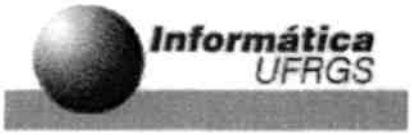
Universidade Federal do Rio Grande do Sul
Instituto de Informática



OFICI0/CPE/PAPED/CAPES No. 699/99

Dissertação de Mestrado

Eletrotutor III - uma abordagem Multiagente para o Ensino à Distância

Por: Francine Bica

Orientadora: Rosa Maria Vicari

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

Eletrotutor III - Uma Abordagem Multiagente para o Ensino à Distância

por

FRANCINE BICA

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

Prof. Rosa Maria Viccari
Orientadora

Porto Alegre, janeiro de 2000. Trabalho beneficiário de auxílio financeiro da CAPES - Brasil.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Bica, Francine

Eletrotutor III: Uma abordagem multiagente para o Ensino à distância / por Francine Bica. - Porto Alegre : PPGC da UFRGS, 2000.

78 f.: il.

Dissertação (Mestrado) - Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR - RS, 1999. Orientador: Viccari, Rosa M.

1. Sistemas multiagentes. 2. Educação à Distância. 3. Modelo de Aluno. 4. Java. I. Viccari, Rosa Maria. II. Título.

"A mente que se abre a uma nova idéia
jamais voltará ao seu tamanho original"
Albert Einstein

Agradecimentos

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

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

Aos professores e colegas Ricardo Silveira, Magda Bercht, Lúcia Giraffa e Michael Móra 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, obrigada pelo incentivo e carinho.

Meus agradecimentos ao apoio financeiro dado pelo CNPq e Capes, entidade do Governo Brasileiro voltada para a formação de recursos humanos, que viabilizou esta dissertação.

Sumário

Lista de Abreviaturas

API	*Aplication Program Interface*
BCV	Base de Conhecimento Virtual
BDI	*Belief, Desire* and *Intention*
BDV	Base de Dados Virtual
EAD	Educação à Distância
ELP	*Extended Logic Programming with explicit negation*
FTP	*File Transfer Protocol*
HTML	*Hyper Text Markup Language*
HTTP	*Hyper Text Transfer Protocol*
IA	Inteligência Artificial
ILE	*Intelligent Learning Environment*
IP	*Internet Protocol*
ITS	*Intelligent Tutoring Systems*
KIF	*Knowledge Interchange Formai*
KQML	*Knowledge Query and Manipulation Language*
KSE	*Knowledge Sharing Effort*
LAN	*Local Área Network*
msg	mensagem
OO	Orientação a objetos
Pg-	Página
RMI	*Remote Method Invocation*
RPC	*Remote Procedures Call*
SLX	*Selected Linear resolution for eXtended programs*
SMA	Sistemas multiagentes
SQL	*Structured Query Language*
SPSS	*Statisical Package for Social Sciences*
URL	*Uniform Resource Locator*
VRML	*Virtual Reality Modeling Language*
vs.	Contra
WAN	*Wide Área Network*
WFSX	*Well-Founded Semantics eXtended for explicit negation*
WWW	*World Wide Web*

Lista de Figuras

Lista de Tabelas

Resumo

Esta dissertação de mestrado está inserida no trabalho desenvolvido pelo grupo de pesquisa GIA/UFRGS, sob a orientação da Prof.a. Rosa Maria Viccari e situa-se na área de Inteligência Artificial aplicada à Educação à Distância apresentando também características de Sistemas Distribuídos e Sistemas multiagentes.

O objetivo deste trabalho é propor uma arquitetura de agente a ser aplicada a uma nova abordagem do ambiente Eletrotutor [BIC 98]. Tal abordagem contempla a utilização de sistemas multiagentes interconectados pela Rede Mundial de Computadores - a *Internet* - como instrumento de Ensino à Distância. As características e a arquitetura interna dos agentes da sociedade do Eletrotutor III são definidas nesta dissertação de mestrado, assim como o protocolo de comunicação entre estes.

O ambiente Eletrotutor foi implementado primeiramente no contexto de uma dissertação de mestrado [SEL 92] e [SIL94], com o objetivo de servir como instrumento de pesquisa. Para verificar a eficácia do uso de ambientes tutoriais inteligentes na escola. Para tanto, buscou-se implementar um sistema que abordasse uma unidade de estudo no campo do ensino da Física, mais precisamente da Eletricidade, no estudo da Lei de Ohm e suas aplicações. Posteriormente, uma segunda versão foi implementada, também com objetivos de avaliação de sua efetividade como ferramenta educacional, porém sob outro paradigma. O conteúdo do Eletrotutor é constituído por oito unidades, formadas por lições, exemplos e exercícios.

A sociedade de agentes implementada no Eletrotutor III propõe agentes autônomos que possam se comunicar uns com os outros. Cada agente possui funções e objetivos dentro de sua especialidade. Como objetivo principal dos agentes tem-se o aprendizado do aluno através da cooperação e comunicação entre os agentes.

Palavras-Chave: Inteligência Artificial, Educação à Distância, Sistemas Multiagentes, Ambiente de Ensino Inteligentes

TITLE: "ELETROTUTOR: MULTIAGENT APPROACH FOR DISTANCE LEARNING "

Abstract

This master's degree dissertation was developed within the research group GIA/UFRGS. The work applies Artificial Intelligence concepts to distance learning and borrows caracteristics from Distributed and Multiagent Systems.

The main goal of this work is to propose an architecture of agents to be applied to a new approach of the Eletrotutor environment [BIC 98]. Such approach contemplates the use of multiagent systems interconnected by the Internet as an instrument for Distance Learning. The characteristcs and the internal architecture of the agents in the society of Eletrotutor **III** are defined as well as the communication protocol between them.

The Eletrotutor environment was firstly implemented in the context of another master's degree dissertation [SIL 92] [SEL94], with the objective of serving as a research instrument whose purpose was to verify the effectiveness of using intelligent tutorials environment in the school. A first version of the system was implemented to support the teaching of an Eletricity study unit - Ohm's Law and its applications. Later, a second version was implemented, also to evaluate its effectiveness as an educational tool, under other paradigm. Eletrotutor is constituted by eight units, formed by lessons, examples and exercises.

The society of agents implemented in Eletrotutor III proposes autonomous agents, that can communicate one with the others. Each agent has specific functions and objectives on its specialty. These agents have the main goal of supporting the student's learning process through cooperation and communication.

Keywords: Artificial Intelligence, Distance Learning, Multiagent System, Intelligent Learning Environment.

1 Introdução

Desde o início da era da informática tem-se pensado no computador como uma ferramenta educacional. Entretanto, a aplicação em larga escala do computador no processo educacional foi permitida apenas recentemente com os custos mais baixos e a disseminação das novas tecnologias de comunicação.

Um importante passo foi dado com o desenvolvimento de Sistemas Tutoriais Inteligentes (ITS - *Intelligent Tutorial System)* [BAR 82] e [CLA 90]. O principal objetivo de um ITS é reproduzir o comportamento inteligente (competente) de um tutor humano e poder adaptar sua maneira de ensinar ao ritmo de aprendizagem do aluno.

Com a evolução da Informática na Educação, os ambientes computacionais de ensino passaram a ser vistos como uma excelente alternativa também para a Educação à Distância (EAD), dando a este campo da ciência um novo vigor. A EAD tem por objetivo o desenvolvimento de ambientes e de metodologias que propiciem o aprendizado remoto, isto é, que um ou mais alunos possam vivenciar experiências de aprendizagem em local fisicamente diferente do qual o ambiente e os recursos instrucionais se encontram. A natureza das pesquisas nesta área têm caráter nitidamente multidisciplinar, unindo esforços das Ciências da Educação, da Psicologia, da Engenharia e da Ciência da Computação.

Empresas e instituições de ensino têm investido grandes recursos em pesquisas relacionadas à utilização de computadores em ambientes de EAD como solução para o atendimento a uma demanda crescente, oferecendo novas oportunidades educacionais. A *World Wide Web* (WWW) se apresenta como uma tecnologia capaz de atender às expectativas dos pesquisadores da área de ensino/aprendizagem à distância, proporcionando soluções para o problema do oferecimento de educação e treinamento em larga escala, a custos mais acessíveis que os atuais, permitindo a publicação de material didático, aplicação de tutoriais, aplicação de provas e testes, comunicação com os estudantes e apresentação de aulas à distância (conferência multimídia).

As pesquisas apontam para o uso de recursos propiciados pela Inteligência Artificial (IA), a fim de prover aos sistemas computacionais de ensino, capacidade de adaptação ao contexto e de personalização do ambiente de acordo com as características do aluno, além de permitir um alto grau de interatividade entre o ambiente e os usuários e um controle de sessões de ensino em ambiente multi-usuários. A introdução das técnicas de IA nestes ambientes tem a finalidade de propiciar mecanismos de modelagem do processo de ensino bem como do estado cognitivo do estudante [MAS 96].

Na Inteligência Artificial Distribuída (IAD), uma sub-área de IA, vem se desenvolvendo com o objetivo de estudar soluções de problemas cooperativos através de um grupo descentralizado de processos ou agentes. A IAD divide-se em três áreas: Resolução de Problemas Distribuídos (RPD), Inteligência Artificial Paralela (IAP) e Sistemas Multiagentes (MAS). RPD interessa-se pela decomposição de problemas em módulos que cooperam entre si e dividem conhecimento e controle. IAP interessa-se principalmente por problemas de performance, preocupando-se em desenvolver

linguagens e algoritmos de computação paralela. MAS caracteriza-se pela existência de um certo número de agentes autônomos, heterogêneos e potencialmente independentes, trabalhando juntos para resolverem um problema.

Os avanços mais recentes no campo dos ambientes de aprendizagem inteligentes, têm proposto o uso de arquiteturas baseadas em sociedades de agentes. Os princípios dos sistemas multiagentes têm mostrado um potencial bastante adequado ao desenvolvimento de sistemas de ensino, devido ao fato de a natureza do problema de ensino-aprendizagem ser mais facilmente resolvido de forma cooperativa. Além disso, ambientes de ensino baseados em arquiteturas multiagentes possibilitam suportar o desenvolvimento de sistemas de forma mais robusta, mais rápida e com menores custos, tornando-os mais atrativos, do ponto de vista de seu aproveitamento real, não ficando restrito a um protótipo.

Assume-se o conceito de [BRA 97] que descreve um agente como uma entidade de software que funciona de forma contínua e autônoma em um ambiente em particular, geralmente habitado por outros agentes, e que seja capaz de intervir no seu ambiente, de forma flexível e inteligente, sem requerer intervenção ou orientação humana constantes. De um modo ideal, um agente que funcione continuamente por longos períodos de tempo, deve ser capaz de aprender com a experiência e, se ele habita um ambiente com outros agentes, seja capaz de comunicar-se e cooperar com eles, e ainda mover-se de um local para outro.

Esta dissertação de Mestrado propõe uma sociedade de agentes baseada em [SIL 98], um protocolo de comunicação entre os agentes da sociedade e uma arquitetura interna de agente a ser aplicada na sociedade elaborada, tal abordagem contempla a utilização de sistemas multiagentes interconectados pela Rede Mundial de Computadores - a *Internet* - como instrumento de Ensino à Distância.

O ambiente Eletrotutor foi implementado primeiramente no contexto de uma dissertação de mestrado [SIL 92] e [SEL94], com o objetivo de servir como instrumento de pesquisa cujo propósito foi verificar a eficácia do uso de ambientes tutoriais inteligentes na escola. Para tanto, foi implementado um sistema que abordasse uma unidade de estudo no campo do ensino da Física, mais precisamente da Eletricidade, no estudo da Lei de Ohm e suas aplicações. Posteriormente, uma segunda versão foi implementada, também com objetivos de avaliação de efetividade como ferramenta educacional, porém sob outro paradigma. O conteúdo do Eletrotutor é constituído por oito unidades, formadas por lições, exemplos e exercícios.

A sociedade de agentes implementada no Eletrotutor III propõe agentes autônomos que possam se comunicar uns com os outros, na qual cada agente possui funções e objetivos dentro de sua especialidade. Como objetivo principal tem-se o aprendizado do aluno propiciado através da cooperação e comunicação entre os agentes humanos e artificiais. As características e a arquitetura interna dos agentes desta sociedade são definidas nesta dissertação de mestrado, assim como o protocolo de comunicação entre estes.

Este trabalho está estruturado da seguinte forma:

- o capítulo 2 apresenta o estado da arte em relação a Educação à Distância, características, vantagens e desvantagens da sua utilização;

- o capítulo 3 introduz o conceito de agentes, salientando aspectos relevantes a este trabalho, como características destes e métodos de comunicação;
- o capítulo 4 apresenta três ambientes de ensino à distância na Web com a abordagem multiagente com a finalidade de aprofundar o conhecimento neste campo;
- o capítulo 5 apresenta as versões anteriores do Eletrotutor, assim como as validações das abordagens anteriores;
- o capítulo 6 descreve a nova versão do Eletrotutor em detalhes;
- o capítulo 7 trata dos aspectos da implementação da nova versão do Eletrotutor e apresenta o estado atual do protótipo e,
- o capítulo 8 apresenta conclusões, limitações e trabalhos futuros.

2 Educação à Distância

A Educação à Distância (EAD) já contabiliza mais de um século de existência. Desta época em diante, a EAD se desenvolveu nos mais variados ferramentais pedagógicos, de acordo com fatores como: características da escola e dos professores, o tipo de curso ministrado, distribuição geográfica entre escola e alunos, principalmente, a tecnologia disponível e a relação custo/benefício para a utilização da mesma.

Walter Perry [PER 87] afirma que a característica básica da educação a distância é o estabelecimento de uma comunicação de dupla via, na medida em que professor e aluno não se encontram juntos na mesma sala, requisitando assim, meios que possibilitem a comunicação entre ambos como correspondência postal, correspondência eletrônica, telefone ou telex, rádio, modem, televisão apoiada em meios abertos de dupla comunicação, etc. Há muitas denominações utilizadas para descrever a educação a distância, como: estudo aberto, educação não-tradicional, estudo externo, extensão, estudo por contrato, estudo experimental, aprendizagem a distância.

Keegan [KEE 91], sumariza os elementos que considera centrais no uso da EAD:

- separação física entre professor e aluno, que a distingue do ensino presencial;
- influência da organização educacional (planejamento, sistematização, plano, projeto, organização dirigida, etc), que a diferencia da educação individual;
- utilização de meios técnicos de comunicação, usualmente impressos, para unir o professor ao aluno e transmitir os conteúdos educativos;
- previsão de uma comunicação de mão dupla, onde o estudante se beneficia de um diálogo, e da possibilidade de iniciativas de dupla via;
- possibilidade de encontros ocasionais com propósitos didáticos e de socialização.

Além dessas, Santos [SAN 96], descreve outras características para a EAD :

- o aluno ganha condições de agente eminentemente ativo, através da auto-aprendizagem, mais do que no processo de ensino presencial;
- o modelo é extremamente flexível, possibilitando o envolvimento dos alunos de variadas características - idade, procedência, nível cultural -situados em locais ou ambientes distintos, atuando individualmente ou em grupos.

A EAD apresenta diversas vantagens sobre o ensino convencional, sendo estas:

- versatilidade;
- pode alcançar um grande número de pessoas ao mesmo tempo;
- pode se adaptar ao ritmo de aprendizagem de cada um;
- se no início o investimento é muito elevado, diminui seus custos pela quantidade de pessoas que o utilizarão posteriormente;
- favorece o uso de recursos tais como o rádio, a televisão e o computador;
- desenvolve o autodidatismo, independência e autonomia;

- satisfaz com rapidez demandas e necessidades educativas ditadas por situações sócio-econômicas específicas de regiões e localidades.

Entretanto, atualmente ainda existem limitações ao seu uso, destacando-se:

- os alunos devem estar motivados a dedicar um tempo razoável para os estudos;
- o conteúdo é igual para todos por ser um material pré-produzido, a menos que se desenvolva ambientes que possam ser configurados pelo aluno;
- limitação em alcançar o objetivo da socialização, pelas escassas ocasiões de iteração aluno-professor e aluno-aluno;
- limitação em alcançar os objetivos da área afetiva/postural, assim como os objetivos da área psicomotora;
- empobrecimento da troca direta de experiências proporcionada pela relação pessoal entre professor e aluno;
- *o feedback* e a retificação de possíveis erros podem ser mais lentos, embora os novos meios tecnológicos possam reduzir estes inconvenientes;
- necessidade de um rigoroso planejamento a longo prazo;
- resultados da avaliação à distância mostram-se menos confiáveis;
- elevado índice de evasão por falta de um bom acompanhamento.

Desde meados de 1980, as atenções têm se voltado para a utilização de redes de computadores para ensino e aprendizagem. Atualmente, as pesquisas têm se concentrado na utilização da *World Wide Web* (WWW) em atividades de ensino à distância.

2.1 WWW e a Educação à Distância

Com o acesso à *Internet* facilitado pela WWW, tem-se um novo recurso tecnológico que, ao permitir a integração de texto, imagem, áudio e vídeo, disponíveis em qualquer parte do planeta e com atualização automática, traz novas possibilidades à tecnologia da educação, no sentido da criação de materiais mais eficientes e da individualização do ensino [MOR 98].

A WWW têm se mostrado eficiente e de fácil utilização para compartilhar informações entre as pessoas. Esta ferramenta também está sendo utilizada na aprendizagem a distância pois, através da sua infra-estrutura transporta quase transparentemente informações dos tutores aos aprendizes.

A WWW oferece a navegação através de hiperdocumentos pela *Internet*, como características principais 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 WWW 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, onde path é o caminho completo onde o documento a ser consultado está armazenado.

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

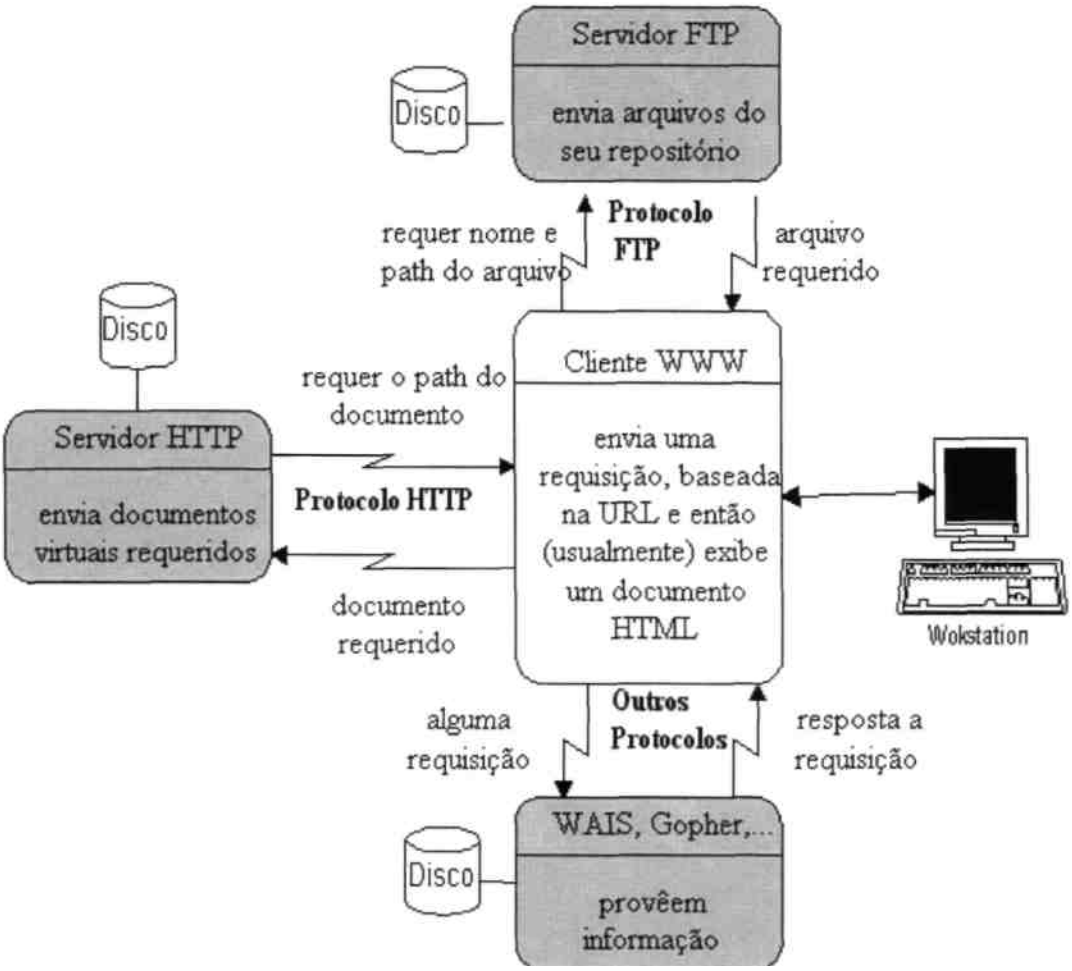


FIGURA 2.1 - *World Wide Web*

O uso dos recursos da rede mundial pode contribuir de várias formas para a melhoria de aspectos importantes de uma aula. Estas contribuições podem ser agrupadas nas seguintes áreas [MOR 98] [CAS 98]:

- dinamizar as aulas expositivas: o emprego adequado de recursos multimídia - imagem, animação, som, vídeo - podem tornar a exposição mais atraente e dinâmica, aumentando o interesse dos alunos. Com o *site* dentro da rede, o aluno pode acessá-lo a qualquer momento, a aula estará sempre disponível, tanto para a preparação prévia do aluno como para posterior resgate de seu conteúdo;
- melhorar a comunicação professor-alunos e entre os alunos: uma aula virtual baseada na WWW permite inúmeras formas de comunicação, como por exemplo: os indivíduos podem falar com outros indivíduos através de *e-mail*, *chats*, listas de discussão, conferências de áudio e de vídeo, etc;
- apoiar o estudo e a pesquisa dos alunos: o conteúdo da matéria pode ser complementado com textos de vários tipos e origens. Uma das fontes

possíveis é a própria rede: muitos órgãos de imprensa, universidades, organizações e empresas têm em seus *sites* artigos e textos que podem ser copiados para o *site* da disciplina ou acessados via *hyperlink.* Além disso, o professor pode propor como atividade aos alunos a pesquisa de um assunto na rede;

agilizar e individualizar o processo de avaliação: com os recursos de preenchimento de formulários *on-line, é* possível criar um banco de testes e exercícios que pode ser individualizado e corrigido automaticamente, dando o *feedback* automático ao aluno. Qualquer trabalho, exercício, relatório ou outra atividade escrita ou com o uso de imagens, pode ser convertida em arquivos de computador e entregues via *e-mail.* Já as atividades que implicam em nota podem trazer riscos quanto à validade e qualidade da avaliação, pois exige um nível de maturidade muito alto por parte dos alunos para que não ocorram fraudes.

