado esquerdo contém as opções cadastradas pelo especialista para a lição e tática apresentada e no lado direito é apresentado o conteúdo da lição. As opções que não são chamadas internas (mudança de estratégia) e sim URLs, são apresentadas em outra janela, como por exemplo, está sendo apresentado um exercício ao aluno e ao clicar na opção calculadora, esta funcionalidade abre em outra janela possibilitando o aluno visualizar as duas telas.

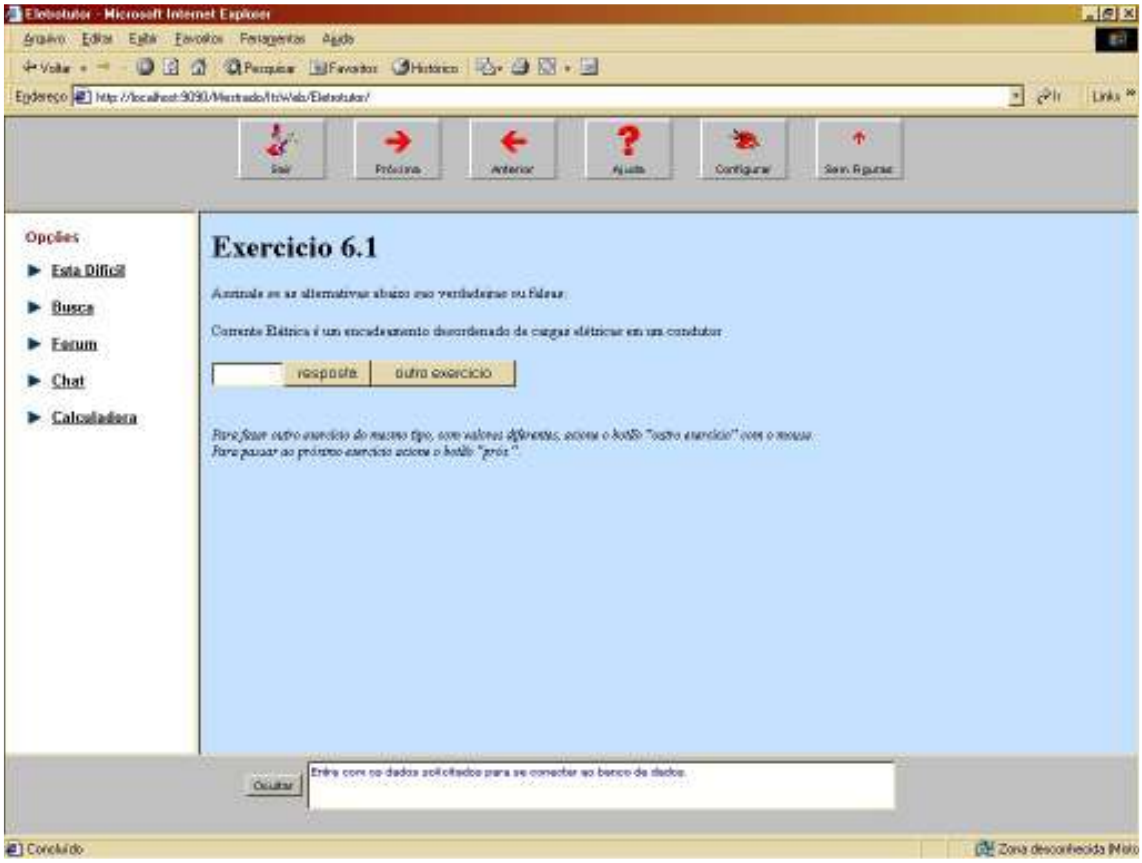

FIGURA 7.8 - Interface do Modo Tutorial

A classe responsável pela interface é um *servlet* denominado C_Servlet.java. Esta classe é responsável por receber requisições via *browser* e retornar conteúdo ao mesmo *browser*. Existe um controle de sessão, através da classe C_Sessão.java, que permite o *servlet* saber qual o *browser* (aluno) que fez determinada requisição e retorná-lo algum conteúdo, como mostra a Figura 7.9.

```
//recebe um objeto c_Kqml que tem a estrutura do login do aluno
C_Content c_Content = ( C_Content ) c_Kqml.getContent();

HttpSession session     = request.getSession( true );
//cria o objeto c_Sessao com os dados do aluno
C_Sessão c_Sessão = new C_Sessao( c_ContentSender.getCdCurso(),
C_ContentSender.getLogin() );
//seta a sessão usando um identificador Mestrado.ItsWeb
session.setAttribute( "Mestrado.ItsWeb", c_Sessao );
```

FIGURA 7.9 - Controle de Sessão

Todas as requisições para o *servlet* sempre contêm um parâmetro chamado FPAGE que identifica qual ação o aluno realizou no *browser*. Por exemplo, para identificar o *login* e a página que requisita a validação chama a URL */Mestrado/servlet/PkModoTutorial.Interface.C_Servlet?FPAGE=PIDENTIFICACAOALUNO*. O *servlet* encapsula os dados no protocolo KQML (ver Figura 7.10) e encaminha ao agente de comunicação para enviar ao agente responsável (neste caso

Modelo do Aluno) para validar o *login* e fica aguardando uma resposta a ser enviada para o aluno.

```
String   sCdCurso = request.getParameter( "FCDCURSO" );
String   sLogin    = request.getParameter( "FLOGIN" );
String    sSenha     = request.getParameter( "FSENHA" );

Vector  v_Params  = new Vector();
 v_Params.add( 0, sSenha );

C_Content        c_Content = new C_Content();
c_Content.setCdCurso( sCdCurso );
c_Content.setLogin( sLogin );
c_Content.setVecParams( v_Params );

c_Kqml            = new C_Kqml( "Identificacao Usuario" );
c_Kqml.setSender( "C_Servlet" );
c_Kqml.setReceiver( "C_ModeloAluno" );
c_Kqml.setOntology( "verificarLogin" );
c_Kqml.setContent( c_Content );

//chama a função que pega o registro do gerenciador de comunicação e repassa o c_Kqml
enviarKqml( c_Kqml );
```