Segundo [SHI 94] existe uma pequena lista de diferentes tipos de ambientes de aprendizagem computacional. Eles representam paradigmas de aprendizagem diferentes e podem ser classificados como aprendizagem por instrução, controle interno-externo, etc. são eles:

- *Programmed Instruction* (passo a passo na transferência do conteúdo);
- *Computer Assisted Instruction* (tutoriais sem inteligência)
- *Intelligent Computer Assisted Instruction* (tutores inteligentes)
- *Computer Based Learning* (simulações, hipertextos, micromundos);
- *Intelligent Learning Environments* (micromundos, tutores, ajudantes, especialistas);
- *Cognitive Learning Support Environments* (alguns hipertextos).

O Eletrotutor III implementa um ambiente de ensino-aprendizagem inteligente à distância, no qual todo o conteúdo instrucional mostrado ao aluno é composto de páginas HTML, *applets* Java e Javascript.

3 Agentes

O conceito de agentes envolve diferentes linhas, resultando em um número de propostas relativas a teorias de agentes, arquiteturas de agentes e linguagens de programação de agentes [WOO 95].

As teorias de agentes são essencialmente especificações, as quais questionam como se concebe um agente, quais as propriedades que os agentes devem possuir, como eles são representados formalmente, etc. As arquiteturas dos agentes representam propostas de especificações que levam às implementações, isto é, estruturação e projeto de cada agente. As linguagens de agentes são linguagens de programação que podem incorporar vários princípios propostos por teóricos que estudam os agentes. Tais linguagens trabalham questões relativas a como programar agentes, quais as primitivas corretas para realizar esta tarefa, como por exemplo, compilar e executar de maneira eficiente os programas de agente.

Existe uma ampla variedade de definições para o termo agente. Segundo Russel [RUS 95], um agente inteligente é alguma coisa que percebe e atua sobre o seu ambiente através de sensores, a Figura 3.1 representa um agente genérico. Um agente humano possui olhos, ouvidos e outros órgãos para perceber seu ambiente, já um agente robô utiliza-se de câmeras, vários motores, raios infra-vermelhos, etc. Um *software* de agente possui codificado em bits suas percepções e ações. Pesquisadores envolvidos com agentes inteligentes oferecem uma variedade de definições, cada um tentando explicar o seu próprio uso para palavra agente.

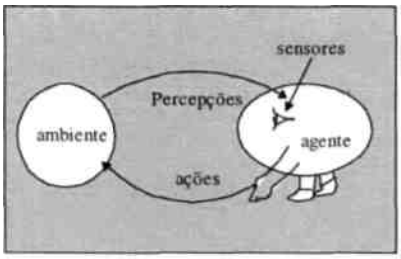


FIGURA 3.1 - Agente Genérico

Um agente é um programa de computador que é capaz de comunicar, cooperar e aprender. Eles possuem a habilidade de tomar conta de algumas tarefas humanas e interagir com pessoas como se assim o fossem, eles estão simplificando o uso do computador.

3.1 *Software* de Agentes Inteligentes

Software de agentes inteligentes são uma nova classe de *software* que atuam a favor do usuário para, por exemplo, encontrar e filtrar informação, negociar por serviços, automatizar com facilidade tarefas complexas, colaborar com outro *software* de agentes para resolver problemas complexos. *Software* de agentes são a melhor forma de abstração para visualização e estruturação de *software* complexo. Mas procedimentos, funções, métodos e objetos são abstrações de *software* familiares para a

maioria dos programadores, enquanto que *software* de agentes são um paradigma desconhecido ou não familiar para muitos destes.

A idéia central em um *software* de agentes é a delegação. O proprietário ou usuário do *software* de agente delega uma tarefa ao agente, e este a executa de forma autônoma. O agente deve se comunicar com o usuário para receber as instruções e prover ao usuário os resultados de suas atividades. Finalmente, um agente deve monitorar o estado de seu próprio ambiente de execução e tomar decisões necessárias para realizar as tarefas delegadas a ele.

Existem duas abordagens para a construção de sistemas baseados em agentes, o desenvolvedor pode utilizar um único agente (agente *stand-alone)* ou implementar um sistema multiagente. Um agente *stand-alone* se comunica apenas com o usuário e provê todas as funcionalidades requeridas para a implementação de um programa baseado em agente. Sistemas multiagentes são sistema computacionais nos quais diversos agentes cooperam para realizarem tarefas que seriam difíceis ou impossíveis de resolver por um agente único.

Segundo a pesquisadora Hayes-Roth [HAY 97], um agente inteligente deve, necessariamente, possuir a capacidade de executar três funções, são elas:

- perceber dinamicamente as condições do ambiente;
- tomar decisões para afetar condições do ambiente;
- interpretar percepções, resolver problemas, extrair inferências e determinar ações.

Para os pesquisadores Wooldridge e Jennings [WOO 95] os agentes inteligentes devem possuir autonomia, percepção e pró-atividade (podem tomar a iniciativa para realizarem um determinado comportamento) e devem exibir um comportamento orientado por objetivo (capaz de realizar tarefas complexas e tomar a decisão de como tais tarefas serão divididas e qual a ordem de execução para alcançar a melhor performance). Nwana argumenta que para um agente ser considerado inteligente, ele deve aprender com os seus atos e/ou interagir com o seu ambiente [NWA 96].

Segundo [WOO 94], existem características gerais que os agentes podem possuir, o primeiro grupo de características fazem parte de uma noção fraca do conceito de agente. Os agentes podem possuir outras características além destas, essas fazem parte de um segundo grupo - noção forte de agente. A tabela 3.1 apresenta as propriedades ditas fracas e fortes do conceito do termo Agente.

Alguns pesquisadores como [SHO 93], [THO 93], [RAO 95], [RA096], [MÜL 96] descrevem os agentes com conceitos aplicados usualmente ao homem, é freqüentemente comum em IA caracterizar um agente com metáforas de noções mentais, como o conhecimento, crenças, desejos, intenções, responsabilidades, etc.

Um segmento da pesquisa em IA tem explorado modelos de agentes baseados em crenças, desejos e intenções. As arquiteturas que seguem este paradigma são conhecidas como arquiteturas BDI *(Belief-Desire-Intention).* As idéias básicas da abordagem BDI são descrever o processo interno de um agente utilizando um conjunto de categorias mentais (crença, desejo e intenções) e definir uma arquitetura de controle

através da qual o agente seleciona racionalmente o curso de suas ações [GIR 99]. Na literatura são encontrados vários conceitos associados à estas categorias. Os conceitos individuais associados à essas categorias são descritos a seguir:

- **Crenças**

Crenças são uma fundamental parte do estado mental do agente, pois representam o possível conhecimento do agente. Um agente pode ter crenças sobre o mundo, sobre crenças de outros agentes, crenças sobre interações com outros agentes e crenças sobre suas próprias crenças. As crenças podem ser contraditórias.

Segundo Corrêa [COR 94], uma crença pode ser representada pela Teoria das Situações, como:

C I= { <Bel, A, P, 1, t, v>], onde:

Bel é a relação para a representação de crença;
A representa o agente;
P é um proposição (i.e, uma situação);
1 é uma localização espacial (local);
t é um instante de tempo;
v 1, se o agente A acredita que a proposição P ocorre o local 1 e no tempo t
0, se o agente A não acredita que a proposição P ocorre no local no tempo t

- **Desejos**

Os desejos de um agente são um conjunto de metas a serem realizadas num período de tempo. Uma meta é tipicamente uma descrição de um estado desejado do ambiente. O desejos motivam o agente a agir de forma a realizar as meta, tais ações são realizadas através das intenções causadas pelos desejos.

Com base na Teoria das Situações, Moussalle [MOU 96] apresenta o desejo D de um agente A como um estado mental intencional e motivador pela seguinte situação:

D 1= { <Des, A, P, e, Io, to,v>}, em que:

Des é a relação para a representação de desejo;
A representa o agente que possui o desejo;
P é uma proposição;
e é uma situação que informa sobre a satisfação, não satisfação, urgência, intensidade e insistência do desejo D;
lo é a localização espacial (local) associada à ocorrência do desejo D;
to é o tempo associado à ocorrência do desejo D;
v 1, se ocorre o desejo D ao agente A
0, se não ocorre o desejo D ao agente A

- **Intenções**

Assim como os desejos, as intenções contêm a representação dos estados que o agente quer que se verifiquem. A base para a definição do conceito de intenção está fortemente associada aos trabalhos filosóficos de Michael Bratman [BRA 84][BRA

89]. Este autor claramente distingue o conceito de fazer as coisas intencionalmente (ação) e possuir intenção de fazê-las (estado mental) [GIR 99].

TABELA 3.1- Características de um Agente

Noção	Propriedades	Descrição
Fraca	Autonomia	agentes operam sem a intervenção direta de um humano e possuem algum tipo de controle sobre suas ações e estados internos
	habilidade social	agentes interagem com outros agentes e humanos através de algum tipo de linguagem de comunicação própria
	pró-atividade	os agentes podem ter a iniciativa para realizarem um determinado comportamento
	reativos	percebem seu ambiente e respondem a mudanças neste
	continuidade temporal	agentes são processos continuamente rodando
	objetivo orientado	um agente é capaz de realizar tarefas complexas e toma a decisão de como tais tarefas serão divididas e qual a ordem de execução para se chegar a melhor performance
Forte	mobilidade	habilidade de locomoção em redes de computadores
	benevolência	supõem-se que entre agentes não ocorra conflito na execução das tarefas
	racionalidade	assume-se que um agente não realizará suas tarefas se estas vão contra suas crenças
	adaptabilidade	um agente pode se adaptar a um ambiente, trabalhando métodos e preferências do seu usuário;
	colaboração	existe colaboração do agente com o seu usuário e com outros agentes

3.2 Classificação de Agentes

Existem diferentes tipos de agentes, segundo [NWA 96], os agentes podem ser classificados como móveis ou estáticos, reativos ou cognitivos e deveriam possuir atributos importantes como autonomia, capacidade de aprendizado e cooperação.

Existem muitas maneiras de classificar os agentes inteligentes, Nwana propôs uma tipologia que define quatro tipos de agentes baseados em suas habilidades para cooperar, aprender e atuar autonomamente. A Figura 3.2 representa como estes quatro tipos de agentes utilizam as capacidades citadas acima.

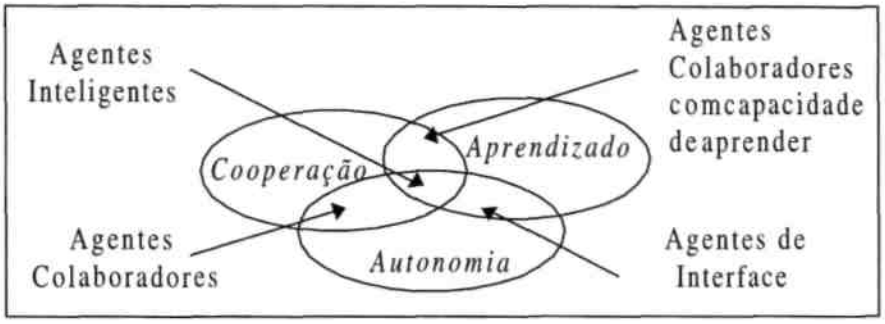


FIGURA 3.2 - Tipologia de Agentes

Os agentes colaboradores enfatizam a autonomia e a cooperação para executarem suas tarefas através da comunicação e possível negociação com outros agentes para alcançarem um entendimento mútuo. Tais agentes são utilizados para resolverem problemas distribuídos, como por exemplo, controle de tráfego aéreo. Esta classe de agentes necessita de uma linguagem de comunicação, tal como KQML, que é descrita na seção 3.4.2 .

Os agentes de interface são autônomos e utilizam aprendizagem para executar as tarefas para os seus usuários. A inspiração para esta classe de agentes é um assistente pessoal que colabore com o usuário. Esta classe é utilizada para implementar assistentes, guias, *helps,* filtros, etc.

As distinções mostradas na Figura 3.2 não são definitivas. Por exemplo, os agentes colaboradores, possuem mais ênfase na cooperação e autonomia do que na aprendizagem, mas isto não implica que agentes colaboradores nunca irão aprender.

Os agentes móveis são processos computacionais capazes de se moverem através de redes (WANs - *Wide Área Network)* como pela WWW, interagindo em *hosts* desconhecidos, agrupando informações em favor do seu usuário e retornando para este depois de ter executado suas tarefas. Este tipo de agente é implementado através de programas remotos, isto é, programas desenvolvidos em uma máquina e entregue para uma segunda máquina para execução subseqüente.

Uma das mais populares utilizações de agentes inteligentes é para encontrar, analisar e recuperar grandes quantidades de informação. Os agentes de informação são ferramentas para ajudar no gerenciamento da informação avaliada em redes como a WWW. Agentes de informação acessam a rede a procura de tipos particulares de informação (selecionada pelo usuário), filtram-na e a retornam aos seus usuários. Este tipo de agente tipicamente utiliza o protocolo HTTP para acessar as informações e pode também utilizar KQML ou outra linguagem de comunicação para interagir.

Sistemas de Agentes Heterogêneos referem-se a coleção de dois ou mais agentes com diferentes arquiteturas. Com a diversidade de domínios de aplicação, os agentes mais apropriados são selecionados. Os agentes neste sistemas podem comunicar, cooperar e interagir uns com os outros.

3.3 Comunicação de agentes

Comunicação é o processo pelo qual a informação é trocada numa transação do agente. Mensagens transmitidas possuem conteúdo e contexto. Conteúdo refere-se aos dados codificados na mensagem; contexto é como as propriedades mudam desde os dados à informação [NIS 95].

A comunicação entre os agentes pode se dar, por exemplo, através do ambiente, da troca de mensagens, da utilização de um quadro negro. Neste trabalho, a comunicação entre os agentes é realizada através da troca de mensagens. Segundo Wooldridge, citado em [BEP 98], a troca ou passagem de mensagens entre agentes, pode ser feita de três formas distintas:

- passagem de mensagem ponto-a-ponto: nesta abordagem, as mensagens são enviadas para um endereço específico (o receptor) que deve ser conhecido pelo emissor. Como vantagens desta abordagem, têm-se: um agente sempre sabe para onde uma mensagem está sendo emitida; e que os controles de segurança são facilmente introduzidos, já que um agente pode assegurar que a mensagem nunca será enviada para agentes desconhecidos;
- passagem de mensagem *broadcast:* a passagem de mensagem *broadcast é* baseada na emissão de uma mensagem não para um endereço específico, mas para todos os agentes da sociedade. Nesta abordagem, um agente particular pode ser substituído por outro agente com conduta equivalente e o procedimento do sistema como um todo será inalterado. A passagem de mensagem *broadcast* não é segura, já que qualquer agente pode examinar os conteúdos de qualquer mensagem;
- passagem de mensagem *multicast:* uma maneira que está sendo utilizada para evitar os problemas do *broadcast* é a estruturação do espaço de agentes em grupos. Assim, cada agente é membro de um grupo menor e se esse agente transmite uma mensagem, esta será emitida para todos os membros de um ou mais (dependendo do sistema) dos grupos de agentes.

As linguagens de comunicação possuem algumas características relevantes na determinação de sua adequação, segundo [MAY 96] alguns destes aspectos são:

- forma: uma boa linguagem de comunicação de agentes deve ser declarativa, sintaticamente simples e legível. Como uma linguagem de comunicação deve ser integrada em uma grande variedade de sistemas, sua sintaxe deve ser extensível;
- conteúdo: uma linguagem de comunicação deve ser estendida de modo que se adapte bem com outros sistemas. A linguagem deve fornecer um conjunto definido de ações de comunicação (primitivas);
- semântica: a descrição da semântica de uma linguagem normalmente é feita através da linguagem natural, a linguagem deve ser baseada numa teoria, não deve ser ambígua e deve considerar tempo e local;
- implementação: a implementação deve ser eficiente, tanto para velocidade como para a utilização da largura da banda. Ela deve prover uma boa adaptação com a tecnologia de *software* existente. A interface deve ser fácil de utilizar, detalhes das camadas de rede que existem abaixo das primitivas de ações de comunicação devem ser escondidas do usuário. A linguagem deve ser apropriada para implementação parcial, para que os agentes inteligentes possam manusear somente um subconjunto de primitivas de ações de comunicação;

- rede: uma linguagem de comunicação de agente deve adaptar-se bem com as tecnologias modernas de rede. A linguagem deve suportar todas conexões básicas - ponto-a-ponto, *multicast* e *broadcast.* As conexões síncrona e assíncrona devem ser suportadas. A linguagem deve conter um conjunto de primitivas bastante ricas que possam servir como um substrato sobre quais linguagens de alto-nível e protocolos possam ser desenvolvidos. Além disso, estes protocolos de alto-nível devem ser independentes dos mecanismos de transporte (ex. TCP/IP, *e-mail,* http, etc.) utilizados;
- ambiente: o ambiente em que agentes inteligentes devem trabalhar deve ser altamente distribuído, heterogêneo e extremamente dinâmico. Ele deve suportar interoperabilidade com outras linguagens e protocolos;
- confiabilidade: a linguagem de comunicação deve suportar uma comunicação confiável e segura entre agentes. Devem ser oferecidos recursos para trocas privadas e seguras entre dois agentes. A linguagem deve suportar mecanismos razoáveis para identificação e sinalização de erros e advertências.

Uma das linguagens de comunicação entre agentes que procura implementar estas características é KQML *{Knowledge Query and Manipulation Language)* que é uma linguagem e um protocolo para troca de informação e conhecimento entre agentes. KQML tem por objetivo desenvolver técnicas e metodologias para a construção de bases de conhecimento de larga escala que são compartilháveis e reutilizáveis. KQML pode ser utilizado como a linguagem que um programa aplicativo utiliza para interagir com um sistema inteligente, ou para dois ou mais sistemas inteligentes compartilharem conhecimento para solução cooperativa de problemas [HÜB 97].

3.4 Linguagens de Agentes

De acordo com as várias áreas de problema nas quais os agentes podem ser úteis, e levando em conta as características de agente mencionadas na Tabela 3.1, Rodrigues [ROD 95], afirma que as ferramentas e linguagens podem ser classificadas de acordo com as seguintes propriedades:

- modularidade: habilidade para criar agentes que são capaz de manipular tarefas simples, bem definidas. Isto prove a reusabilidade do agente, esta é uma propriedade fundamental não só associada com o desenvolvimento de agente mas com código bem escrito;
- interação entre agentes: a interface deveria prover uma comunicação entre agentes flexível e robusta;
- mobilidade: em alguns domínios, os agentes serão mais eficientes se forem capazes de migrarem para outras redes;
- dados e recursos distribuídos: um agente deveria poder ter acesso a dados e conhecimento distribuídos;
- Conhecimento: o agente deveria poder armazenar conhecimento, atualizando-o e planejando suas ações levando em consideração tais mudanças.

Existe várias linguagens de programação amplamente distribuídas e ferramentas de desenvolvimento que estão sendo usadas para construir os agentes atualmente. Algumas destas linguagens são descritas nesta seção:

- Smalltalk: desenvolvida pela Xerox Parc. Esta linguagem é particularmente popular nas indústrias de telecomunicações para o desenvolvimento de soluções cliente/servidor;
- **Tcl/Tk:** uma máquina independente interpreta o código da linguagem, implantada como uma biblioteca do procedimentos em C que são incorporadas às aplicações. É de livre distribuição na *Internet.* Esta linguagem é adequada em menor escala para aplicações comerciais e prototipagem;
- KSE: A ARPA *Knowledge Sharing Effort* (KSE) desenvolve técnicas e metodologias para a construção de conhecimento compartilhavel e reutilizável. Foram desenvolvidas várias ferramentas para alcançar esta meta, entre elas: **KQML** *(Knowledge Query and Manipulation Language)* que consiste em uma linguagem com o propósito de desenvolver bases de conhecimento compartilhado em larga escala para habilitar a comunicação entre agentes e troca de conhecimento e **KIF** *(Knowledge Interchange Format)* é um idioma formal para o intercâmbio de conhecimento entre programas discrepantes (escrito por programadores diferentes, a tempos diferentes, em idiomas diferentes e assim sucessivamente);
- **Telescript:** uma linguagem de programação orientada a objetos para a construção de agentes móveis. É adequada para aplicações comerciais em radiotelefonia e ambientes de rede;
- **Lisp:** é uma das mais velhas linguagens de programação existentes, recentemente o comitê de programadores Lisp aprovou o padrão ANSI CLOS *(Common Lisp Object System)* que define um idioma orientado a objeto dinâmico, provendo uso automático e eficiente de herança múltipla, administração de memória automática, etc;
- **Java:** principal linguagem para aplicações de agentes na Web. Java é interpretada, *multithread* (várias linhas de execução). O interpretador Java executa o Java *bytecode* (código gerado pelo compilador Java) diretamente de alguma máquina, na qual o interpretador e o sistema *runtime* estejam. A capacidade *multithread* permite que Java implemente aplicações de comércio eletrônico interativo.

A Tabela 3.2 [ROD 95], apresenta as linguagens e algumas características importantes no desenvolvimento de agentes, descritas nesta seção, na qual '*' simboliza que a linguagem suporta completamente tal característica, ou suporta em maior grau '+' ou menor grau '-'.

TABELA 3.2 - Linguagens X Características

Característica	Java	Telescript	Tcl	SmallTalk	KSE	Lisp
Modularidade	*	*	+	*	-	+
Robustês, flexibilidade na comunicação entre agentes	+	*		*	*	
Mobilidade	+	*	+	-	-	-

Dados e recursos distribuídos	*	*	*	*	+	+
Conhecimento	-	-	-	-	+	+

Os agentes do Eletrotutor III são implementados na linguagem Java e utilizam como linguagem de comunicação o KQML, as seções 3.4.1 e 3.4.2 se aprofundam nestas linguagens.

3.4.1 Java

Além da independência de plataforma e das fortes capacidades de rede, Java oferece os benefícios da orientação a objetos (OO) e de múltiplas linhas de execução *{multithreading).* A linguagem também é dinâmica: pequenas partes do código Java são montados em tempo de execução *(runtime)* dentro do programa [HOP 97].

Atualmente, na *Internet* muito se fala sobre o Java, grande parte desta agitação foi gerada pelas *applets* que são pequenos programas que podem ser embutidos em páginas Web. A Figura 3.4 apresenta como uma *applet* é executada. A execução das *applets* no lado cliente é uma grande inovação na programação na Web.

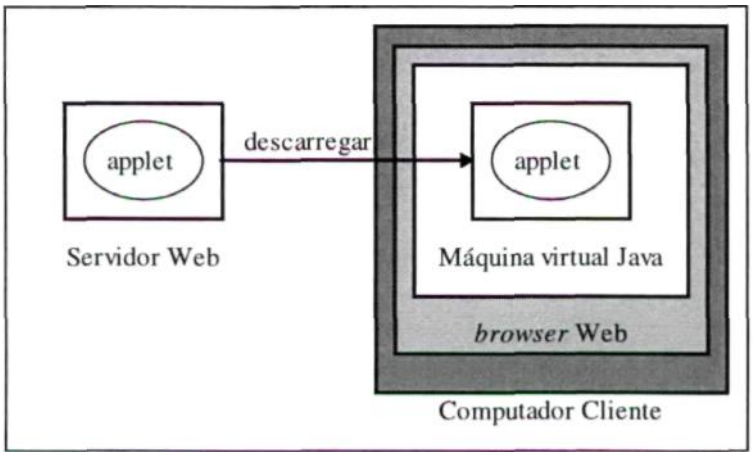


FIGURA 3.4 - Como as *applets* rodam

A linguagem Java foi projetada para ser segura no sentido de que o seu código não prejudicaria a si mesmo ou a outros componentes de *software*, por exemplo, não é permitida a manipulação explícita de ponteiros.