FIGURA 7.10 - Enviando dados do *servlet* para o Gerenciador de Comunicação

7.3.2 Gerenciador de Comunicação

A classe responsável pelo gerenciamento da comunicação é C_Comunicação.java. A classe implementa o protocolo RMI para comunicação (ver Figura 7.11) entre as classes do sistema.

```
//registrando a classe C_Comunicacao
try {
 C_ComunicacaoImpl c_ComunicacaoImpl = new C_ComunicacaoImpl();
 Registry registry = LocateRegistry.createRegistry( 1234 );
 registry.rebind( "C_Comunicacao", c_ComunicacaoImpl );
}
catch( Exception exc ) {
}

//recuperando a referência para C_Comunicação
try {
 Registry registry = LocateRegistry.getRegistry( endereco, porta );
 C_Comunicação c_Comunicacao  = ( C_Comunicacao ) registry.lookup( C_Comunicacao"
);
}
catch( Exception exc ) {
}
```

FIGURA 7.11 - Registrando e recuperando o objeto RMI Comunicação

A classe quando é inicializada carrega da base de dados todos os agentes pertencentes ao sistema em um dicionário de dados e através do nome passado na estrutura KQML (sender e receive) identifica o agente que receberá a comunicação. Esta estrutura de agentes é modelado na classe C_Agente.java que contém o nome, endereço e a porta do agente.

7.3.3 Banco de Dados

A classe responsável pela comunicação com a base de dados é a C_DB.java. Esta classe inicializa (ver Figura 7.12) a conexão com o banco de dados (neste protótipo foi utilizado o banco MYSQL) e depois efetua requisições *SQLs* na base.

```
try {
 String sConexao  = "jdbc:mysql://" + sIp + ":" + sListen + "/" + sInstance + "?user=" +
sUsuario + "&autoReconnect=true";
 Class.forName( "org.gjt.mm.mysql.Driver" );
 this.c_Connection  = DriverManager.getConnection( sConexao );
}
catch( Exception exc ) {
}
```

FIGURA 7.12 - Conexão com o banco de dados

A implementação dos *SQLs* é uma parte interessante do trabalho pois foi retirado todo e qualquer SQL do código fonte. Este mecanismo utiliza a classe *ResourceBundle* que referencia um arquivo de requisições SQLs . Este arquivo possui um álias (nome) da requisição SQL e os parâmetros são representados pelo caractere de interrogação (?). Assim quando o C_DB for utilizar o SQL são repassados os parâmetros em um vetor e substituído no SQL como mostra a figura 7.13.

```
private ResourceBundle  rb_SQLs  = null;
//no pacote DB existe um arquivo chamado SQLs.properties
this.rb_SQLs = ResourceBundle.getBundle( "PkModoTutorial.BancoDados.SQLs" );
//exemplo de uma consulta no SQL.properites
//getNomeAluno = SELECT nm_aluno FROM alunos WHERE login = ?
//chamada do SQL
String  sQuery = this.rb_SQLs.getString( "getNomeAluno" );
//função que executa o SQL com os parâmetros, no caso o login do usuário.
//Vector v_Params = new Vector();
//v_Params.add( 0, sLogin );
getResultSet( sQuery , v_Params );
```

FIGURA 7.13 - Utilização de SQLs

Uma das maiores vantagens desta estrutura é que não existe a preocupação do parâmetro ser texto, número, data, etc, o que às vezes exige apóstrofos ou alguma formatação especial, pois o Java converte automaticamente.

7.3.4 Agente Modelo do Aluno e Agentes Pedagógicos

A principal classe do modo tutorial é a C_ModeloAluno.java, que é a responsável por modelar o aluno individualmente, instanciar a classe C_DB.java para acessar a base de dados e buscar todas as informações relevantes ao aluno.

O tutor tem o seu início a partir do *login* do aluno (*servlet* recebe a requisição e envia ao Modelo do Aluno). Após a validação de seus dados pelo agente Modelo do Aluno na base de dados, o tutor busca no modelo individual do aluno, qual a lição e tática atual utilizada e apresenta na tela. Caso seja o primeiro acesso do aluno no curso, o tutor busca na base de dados quais as táticas disponíveis para a primeira lição e pergunta ao aluno selecionar a melhor forma que este gostaria de começar seu estudo. Esta tela também é apresentada quando o aluno passa para a próxima lição que não tenha a tática que estava sendo apresentada ao aluno.

Após setar a tática o tutor apresenta o conteúdo da lição sobre esta tática e seta no seu ambiente o agente atual desta tática.

As estratégias são implementadas através das regras estipuladas na base de dados. A requisição de troca de estratégia pode acontecer diretamente pelo aluno que pode não estar compreendendo a forma de como o conteúdo da lição está sendo apresentado ou partir do próprio tutor a chamada para uma estratégia como, por exemplo, no caso do aluno ser reprovado em uma avaliação. O tutor busca na tabela de estratégias (pelo nome da estratégia) a lição e a tática que está sendo utilizada e passa a usar a lição e tática estipulada por esta estratégia. Esta mudança é feita através do histórico do aluno que é setado para a lição e tática a usar.

Os agentes pedagógicos (C_Domínio, C_Exemplo, etc.) sempre requisitarão a classe C_ModeloAluno.java para setar e buscar informações sobre o aluno. O agente assumido como padrão é o Modelo do Aluno, porém a partir do momento que o aluno opta por uma tática, o agente relacionado a esta tática torna-se ativo.

A mudança de agente pode ocorrer a qualquer momento em virtude das estratégias de ensino estipuladas pelo especialista. Assim cada agente no sistema tem uma *thread* que fica esperando alguma requisição chegar no Gerenciador de Comunicação (conforme a Figura 7.14) de maneira que o agente tem o seu lado ativo (enviar uma ação ao tutor) e o lado passivo (receber uma chamada para si).

```
//NO AGENTE
try {
    Registry registry = LocateRegistry.getRegistry( this.sEnderecoComunicacao,
this.iPortaComunicacao );
    C_Comunicacao c_Comunicacao  = ( C_Comunicacao ) registry.lookup( "C_Comunicacao"
);

    for( ; ; ) {
     synchronized( c_Comunicacao ) {
       while( c_Comunicacao.temKqml( "nome_do_agente" ) ) {
         //captura o Kqml e verifica o ontology a ser executado pelo agente
         verificarFuncao( this.c_Comunicacao.getKqml() );
       }
     }
     // se não tiver KQML para o agente fica na espera
     c_Comunicacao.setEspera();
    }
}
catch( Exception exc ) {
}

//NO GERENCIADOR DE COMUNICAÇÃO
//adormece o agente que chamou
public synchronized void setEspera() throws RemoteException {
   try {
     this.wait();
   }
   catch( Exception exc ) {
   }
}

//notifica todos a acordar e verificar se o KQML é para si
public synchronized void enviarKqml( C_Kqml c_Kqml ) throws RemoteException {
  try {
    this.c_Kqml  = c_Kqml;
    this.notifyAll();
 }
  catch( Exception exc ) {
  }
}
```

FIGURA 7.14 - Comunicação entre agentes

A grande vantagem da implementação dos agentes foi a criação de método chamado verificarFuncao() ilustrada na Figura 7.15. Este método procura na própria classe (parâmetro um) se existe o método (parâmetro dois) e seus argumentos (parâmetro três). O nome do método é passado no ontology da classe KQML e o parâmetro é sempre um objeto KQML. Sendo assim é necessário apenas registrar o método verificarFuncao() no objeto RMI, uma vez que este faz a tarefa de procurar os outros métodos da classe. A vantagem na programação é que não existe a necessidade de criar um método na classe e gerar os *stubs* e *skeletons* novamente.

```
/*verifica se a classe contém tal função com tais parâmetros e a chama caso exista*/
public void verificarFuncao( C_Kqml c_Kqml ) throws RemoteException {
  Vector  c_VecParams  = new Vector();
  c_VecParams.addElement( c_Kqml );
  //classe C_ArgumentHolder seta o tipo do objeto para cada instância no vetor
  C_ArgumentHolder c_ArgumentHolder = new C_ArgumentHolder( c_VecParams );
  //invoca a própria classe sobre o método (getOntology) e seus argumentos
  C_Invoker.dynaCall( this, c_Kqml.getOntology(), c_ArgumentHolder );
}
```

FIGURA 7.15 - Função verificarFuncao()

7.3.5 Avaliação

O tutor ao verificar o término da apresentação de todo o conteúdo de uma lição procura na base de dados a avaliação adotada para a lição e tática que acabou de ser apresentada. Como comentado na seção 7.3, o tutor verifica se existe uma prova de avaliação ou se apenas passando por todo conteúdo o aluno já está aprovado. Neste caso, o tutor seta o histórico de aprovações como aprovado e passa para a próxima lição. Caso exista uma prova a ser aplicada o tutor busca a configuração, ou seja, quantas questões de um determinado nível à prova contêm e que tipos de objetos de tela deverá ser mostrado. A seleção das questões é realizada aleatoriamente para evitar que seja montada sempre uma mesma prova (ver Figura 7.16). Esta seleção é realizada através de um vetor que contém todas estas questões e depois é criado um objeto random para ser utilizado sobre as posições no vetor. Toda a questão selecionada é retirada do vetor, aplicando-se novamente o método de escolha aleatória do índice no vetor, até a prova ser preenchida com o seu número de questões. Após este processo busca-se as alternativas da questão (caso exista).

```
//o vetor v_Questoes contém todas as questões para a avaliação
Random r_Questoes = new Random();
int iIndice;

//retorna um índice de 0 ao tamanho do vetor - 1
iIndice = r_Questoes.nextInt(  v_QuestoesNivel.size() - 1 );
//monta a questão sobre v_Questões( iIndice )
//retira do vetor a questão para não repetí-la na prova
v_Questões.remove( iIndice );
```

FIGURA 7.16 - Seleção aleatória de questões

Na hora do envio das respostas do aluno, o tutor verifica com a base a resposta correta, computa os acertos do aluno através do peso de cada questão, adiciona o peso sobre o acesso do conteúdo informado pelo especialista para esta avaliação e verifica a média da prova. Se o aluno for aprovado passa-se para a próxima lição, caso contrário o tutor verifica qual a estratégia a ser adotada para reprovação desta avaliação.

É importante destacar que uma vez o aluno aprovado em uma lição, caso ele retorne a lição para verificar algum conteúdo, ao avançar para o término da lição, mesmo se for outra tática, o aluno passará para a próxima lição sem ter que efetuar uma nova avaliação sobre a mesma lição.

7.4 Ferramentas Visuais de Auxílio ao Aluno

Como a implementação deste trabalho é voltada para ambientes *Web*, foram desenvolvidas ferramentas que provêem uma maior característica de um ambiente integrado a Internet. Outra preocupação foi a de desenvolver ferramentas que evitasse o aluno ter que sair de seu ambiente de ensino para utilizar uma determinada ferramenta, apesar que pela quantidade de recursos disponíveis na Internet nos tempos atuais, é normal o aluno procurar outras ferramentas de auxílio ao seu estudo para o complemento de seu aprendizado que não foram implementadas no tutor.