Antes que algum código de classe Java seja executada, seus *bytecodes* são verificados. O verificador de bytecode utiliza um teorema simples para assegurar que:

- ponteiros não foram esquecidos;
- restrições de acesso não foram violadas;
- as chamadas dos métodos contém o número e tipo corretos de parâmetros;
- a pilha não está em *overflow*.

Este passo de verificação é muito desejável do ponto de vista da segurança, e tem adicionado benefícios para que o interpretador execute mais rapidamente.

Os programas Java rodam dentro de máquinas virtuais, as quais ficam dentro do computador no qual eles estão rodando. A máquina virtual age como uma parede entre o *host* e o programa Java. Um programa Java nunca acessa os dispositivos de entrada e saída, o sistema de arquivos ou mesmo a memória do computador no qual está rodando. Em vez disso, ele pede que a máquina virtual os acesse. A máquina virtual contém ferramentas de manipulação de *strings*, rotinas gráficas e de interface com o usuário, estruturas básicas de dados e funcionalidades matemáticas. Para utilizá-los é necessário aprender sobre a Interface de Programação de Aplicativos (API), que é uma coleção de classes, interfaces e exceções prontas. Dentro da API, essas classes, interfaces e exceções estão agrupadas em oito pacotes: java.applet, java.awt, java.awt.image, java.awt.peer, java.io, java.lang, java.net e java.util. A Tabela 3.3 [THO 97] descreve brevemente o propósito de cada pacote.

TABELA 3.3 - Os pacotes da API

Pacote	Descrição
java.applet	contém classes e interfaces que ativam as *applets*
java.awt	permite escrever interfaces gráficas de usuário
java.awt.peer	composto totalmente de interfaces que permitem que o sistema de janelas do Java seja facilmente portado entre plataformas
java.awt.image	dedicado à criação e manipulação de imagens
java.io	trata a entrada e saída do programa
java.lang	contém os elementos centrais da linguagem Java, tais como Object, String, Exception e Thread
java.net	Trata a interação com a rede
java.util	Contém várias classes de utilitários que tornam mais fácil a programação

3.4.1.1 *Sockets*

Uma forma dos programas Java interagirem com outros programas é a utilização de *sockets.* Um *socket é* uma conexão de dados transparente entre dois computadores numa rede. O pacote java.net prove duas classes - *Socket* e *ServerSocket* - que implementam, respectivamente, o lado da conexão do cliente e o lado do servidor.

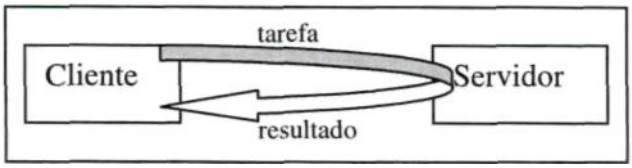
Um *socket é* identificado pelo endereço de rede e por um número de porta do computador, só assim, a camada TCP poderá identificar a aplicação na qual os dados serão transmitidos.

3.4.1.2 **RMI**

Além dos *sockets*, existe outra forma dos programas Java interagirem uns com os outros: a utilização do *Remote Method Invocation* (RMI) [SUN 98]. RMI prove um modelo simples e direto para computação distribuída com objetos Java. Para utilizá-lo é necessário implementar um Servidor com todos os métodos necessários para a troca de informações e os Clientes que irão acessá-lo.

FIGURA 3.5 - RMI Ciente/Servidor

Quando um cliente deseja enviar uma mensagem ao servidor, ele invoca um método do servidor e o mesmo se dá quando o desejo é o de receber uma mensagem, como mostra a Figura 3.5.



RMI pode passar objetos como argumento e retornar valores de tipos de dados não pré-definidos, portanto, os objetos podem fazer parte da API ou serem criados pelo implementador.

O sistema de RMI consiste em três camadas: a camada de stub/skeleton, a camada de referência remota e a camada de transporte. Cada camada é definida por uma interface específica e protocolo, portanto cada uma delas é independente da próxima e pode ser substituído por diferentes tipos de implementação sem afetar as outras camadas no sistema.

3.4.1.3 JDBC

O JDBC consiste em uma API Java, composta por um conjunto de classes e interfaces escritas na linguagem Java e que são responsáveis pela execução de operações SQL *(Structured Query Language)* [JEP 97]. O JDBC provê uma forma simples de desenvolver aplicações com banco de dados utilizando a linguagem Java, são três as suas funções:

- estabelecer a conexão com o banco de dados;
- enviar ao banco consultas SQL;
- processar os resultados das consultas.

O banco de dados Oracle foi utilizado na implementação do protótipo. A Oracle [ORA 99] desenvolveu *drivers* que possuem extensões de propriedades, tipos e performance, melhorando assim o desempenho do JDBC fornecido pelo Java. Os *drivers* existentes são: JDBC Thin (utilizado em *Applets)* e o JDBC OCI (utilizado em aplicações cliente-servidor).

3.4.2 KQML

KQML é uma linguagem de alto nível que suporta interoperabilidade entre agentes em aplicações distribuídas. O formato da mensagem e o protocolo de manuseio da mensagem é suportado em tempo de execução e o conhecimento é compartilhado entre os agentes. Esta linguagem enfoca um grande conjunto de mensagens pré-definidas *(performatives),* as quais definem as operações possíveis de serem executadas pelos agentes. Assim, aqueles que estão em conformidade com a KQML podem responder a estas mensagens de maneira adequada independentemente da estrutura do seu emissor.

Essa tecnologia se torna importante em virtude da crescente necessidade de uma linguagem comum entre agentes principalmente numa rede de grande magnitude como a *Internet.* Um agente ao receber uma mensagem KQML toma uma ação com base no seu significado e conteúdo [FIN 96].

KQML é uma linguagem projetada para suportar interações entre agentes inteligentes. Ela foi desenvolvida pelo ARPA no Programa de Compartilhamento de Informações *(Knowledge Sharing Effort)* e implementada independentemente por vários grupos de pesquisa. Foi utilizada com sucesso para implementar uma variedade de sistemas de informações usando diferentes arquiteturas de *software* [FIN 96].

Na especificação da KQML, os agentes que participam através da troca de mensagens podem ser comparados com entidades que possuem comportamento próprio e autônomo agindo em sociedade através da interação com outros agentes. Esse comportamento está intimamente ligado ao seu estado mental que pode ser representado por uma base de conhecimento onde informações que descrevem suas crenças e objetivos estão armazenadas.

As crenças do agente representam as informações sobre si próprio e sobre o ambiente externo, incluindo as bases de conhecimentos de outras entidades. Todas as *performatives* giram em torno das bases de conhecimentos, ou seja, as trocas de mensagens estão normalmente associadas com as crenças ou objetivos contidos na base do agente chamada de *Virtual Knowledge Base* (VKB), de forma que sua implementação não deve necessariamente ser estruturado como uma base de conhecimento [FIN 96].

O conjunto de mensagens KQML pode ser estendido desde que as novas *performatives* criadas obedeçam a mesma forma da especificação original da linguagem. Os agentes que estão de acordo com a KQML não precisam reconhecer todas as mensagens, de forma que um pequeno subconjunto pode ser suficiente num determinado sistema multiagente, ou seja, dependendo da necessidade pode-se escolher somente algumas *performatives* a serem utilizadas na comunicação.

Uma mensagem KQML, também chamada de *performative,* é expressa como uma *string* de caracteres *ASCII* usando a sintaxe definida, segundo [DAR 93], na Figura 3.6.

```
<performative>        : = <word> {<whitespace> : <word> <whitespace> <expression>)*)

<expression>          := <word> 1 <quotation> 1 <string> 1
                      ((<word> {<whitespace>   <expression>}*)

<word>                := <character><character>*

<character>           := olphabetio 1 <numeric> 1 <special>

<special>             .= < 1 > 1 = I + 1 -1 * I / 1 & 1 ^I -1 _ 1 @ 1 $ 1 % 1: 1 . I! I ?

<quotation>           := '<expression> 1 '<comma-expression>

<comma-expressior  > ::= <word> 1 <quotation> 1 <string> 1 < , <comma-expression>
                      (<word> {<wfiitespace> <comma-expression>}*)

<string>              := "<stringchar>*" 1 #<digitxdigit>" "<ascii>*

<stringchar>          := \ascii 1 <ascii>-\-<double-quote>
```

FIGURA 3.6 - Sintaxe KQML em BNF

A semântica das *performatives* KQML deve ser compreendida para que as mensagens trocadas pelos agentes sejam adequadamente usadas. Para isso, é proposta uma descrição textual das principais mensagens, destacando sua aplicabilidade para a interação entre dois ou mais agentes.

4 Aplicações na WWW com abordagem de agentes

Para construir um ambiente educacional interativo, em vez de um mero hiper-livro eletrônico, é necessário prover um mecanismo de adaptação individual que, de forma ativa, leve o aluno através do hiper-espaço, tendo em consideração o conhecimento prévio deste aluno, sua habilidade de compreensão, sua área de interesse, seus planos e intenções.

Nos últimos tempos, tem sido considerável o número de trabalhos que vêm sendo desenvolvidos na área de Informática aplicada a Educação à Distância. Nesta seção serão descritos alguns exemplos de arquiteturas de ambientes distribuídos de ensino baseados em sistemas multiagentes.

4.1 ADELE

ADELE *(Agentfor Distance Education - Light Editiorí)* consiste em um agente pedagógico projetado para trabalhar na WWW em resoluções de problemas em cursos na área médica, como diagnósticos clínicos e treinamento na área de traumatologia. O agente deste trabalho possui características citadas na seção 3.1, tais como: adaptabilidade, autonomia e pró-atividade.

Agentes pedagógicos animados que habitam ambientes de aprendizagem interativos podem exibir comportamentos notavelmente naturais. Além de prover resolução de problemas, podem aconselhar a respeito das atividades dos estudantes no ambiente de aprendizagem, estes agentes também podem aumentar a motivação do estudante e sua atenção.

Para projetar um *software* de aprendizagem baseado em agente, é essencial entender como os estudantes percebem um agente pedagógico animado com respeito a dimensões afetivas como encorajamento, utilidade, credibilidade, etc. [LES 97].

Um módulo de um curso, tipicamente, possui os elementos mostrados na Figura 4.1. Cada módulo é centrado em algum caso ou problema, o qual o estudante deve resolver. Primeiro, os estudantes recebem uma narrativa instrutiva, em forma textual ou de vídeo, após o caso/problema é apresentado ao estudante na forma de simulação bidimensional ou tridimensional. São providos materiais de referência on-line e o estudante pode discutir o caso on-line com outros estudantes e com instrutores, utilizando ferramentas de teleconferência.

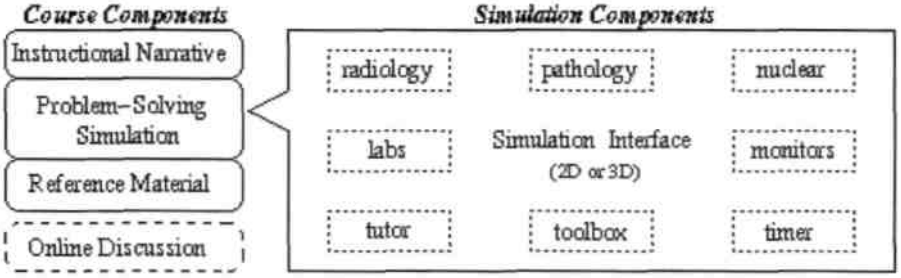


FIGURA 4.1 - Elementos fundamentais de um módulo de curso [LES 97]

Para suportar as simulações descritas acima, foi adotado uma arquitetura geral descentralizada para o sistema, ou seja, existe o servidor central que consiste em um servidor Web, banco de dados e administração do curso e o lado cliente, no qual o material do curso é mostrado localmente por um *software* na máquina do estudante, são providos também, a interatividade e as capacidades de ensino inteligentes. O banco de dados inclui informação sobre o estado de cada estudante. O servidor Web age como uma interface entre o *software* do lado cliente e servidor central. Quando os estudantes iniciam ou terminam uma sessão no computador, o *software* cliente automaticamente notifica o servidor central.

O *software* cliente foi construído a partir de um *browser* Web convencional. O material de referência é armazenado em páginas HTML e acessado localmente. A interatividade é provida pela linguagem Java incorporado nas páginas Web. Os aspectos mais sofisticados do *software* cliente é que ele apresenta e analisa estudos de caso, permite ao estudante explorar o caso, examinar o paciente de diferentes formas e coletar dados, simular os efeitos das suas ações no paciente, monitorar e interagir com o estudante através do agente pedagógico Adele e permitir a inserção de novas lições sem muito trabalho de programação.

O *software* cliente é chamado de *Virtual Environments for Training* (VET) e é composto de quatro partes, um *software* de visualização 3D, utilizado pelos estudantes para navegar através do ambiente virtual e interagir com os objetos dentro deste. Um *software* de simulação, que mantém o modelo de estados do dispositivo que é simulado e simula os efeitos das ações do estudante e o agente pedagógico Adele, que monitora as mudanças no modelo de estados e das ações dos estudantes. A comunicação entre tais partes é feita através de eventos *broadcast.*

Questionando e examinando o paciente virtual e estudando os dados clínicos o estudante poderá praticar diagnósticos. Adele provera a avaliação e uma revisão do progresso do estudante, com base no melhor diagnóstico e critérios de custo deste.

A arquitetura da ADELE implementa as funções de apresentação do material didático, monitoramento do aluno, *feedback,* sugestões, testes e explanações, adaptando a apresentação do curso ao estudante e informando o desempenho do aluno ao servidor central ao término da sessão. Tais capacidades são acopladas a uma personagem animada que suporta contínua interação com o aluno.

ADELE possui um repertório de expressões faciais e posturas corporais que representam suas emoções, tais como surpresa e desapontamento, isso permite que ADELE responda com mais "vida" às ações dos alunos, como ilustra a Figura 4.2. Além disso, baseado nas ações do aluno ADELE pode escolher como intervir, oferecer conselhos e fazer um teste ao aluno para assegurar que este está entendendo as implicações ao paciente, devido as suas ações.

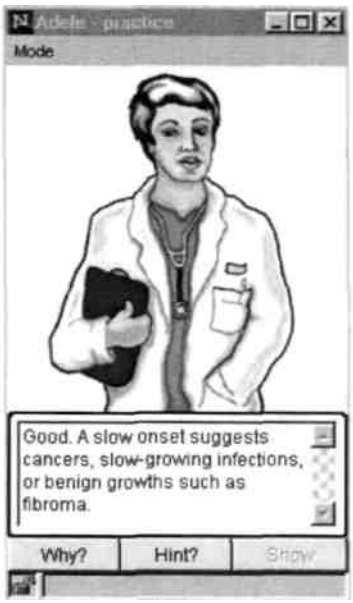


FIGURA 4.2 - Interface do agente Adele

Do ponto de vista do estudante a simulação inclui os seguintes componentes:

- uma visão do paciente, em forma dimensional ou tridimensional;
- se requisitado, Adele informa ao estudante informações sobre o paciente como teste de laboratório, resultado da patologia, e são exibidas radiografias, como também monitores do estado atual do paciente.
- um cronômetro mostra a quantia de tempo que passou ao longo do caso;
- uma caixa de ferramentas de manipulação e instrumentos como estetoscópios que permitem ao estudante examinar e trabalhar no paciente.

Um estudo sobre o impacto afetivo da utilização de agentes pedagógicos animados em experiências de aprendizagem de estudantes foi realizado em Adele. O estudo revelou que a presença do agente animado pode ter um forte efeito positivo na percepção do estudante no processo de aprendizagem. Levando em consideração os resultados positivos obtidos neste estudo, a utilização de agentes pedagógicos animados poderia acrescentar ao ambiente Eletrotutor uma maior interatividade entre aluno e agentes.

4.2 AMBIENTE AME-A

AME-A [DAM 99] é um ambiente de ensino-aprendizagem, o qual se propõe ao estudo e o desenvolvimento de um sistema educacional interativo para o ensino à distância, e o ensino genérico e adaptável às características psico-pedagógicas do aprendiz. Entre as características do ambiente pode-se citar:

- Aprendizagem Estática, a qual é utilizada para classificar o aprendiz entre 'N' possíveis modelos psico-pedagógicos e para determinar entre as 'M' possíveis estratégias de ensino a que melhor se adapta ao mesmo;
- Aprendizagem Dinâmica, a qual é realizada durante a interação do aprendiz e será utilizada para corrigir falsas concepções à respeito do aprendiz inferidas pelos demais agentes atuantes no processo de ensino-aprendizagem.

Este ambiente utiliza a abordagem de sistemas multiagentes [CHE 96] conforme mostra a Figura 4.3.

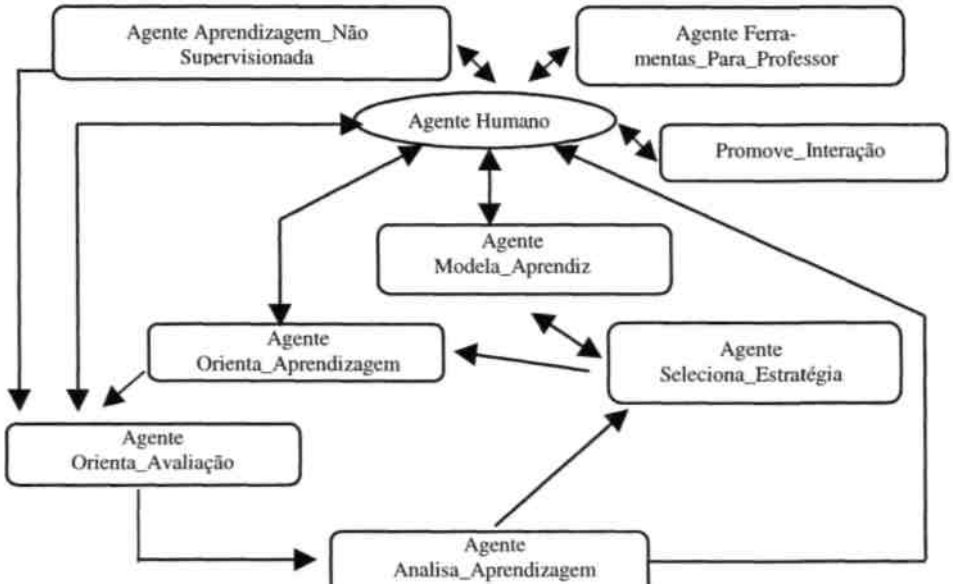


FIGURA 4.3 Ambiente A-MEA

Os agentes que atuam no ambiente estão distribuídos em dois processos: o de ensino e o de aprendizagem. Os agentes são autônomos porque as atividades que eles executam não necessitam supervisionamento externo constante e, não existe uma autoridade central para controlar todas as interações entre os mesmos, além da autonomia eles possuem as seguintes características: colaboração, habilidade social e adaptação.

A análise deste ambiente mostrou a possibilidade de utilização de múltiplas estratégias de ensino, selecionadas em função de parâmetros que o agente seleção_estratégia recebe de outros agentes. Este trabalho foi importante, pois reforça a idéia de estratégia de ensino em função de um modelo diferenciado de aluno. Tal idéia pode ser uma grande contribuição a trabalhos futuros no Eletrotutor.

4.3 Virtual Cell

O Virtual Cell é um ambiente de biologia interativo e tridimensional. Ele foi projetado através da *Virtual Reality Modeling Language* (VRML) e está disponível na *Internet.* Para o estudante o Virtual Cell assemelhar-se a um enorme espaço navegacional populado pelos componentes da célula: núcleos, endoplasma, retículo, complexo de Golgi, mitocôndria, cloroplasto, vacúolos, como mostra a Figura 4.3. Cada estrutura é renderizada como um objeto 3D utilizando a VRML.

O Virtual Cell é um mundo virtual multi usuário no qual os estudantes "navegam" ao redor de células e interagem com um ambiente orientado a objetivos e regras. Os estudantes aprendem fundamentalmente conceitos de células e estratégias para resolução de problemas a partir de deduções através de suas experiências no ambiente explorado. Esta abordagem pedagógica permite os estudantes uma

experiência autêntica que inclua elementos práticos, projeto experimental e tomada de decisões, enquanto é introduzido a eles o conteúdo da disciplina. Na prática, os estudantes aprendem como pensar, agir e reagir às células [WHI99].

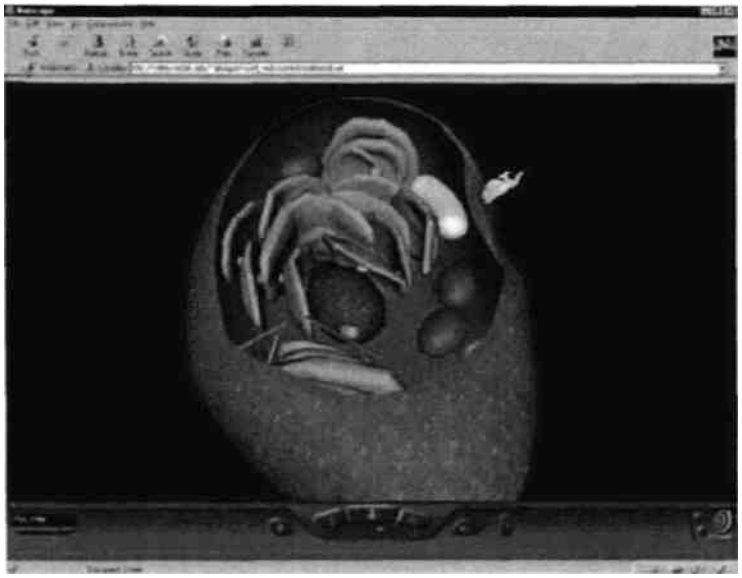


FIGURA 4.3 - o Ambiente Virtual Cell

O primeiro nível do Virtual Cell consiste de um laboratório baseado em VRML. Neste laboratório, o estudante recebe um tarefa específica, podendo fazer experimentos simples, e aprender o básico sobre o fisiologia e a química da célula e seus componentes. O estudante possui um barra de ferramentas para medir e analisar os vários processos celulares. Tais ferramentas incluem medição de 02, C02, PH, açúcar, proteína, várias manchas e enzimas. Conforme progridem, os estudantes visitam o laboratório, desenvolvem amostras celulares para experimentação, e subseqüentemente recebem mais tarefas.