7.4.1 Calculadora

A ferramenta de calculadora foi colocada à disposição para o aluno caso seja necessário na resolução de exercícios. O desenvolvimento desta ferramenta utilizou *JavaScript* em uma página HTML, aplicando-se as funcionalidades de uma calculadora científica (ver Figura 7.17).

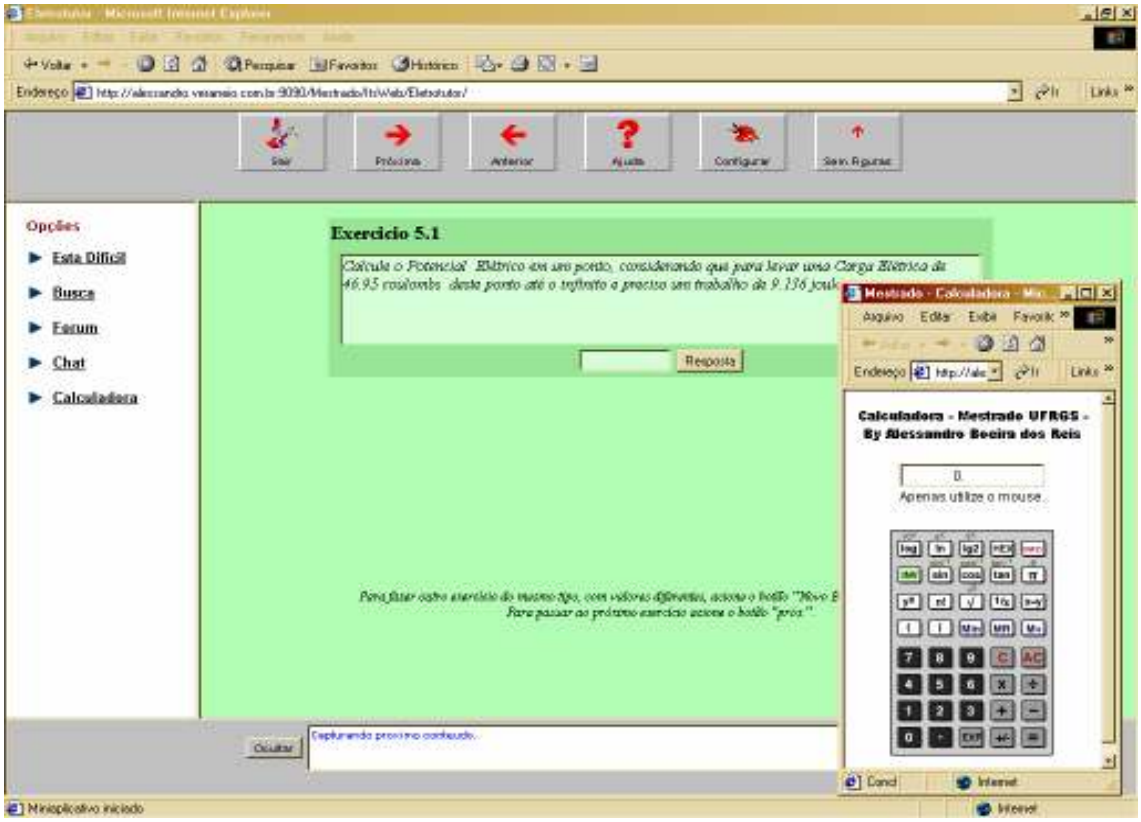

FIGURA 7.17 - O uso da calculadora

7.4.2 *Chat*

A ferramenta de *chat* (bate-papo) tem o propósito de permitir aos alunos logados de conversarem *on-line* diretamente no tutor (ver Figura 7.18). O *chat* é inicializado automaticamente com o *login* do aluno e caso este saia do tutor, automaticamente o *chat* invalidará este aluno de usá-lo.