Em um segundo nível, ocorre uma simulação sobre as modificações na célula pela introdução da mutação ou através de um inibidor que rompe o processo celular. Um alarme soa quando um processo celular será mal formulado e o estudante tem como meta diagnosticar o problema. Utilizando as mesmas ferramentas do primeiro nível o estudante pode navegar através da célula, fazer observações, e executar medidas e experiências. O estudante tentará identificar a área afetada, o processo perturbado, e a natureza da mutação ou inibidor que estão causando o problema. Como resultado o usuário aprenderá detalhes das funções e do processo celular.

No terceiro nível, o estudante recebe um conjunto de metas sobre o processo e a estrutura celular para investigar. O estudante tem a sua disposição as ferramentas do primeiro nível e as mutações e inibidores do segundo nível. Utilizando várias combinações das ferramentas disponíveis, o estudante poderá formular hipóteses, desenvolver experimentos, e utilizar a barra de ferramentas para executar estas experiências.

Como o Virtual Cell é multiusuário, os estudantes podem interagir diretamente uns com os outros, trabalhar em conjunto para resolver uma meta. Os agentes são pró-ativos e agem como tutores, monitorando as ações dos estudantes. Os agentes provêem

conselhos na escolha de equipamentos, navegação e conclusões científicas, mas não bloqueiam ações nem previnem os estudantes.

O Virtua Cell será referenciado no Eletrotutor III, pois é um exemplo de ambiente que utiliza VRML para que o aluno sinta-se imerso durante a interação com o sistema.

5 Eletrotutor

O objetivo das diferentes implementações do Eletrotutor [SIL 92] [SIL 94] e [SDL 96] foi desenvolver um instrumento para verificar a eficácia do uso de diferentes abordagens de ambientes de ensino por computador na escola. O ambiente Eletrotutor aborda o conteúdo constituído por alguns capítulos de Eletrodinâmica, um capítulo da Física que estuda alguns fenômenos da Eletricidade e aborda as relações entre algumas grandezas elétricas como Corrente Elétrica, Tensão, ou Diferença de Potencial, Resistência e Potência Elétrica.

As versões do Eletrotutor são constituídas por oito unidades, descritas na Tabela 5.1, cada uma delas constituídas por lições, algumas possuem exemplos e exercícios.

TABELA 5.1- Unidades do EletroTutor

Unidade	Objetivos
1	Descreve a estrutura da Matéria
2	Conceitua cargas elétricas, eletrização, condutores e isolantes
3	Conceitua carga elétrica elementar e aplica a lei de Coulomb
4	Conceitua campo elétrico
5	Conceitua potencial elétrico e diferencial de potencial
6	Conceitua corrente elétrica
7	Aplicação da Lei de Ohm
8	Aplicação da Lei de Joule

As lições do Eletrotutor são estruturadas em forma de telas apresentadas seqüentemente. Cada tela é compostas por pequenos textos previamente elaborados. A base de conhecimento do Eletrotutor é constituída em oito arquivos de lições, os quais são apresentados aos alunos através de um mecanismo de controle que gera a seqüência de telas.

Os exemplos são formados de aplicações práticas sobre a maioria das lições. Cada exemplo foi elaborado com o intuito do aluno participar através da entrada de dados via teclado. O sistema analisa a entrada de dados do aluno, aplica-os na fórmula que esta sendo exemplificada e apresenta o resultado juntamente com o processo de cálculo utilizado.

Cada exercício tem a capacidade de gerar números randomicamente, isto permite que o aluno possa fazer um número exaustivo de exercícios.

5.1 Eletrotutor I

A primeira versão do Eletrotutor foi concebida como um sistema tutorial inteligente tipo *stand-alone,* desenvolvido em linguagem Arity-Prolog seguindo o modelo de Tutor Inteligente proposto por [VIC 89].

Além dos módulos descritos na seção anterior, esta versão dispõe de um módulo denominado arquivo de introdução que é constituído por uma seqüência de telas com o

objetivo de estabelecer um diálogo com o aluno, explicando-lhe o propósito do programa, como utilizá-lo e fazendo-lhe uma série de perguntas para o levantamento de dados a respeito do seu conhecimento sobre o assunto a ser abordado.

Ao final do diálogo, o tutor apresenta uma tela com os objetivos a serem alcançados, entre os quais o tutor assinala os que julga adequados para este aluno participar. O aluno é convidado a seguir adiante nas lições que lhe foram prescritas ou reformular o assinalamento dos objetivos, mudando os passos a serem seguidos.

A partir dos objetivos traçados pelo tutor é montado uma base de dados com as informações sobre o aluno e com a seqüência de lições, exemplos e exercícios que devem ser seguidos. Esta base de dados é gravada em disco e vai sendo reformulada a medida em que o aluno avança na matéria.

Existe um módulo que é responsável por controlar a seqüência de procedimentos de ensino e atualizar a base de dados do aluno. Este módulo define, a partir da performance do aluno nos exercícios, o número de exemplos e até a recomendação de revisão dos conteúdos.

5.2 Eletrotutor II

O Eletrotutor II foi desenvolvido na linguagem HTML e *Java Script* em uma arquitetura cliente servidor, representada na Figura 5.1. Em cada uma das unidades foi desenvolvido um conjunto de páginas HTML contendo os textos e as figuras que constituem o conteúdo apresentado pelo Eletrotutor. Os exercícios e exemplos foram incorporados a cada uma das unidades através do uso de *Java Script.* Com isso o controle sobre o andamento das lições fica sob a responsabilidade do ambiente do usuário. Os *scripts,* ao serem executados, constróem as questões e as apresentam ao aluno. Em seguida o *script* processa a resposta fornecida pelo aluno e prove *o feedback* adequado.

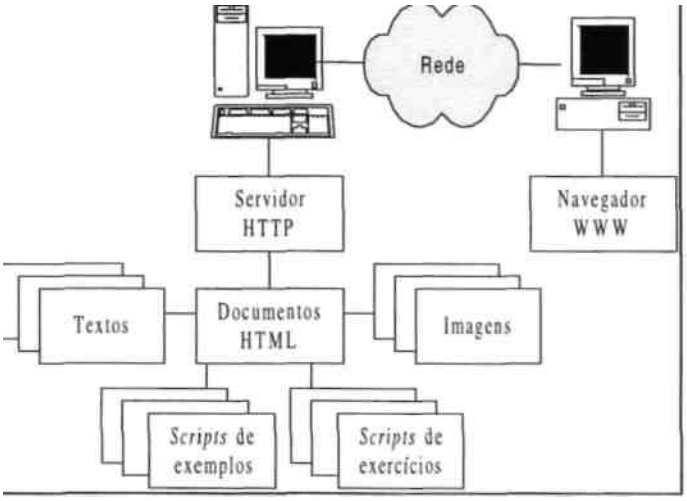


FIGURA 5.1 - Arquitetura Eletrotutor II

A interface com o usuário é dividida em três *frames:* um *frame* principal onde os textos e figuras são apresentados, *um frame* de menu, através do qual o aluno pode controlar o andamento das lições e um *frame* separado onde são apresentados os exercícios e exemplos.

Nesta implementação, não existe nenhum mecanismo de identificação do usuário ou de modelagem do aluno.

O ambiente Eletrotutor II está disponível no endereço *http://www. inf. ufrgs. br/~rsilv/eletro.html.*

5.3 Avaliações do Eletrotutor

Com a finalidade de verificar a eficácia das duas abordagens do Eletrotutor foi elaborada uma avaliação de pesquisa experimental para cada uma das versões. Os resultados são mostrados na Tabela 5.1 [SIL 96].

Na avaliação do Eletrotutor I [SIL 92], os alunos foram divididos em dois grupos: o primeiro foi submetido ao ensino através do tutor e o segundo submetido ao ensino do mesmo conteúdo através de aula expositiva. Após, os alunos realizaram testes de conhecimento, relativos à matéria, através dos quais foram feitas comparações de performance de cada um dos três grupos em um pré-teste e um pós-teste. Além disto foram realizadas observações sistemáticas e entrevistas com os alunos submetidos ao uso do programa com o objetivo de levantar dados complementares sobre o uso do tutor. Esta avaliação foi feita em uma escola de Porto Alegre com alunos do terceiro ano do segundo grau.

A avaliação do Eletrotutor II foi feita com uma amostra de 149 alunos entre o segundo e o terceiro ano do segundo grau, sendo 58 de uma escola particular da cidade de Viamão (Escola 1) e 91 alunos de uma escola particular de Porto Alegre (Escola 2). Em ambas escolas os alunos foram divididos em dois grupos: um participando de aulas de ensino com o ambiente e a outra participando de aulas expositivas.

Após as aulas, todos os alunos foram submetidos a uma prova sobre duas unidades (Lei de Ohm e Corrente Elétrica). Tal prova foi elaborada com o auxílio de dois professores regentes da disciplina de Física da Escola 1 e Escola 2, baseada na prova utilizada em [SIL 92]. O critério de atribuição das notas foi feito da seguinte forma: para cada questão correta foi atribuído 1,0 ponto, chegando-se a pontuação máxima de 10,0 pontos. A correção foi feita pela pesquisadora, as provas foram devolvidas aos alunos e as notas destas foram utilizadas como avaliação bimestral nas duas escolas.

Para avaliar os resultados obtidos durante o experimento, foram aplicados dois teste estatísticos, utilizando-se a ferramenta SPSS *(Statisical Package for Social Sciences):* o teste paramétrico T de Student e o teste não paramétrico U de Mann-Whithney.

O desvio padrão do Eletrotutor I foi menor do que o Eletrotutor II, manifestando a homogeneidade da turma submetida ao primeiro tutor. Quanto aos resultados advindos dos testes estatísticos usados, têm-se que as probabilidades P encontradas demonstram que para os testes realizados no Eletrotutor I a diferença não é significativa, pois os valores encontrados foram $P<0.26$ e $P < 0.18$, que são maiores que 0,05 (valor de significância estipulado para este tipo de teste), o mesmo não acontece para o Eletrotutor n, o valor encontrado de P pode ser considerado muito significativo.

TABELA 5.1 - Resultados das Avaliações do Eletrotutor

		N°de **Alunos**	**Médias**	**Difer. das médias**	**Desvio Padrão**	**Valor das** Estatísticas	P	**Signifi-cância**
Eletro-tutor I (1992)	Aulas com o Tutor	**15**	7,80	0,58	1,32	Teste T de Student calculado **T=2,29** **Z=l,33** U de Mann-Whitney calculado	0.26 0,18	Não Signifi-**cativo**
	Aulas Exposi-tiva	**13**	8,38		1,39			
Eletro-tutor li (1997)	Aulas com o Tutor	**73**	7.90	0,61	**1,51**	Teste T de Student calculado **T=2,58** **Z=2,54** U de Mann-Whitney calculado	0.011 0,011	Signifi-cativo
	Aulas Ex posi-tiva	**76**	**8,51**		1,36			

Os resultados da avaliação do Eletrotutor I demonstraram que a utilização do tutor como ambiente de aprendizagem foi tão eficiente quanto o ensino com a aula expositiva. Com a utilização de técnicas de IA, ocorre um controle sobre a quantidade mínima de exemplos e exercícios que o aluno deve fazer, de acordo com o seu desempenho, obtendo-se assim melhores resultados.

No Eletrotutor II não existe controle de desempenho, como na primeira versão. O novo ambiente, embora mais atrativo do ponto de vista da interface, não atua no monitoramento do aluno, isto pode ser apontado como o motivo pelo qual os alunos submetidos a aula tutorial não obtiveram notas tão boas quanto os alunos que tiveram aula expositiva.

A terceira versão do Eletrotutor agrega aspectos das versões anteriores, como: a utilização de uma metodologia de ensino e o controle do aluno baseados no EletroTutor I; uma arquitetura distribuída e uma interface gráfica, como no Eletrotutor II. Além disso, o Eletrotutor III incorpora uma arquitetura baseada em agentes artificiais e humanos; a partir dos elementos citados espera-se que o aluno assimile os conteúdos abordados pelo sistema.

6 Eletrotutor III

O Eletrotutor III implementa um ambiente distribuído de ensino-aprendizagem inteligente (ITLE) baseado em uma arquitetura multiagente, na qual os agentes possuem características abordadas na seção 3.1, citadas por [HAY 97]. 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. Todos os agentes são baseados no Agente Genérico, descrito na seção 6.2. As características individuais dos agentes são citadas na seção 6.2.5.

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, tomar 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 99].

A fim de poder atuar sobre o ambiente, cada agente possuirá uma representação interna parcial do mundo que o rodeia. Para isso, empregou-se a metáfora de estados mentais para modelar a base de conhecimento que representa os estados do ambiente onde o agente está inserido.

As definições formais dos estados mentais seguem a especificação de Mora. et ai. [MOR 99] *(X-BDI - eXecutable BDI)*, seu detalhamento é apresentado no item a seguir, no qual as descrições foram elaboradas utilizando o texto original de Michael Mora [MOR 99].

6.1 O formalismo lógico de X-BDI

Assim como em [GIR 99], a escolha do formalismo proposto por Mora et al., produto do PPGC/UFRGS, deve-se a dois fatores:

- é uma teoria de agentes computacional disponível para uso imediato. Embora questões como performance e eficiência não sejam consideradas, a partir das especificações utilizadas para formalizar o agente é possível executá-los e verificar o seu comportamento;
- possui baixa complexidade para descrição dos agentes. Se define formalmente um agente apenas em termos dos seus estados mentais (crença, desejos e intenções), as propriedades que se deseja que o agente possua e verificar se estas propriedades acontecem de fato.

Segundo Mora [MOR 99], sua proposta disponibiliza um sistema formal cuja linguagem seja adequada para a representação de conhecimento e suporte a diversos tipos de raciocínio, tratados de forma computacional, além de fornecer as ferramentas necessárias para se modelar os vários estados mentais.

O ambiente base utilizado para construção do X-BDI é a programação lógica estendida com negação explícita - ELP *{Extended Logic Programming with explicit negatiori),* com a semântica bem fundada estendida - WFSX *{Well-Founded Semantics eXtended for explicit negation).* Para este formalismo lógico, existe um procedimento de derivação descendente, o SLX *(Selected Linear resolution for eXtended programs),* que é completo e correto em relação a WFSX. Segundo [MOR 99] este procedimento permite provar que determinado literal pertence ou não ao modelo do programa, sem exigir o cálculo do modelo completo do programa. Existe um algoritmo bem definido para o procedimento de revisão do programa, sobre o qual se baseiam os diversos tipos de raciocínio.

A estrutura da ELP, além de prover procedimentos computacionais para a prova de teorias que se expressam em sua linguagem, também provê um mecanismo para determinar como minimizar a remoção de contradições. Os benefícios da utilização de um modelo com estas características é provido pelo formalismo. Então, inicialmente são definidos estes três estados mentais e as relações estáticas entre eles, isto é constrangimento em consistência entre esses estados mentais. Depois, avança com a definição de aspectos dinâmicos de estados mentais, isto é como o agente escolhe suas intenções, e quando e como revisa suas intenções.

A lógica de programação estendida (ELP) é um conjunto de regras.$H \leftarrow B_1,\mathrm{K}\ ,B_n,not\ C_1,\mathrm{K}\ ,not\ C_m \quad (\mathrm{m},\mathrm{n} \geq 0)$, onde são $H,B_1,\mathrm{K}\ ,B_n,C_1,\mathrm{K}\ ,C_m$ literais objetivos. Um literal objetivo é um átomo *A* ou sua negação explícita —*A.* O símbolo *not* significa a negação por omissão ou negação por falha, e *not L* é uma literal por omissão. Literais podem ser literais objetivos ou literais omissão, sendo que e ——*L = L.*

A linguagem também permite tratar da consistência das restrições através de $A \Leftarrow B_1,\mathrm{K}\ B_n,not\ C_1,\mathrm{K}\ ,not\ C_m\ (\mathrm{m},\mathrm{n} \geq 0)$ onde $A,B_1,\mathrm{K}\ ,B_n,C_1,\mathrm{K}\ ,C_n$ são objetivos literais, instanciando que *A* pode suportar se o seu corpo Bi,K ,*Bn,*Ci,K , Cn suporta também. Particularmente, quando *A*- ±, onde $\perp$ instância por *contradição,* isto significa que uma contradição aparece quando o corpo da contradição a suporta.

Quando raciocinamos sobre pró-atitudes tais como desejos e intenções, necessitamos lidar com propriedades que possam ser suportadas certo instante de tempo e com ações que devem ser executadas num determinado tempo.

Entretanto, para representá-las e raciocinar a seu respeito, necessitamos de um formalismo lógico que lide com ações e tempo. Móra utiliza uma versão modificada do Event Calculus (EC) proposto por [MOR 95]. O predicado *holds_at(P,T),* define que a propriedade *P* é verdadeira no tempo *T* se:

O predicado *happens(E.T)* significa que o evento *E* ocorreu num tempo *T; initiates(E,P)* significa que o evento *E* inicia a propriedade *P* no tempo que o evento *E* ocorre; *terminates(E,P)* significa que o evento P terminou; *persists(Te,P,T)* significa que *P* persiste desde *Te* até *T* (pelo menos). Existe uma variável especial denominada de *Now* que representa o tempo presente. Observe que a propriedade *P* é verdadeira num tempo *T (holds_at(P, T))* se tiver um evento prévio que inicia em Pese P persiste até *T. P* persiste até *T* se não pode ser provado por omissão a existência de outro evento

que termina *P* antes do tempo *T.* A Figura 6.1, apresenta alguns aspectos da linguagem. Maiores detalhes podem ser obtidos em [MOR 97], [MOR 98], [MOR 99].

```
¬ holds _ AT (P , T )   ←   not holds _ at (P , T ).
holds _ at (P , T )     ←   initially (P ),
                            persists( 0 ,P,T).
holds_at(P ,T)          ←   happens (E , Te ),
                            initiates (E , P ),
                            Te < T ,
                            persists (Te , P , T ).
holds _ at (P , T )     ←   senses (P,Ts),
                            Ts < T ,
                            persists (Ts , P , T ).
persists (Te , P , T )  ←   not clipped (Te , P , T ).
clipped (Te , P , T )   ←   happens (Ie , TIe ),
                            terminates ( Ie , P ),
                            not out (TIe , Te , T ).
out (TIe , Te , T )     ←   T ≤ TIe .
out (TIe , Te , T )     ←   TIe < Te .
```

FIGURA 6.1 - Elementos da linguagem do X-BDI

ELP com o EC provê os elementos que são precisos para se modelar estados mentais. Não obstante, o uso de linguagem de mais alto nível, ela nos permitiria ter uma descrição mais simples e mais clara de ações e seus efeitos. Então, Mora et al. lançam a linguagem de ELP e as primitivas de EC em uma sintaxe mais simples que é semelhante a linguagem ONE proposta por Gelfond e Lifschitz [GEL 92]. Neste idioma são traduzidas as orações ELP e as primitives de EC, durante a argumentação segundo [QUA 97]. As orações são:
A causa F ifPj,...,P_n - ação *A* causa propriedade *F* se suporta a proposição *P; F after A if Pi,...,Pn- suporta a* propriedade *F* após a ação *A* se suporta a proposição *P;*
A occurs_in E - ação *A* ocorre quando o evento *E* ocorre;
Epreceeds E' - o evento *E* ocorre antes do evento E".

A linguagem ainda provê meios para referência de crenças que podem conter o futuro ou que deveriam ter contido o passado. Isto é feito através do operador *next(P)* declarando que a propriedade P deveria ser suportar no próximo momento de tempo. Uma vez tendo o formalismo lógico fixado, pode-se definir o modelo de BDI.

Uma vez construída a base formal necessária, passa-se a definição dos estados mentais. Inicialmente são definidos os aspectos estáticos do modelo: quais são os três estados mentais básicos, seu conteúdo, e as condições e restrições que tais estados devem satisfazer.

O *X-BDI* possui um ambiente de implementação com alto nível de abstração (os estados mentais), o que reduz a complexidade no desenvolvimento de sistemas baseados em agentes.

A linguagem do *X-BDI* possui baixa complexidade no que diz respeito à sintaxe, facilitando sobremaneira o trabalho de especificação e implementação de agentes baseados na arquitetura BDI. A arquitetura BDI passa ser uma paradigma de implementação de agentes cognitivos devido à possibilidade de se executar o modelo formal.

6.2 Agente Genérico

Baseado nos trabalhos de Shoham [SHO 91][SHO 93], Móra [MÓR 98] e Brazier [BRA 97], os agentes cognitivos propostos possuem uma estrutura composta por vários elementos: primitivas de estados mentais: crenças, desejos e intenções (arquitetura BDI - *Belief, Desire* and *Intention)* e dois tipos de ação: as capacidades pessoais e de comunicação, como mostra a Figura 6.2.

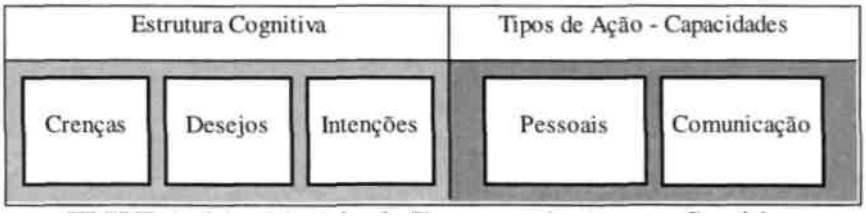


FIGURA 6.2 - Modelo da Estrutura do Agente Genérico

6.2.1 Estrutura Cognitiva

A estrutura cognitiva contém os estados mentais que compõem o agente genérico e as regras que governam as interações entre tais estados mentais, e conseqüentemente, o comportamento do agente. Tal estrutura segue a seguinte definição: B é o conjunto de crenças do agente, D é o conjunto de desejos do agente, I é o conjunto das intenções do agente.

Crenças constituem a informação sobre a atitude do agente, elas representam as informações que o agente tem sobre o ambiente e sobre si mesmo. O conjunto B contém orações que descrevem o domínio do problema e utilizam ELP. Um agente Ag acredita em uma propriedade P, num tempo T se, a partir de B e T, o agente deduzir BEL(Ag, P) durante o tempo T. Assume-se que o agente atualiza continuamente suas crenças para refletir mudanças que detecta no ambiente e que, sempre quando uma nova crença é somada ao conjunto de crenças, a consistência é mantida.

Os desejos do agente são um conjunto de orações DES(Ag,P,Atr), sendo que Ag é a identificação do agente, P é uma propriedade e Atr é uma lista de atributos. Os desejos são relacionados ao estado de eventos que o agente quer provocar. O agente irá escolher os desejos em função das crenças válidas a respeito do ambiente, no dado momento.

Intenções são caracterizadas como algo que o agente se dedica a tentar cumprir, neste trabalho elas são geradas pelos desejos. Uma intenção não pode ser contraditória com outras intenções.

6.2.2 Capacidades

A capacidade é utilizada pelo agente para associar uma ação com pré-condições necessárias para realizar uma ação. Uma lista das capacidades do agente define as ações, as quais, o agente pode executar provido das pré-condições necessárias satisfeitas. As ações, aqui, são consideradas como pessoais (ações que afetam ou interagem com o ambiente do agente e não dependem da interação com outros agentes) e de comunicação (ações definidas como envio de mensagens para outros agentes, explicitadas na seção 7.6).

6.2.3 Regras de Comportamento

Todas as ações são executadas como resultado de intenções. Regras de comportamento determinam o curso da ação que um agente deve tomar em todo o ponto do início ao fim da execução do agente. Regras de comportamento podem ser vistas como *WHEN-IF-THEN*.

A parte *WHEN* da regra endereça um novo evento ocorrendo no ambiente do agente e inclui novas mensagens recebidas de outros agentes. O *IF* compara o estado mental corrente ou condições do ambiente com as condições requeridas para que a regra seja aplicada. A parte *THEN* define as ações do agente e mudanças mentais executadas em resposta do evento corrente, modelo mental e ambiente externo. Isto inclui atualização do modelo mental, ações de comunicação e ações pessoais.

Exemplo de formato de regras de comportamento:

NAME nome regra
WHEN
    Condições das Mensagens
IF
    Condições Mentais OU
    Condições do Ambiente
THEN
    Ações Pessoais
    Atualiza Estados Mentais
    Ações de Comunicação

6.2.4 Ciclo do Agente

O primeiro passo é a iniciação das crenças iniciais, desejos iniciais, intenções iniciais, capacidades e regras de comportamento. Apesar das regras e capacidades serem estáticas, as crenças, desejos e intenções do agente são dinâmicos e portanto,

podem mudar no tempo de vida do agente, tais primitivas de estados mentais são revisadas no ambiente X-BDI.

Na arquitetura do agente genérico as tarefas são decompostas e executadas individualmente ou em grupo de agentes. O modo como a tarefa será decomposta é definido pelo conteúdo das mensagens trocadas entre os agentes.

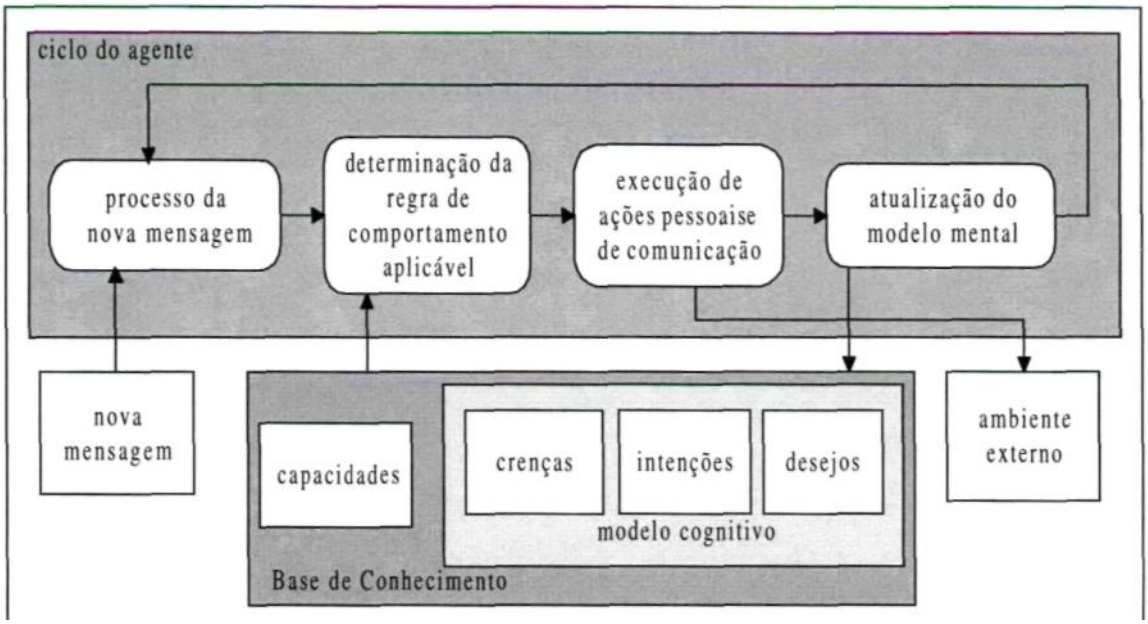


FIGURA 6.3 - Processo de Execução do Agente Genérico

O ciclo de execução do agente, mostrado na Figura 6.3 consiste dos seguintes passos:

- processamento de novas mensagens: neste momento, o agente decompõe a mensagem e reconhece qual a tarefa que esta propõe;
- determinação da regra de comportamento: a partir da análise da mensagem, o agente determina qual regra de comportamento é aplicável para a corrente situação e, se necessário, delega a tarefa a outro(s) agente(s);
- execução de ações pessoais e de comunicação: cada regra de comportamento possui ações pessoais e de comunicação, e é neste momento que tais ações são executadas pelo agente;
- atualização do estado mental: as ações executadas pelo agente no nível anterior podem alterar o seu conhecimento a respeito do mundo, então, é necessário que ocorra um gerenciamento do seu conhecimento a respeito do mundo;

6.2.5 Arquitetura do Eletrotutor III

A sociedade de agentes proposta na terceira versão do Eletrotutor contém agentes autônomos que comunicam-se uns com outros, cada agente possui funções e objetivos dentro de sua especialidade. A Figura 6.4 representa a arquitetura elaborada neste trabalho, baseada em [SIL 98].

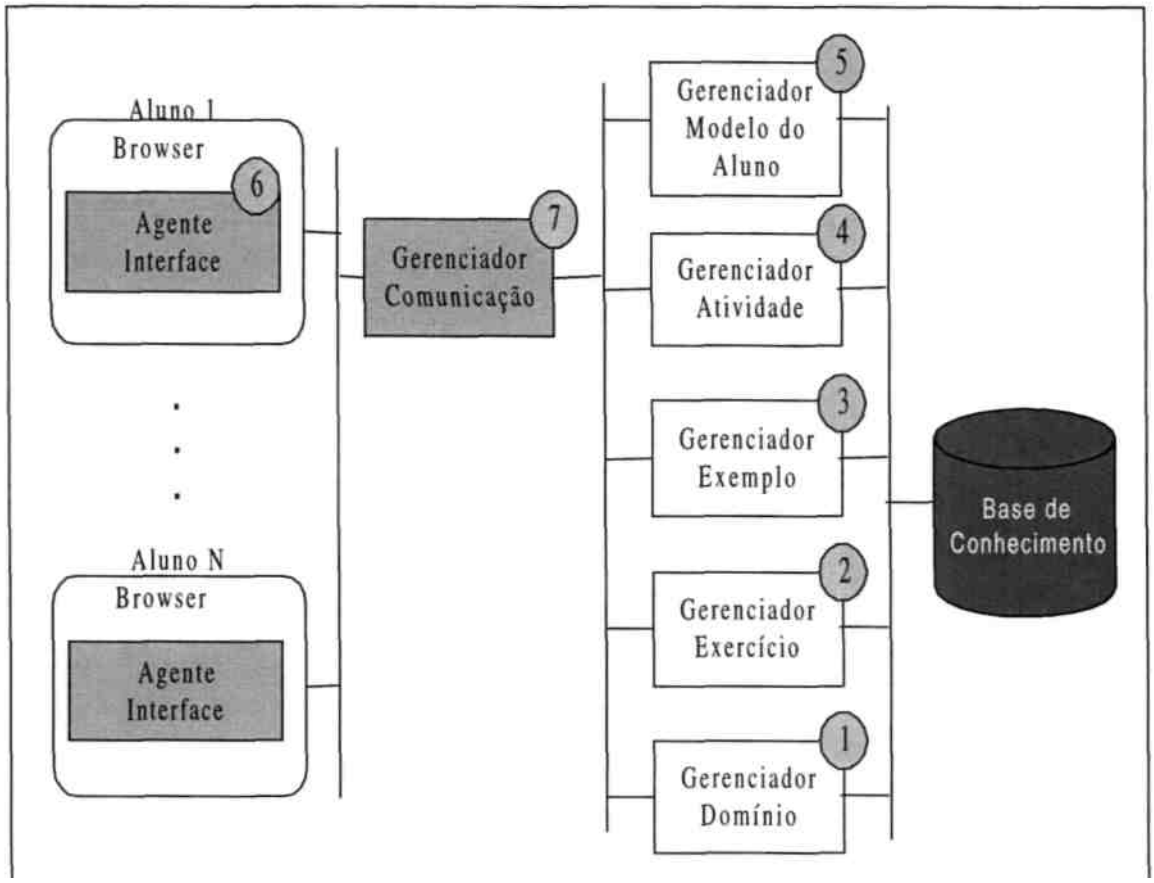


FIGURA 6.4 - Arquitetura do Eletrotutor III

O ambiente de aprendizado inteligente proposto, seguindo a Figura 6.4 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. Todos os agentes propostos são baseados na arquitetura do Agente Genérico, abordada nesta seção, mas apenas o agente Gerenciador Modelo do Aluno necessita utilizar a arquitetura do Agente Genérico por completa, os demais agentes, por serem mais simples, não necessitam de estrutura cognitiva.

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

6.2.5.1 Agente Gerenciador Domínio

O agente Gerenciador Domínio (agente 1) possui a função de recuperar informações referentes ao domínio sobre cada ponto a ser apresentado ao aluno. O agente 1 pode trocar mensagens com os agentes Gerenciador Modelo do Aluno (agente 5) e Agente Interface (agente 6). Como características básicas, o agente 1 é colaborador (autonomia, habilidade social, cooperação), orientado por objetivo e possui continuidade temporal.

6.2.5.2 Agente Gerenciador de Exercício

O agente Gerenciador Exercício (agente 2) possui a tarefa de propor exercícios e avaliar respostas do aluno. A troca de mensagens pode ser feita com o agente 5 ou o agente 6. O agente 2 possui como características básicas a colaboração, adaptabilidade e continuidade temporal.

6.2.5.3 Agente Gerenciador de Exemplo

O agente Gerenciador Exemplo (agente 3) é responsável pela tarefa de propor exemplos ao aluno. O agente 3 pode trocar mensagens com os agentes 5 e 6. Como características básicas o agente 3 é colaborador, adaptável e continuidade temporal.

6.2.5.4 Agente Gerenciador de Atividades

O agente Gerenciador Atividade (agente 4) é responsável pela tarefa de propor atividades extras ao aluno. O agente 4 pode trocar mensagens com os agentes 5 e 6. As características básicas do agente 4 são: colaboração, adaptabilidade e continuidade temporal.

6.2.5.5 Agente Gerenciador do Modelo do Aluno

O agente Gerenciador Modelo do Aluno (agente 5) 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 5 recebe informação do agente interface (agente 6), sobre a identificação do aluno, para que possa recuperar o Modelo do Aluno correspondente e conforme o andamento das lições decidir qual a estratégia de ensino a ser utilizada com este aluno.

O agente 5 possui como características a colaboração, adaptabilidade, racionalidade e continuidade temporal.

6.2.5.6 Agente Interface

O agente interface (agente 6) 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 (agente 7). O agente 6 não possui inteligência, apenas repassa as atividades do aluno na interface do sistema. Como características, o agente 6 possui reatividade e autonomia.

6.2.5.7 Agente Gerenciador de Comunicação

O agente Gerenciador de Comunicação é responsável pela comunicação do agente Interface *(applet)* com os demais agentes. Recebe e envia mensagens ao agente 1 ao 5, conforme necessidades do Agente 6, não trata mensagens, só as repassa. É executado na mesma máquina que possui a *applet* (agente 6). Não possui nenhuma inteligência. Como características, o agente 7 possui reatividade, habilidade social, autonomia e continuidade temporal.

6.3 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, descrita na seção 3.4.2, 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 mensagem são definidas em duas partes, explicitadas na seção 7.6, são elas:

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

Todas as mensagens que os agentes necessitam enviar e receber foram mapeadas, já que é através da troca das mensagens entre os agentes que o ciclo interno destes, ilustrado na Figura 6.3, se baseia. Portanto, os agentes sempre terão a capacidade de decompor as mensagens e decidirem qual as ações pessoais e de comunicação que devem executar.

6.4 Metodologia de Ensino

A metodologia de ensino empregada no Eletrotutor III segue a descrição de [SIL 92], a qual procura se adaptar à sistemática de ensino do estudante, assim, a forma do conteúdo é apresentada de acordo com as características que o aluno apresenta, num dado momento.

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.

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. Para levantar este diagnóstico, o Eletrotutor deverá lançar mão da seguinte estratégia: o tutor formulará uma seqüência de estratégias de ensino a serem desenvolvidas para que aquele aluno em particular atinja o objetivo de assimilar o conteúdo ministrado pelo tutor. Estas estratégias

consistem na seqüência de conteúdos, de exemplos e de exercícios que serão propostos ao aluno.

O conjunto de informações a respeito do aluno, constitui-se no denominado Modelo Cognitivo do Aluno. Este modelo de aluno deverá ser atualizado a todo momento, através da verificação da performance deste aluno, por meio dos seus erros e acertos nos exercícios propostos, nas necessidade de reforço instrucional que ocorrem, etc.

O tutor, para atingir o seu objetivo, deverá, durante a sessão de estudo adaptar dinamicamente a sua estratégia de ensino, de acordo com seu próprio diagnóstico, assim como por solicitação do aluno, explanando conceitos, gerando exemplos, propondo e corrigindo exercícios, fazendo revisões, remetendo o aluno a estudar outros pontos, enfim, procurando fazer exatamente o que um tutor humano faria para ajudar o aluno, passo a passo, a ir superando os obstáculos até atingir um patamar que ambos julguem satisfatório.

A dinâmica do modo tutorial funciona de um modo tal que, a cada avaliação, é proposta uma série de até, no máximo, sete vezes o mesmo tipo de exercícios, mas sempre com valores diferentes, escolhidos randomicamente em uma tabela de valores pré-determinada. O procedimento seguinte do tutor é determinado pela performance do aluno, sendo que o tutor se comporta de acordo com o resultado obtido em cada série de exercícios de um mesmo tipo, conforme um dos casos descritos na Tabela 6.1.

TABELA 6.1 - Casos de performance do Aluno

Nível	Situação	Estado do Aluno
1	três acertos consecutivos ou cinco acertos alternados em sete tentativas	aluno com conhecimento satisfatório
2	quatro acertos alternados em sete tentativas	aluno com conhecimento razoável
3	três acertos alternados em sete tentativas	aluno com pouco conhecimento
4	três erros consecutivos ou cinco erros alternados em sete tentativas	aluno com conhecimento insatisfatório

Segundo a Tabela 6.1, no primeiro caso, o aluno encontra-se num estado que foi denominado Nível 1, no qual o tutor encaminha o aluno para a próxima série de exercícios ou a próxima lição, dependendo do ponto onde ele se encontre. No segundo caso, denominado Nível 2, o aluno é remetido a uma nova série do mesmo tipo de exercício para submeter-se a uma nova avaliação. No terceiro, o aluno encontra-se num estado denominado Nível 3, onde o aluno é remetido novamente a uma nova série de exemplos da lição. No quarto e último caso, denominado Nível 4, ele é remetido novamente ao início da lição a fim de recapitular a matéria. Esta nomenclatura adotada é uma forma de representar o estado do aluno, durante a sua avaliação, internamente para o programa, não sendo portanto acessível ao aluno.

Deste modo, o tutor encaminha o aluno a percorrer todas as lições, executando os exercícios até que ele obtenha uma boa performance em cada um deles. No entanto,

a todo momento, o aluno tem liberdade para interromper o processo tutorial, modificando a seqüência estabelecida e acompanhando a lição que desejar, desde que a lição desejada já tenha sido previamente estudada por este aluno.

Este modelo adotado é essencialmente quantitativo, como pode ser percebido. Como complemento futuro para este trabalho poderia ser aqui adotado, um modelo mais sofisticado, no qual o tutor percebesse o contexto dos erros cometidos pelo aluno, o tipo e a freqüência de erros cometidos, apresentação de lições alternativas, revisões de matemática e formas mais sofisticadas de avaliação.

7 Implementação

O protótipo inicial implementa o Modo Tutorial, tal protótipo está sendo desenvolvido na linguagem Java, com o objetivo de manter a independência entre plataformas e o acesso através da *Internet.*

Os agentes trocam mensagens através do *Remote Method Invocation* (RMI), citado na seção 3.4.1.2, pois este meio permite a passagem de objetos criados pelos implementadores. As mensagens trocadas entre os agentes são descritas na seção 7.6 e seguem um protocolo baseado na linguagem KQML.

A base de conhecimento está armazenada em um banco de dados relacionai, o Oracle versão 8.0.4. Para a comunicação do Java com o banco de dados, utilizou-se o driver JDBC-Thin da Oracle.

O material instrucional apresentado no Eletrotutor III é formado por lições (páginas HTML), exemplos (páginas HTML e *Javascript)* e exercícios *{applets* Java), grande parte deste material é o mesmo utilizado na segunda versão do Eletrotutor.

Como pontos da implementação do protótipo têm-se:

- o número de acertos/erros de cada lição e o número de exemplos e exercícios realizados são utilizados como principal método de avaliação para obter o grau de compreensão do aluno;
- o agente Interface é uma *applet-Client.* Os demais agentes são aplicações Java-Server, executadas de forma distribuída; exceção - agente Gerenciador de Comunicação, executado na mesma máquina que agente Interface, pois sua função é enviar mensagens do agente Interface aos demais agentes localizados de forma distribuída e receber mensagens destes agentes e repassá-las ao agente Interface;
- o agente Gerenciador de Atividades não será implementado, este agente propõe atividades como VRML, citada na seção 4.1 (Virtua Cell), animações, e em princípio não possui inteligência, isso dependerá de um estudo de avaliação dos tipos de atividades extras que serão utilizadas e se estas possuirão algum tipo de avaliação, portanto este agente não é necessário neste primeiro momento para a validação da arquitetura de agentes proposta.
- O Modelo do Aluno é simples, sendo composto por:
    - identificação do aluno;
    - identificação do agente que está no controle da lição;
    - número de acertos/erros de cada lição;
    - último número de exemplos requisitados pelo aluno;
    - último número de exemplos requisitados pelo aluno;
    - histórico de todo os passos do aluno (quais lições, exemplos, exercícios esteja realizou).

O sistema segue a seguinte ordem de execução: 1. Quando o usuário se conecta ao sistema pela primeira vez, o agente Gerenciador Modelo do Aluno cadastra o usuário e cria o modelo de aluno inicial para este;

2. Depois de recuperado o modelo do aluno, no caso deste já ter se conectado anteriormente, o Agente Gerenciador Modelo do Aluno envia uma mensagem ao Agente que possui o controle da lição (exemplo: se o aluno estava respondendo a exercícios quando encerrou a sessão, quando esta é reiniciada, o Agente responsável pelos exercícios recupera o controle e continua seu trabalho). O agente que possui o controle troca mensagens com o Agente de Interface que por sua vez, monta a tela de acordo com o conteúdo destas mensagens;
3. Após o encerramento de uma sessão de exercícios/exemplos/atividades, o Agente correspondente envia uma mensagem ao Agente Gerenciador Modelo do Aluno para que este atualize os passos do aluno e a partir da performance deste decidir qual a próxima atitude a tomar (mais exemplos, mais exercícios, próxima lição,...);
4. As mensagens são enviadas apenas ao agente destinatário que lê/executa as ações que a compõem, se estas vão de acordo com suas crenças.

7.1 Classes do ambiente Eletrotutor III

A Figura 7.1 representa o diagrama das principais classes implementadas que compõem o ambiente Eletrotutor III, mostrando as relações **utiliza** e **implementa** (conceito Java - para que uma classe *implements* outra classe do tipo *interface* ela deve conter todos os métodos da classe *interface* implementados), as classes são as seguintes:

- **Agent:** classe que possui as especificações dos agentes, explicitada na seção 7.3;
- **KQMLMessage:** implementa o cabeçalho das mensagens, explicitado na seção 7.6;
- **ContentMessage:** implementa o conteúdo das mensagens, explicitada na seção 7.6;
- **server:** *interface* Java RMI, explicitada a seção 7.2.**1**;
- **Interface:** implementa o agente Interface, é o único RMI *client* do ambiente, representa a interface gráfica deste *(applet),* explicitada na seção 7.8;
- **ServerAppIet:** implementa o RMI *server* que se comunica com o agente Interface (classe Interface.java) e com demais agentes do ambiente, eqüivale ao agente Gerenciador de Comunicação, deve ser executado na mesma máquina que o agente Interface;
- **Serverlmpl:** implementa os demais agentes RMI *servers* através da especificação do agente (classe agent.java), ou seja, este programa é executado diversas vezes com parâmetros diferentes, tais como, nome do agente, nome da máquina, porta;
- **BuscaMensagem:** implementa todos os tipos de mensagem mapeados no ambiente, é nesta classe que o agente reconhece e trata as mensagens, é utilizada pelos agentes do tipo Serverlmpl;
- Manda **Resposta:** é utilizada sempre que um agente do tipo Serverlmpl deseja enviar uma mensagem.

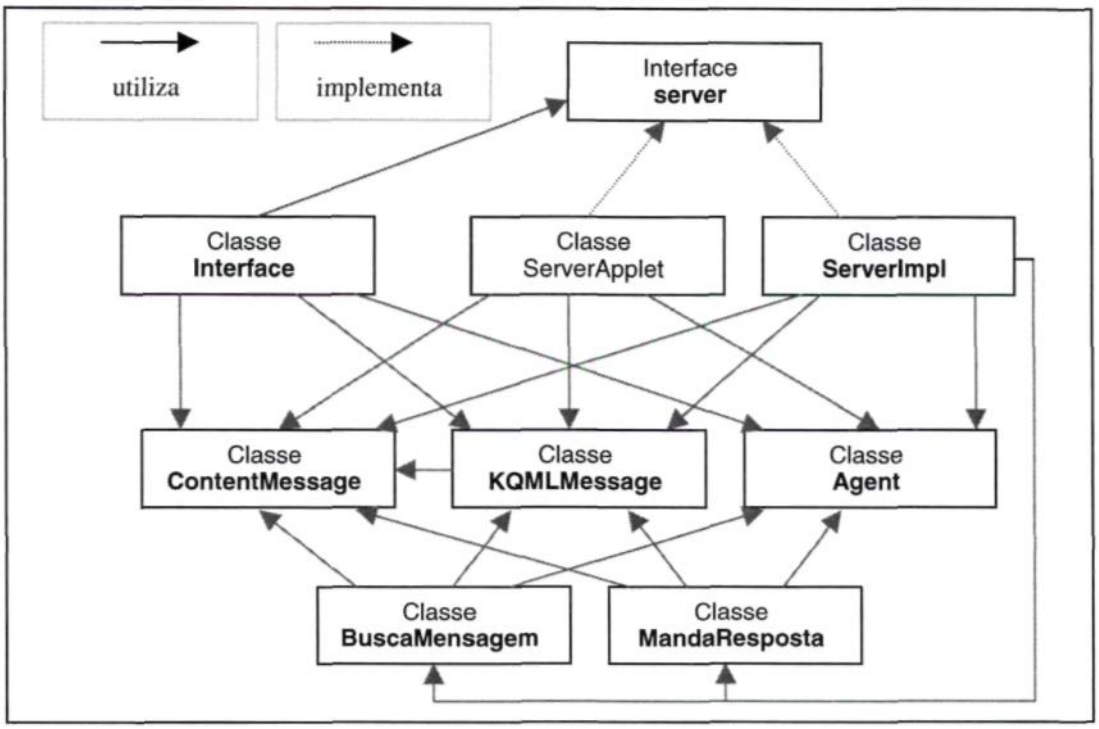


FIGURA 7.1 - Diagrama das principais classes

Além das classes descritas acima, existe outras classes complementares, como a *applet* de exercício, que consiste em um *template* que contém fórmulas para a aplicação dos exercícios numéricos. O arquivo HTML de chamada de um exercício contém o enunciado e o tipo de formula que deve ser utilizada, assim, o *template* gera um exercício com o enunciado especificado, valores randômicos e aplica-os a fórmula do exercício especificado para obter a resposta correta deste e podendo assim, corrigir a resposta informada pelo aluno.