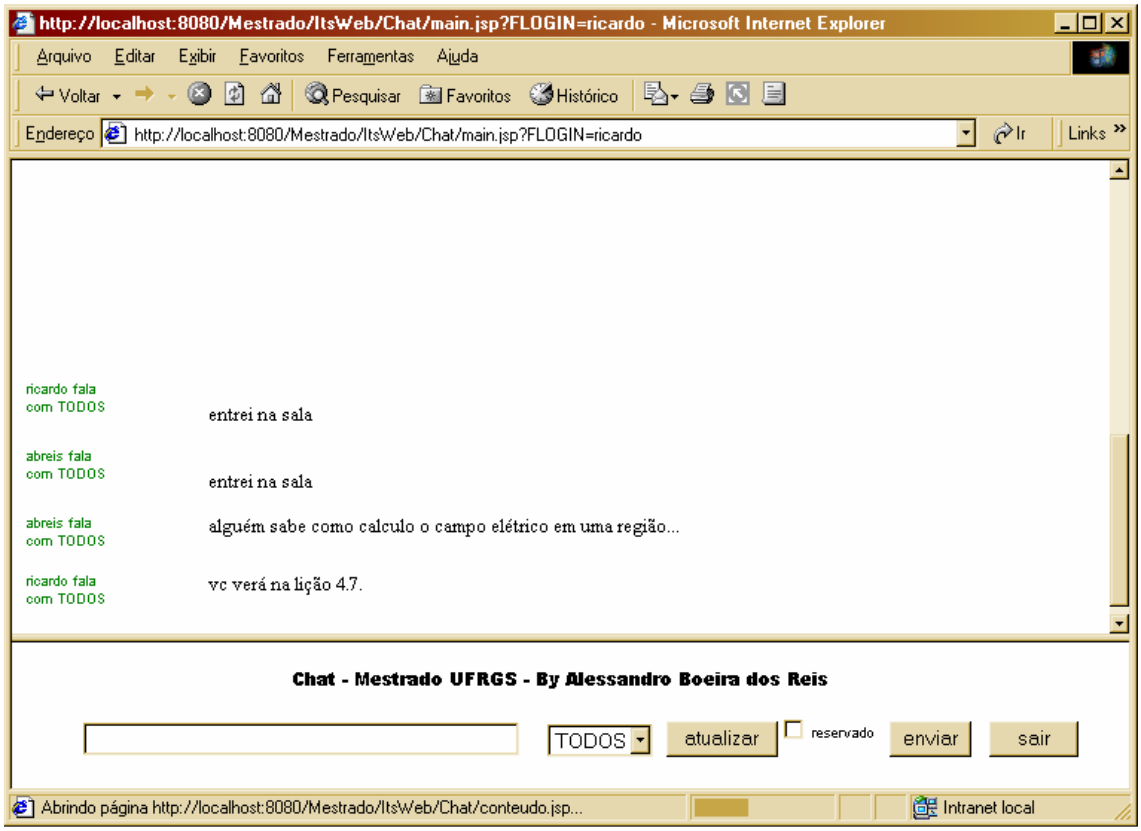

FIGURA 7.18 - Ferramenta de *chat*

A implementação utilizou os recursos de JSP. A estrutura JSP contém um controlador de sessão no *browser* do aluno que verifica a todo instante se tem alguma mensagem para este aluno. Esta verificação é feita por um controlador de aplicação que recebe todas as mensagens inserindo em um vetor. Quando chega uma nova mensagem, este controlador notifica todas as sessões abertas. Estas sessões têm o *login* do aluno e analisam se a mensagem contém o nome do aluno ou é uma mensagem para todos, sendo que nestes casos apresenta a mensagem para o aluno. Caso contrário, a mensagem é dita como reservada e é apresentada apenas para o aluno que está na mensagem.

7.4.3 Fórum

A ferramenta de fórum possibilita aos alunos discutirem o curso por assuntos conforme ilustrado na figura 7.19. O aluno sempre tem a possibilidade de enviar uma nova mensagem ou replicar uma mensagem existente apenas clicando sobre a mensagem e digitando o texto.

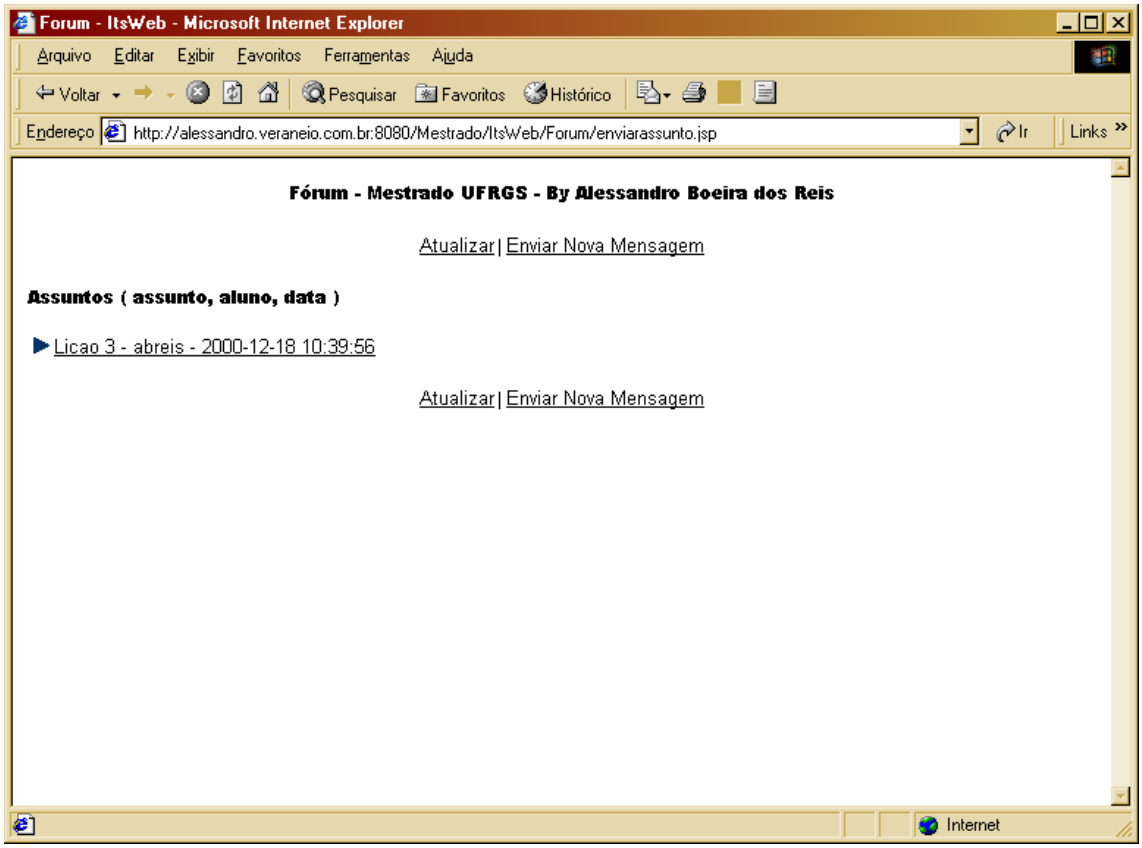

FIGURA 7.19 - Ferramenta de fórum

A implementação utiliza a tabela "fórum" (ver Tabela 8.1) como geração da página que mostra os assuntos. Cada assunto é cadastrado por uma ordem e a sua referência. No caso do assunto ser raiz a referência é nula. Assim o assunto com a maior ordem (último assunto inserido) e que tenha referencia nula é inicialmente inserido na árvore. Após é chamado uma função recursiva que procura todos os assuntos que tenham como referência o assunto atual, até todos os assuntos selecionados serem montados na árvore.