7.2 Implementação dos Métodos de Comunicação

Nesta implementação, são utilizados três métodos distintos de comunicação, para a comunicação entre os agentes foi utilizado o RMI Java, para a comunicação entre os agentes e o banco relacionai foi utilizado o *driver* JDBC-Thin da Oracle, e para a comunicação do ambiente X-BDI com o ambiente Eletrotutor está sendo estudada a utilização de *sockets* Java.

A Figura 7.2 ilustra a arquitetura do Eletrotutor III em relação aos métodos de comunicação utilizados, os quais serão detalhados a seguir em alguns aspectos relevantes à implementação. Além dos métodos citados acima, também foi implementado a comunicação entre *applets* através de programação *Java* e *JavaScript,* isso ocorre na interação entre um aluno e exercícios, pois o agente Interface *{applet)* deve receber dos exercícios *{applets)* o resultado da avaliação.

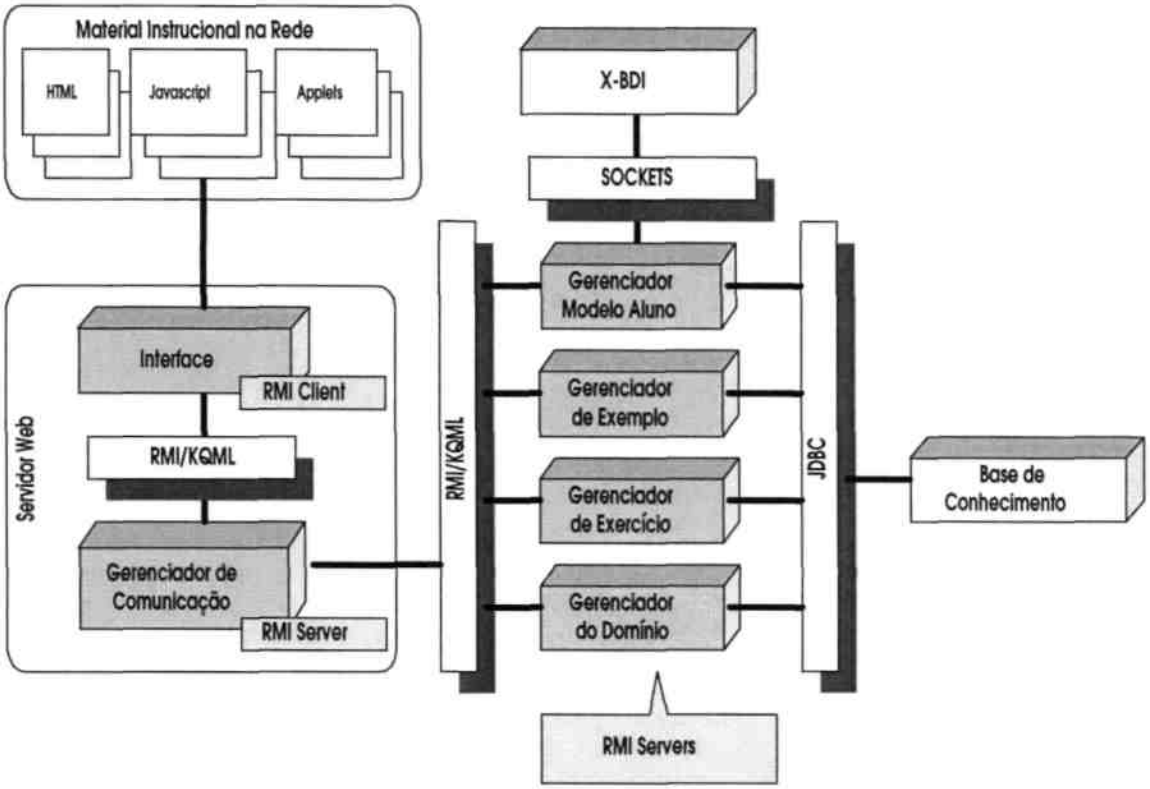


FIGURA 7.2 - Arquitetura do Eletrotutor III e métodos de comunicação

7.2.1 RMI

Para implementar o RMI foi necessário, primeiramente, analisar as possibilidades de comunicação entre os agentes da sociedade. Na Figura 7.2, pode-se reconhecer um RMI *client* que se comunica apenas com o seu RMI *server* (agentes Interface e Gerenciador de Comunicação respectivamente) e os demais agentes são RMI *servers* e podem trocar mensagens com todos os RMI *servers* existentes.

As classes necessárias para a implementação do RMI, neste trabalho, são compostas por uma *interface* (server.class) que contém as chamadas dos métodos que obrigatoriamente devem ser implementados nas classes que utilizarem RMI.

O agente Gerenciador de Comunicação apenas repassa as mensagens que o recebe, ele não necessita de todo o controle de decomposição e composição das mensagens, sendo assim, foram implementadas duas classes RMI *servers*, a ServerApplet (Gerenciador de Comunicação) e os demais agentes (Serverlmpl).

Exemplos de linha de execução de um agente são as seguintes:

(1) C:/>java Serverlmpl agentel jacui.inf.ufrgs.br 1234

(2) C:/>java Serverlmpl agente2 piratini.inf.ufrgs.br 1230

(3) C:/>java ServerApplet agente6 jacui.inf.ufrgs.br 1246

no exemplo (1):
agente1 representa o nome do agente dentro do programa e é através deste nome que ele é registrado no RMI;
jacui.inf.ufrgs.br identifica o endereço de rede na qual o agente será executado;
1234 identifica a porta de comunicação.

A Figura 7.3 apresenta um trecho do código que representa um agente do tipo Agent (classe Agent.java - seção 7.3) sendo registrado pelo seu nome e porta para que outros agentes possam se comunicar com ele, tais dados são passados por parâmetro como mostrado no exemplo acima.

```
Agent AgCorrente = new Agent(args[0],"",args[1],Inteqer.parseInt(args[2]));

ServerImpl server = new ServerImpl(AgenteCorrente);
Registry registry = LocateRegistry.createRegistry(AgenteCorrente.getPort());
registry.rebind(AgCorrente.getName(), server);
```

FIGURA 7.3 - agente registrado no RMI

7.2.2 JDBC-Thin

A Figura 7.4 apresenta um trecho do código da implementação da conexão ao banco de dados. Para realizar a conexão ao banco de dados é preciso definir o tipo de *driver,* o qual depende do banco de dados que está sendo utilizado, e especificar uma *string* a qual contém as informações sobre o tipo de *driver* (thin), o endereço IP da máquina onde se encontra o banco *(host),* a porta de comunicação (port = 1521), o nome do banco (sid = orei), o usuário e a senha de acesso.

```
try{
DriverManager.registerDriver (neworacle.jdbc.driver.OracleDriverO);
conn= DriverManager.getConnection("jdbc:oracle:thin:francine/eletro@
 143.54.8.177:1521:orcl");
stmt ■ conn.createStatement();
} catch (Exception e) {System.out.println("##Erro conexão Banco");}
```

FIGURA 7.4 - conexão ao banco de dados

7.3 Agentes

Os agentes foram implementados na classe chamada Agent, como mostra a Figura 7.5. Os dados que ela possui também estão no banco de dados. Uma vez inicializado, o agente busca na base de conhecimento os agentes que compõem o sistema e seus respectivos dados, evitando assim, acessos desnecessários na base de dados.

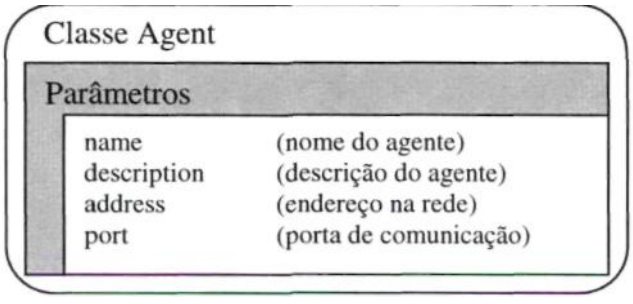


FIGURA 7.5 - Classe Agent

7.4 Os estados mentais

O agente Gerenciador Modelo do Aluno é o único agente desta arquitetura que necessita de um conjunto de crenças e desejos a respeito dos alunos e do seu próprio comportamento. Neste protótipo, a estrutura cognitiva deste agente é implementada por regras.

A fim de ilustrar a forma como os estados mentais do agente Gerenciador do Aluno se comportariam pelo formalismo X-BDI durante a interação entre os agentes e o aluno, selecionou-se a avaliação dos exercícios no modo tutorial, abordada na seção 6.3, que determina ao fim de um conjunto de sete exercícios como o agente Gerenciador Modelo do Aluno deve proceder. Tais crenças são expressas no formalismo abordado na seção 6.1. O agente Gerenciador Modelo do Aluno, nesta etapa do projeto, possui somente um desejo: ajudar o aluno a compreender o material instrucional que está sendo mostrado. Para isso ele acredita que pode ajudar modificando a estratégia de ensino. Esses desejos e crenças são modelados como segue no ambiente X-BDI:

```
des(agente, ajudar).

bel(agente, ajudar) if
            bel(agente,nivel aluno(X)),
            X > 1,
            bel(agente,trocou_estrategia).

bel(agente, trocou_estrategia) if
            bel(agente,enviar_msg(Y)).

bel(agente, enviar msg(1)) if
            bel(agente,nivel aluno(X)), X
            = 2.

bel(agente, enviar msg(2)) if
            bel(agente,nivel aluno(X)), X
            = 3.

bel(agente, enviar msg(3)) if
            bel(agente,nivel aluno(X)), X
            = 4.
```

Suponhamos que o agente acredita que o aluno encontra-se no Nível 2, isso significa que ele, por exemplo, enviará uma mensagem ao ambiente X-BDI com o seguinte conteúdo:

```
[sense(bel(agente,nivel_aluno(2)),2)].
```

portanto, se as pré-condições para a ação de enviar a mensagem 1 forem satisfeitas, neste exemplo existe apenas uma pré-condição de o aluno estar no nível 2, o agente estará em condições de ajudar este aluno. O ambiente X-BDI transforma o desejo em

uma intenção e a satisfaz enviando a mensagem 1 ao agente, e este por sua vez, executará as ações necessárias para dar continuidade ao processo.

A integração entre o ambiente X-BDI e o protótipo está sendo trabalhada no momento, tal integração se fará através de *sockets* Java, ou seja, a troca de mensagens entre o protótipo e o X-BDI será via *sockets.* Esta integração é uma tentativa de ligar os estados mentais ao protótipo e verificar a eficácia destes em um ambiente na Web com as características do Eletrotutor III. No momento atual, o X-BDI está sendo utilizado como formalismo para os estados mentais dos agentes do sistema, mas esta integração terá continuidade em trabalhos futuros no GIA.

7.5 Base de Conhecimento

Para armazenar todas as informações necessárias para a geração das telas de lições, exemplos, exercícios e atividades extras, assim como, o armazenamento dos dados do aluno, foi necessário criar um modelo de Entidade e Relacionamento (ER) mostrada na Figura 7.6. Este modelo é composto por tabelas ditas básicas e compostas descritas respectivamente na Tabela 7.1 e Tabela 7.2. Este modelo comporta várias matérias, portanto, poderá ser utilizado para outras propostas/implementações de ambientes de ensino/aprendizado que comportem esta modelagem.

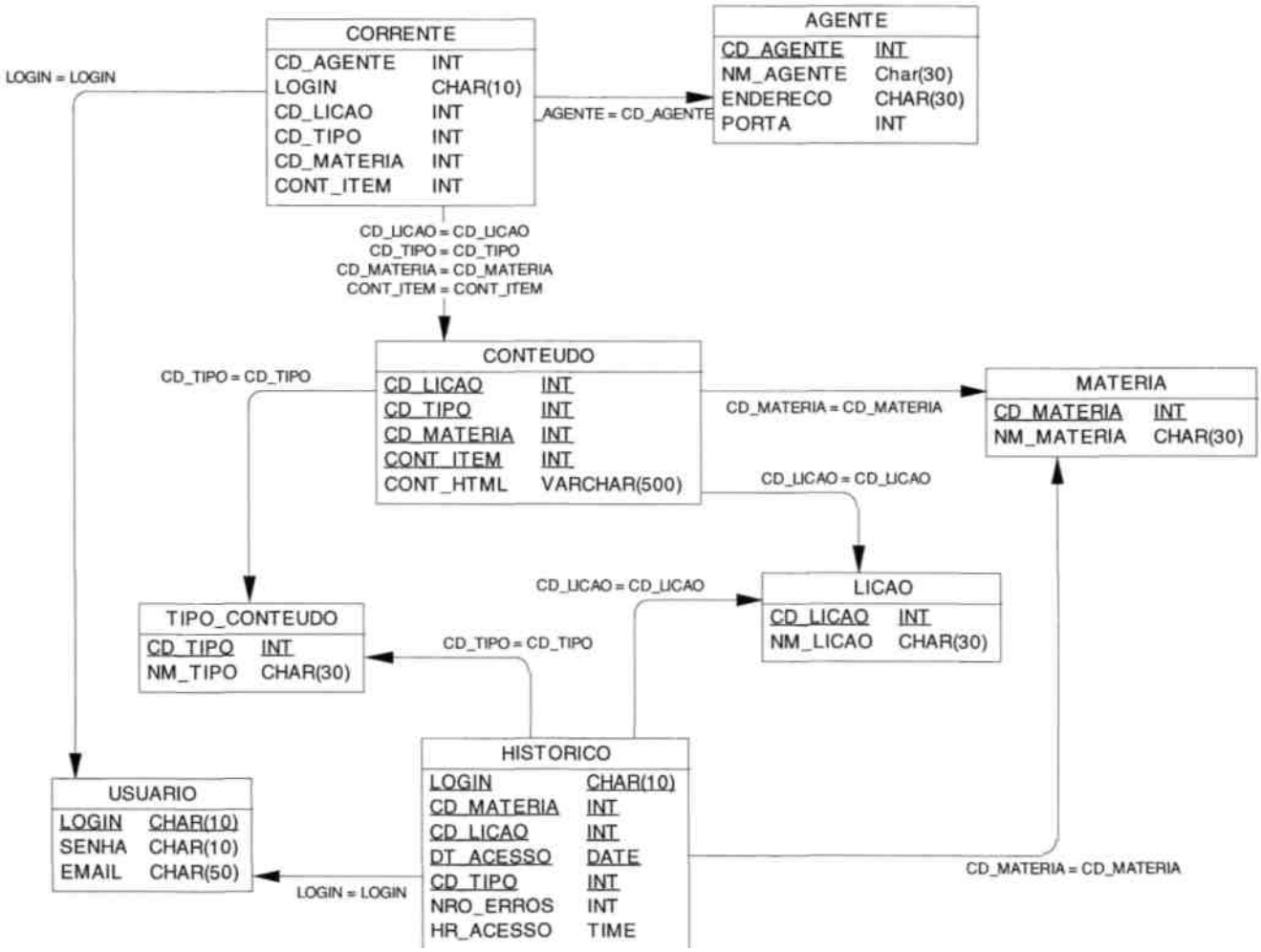


FIGURA 7.6 - ER do Eletrotutor III

Seguindo os relacionamentos mostrados na Figura 7.6, podem existir várias matérias, que por sua vez podem conter várias lições de diversos tipos de conteúdo (tipos de conteúdo são os 'exemplos', 'exercícios', 'atividades extras' ou 'domínio'), portanto, o conteúdo é formado por várias matérias, lições e tipos de conteúdo. A tabela Histórico armazena todos os acessos do usuário, já a tabela corrente, apenas o último acesso e qual agente estava no controle naquele momento. O modelo descrito é simples mas suficiente para as necessidades atuais.

TABELA 7.1 - Tabelas básicas

Tabelas Básicas	Descrição
AGENTE (CDAGENTE, NMAGENTE, ENDEREÇO, PORTA)	• Descreve os Agentes do sistema. • Possui código e nome do agente Exemplo: (1,'Ag.Domínio',' ijui.inf.ufrgs.br', 1234)
USUÁRIO (LOGIN. SENHA, EMAEL)	• Identificação do Usuário do Sistema. • Possui login(identificação), senha e email do usuário Exemplo: ('Francine','fr','fran@inf.ufrgs.br')
TIPO_CONTEUDO (CDTIPO, NMTIPO)	• Descreve tipos de conteúdo do sistema, se exemplo, exercício, atividade extra ou domínio. • Possui código e nome do Tipo Exemplo: (2, 'Exercício')
LICAO (CDLICAO, NMLICAO)	• Descreve lições do sistema. • Possui código e nome da Lição Exemplo: (32,'Eletrodinâmica')
MATÉRIA (CDMATERIA. NMMATERIA)	Descreve matérias do sistema. Exemplo: (18,'Física')

TABELA 7.2 - Tabelas Compostas

Tabelas Compostas	Descrição
CONTEÚDO (CD MATÉRIA, CD LICAO, CD TIPO. CONTITEM. HTMLTXT)	• conteúdo formado por uma Matéria que é composta por lições, cada lição é composta por tipos de conteúdo, cada tipo de conteúdo possui páginas (CONTITEM) e estas páginas são links(.HTML). Exemplo: (18,32.2,l,'http://www....')
CORRENTE (CD AGENTE, LOGIN, CD MATÉRIA, CD LICAO,	• contém uma ocorrência por usuário, possuindo a última ação deste no sistema. Exemplo:

CD TIPO, CONT ITEM	(1,'Francine',18,32,1,1)
HISTÓRICO (LOGIN. CDMATERIA, CDLICAO, DTACESSO, HRACESSO, NRERROS)	• o Histórico possui todos os passos do usuário. Exemplo: ('Francine', 18,32,1999/12/01,12:00,1)

7.6 Mensagens

No Eletrotutor III 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, além das vantagens descritas na seção 3.3.

As mensagens e a troca destas entre os agentes são uma das principais contribuições deste trabalho. A implementação da mensagem, ilustrada na Figura 7.7, é composta de dois módulos: um formato geral que possui especificação KQML (classe KQMLMessage) e um formato específico que compõem a tarefa que o agente deve executar (classe ContentMessage). A especificação da mensagem foi elaborada desta maneira pois o RMI permite a passagem de objetos criados pelo implementador.

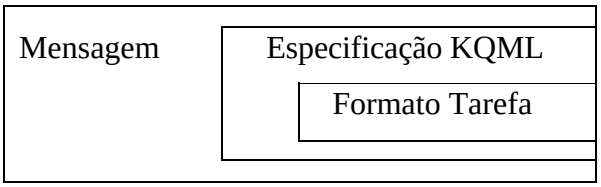


FIGURA 7.7 - Módulos da Mensagem

7.6.1 Implementação KQML

KQML possui um conjunto de parâmetros chaves reservados, sendo útil para estabelecer uma uniformidade dos parâmetros e suporte para que programas entendam *performatives* desconhecidas mas com parâmetros conhecidos. Os parâmetros reservados e seus significados estão demonstrados na Tabela 7.3, tais parâmetros foram implementados na forma de uma classe Java, chamada KQMLMessage, como mostra a Figura 7.8.

TABELA 7.3 - Parâmetros chaves das *Performatives*

Parâmetros-Chave	**Significado**
:sender	O atual transmissor de uma *performative*
.receiver	o atual receptor da *performative*

language	o nome da linguagem de representação do parâmetro *:content*
:ontology	o nome da ontologia usada no *: content*
:in-reply-to	o nome esperado para resposta
:reply-with	Se o transmissor espera uma resposta, e se sim, um rótulo para resposta
:force	se o transmissor irá sempre negar o significado da *performative*
:content	a informação sobre o que a *performative* expressa

Classe KQMLMessage

Parâmetros	
performative	(nome da *performative)*
sender	(transmissor)
receiver	(receptor)
language	(linguagem do conteúdo)
ontology	(ontologia do conteúdo)
replay_to	(re-enviar a mensagem para)
replay_with	(rótulo da resposta)
force	(se *performative* negada)
content	(conteúdo) Objeto ContentMessage, seção 7.6.2

FIGURA 7.8 - Classe KQMLMessage

A Tabela 7.4 identifica as *performative* KQML que estão em fase de implementação, 'S' significa o remetente e 'R' o receptor. A lista completa de *performatives* KQML pode ser encontrada em [DAR 93].

TABELA 7.4 - *Performatives* utilizadas pelos agentes

Nome	**Resposta requerida**	**Significado**
Achieve		S deseja que R torne algo de verdadeiro em seu ambiente
Break		S deseja que R rompa um caminho estabelecido
DeletejLill		S deseja que R remove todas as sentenças que se encontram na BCV (Base de Conhecimento Virtual)
Delete_one		S deseja que R remove uma sentença da BCV
Error		S considera uma mensagem de R mal formulada
Forward		S deseja que R repasse a mensagem para outro receptor
lnsert		S pergunta à R para adicionar o conteúdo (: content) na BCV
Sorry		S não pode mais prover informações para o reply da *performative*
Tell	**X**	a sentença está no VKB do S

7.6.2 Conteúdo das Mensagens

Seguindo o formato de mensagem KQML, é necessário especificar o parâmetro *content*, para isso, foi implementada a classe ContentMessage, como mostra a Figura 7.9, 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.

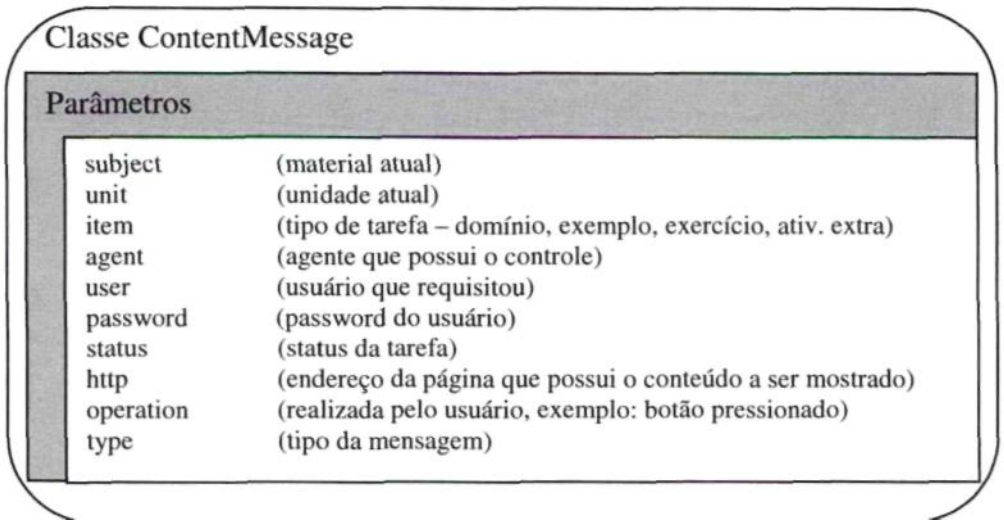


FIGURA 7.9 - Classe ContentMessage

Os tipos de conteúdo foram mapeados na Tabela 7.5. De acordo com o conteúdo, diferentes parâmetros são utilizados na mensagem.