TABELA 7.1 - Tabela "forum"

CAMPOS	**DESCRIÇÃO**
CD_FORUM	Código do assunto. Este código é gerado por uma seqüência e indica a ordem de inserção. Quanto maior a ordem mais recente é o assunto (chave primária).
#CD_FORUM_REFERENCIA	Código do fórum que faz referência a este assunto (referência estrangeira).
LOGIN	Contém o *login* do aluno que está enviando uma mensagem.
DT_HR_PUBLICADO	Data e hora de quando a mensagem foi enviada.
ASSUNTO	Título principal da mensagem que posteriormente será apresentado com *link* da árvore de assuntos.

A página que mostra os assuntos, é um *servlet* que verifica o nível do assunto. Cada nível recebe uma imagem no início da linha. Se o assunto for do nível zero (0) a

imagem é uma seta indicando o inicio de um fórum, caso contrário se o nível for um (1) a linha recebe no início uma imagem transparente, se for dois (2) duas imagens transparentes e assim por diante.

Os textos das mensagens enviadas são gravados em um arquivo HTML nomeado com a ordem da mensagem. Estes arquivos ficam em um diretório no servidor e quando o aluno deseja replicar ou visualizar um assunto a *servlet* busca e apresenta o arquivo no *browser*. Todos estes arquivos no seu final contêm um espaço (*textarea*) para replicar a mensagem visualizada.

7.4.4 Busca

A ferramenta de busca permite ao aluno efetuar uma pesquisa sobre qualquer assunto como mostra a Figura 7.20.

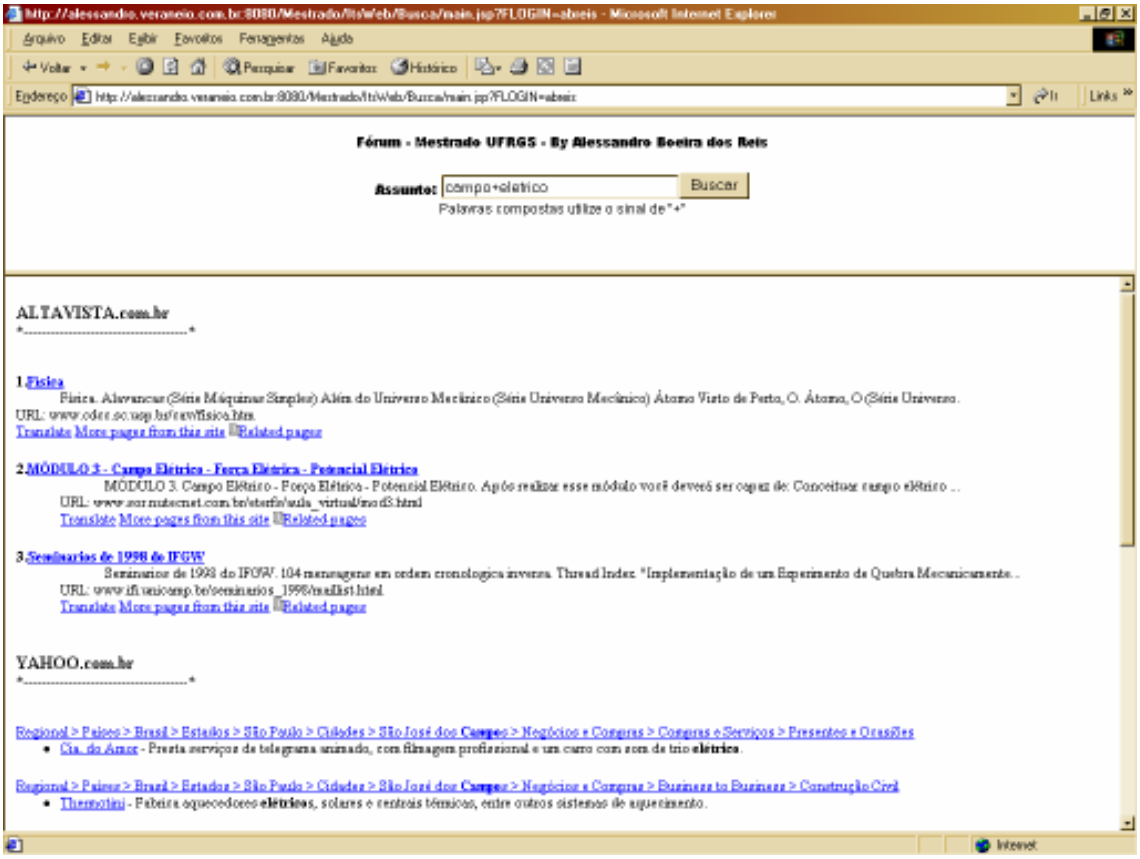

FIGURA 7.20 - Ferramenta de Busca

Este recurso dispara a busca em três (3) *sites* de busca na rede e após obter as respostas coleta os três primeiros *links* (caso exista) de cada *site* e monta em uma única página. A configuração dos *sites* foi feita aleatoriamente apenas para disponibilizar este recurso, porém a implementação é flexível e pode absorver qualquer *site* de busca.

Neste trabalho foi utilizado os *sites* AltaVista, *Yahoo* e *Google*. A tecnologia utilizada foi uma página JSP que chama uma classe em Java que dispara a URL de pesquisa sobre cada *site* repassando o assunto informado pelo aluno. A classe utiliza os mecanismos de conexão conforme ilustrado na figura 7.21.

```
URL url = newURL(http://www.altavista.com/cgi-bin/query?q= + sAssunto +
"&kl=XX&pg=q&Translate=on" );
URLConnection urlConn = url.openConnection();
InputStream is_Busca = urlConn.getInputStream();
InputStreamReader isr_Busca = new InputStreamReader( is_Busca );
//abre a resposta do site como um arquivo html
BufferedReader br  Busca = new BufferedReader( isr  Busca );
```

FIGURA 7.21 - Conexão com o site de busca AltaVista

8 Conclusões

A proposta de um sistema tutorial inteligente para *Web* que se adapta ao aluno, surge da necessidade crescente da Internet e a sua infiltração nos sistemas de educação à distância. Neste contexto, o presente trabalho apresentou um modelo, que posteriormente foi implementado, para dispor ao aluno o máximo de recursos disponíveis na rede Internet e fazer com que a diferença entre uma sala de aula e o computador seja a menor possível.

Porém tem-se a consciência que à presença do professor humano é de vital importância para o ensino de um aluno e que o ensino à distância ainda é utilizado mais para reforçar o aprendizado do aluno.

Um dos objetivos principais do trabalho foi sempre em prover a um especialista a possibilidade de comandar um curso conforme a sua experiência de ensino. Isto se tornou viável a partir da modelagem da base de dados e o desenvolvimento de uma ferramenta que possibilitasse o especialista de criar e gerenciar seu próprio curso.

O desenvolvimento das estratégias de ensino e dos recursos de rede proporcionam ao aluno recursos que a maioria dos sistemas similares ainda não conseguem prover. Esta afirmação é validada pelos diferentes estudos de casos realizados, na qual o presente trabalho tentou agregar funcionalidades interessantes que cada um podia contribuir e que dessem um diferencial no trabalho.

As estratégias ficaram totalmente centradas em relação à lição e a tática usada. A principal preocupação é de que os alunos aprendem de formas diferentes e que esta diversificação deve ser levada em consideração pelo tutor, ou seja, algum mecanismo de individualização deveria ser implementado.

Com o uso das ferramentas de rede disponibilizadas no protótipo, os alunos possuem recursos a mais no seu aprendizado, que auxiliam a compartilhar o conhecimento em uma amplitude maior do que uma restrita sala de aula. Outra vantagem é de integrar em um único ambiente estas ferramentas sem que o aluno tenha que abrir outras janelas para navegar em *chats*, buscas, etc., o que poderia acarretar do aluno se perder no ambiente.

Um grande apoio a este trabalho foi o estudo e a utilização do Eletrotutor III. Com este sistema pode-se avaliar e testar o presente trabalho de forma a validar um curso já existente e não fictício.

A sociedade de agentes e comunicação basicamente seguiu a estrutura do Eletrotutor III, porém provendo uma capacidade maior aos agentes em modificarem o estado atual de ensino. Uma mudança relevante é que para cada tática utilizada pelo especialista deverá ser implementado o agente responsável por atribuir a está tática suas funcionalidades.

A avaliação implementada permite ao especialista avaliar o aluno a cada lição apresentada. Como discutido no trabalho, não é interessante fazer uma avaliação no final do curso, pois seria complicado identificar em que lição o aluno não compreendeu.

Apesar de simples, o mecanismo desenvolvido para avaliação permite ao especialista criar diferentes tipos de avaliações para cada uma das lições e a sua forma de apresentação. A avaliação não foi um dos objetivos do trabalho, porém foi desenvolvida para o especialista dispor alguma forma de identificar o que o aluno está aprendendo dentro do curso e prover mais um recurso para adaptar as estratégias de ensino.

Como técnicas de programação utilizou-se basicamente a linguagem Java que hoje em dia se tornou uma das principais linguagens voltadas ao ambiente *Web* por prover diferentes ferramentas, além da sua estrutura de orientação a objetos amplamente divulgada.

As contribuições do trabalho estão inseridas nos projetos do Grupo de Inteligência Artificial (GIA) do Instituto de Informática e o seu resultado poderá ser agregado aos presentes projetos que estão em desenvolvimento.

8.1 Limitações e Trabalhos Futuros

As limitações observadas neste trabalho são:

- não se tornou viável a implementação sobre o tempo gasto pelo aluno para aprender determinado conteúdo;
- a avaliação foi implementada apenas para validar um aluno sobre uma determinada lição mas nenhum estudo aprofundado sobre este tópico foi realizado;
- a criação automática dos agentes a partir de uma tática ficou restrito em virtude de cada agente ter suas funcionalidades específicas, o que limita ao tutor de criar uma classe genérica;

Como trabalhos futuros pretende-se:

- dar continuidade dos estudos sobre sistemas adaptáveis na *Web*, principalmente em relação às limitações levantadas;
- avaliar o conteúdo que o aluno está utilizando dentro das ferramentas de rede disponíveis;
- testar a real produtividade do sistema em salas de aulas;
- prover algum mecanismo que permita ao aluno verificar o resultado de seu aprendizado;

Bibliografia

[BIC 99] BICA, Francine. **Eletrotutor III:** uma abordagem multiagente para o ensino à distância. 1999. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

[BRU 98] BRUSILOVSKY, P.; EKLUND, J. A Study of User Model Based Link Annotation in Educational Hypermedia. **Journal of Universal Computer Science,** [S.l.], v. 4, n. 4, p. 429-448, 1998. Disponível em: <http://www.jucs.org/jucs_4_4/a_study_of_user>. Acesso em: 16 out. 2000.

[BRU 97] BRUSILOVSKY, P. Distributed intelligent tutoring on the Web. In: INTELLIGENT EDUCATIONAL SYSTEMS ON THE WORLD WIDE WEB, 8., 1997, Kobe, Japan. **Proccedings…** [S.l.]: AIED, 1997.

[DAM 99] D'AMICO, C. B. **Aprendizagem Estática e Dinâmica em Ambientes Multiagentes de Ensino-Aprendizagem.** 1999. Tese (Doutorado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

[EKL 98] EKLUND, J.; BRUSILOVSKY, P. The Value of Adaptivity in Hypermedia Learning Enviroments: A Short Review of Empirical Evidence. In: ACM CONFERENCE ON HYPERTEXT AND HYPERMEDIA, 9., 1998, Pittsburgh, USA. **Proccedings…** [S.l.]: ACM, 2001.