TABELA 7.5 - Conteúdo das mensagens

Conteúdo das Mensagens		
Agente	Conteúdo de Entrada	Conteúdo de Saída
Gerenciador de Domínio Gerenciador de Exercícios Gerenciador de Exemplos	*(Mensagem Mostra Conteúdo)* **Recebendo Tarefa** de **Buscar Conteúdo** • usuário que solicitou; • matéria; • unidade; • item de conteúdo; • tipo de conteúdo. Esta mensagem resulta em uma busca na base de dados que retorna o conteúdo a ser mostrado (http://...) ao usuário, após esta busca, o agente envia uma mensagem ao usuário com o conteúdo a ser mostrado. Se o agente não encontrar o conteúdo na base de dados, este envia a mensagem de controle ao agente Gerenciador Modelo do Aluno, para que este decida qual o próximo passo	*(Resposta Mensagem Mostra Conteúdo)* *(Resposta Mensagem Próximo/A nterior)* **Enviando uma tarefa ao usuário** • usuário; • matéria; • unidade; • item de conteúdo • tipo de conteúdo • Conteúdo do item da Unidade (http://...) *(Mensagem Controle)* **Enviando fim de tarefa ao Agente Gerenciador Modelo do Aluno** • agente • usuário.

	Recebendo Requisição do Usuário - Mensagem do Agente Interface) • usuário que solicitou; • requisição de próximo conteúdo ou conteúdo anterior. Esta mensagem resulta em uma busca na base de dados que retorna **o** conteúdo a ser mostrado (http://...) ao usuário, após esta busca, o agente envia uma mensagem ao usuário com o conteúdo a ser mostrado. Se o agente não encontrar o conteúdo requisitado pelo usuário (próximo ou anterior) na base de dados, este envia a mensagem de controle ao agente Gerenciador Modelo do Aluno, para que este decida qual o próximo passo.	
Gerenciador do Aluno	***(Mensagem Controle)*** **Recebendo fim de Tarefa de um agente para um usuário** • agente, • usuário. O agente Gerenciador do Aluno, baseado na estratégia de ensino que estamos utilizando, irá decidir qual o próximo conteúdo a ser apresentado ao usuário. Tal decisão também leva em consideração de qual agente ele recebeu a mensagem. ***(Mensagem Login Usuário)*** **RECEBENDO LOGIN USUÁRIO** • usuário; • senha. Esta mensagem desencadeia os seguintes procedimentos: • o agente Gerenciador Modelo do Aluno procura na base de dados informações sobre o usuário; • se o usuário já estiver cadastrado, o agente envia uma mensagem de Mostra Conteúdo para o agente que possuía o controle da Lição anteriormente; • se o usuário não existir, ele é cadastrado na base de dados e o agente Gerenciador de Modelo do Aluno envia uma mensagem de Mostra Conteúdo para o agente Gerente do Domínio.	***(Mensagem Mostra Conteúdo)*** **Enviando Tarefa de Buscar Conteúdo** • usuário; • matéria; • unidade; • item de conteúdo; • tipo de conteúdo.

Interface	0 agente Interface recebe suas mensagens via o agente Gerenciador de Comunicação, tal agente foi implementado apenas para repassar as mensagens do agente Interface para os demais agentes e vice-versa. *(Resposta Mensagem Mostra Conteúdo)* *(Resposta Mensagem Próximo/Anterior)* **Recebendo uma tarefa para o usuário** • usuário; • matéria; • unidade; • item de conteúdo • tipo de conteúdo • Conteúdo do item da Unidade (http://...) Através do conteúdo desta mensagem, o agente Interface mostra ao usuário o material instrucional.	*(Mensagem Próximo/Anterior)* **Enviando Requisição do usuário)** • usuário; • requisição (próximo/anterior).

A Figura 7.10 ilustra uma mensagem enviada pelo Agente Interface ao agente Gerenciador Modelo do Aluno com o conteúdo Login do Usuário, para enviar uma mensagem é necessário criar dois objetos respectivamente herdados das classes KQMLMessage e ContentMessage.

KQMLMessage mensagem = new KQMLMessage();
ContentMessage conteúdo = new ContentMessage( );

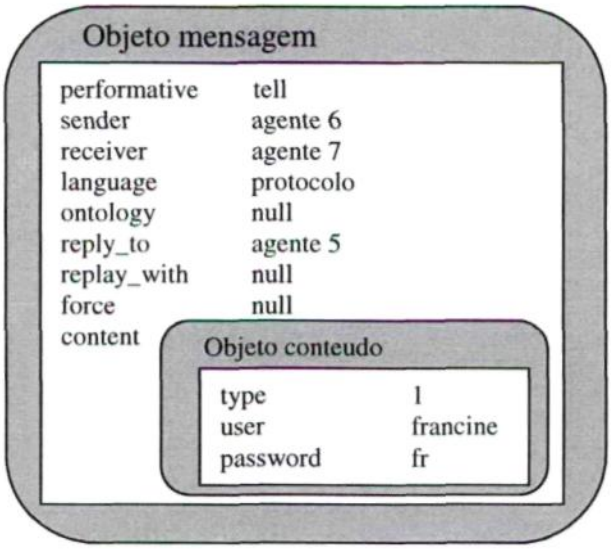


FIGURA 7.10 - Mensagem com conteúdo login do usuário

O objeto conteúdo será encapsulado pelo parâmetro *content* do objeto mensagem, que por sua vez será repassado por RMI ao agente especificado pelo parâmetro *receiver.*

A Figura 7.11, exemplifica o fluxo de mensagens, ou seja, os passos e conteúdos de mensagens, a partir do login de um usuário até a recuperação do conteúdo a ser apresentado.

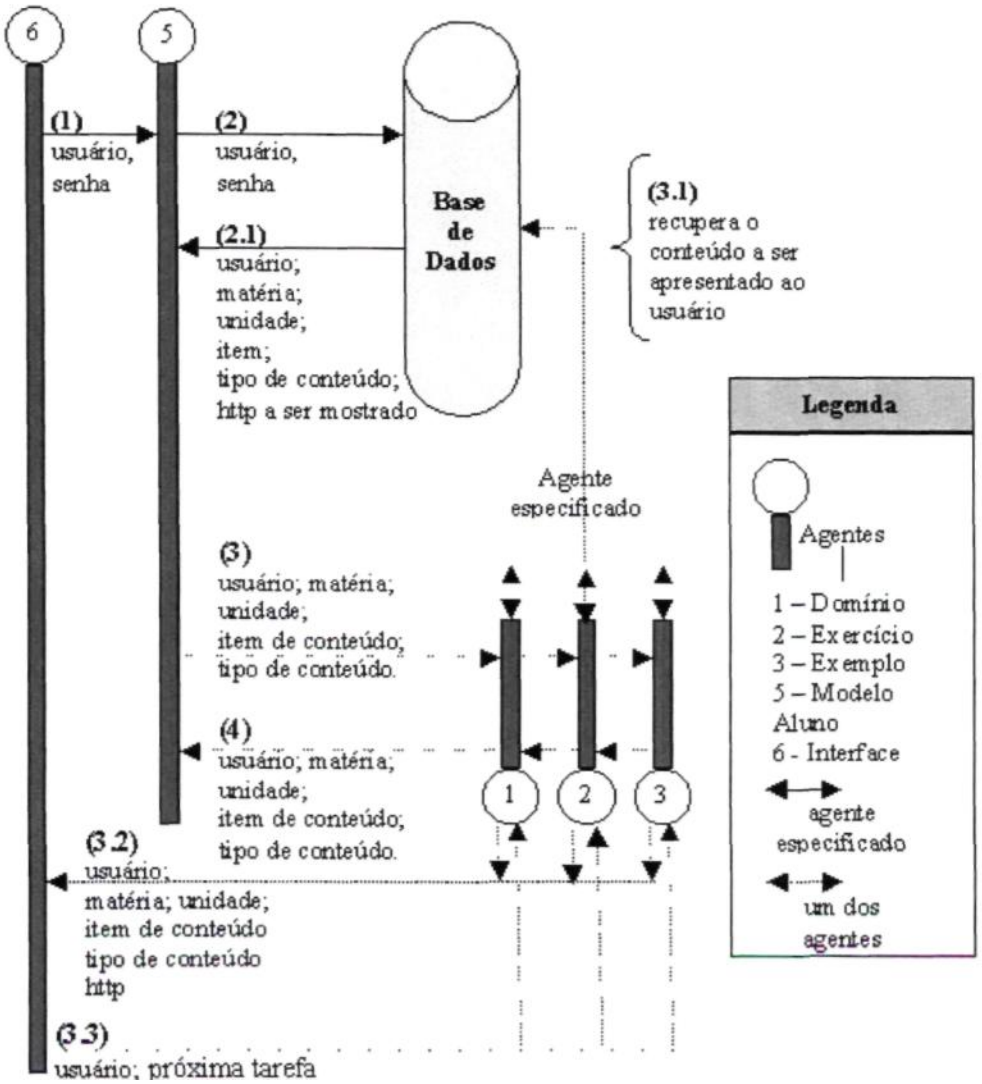


FIGURA 7.11- Fluxo de mensagens entre os agentes

As mensagens trocadas entre os agentes (1 ao 6) são especificadas a seguir:

- (1) Agente Interface envia Identificação do Usuário e sua Senha;
- (2) Agente Gerenciador do Modelo do Aluno busca validação da identificação do usuário na BC. Se este usuário não existir, o agente Gerenciador do Modelo do Aluno o cadastra na BC.
- Se o usuário já estava cadastrado na BC (2.1), é recuperado o modelo de aluno deste e o Agente Gerenciador do Aluno envia ao agente correspondente (3) o controle do processo. Este agente, por sua vez, recupera na BC o conteúdo que deve ser apresentado a este usuário (3.1) e o

envia ao usuário (3.2). Através do Agente Interface o usuário especifica o que deseja, como por exemplo, mais um exercício (3.3).

- Quando a tarefa que o agente se propunha se encerra ele avisa o Agente Gerenciador do Modelo do Aluno (4), para que este com base no desempenho da última tarefa possa decidir qual o próximo passo. Antes disso, porém, é atualizado o Modelo do Aluno na BC (5).

7.7 Regras de Comportamento

As regras de comportamento, definidas na seção 6.2.3, possuem uma estrutura que se decompõe em WHEN-EF-THEN. As condições das mensagens estão ligadas as mensagens KQML, descritas na seção 7.6. As regras existentes são cinco e funcionam da seguinte forma:

- **login_usuario:** esta regra acontece quando o agente Interface envia uma mensagem ao agente Gerenciador Modelo do Aluno, informando que um aluno se conectou ao sistema, como resultado desta regra tem-se o cadastramento do aluno, se tal aluno não foi cadastrado previamente. Esta regra desencadeia a regra mostra_conteúdo;
- **mostra_conteudo:** o agente Gerenciador Modelo do Aluno envia uma mensagem para o agente que receberá o controle, este por sua vez, tenta encontrar o conteúdo, se obtém sucesso envia-o ao agente Interface, se não o encontra desencadeia uma mensagem ao agente Gerenciador Modelo do Aluno para que este informe qual o próximo passo a seguir;
- próximo: tal regra é desencadeada quando o aluno pressiona o botão próximo da interface do Eletrotutor III. O agente Interface, então, envia uma mensagem ao agente que possui o controle, este por sua vez, tenta encontrar na BC o próximo conteúdo a ser mostrado, se obtém sucesso, envia tal conteúdo ao agente Interface, mas se não encontra, envia mensagem ao agente Gerenciador Modelo do Aluno desencadeando assim, outra regra de comportamento: controle;
- **anterior:** essa regra é similar a **próximo,** e é desencadeada quando o aluno pressiona o botão anterior da interface do Eletrotutor III. O agente Interface, então, envia uma mensagem ao agente que possui o controle, este por sua vez, tenta encontrar na BC o conteúdo anterior ao atual, se obtém sucesso, envia tal conteúdo ao agente Interface, mas se não encontra, envia mensagem ao agente Gerenciador Modelo do Aluno desencadeando assim, outra regra de comportamento: controle;
- **controle:** o agente Gerenciador do Modelo do Aluno recebe uma mensagem de um agente que não conseguiu recuperar o próximo conteúdo ou o conteúdo anterior ao atual. Para descobrir o conteúdo próximo ou anterior o agente deve saber qual o agente que está requisitando esta ajuda, já que, por exemplo, se acabaram as lições o próximo conteúdo a ser mostrado, é um exemplo, se este existir, senão um exercício.

Para que novas mensagens sejam incorporadas ao protótipo, é necessário a inclusão de novas regras de comportamento, já que, cada regra de comportamento contém as ações que os agente devem executar quando recebem determinada mensagem.

7.8 A Interface

A interface proposta no protótipo do Eletrotutor III é dividida em dois *frames:* um *frame* que contém o agente Interface *(applet java)* e um *frame* no qual o material instrucional, formado por *applets*, *javascript* e HTML, é apresentado, como ilustra a Figura 7.12.

Neste primeiro momento os botões Lição, Exemplo e Exercício não estão sendo utilizados, pois são os agentes que decidem o momento e a forma como o material instrucional é apresentado, estes botões serão utilizados no modo de execução Autônomo, descrito na seção 6.3.

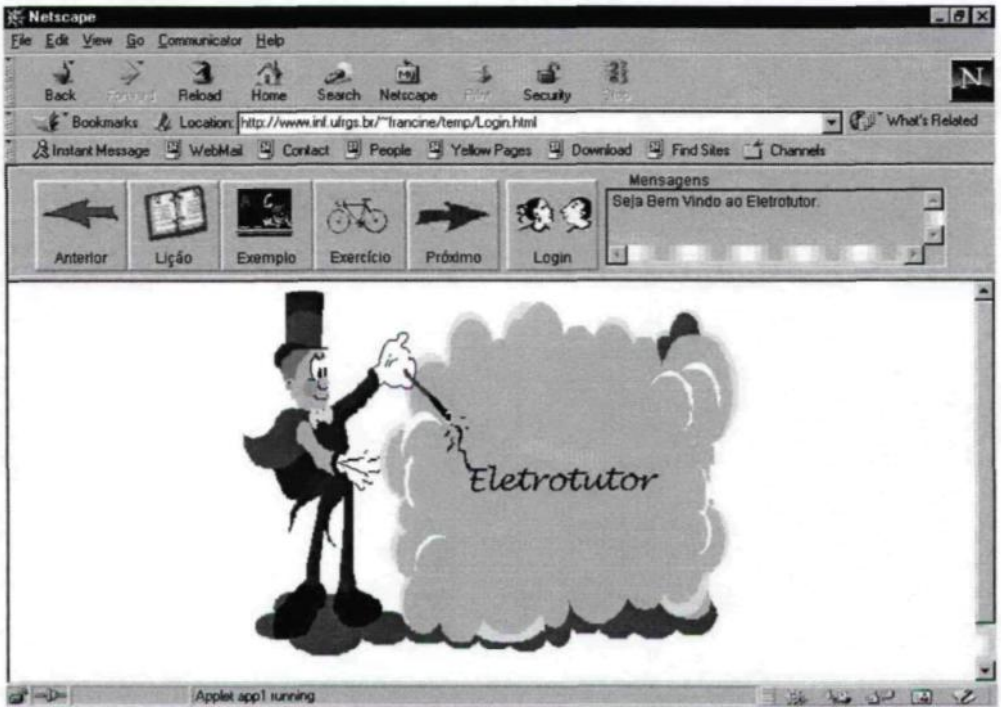


FIGURA 7.12 Interface Eletrotutor III

A Figura 7.9 representa a tela de entrada (Tela Principal) do Eletrotutor III, para se conectar ao Modo Tutorial, é necessário pressionar o botão Login, logo a seguir, o sistema abrirá uma janela de identificação, na qual o usuário deve preencher os dados requeridos, como ilustra a Figura 7.13. O usuário deve pressionar o botão Ok para confirmar seus dados e aguardar enquanto o sistema verifica a identificação, após tal verificação o sistema segue a ordem ilustrada na Figura 7.8.

FIGURA 7.13 - Janela de Login

Para ocorrer uma maior interação entre o aluno e os agentes foi elaborado um quadro de mensagens, no qual os agentes enviam mensagens aos alunos, como ilustrado na Figura 7.12, Figura 7.14 e Figura 7.15. Tais mensagens informam ao aluno, por exemplo, qual a lição, exemplo ou exercício que ele se encontra, assim como o andamento da correção dos exercícios.

A maior preocupação na composição desta interface foi a de conseguir implementar um ambiente agradável de fácil utilização. Assim como na segunda versão do Eletrotutor, cada vez que um exercício é respondido, o sistema retorna ao usuário a resposta correta e o resultado da correção, como ilustra a Figura 7.12.

FIGURA 7.14- Interface com conteúdo de Lições

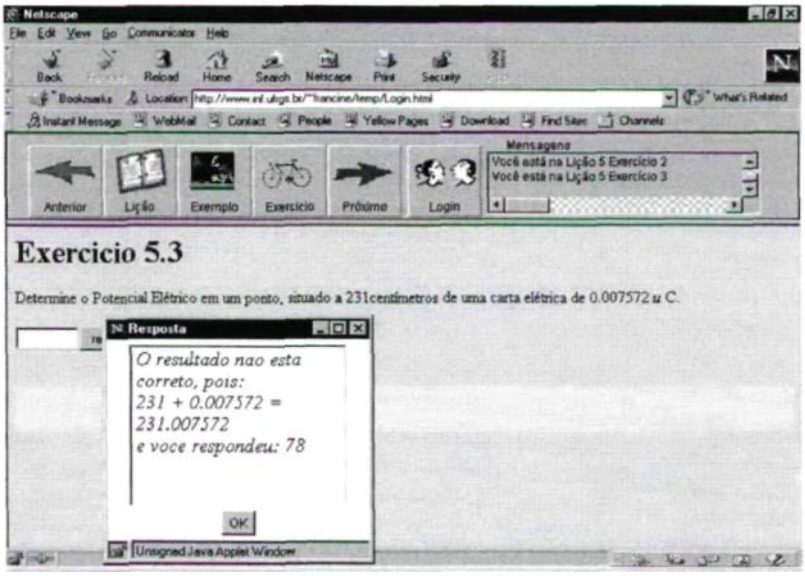


FIGURA 7.15- Interface com Exercício e Janela com o resultado da correção

7.9 Requisitos para executar o protótipo

Para executar o protótipo são necessários alguns requisitos, são eles:

- o *Java* para executar os agentes de forma distribuída;
- um Banco de Dados Relacionai (BDR) onde são armazenados os dados dos alunos, agentes e material instrucional;
- um *driver* de conexão do BDR com o *Java* (JDBC) e
- uma máquina com um *Web Server* instalado para executar os agentes Interface e Gerenciador de Comunicação.

As classes implementadas devem ser compiladas em uma ordem pré-estabelecida, esta ordem deve ser respeitada, conforme a Figura 7.1, algumas classes necessitam de métodos de outras. A Figura 7.16 ilustra a ordem correta de compilação, a qual pode ocorrer em paralelo, como no caso, por exemplo, das classes KqmlMessage e Agent.

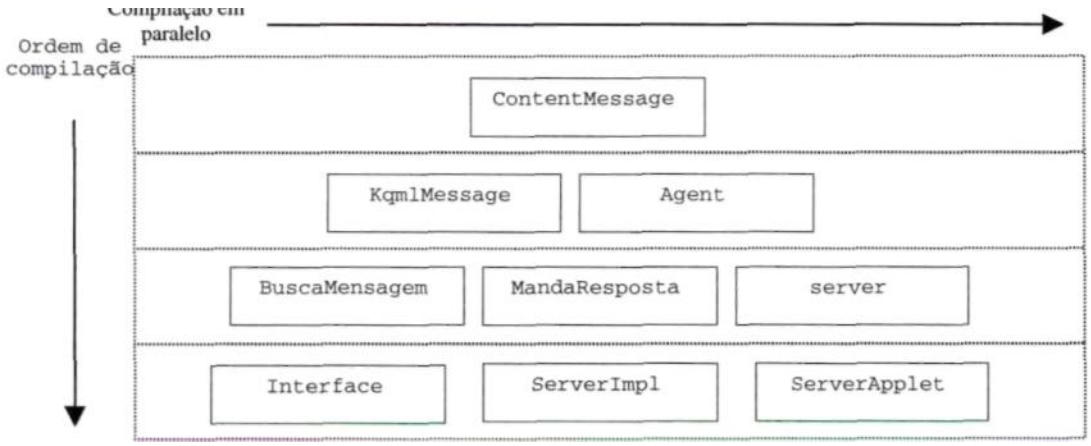


FIGURA 7.16 - Ordem de execução das classes do protótipo

8 Conclusões

A aplicação de novas abordagens de ensino-aprendizagem e estratégias pedagógicas têm sido possível graças ao uso, cada dia mais crescente, da Informática e de novas tecnologias de comunicação. A união da computação aos meios de comunicação facilita o acesso às informações, gerando assim novas perspectivas para o sistema educacional vigente.

Com o acesso à *Internet* facilitado pela WWW, tem-se um novo recurso tecnológico que, ao permitir a integração de texto, imagem, áudio e vídeo, disponíveis em qualquer parte do planeta, traz novas possibilidades à tecnologia da educação, no sentido da criação de materiais mais eficientes e da individualização do ensino [MOR 98].

Um aspecto importante a considerar é que o estudante é um indivíduo com características próprias, que devem ser respeitadas, merecendo portanto atenção ao ritmo de estudo individual. Deve-se levar em conta seu comportamento e os mecanismos que podem facilitar sua aprendizagem. Os alunos, geralmente, têm forte influência dos métodos presenciais tradicionais e, principalmente, são pouco educados a estudar a partir de seu próprio esforço individual. É fundamental que se oriente o estudante a estudar por conta própria, desenvolvendo habilidades de independência e iniciativa [NUN 96].

Portanto, para construir um ambiente educacional interativo, em vez de um mero hiper-livro eletrônico, é necessário prover um mecanismo de adaptação individual que, de forma ativa, leve o aluno através do hiper-espaço, tendo em consideração o *backgroung* de conhecimento dele, sua habilidade de compreensão, sua área de interesse, seus planos e intenções.