[FUK 95] FUKUHARA, Y. et al. A Knowledge-based educational environment integrating conceptual knowledge and procedural knowledge in telecommunication service field. In: WORLD CONFERENCE ON EDUCATIONAL MULTIMEDIA AND HYPERMEDIA, ED-MEDIA, 1995, Graz, Austria. **Proceedings...** Charlottesville, VA USA: Association for the Advancement of Computing in Education (AACE), 1995.

[GIR 98] GIRAFFA, L. M. M. **Selecting Teaching Strategies using Pedagogical Agents.** 1998. Proposta de Tese (Doutorado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

[GIR 95] GIRAFFA, L. M. M. **Fundamentos de Teorias de Ensino-Aprendizagem e sua Aplicação em Sistemas Tutoriais Inteligentes**. 1995. Trabalho Individual I (Mestrado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

[LAN 97] LANDSTROM, M.; MAYER, D.; SHOBE, C. Indicators to Measure Performance in Distance Education, a Double-Edged Sword. In: WORLD ICDE CONFERENCE, 18., 1997. **Proceedings...** Pennsylvania: Pennsylvania State University, 1997.

[LUC 97] LUCENA, M. **Um Modelo de Escola Aberta na Internet**: Kidlink no Brasil. Rio de Janeiro: Brasport, 1997.

[KAY 96] KAY, J.; KUMMERFELD, R.J. User model based filtering and customisation of web pages. In: USER MODELING, UM, 5., 1996, Kailua-Kona, Hawai. **Proceedings…** Wien: Springer-Verlag, 1996.

[MAR 99] MARINILLI, M.; MICARELLI, A.; SCIARRONE, F. A Case-Based Approach to Adaptive Information Filtering for the WWW. In:

USER MODELING, UM, 7., 1999, Banff, Canadá. **Proceedings…** Wien: Springer-Verlag , 1999.

[MOO 96] MOORE, M. G.; KEARSLEY, G. **Distance Education**: a Systems View. Belmont, California, USA: Wadsworth Publishing Company, 1996.

[MOR 97] MORAES, D. **A Dialética das mídias Globais in Globalização, Mídia e Cultura Contemporânea**. Campo Grande, MS: Letra Livre Ed., 1997.

[MUL 98] MULLIER, D. J.; MOORE D. J. A Web based Intelligent Tutoring System. In: NETWORKING FOR THE MILLENNIUM, NETIES, 1998, Leeds, West Yorkshire, Reino Unido. **Proccedings…** Leeds: EATA, 1998.

[NAK 95] NAKABAYASHI, K. et al. An intelligent tutoring system on World-Wide Web: Towards an integrated learning environment on a distributed hypermedia. In: WORLD CONFERENCE ON EDUCATIONAL MULTIMEDIA AND HYPERMEDIA, ED-MEDIA, 1995, Graz, Austria. **Proceedings…** Charlottesville, VA USA: Association for the Advancement of Computing in Education (AACE), 1995.

[NUM 92] NUNES, I. B. **Noções de educação à distância**. 1992. Disponível em: <http://www.ibase.org.br/~ined/ivonio1.html>. Acesso em 13 out. 2000.

[RUS 95] RUSSEL, S.; NORVIG, P. **Artificial Intelligence a Modern Approach**. [S.l.]: Prentice-Hall, 1995.

[PER 99] PEREIRA, A. S. **Um Agente para Seleção de Estratégias de Ensino em um Ambientes Educacionais na Internet**. 1999. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

[RIB 98] RIBEIRO, M. B.; NOYA, R. C.; FUKS, H. Um Ambiente de Aprendizado Cooperativo para Web. In: CONGRESSO RIBIE, 4., 1998, Brasília, DF. **Anais…** Brasília: [s.n.],1998.

[RUS 95] RUSSEL, S.; NORVIG, P. **Artificial Intelligence a Modern Approach**. [S.l.]: Prentice-Hall, 1995.

[RIT 97] RITTER, S. PAT Online: A Model-tracing tutor on the World-Wide-Web. In: INTELLIGENT EDUCATIONAL SYSTEMS ON THE WORLD WIDE WEB, 8., 1997, Kobe, Japan. **Proccedings…** [S.l.]: AIED, 1997.

[SIL 97] SILVEIRA, R. A. **Ambientes inteligentes distribuídos de aprendizagem**. 1997. Exame de Qualificação (Doutorado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

[SIL 96] SILVEIRA, R. A. **Eletrotutor II:** Um tutor na Web. 1996. Trabalho Individual (Doutorado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

[SIL 94] SILVEIRA, R. A. Sistemas tutoriais inteligentes: Inteligência Artificial em Educação. **Educação**, Porto Alegre, v. 17, n.27, p.127 - 135, 1994.

[SIL 92] SILVEIRA, R. A. **Inteligência Artificial em Educação**: um modelo de sistema tutorial inteligente para microcomputadores. 1992. Dissertação (Mestrado em Educação) - Faculdade de Psicologia - PUCRS, Porto Alegre.

[WEB 97] WEBER, G.; SPECHT, M. User Modeling and Adaptive Navigation Support in WWW-based Tutoring Systems. In: USER MODELLING, UM, 6., 1997, Sardinia, Italy. **Proceedings…** Sardinia: Springer-Verlag, 1997.

[WEB 95] WEBER, G.; MÖLLENBERG, A. ELM programming environment: A tutoring system for LISP beginners. In: WENDER, K. F.; SCHMALHOFER, F.; BÖCKER, H.-D. (Ed.). **Cognition and Computer Programming**. Norwood, NJ: Ablex Publishing Corporation, 1995.

[ZUK 99] ZUKERMAN, I.; ALBRECHT, D. W.; NICHOLSON, A.E. Predicting User's Requests on the WWW. In: USER MODELING, UM, 7., 1999, Banff, Canadá. **Proceedings…** Wien: Springer-Verlag , 1999.