Este projeto tem caráter nitidamente multidisciplinar e busca unir alguns avanços obtidos em diversas áreas afins, buscando obter resultados no campo da pesquisa experimental aplicada. Esperamos com ele trazer algum avanço no sentido de aprimorar a eficácia dos ambientes computacionais voltado ao processo educacional, agregando conceitos e avanços de outras áreas através da implementação de um ambiente de aprendizado inteligente, utilizando uma arquitetura baseada em uma sociedade de agentes, na qual todos os elementos desta sociedade, humanos ou não, tenham capacidade de ensinar e aprender.

A terceira versão do Eletrotutor agrega aspectos das versões anteriores, como: a utilização de uma metodologia de ensino e o controle do aluno baseados no Eletrotutor I; uma arquitetura distribuída e uma interface gráfica, como no Eletrotutor II. Além disso, o Eletrotutor III incorpora uma arquitetura baseada em agentes artificiais e humanos; a partir dos elementos citados espera-se que o aluno assimile os conteúdos abordados pelo sistema.

O protótipo desenvolvido compõem um ambiente multiagente, na qual, os agentes interagem entre si, cada agente possui uma função específica e o objetivo principal deste conjunto de agentes cooperativos é a aprendizagem deste aluno. O material instrucional é composto por páginas HTML, Javascript e *applets* Java

executadas em um servidor WWW, que geram uma interface gráfica para a disponibilização de uma determinada disciplina na rede.

Com este trabalho obteve-se os seguintes resultados:

- proposta de uma sociedade de agentes inteligentes que possa ser aplicada para o projeto de ambientes de ensino em diversos domínios;
- proposta de uma arquitetura interna de agente compatível com a metodologia de ensino e com o modelo da sociedade de agentes estabelecida;
- proposta de um modelo de mensagens tocadas entre os agentes adequado a sociedade de agentes estabelecida;
- implementação de um protótipo de ambiente em **IAD** na área de Tutores Inteligentes;
- a arquitetura, na forma como foi projetada e implementada, pode ser utilizada em outros domínios através da implementação de páginas HTML e inserções na base de dados.
- implementação da sociedade dos agentes em uma arquitetura na qual os agentes estão fisicamente distribuídos, isto é, suportada por uma estrutura de rede.

Contribuições deste trabalho:

- os resultados deste trabalho contribuirão para um projeto em conjunto com a UFRGS, CRT e Unisinos - Projeto Tapejara e para outros projetos do Grupo de Inteligência Artificial (GIA) do Instituto de Informática;
- o protótipo implementado servirá para agregar outros projetos do GIA.

Espera-se com isto contribuir para que a Educação a Distância possa dar mais um passo para ser uma alternativa viável no processo educacional.

8.1 Eletrotutor III em relação aos outros trabalhos estudados

ADELE consiste em um agente pedagógico projetado para trabalhar na WWW. O conteúdo do curso que ADELE apresenta fica restrito a resolução de problemas na área da medicina. Como o ambiente no qual ADELE está inserido utiliza simulações complexas, foi necessário colocar o material de referência na máquina do estudante, no caso do Eletrotutor III, todo o material está na Web, sendo que o estudante só necessita de um Web *browser* para se conectar ao ambiente.

O ambiente na qual ADELE está inserido é bastante complexo e mostra as possibilidades existentes da utilização de recursos como a simulação e agentes pedagógicos animados. Levando em consideração os resultados positivos obtidos no estudo sobre o impacto afetivo da utilização de agentes pedagógicos animados em experiências de aprendizagem, a utilização de agentes pedagógicos animados poderia acrescentar ao ambiente Eletrotutor uma maior interatividade entre aluno e agentes.

O ambiente AMEA, assim como no Eletrotutor III, é distribuído e está sendo desenvolvido na linguagem Java, o domínio do conhecimento está armazenado em um banco de dados relacionai e os agentes colaboram na aprendizagem do aluno.

Apesar dos objetivos e características similares, na arquitetura desenvolvida no ambiente AMEA, após a seleção da estratégia de ensino para um determinado aluno, apenas um agente mostra o conteúdo ao aluno, não ocorrendo, assim, a interação e cooperação entre os agentes para mostrar o material instrucional, como ocorre no ambiente Eletrotutor III.

O Eletrotutor III contém a opção de exercícios e a correção destes em tempo real. Além disso, no ambiente AMEA não existe um protocolo especificando mensagens e troca de mensagens entre os agentes.

O ambiente AMEA, com base em um diálogo inicial com o aluno, traça o perfil deste e define qual a estratégia de ensino que deve ser utilizada. Esta análise poderia ser incorporada ao ambiente Eletrotutor, com isso, haveria um aprimoramento das estratégias de ensino, assim como, mais opções de material instrucional para o mesmo conteúdo.

O Virtua Cell foi referenciado no Eletrotutor III, pois é um exemplo de ambiente que utiliza VRML para que o aluno sinta-se imerso durante a interação com o sistema, o VRML pode ser utilizado como um dos recursos para o agente Gerenciador de Atividades Extras, não implementado neste versão. Na bibliografia estudada não foi encontrado dados suficientes sobre a arquitetura dos agentes do Virtua-Cell.

8.2 O Protótipo

Na fase atual, o protótipo possui todos os agentes implementados (com exceção do Agente Gerenciador de Atividades Extras).

O protocolo de comunicação entre os agente é baseado na linguagem KQML e as mensagens trocadas entre os agentes são enviadas por RMI.

A base de conhecimento está armazenada em um banco de dados relacionai, o Oracle versão 8.0.4.

A comunicação dos programas Java com o banco de dados foi realizada através do driver JDBC-Thin da Oracle.

As lições e exemplos são formados por páginas HTML, os exemplos possuem JavaScript e os exercícios são *applets* Java.

8.2.1 Limitações

As limitações observadas neste trabalho foram as seguintes:

- devido a existência de pouco material instrucional a estratégia de ensino é simples;
- os exemplos estão voltados para tipos numéricos de aplicação de fórmulas, outras alternativas devem ser implementadas;

- como o protótipo será utilizado na Web, existe uma forte dependência da performance desta.

8.2.2 Validação do protótipo

Com a finalidade de verificar a eficácia da abordagem apresentada será elaborada uma avaliação de pesquisa experimental, na qual os alunos serão divididos em dois grupos: o primeiro será submetido ao ensino através do tutor e o segundo submetido ao ensino do mesmo conteúdo através de aula expositiva. Após, os alunos realizarão testes de conhecimento, relativos à matéria, através dos quais serão feitas comparações de performance de cada um dos grupos em um pré-teste e um pós-teste. Além disto serão realizadas observações sistemáticas e entrevistas com os alunos submetidos ao uso do programa com o objetivo de levantar dados complementares sobre o uso do tutor.

8.3 Trabalhos Futuros

Como trabalhos futuros têm-se os seguintes pontos:

- elaboração de um diálogo inicial entre o tutor e o aluno, com o objetivo de traçar o perfil do aluno, além de descobrir o quê o aluno sabe do conteúdo a ser estudado, este deveria tentar enquadrá-lo em uma forma de aprendizagem, ou seja, qual a forma que este aluno gosta de aprender, por exemplos, lições com mais figuras, mais textos, animações, etc;
- com base em um diálogo inicial entre tutor e aluno, o material didático (domínio, exercícios e exemplos) deve possuir diferentes abordagens de aprendizagem para suprir as necessidades dos diferentes perfis de alunos;
- elaboração das atividades extras, as quais serão controladas pelo Agente Gerenciador de Atividades Extras, e formas de avaliação destas, se for necessário;
- estudo de um modelo mais sofisticado de avaliação do aluno, no qual o tutor percebesse o contexto dos erros cometidos pelo aluno, o tipo e a freqüência de erros cometidos, apresentação de lições alternativas, revisões de matemática e formas mais sofisticadas de avaliação, etc.
- testar a utilização de agentes pedagógicos animados, de acordo com estudos realizados por [LES 97] tais agentes criam um ambiente de maior interação com o aluno;
- a navegação dentro das páginas do Eletrotutor **III** deve ser estudada e ser simples e interessante;
- implementar uma forma de comunicação entre alunos e professores *(mail)*, a qual não precisa, necessariamente, estar acoplada ao Eletrotutor;
- realizar uma avaliação do sistema implementado em um ambiente real de instituição de ensino, comparando a sua performance com as versões anteriores do Eletrotutor;
- elaboração de um gerador de conteúdo (lições), exercícios e exemplos de fácil manipulação para compor mais material instrucional.

- permitir que o aluno consulte a sua performance durante a sessão Tutorial no Eletrotutor, assim como elaborar uma consulta para o professor conferir as performances dos alunos.

Anexo 1 - Interface Adaptativa

Em um trabalho do GIA [GOM 00], foi proposto um modelo heurístico para projetar interfaces adaptativas, um dos resultados deste trabalho foi a concepção de uma interface para o Eletrotutor no modo Autônomo. O modelo heurístico é dividido em heurísticas de projeto, heurísticas de avaliação e *guidelines.*

Os *guidelines* são guias que descrevem algumas regras que devem ser seguidas na composição de uma interface, levando em consideração que a construção de uma interface é um processo que depende da aplicação e dos tipos de usuários que irão utilizá-la. A pesquisara [GOM 00] sugeriu *guidelines* quanto aos usuários, aos estilos de interação, a entrada de dados, ao *feedback*, as ações necessárias, ao controle do usuário, aos elementos da hipermídia, as opções visuais disponíveis, a adaptabilidade, a consistência, aos erros e mensagens de erros e quanto as formas de ajuda.

O conjunto de heurísticas de projeto é dividido da seguinte forma:

- facilidade de usar: indicativo que a interface deve ser **eficiente** (minimizar o esforço do usuário para executar uma tarefa); **conveniente** (permitir fácil acesso a todas as operações); **auto-descritiva** (facilidade de lembrar ao usuário como uma tarefa é realizada); **prestativa** (a interface deve aconselhar, orientar, informar e conduzir o usuário durante a interação); **confortável** (a interface deve proporcionar uma diminuição da carga de trabalho perceptiva e cognitiva do usuário durante a interação);
- facilidade de aprendizagem: **consistente** (refere-se a forma na qual as escolhas na concepção da interface, códigos, denominações, formatos, entre outros, são conservados idênticos em contextos idênticos e diferentes em contextos diferentes); **obediente** (o usuário deve ter controle sobre o sistema); **códigos e denominações significativos** (adequação entre a informação e sua referência); **recursos hipermídia** (utilização de som, imagem, texto, etc);
- gestão e manipulação de erros: **complacente** (facilidade na recuperação de erros); **tolerante a falhas** (mecanismos que permitam evitar, reduzir e prevenir a ocorrência de erros); **mensagem de erros de qualidade** (refere-se a pertinência, legibilidade e exatidão da informação dada ao usuário a respeito do erro cometido e ações que este deve realizar);
- subjetiva satisfação e atração do usuário: uma interface deve ser **prestativa** e **satisfazer** o usuário (a interface deve fornecer ajuda quando requisitada ou quando observar que o usuário encontra-se em dificuldades); **natural** e **passiva** (a interface deve se comunicar com o usuário de maneira natural, não exigindo o conhecimento de terminologia não referente a tarefa); a interface deve fornecer *feedback* imediato e trabalhar com ações explícitas do usuário, isto significa que todas as ações do usuário devem receber uma resposta do sistema e esta deve ser rápida, fornecendo informação sobre a transação solicitada e o seu resultado; **clara** e **compatível:** o usuário não deve Ter dúvidas sobre a leitura da interface e isto é obtido através dos *guidelines* que dizem respeito às características léxicas das informações apresentadas na tela;

- Adaptabilidade: a interface deve ser **diversa** (suportar os diferentes tipos de usuários); **flexível** (a interface deve apresentar ao usuário meios que lhe permitam personalizar a interface);

As heurísticas de avaliação são utilizadas para examinar uma interface, são elas:

- Visibilidade do estado do sistema;
- Correspondência entre sistema e o mundo real;
- Liberdade e controle do usuário;
- Consistência e padrões;
- Prevenção de erros;
- Flexibilidade e eficiência de uso;
- Projeto estético;
- Ajuda na linguagem do usuário, diagnósticos e recuperação de erros;
- Ajuda e documentação;
- Revisão de *guidelines.*

A interface elaborada para o Eletrotutor é ilustrada nas figuras abaixo, nas quais as setas vermelhas indicam as heurísticas utilizadas. A Figura A.l(a) mostra a tela inicial do sistema, exemplificando as ações disponíveis ao usuário no Eletrotutor, tais ações são acessar lições, exemplos, ajuda, configurações, mostrar/retirar figuras dos botões e ocultar/exibir o campo de mensagens (interface eficiente).

Na Figura A. l(b) é ilustrado as opções que o usuário pode acessar em um dado momento, e que tais acessos podem ser realizados de formas diferentes, como por um menu ou através da barra de ferramentas (interface conveniente). O usuário é informado sobre seu posicionamento através das barras de navegação e através do campo de mensagens que está disponível na parte inferior de todas as telas (interface prestativa). Cada tarefa pode ser acessada por botões e foram disponibilizadas duas formas de atalho (botão direito do *mouse* e teclado). Ajudas e conselhos são oferecidos através do campo de mensagens e as características da interface podem ser alteradas pelo usuário através do botão configurar (interface confortável).

FIGURA A.l - Telas do Eletrotutor indicando as heurísticas eficiente, diversa, confortável, conveniente e prestativa.

A Figura A.2(a) mostra que a interface oferece um help estruturado e um campo de mensagens que descreve toda a ação disponível ao usuário (interface auto-descritiva). A interface permite que o usuário troque de lição, exemplos ou exercícios a qualquer momento, como ilustra a Figura A.2(b) (interface obediente). Para adequar a

informação apresentada com sua referência foram utilizados imagens e textos representativos (interface com códigos e denominações significativos). As configurações (por exemplo, de cor, fonte, imagem) realizadas pelo usuário são mantidas durante toda a sua navegação (interface consistente)

(a) (b)

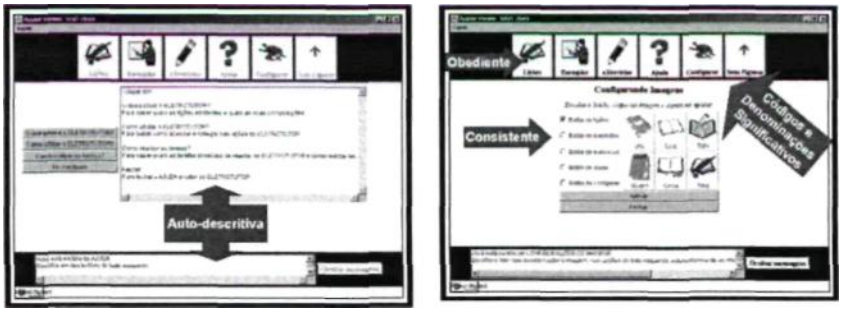


FIGURA A.2 - Telas do Eletrotutor indicando as heurísticas auto-descritiva, obediente, consistente, códigos e denominações significativas.

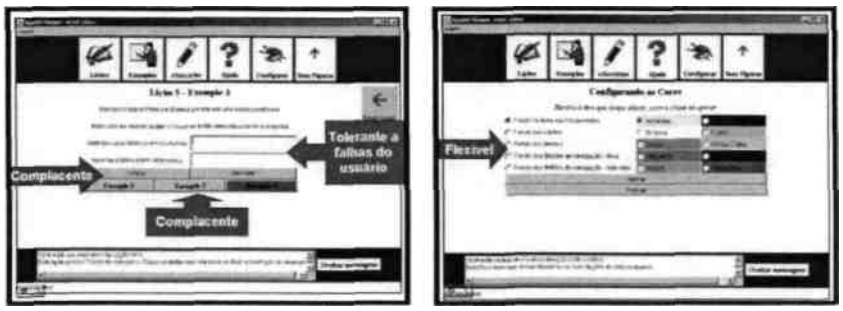
A Figura A. 3 representa a utilização de recursos hipermídia na interface do Eletrotutor. O som é utilizado para chamar a atenção do usuário sobre feitos positivos e negativos, o usuário pode desabilitar o som. As imagens são utilizadas para ilustrar o conteúdo das lições e botões de ferramentas. A fonte, cor e estilo do texto podem ser configurados.

FIGURA A.3 - Telas do Eletrotutor indicando a heurísticas de utilização de recursos hipermídia

A Figura A.4(a) ilustra que a interface possui uma barra de navegação, na qual o usuário pode voltar ao ponto anterior a qualquer momento. Nos exercícios existe um botão limpar para que o usuário possa modificar o valor digitado (interface complacente). As mensagens de erro dizem o que o usuário fez de errado, o que ele deveria ter feito e o que ele deve fazer para corrigi-lo (interface tolerante a falhas). Existe várias maneiras de executar uma tarefa, de acessar estas e a interface também permite que o usuário configure-a, escolhendo fonte, cor, tamanho, imagens, entre outros (interface flexível).

FIGURA A.4 - Telas do Eletrotutor indicando a heurísticas de utilização de recursos hipermídia

(a) (b)



Estão sendo estudadas maneiras de modificar a interface do Eletrotutor no modo Tutorial para os moldes propostos nesta seção.

Bibliografia

[ALF96] ALFERES, J.J.; PEREIRA, L.M. **Reasoning with Logic Programming.** Berlin: Springer, 1996. [BAR 82] BARR, A.; FEIGENBAUM, E. A. **The handbook of Artificial Intelligence.** Los Altos, Califórnia: [s.n.], 1982.

[BEP98] BEPPLER, F. Duarte. **Comunicação entre agentes em Kqml, aplicado a problema de logística:** Trabalho de Conclusão. Blumenau: Universidade regional de Blumenau, 1998.

[BIC 98] BICA, F.; SILVEIRA,R; VICCARI, R. M. Educação a Distância: do paradigma de Tutores Inteligentes a uma arquitetura Multiagentes. In: CONGRESSO INTERNACIONAL DE EDUCAÇÃO À DISTÂNCIA, 5., 1998. **Anais...** São Paulo: [s.n.], 1998. [BRA 84] BRATMAN, M. Two Faces of Intention. **The Philosophical Review, [S.I.],** v.93, n.3, p.275-405, July 1984.

[BRA 89] BRATMAN, M. What is Intention. In: COHEN, P., MORGAN, J. ; POLLACK, M. (Eds.). **Intentions in Communication. [S.l.]:** MIT Press, 1989.

[BRA 97] BRADSHAW, J. M. An introduction to software agents. In: BRADSHAW, **J.** M. (Ed.). **Software Agents.** Massachusetts: MIT Press 1997.

[CAS 98] CASTANHO, J.E.C. **Ambiente de apoio a cursos de Educação a Distância mediada por computador (EDMC).** Disponível por WWW em http://www.abed.org.br/artigos2/artigos/35/ambapoioedmc.html (1998).

[CHE 96] CHEONG, F. **Internet Agents:** Spiders, Wanderes, Brokers and Bots. Indianopolis, USA: New Riders Publishing, 1996.

[CLA 90] CLANCEY, W. J.; SOLOWAY, E. Artificial Intelligence and learning environments. **Artificial Intelligence,** Amsterdam, n.42, v.l, p.l - 6,Feb. 1990.

[COR 94] CORRÊA, M. **The architecture of dialogues among distributed cognitive agents.** Rio de Janeiro: UFRJ, 1994. [DAR 93] THE DARPA KNOWLEDGE SHARING INITIATIVE EXTERNAL INTERFACES WORKING GROUP. **Specification of the KQML Agente-Communication Language.** Baltimore: University of Maryland, 1993. [DAM 99] D'AMICO, C.B. **Aprendizagem estática e dinâmica em ambientes multiagentes de ensino-aprendizagem.** Porto Alegre: PPGC da UFRGS, 1999. Tese de Doutorado.

[FIN 96] FTNIN, Tim. UMBC KQML Web. **Lab for Advanced Information Technology.** 1996. Disponível por WWW em http://www.cs.umbc.edu/kqml

[GIR 99] GIRAFFA, L.M.M. **Uma arquitetura de tutor utilizando estados mentais.** Porto Alegre: PGCC/UFRGS, 1999. Tese de Doutorado.

[HAY 97] HAYES-ROTH, B. et ai. Acting in Character. In: TRAPPL,R; PETTA, P. **Creating Personalities for Synthetic Actors:** towards autonomous personality agents. (Eds.). Berlin: Springer-Verlag, 1997.

[HÜB 97] HÜBNER, Jomi Fred. **Linguagem para Especificação de Protocolos de Comunicação entre agentes.** Blumenau : FURB, 1997.

[JEP 97] JEPSON, Brian. **Programando banco de dados em Java.** São Paulo: Makron Books, 1997. p. 27 - 29.

[KEE91] KEEGAN, D. **Foundations of distance education.** 2.ed. Londres: Routledge, 1991.

[LES 97] LESTER, J.C. et ai. The persona effect: Affective impact of animated pedagogical agents. In: CHI, 1997. **Proceedings...** [S.l.]: ACM Press, 1997.
Disponível por WWW em http://wwwl.acm.org:82/sigs/sigchi/chi97/proceedings/paper/jl.htm

[MAT96] MATHOFF, J.; VAN HOE, R. A Multi-agent approach to interactive learning environments. In: EUROPEAN WORKSHOP ON MODELLING AUTONOMOUS AGENTS IN A MULTIAGENT WORLD, 6., 1996, Odinse, DK. **Proceedings...** Berlin: Springer-Verlag, 1996.

[MAY 96] MAYFIELD, James; LABROU, Yannis; FININ, Tim. Evaluation of KQML as an Agent Communication Language. In: WORKSHOP ON AGENT THEORIES, ARCHITECTURES & LANGUAGES, ECAI, 1995, Montreal, Canada. **Proceedings...** Berlin: Springer-Verlag, 1996.

[MOR 98] MORAES, S. G. **Uso da Internet como apoio para cursos presenciais.** Disponível por WWW em http://www.abed.org.br/artigos2/artigos/an06.htm (1998).

[MÓR97] MORA, Michael; LOPES, J. G.; VICCARI, R. Modelling dynamic aspects of intentions. In: PORTUGUESE CONFERENCE ON ARTIFICIAL LNTELLIGENCE, 8., 1997. **Proceddings...** Berlin: Springer-Verlag, 1997.

[MÓR 98] MORA, Michael C. et ai. BDI Models and Systems: Reducing the Gap. In: WORKSHOP ON AGENT THEORIES, ARCHITECTURES, AND LANGUAGES, 5., 1998. **Proceedings...** Berlin: Springer-Verlag, 1998.

[MÓR 99] MORA, Michael C. **Um modelo de agente executável.** Porto Alegre: PPGC/UFRGS, 1999. Tese de Doutorado.

[MOU96] MOUSSALLE, N.M; VICCARI, R.M.; CORRÊA, M. Intelligent Tutoring Systems Modelled Through the Mental States. In: SBIA, 1996. **Proceedings...** Berlin: Springer-Verlag, 1996. p. 221-230.

[MUL 96] MÜLLER, **J. P. The design of intelligent agents:** a layered approach. Heidelberg: Springer-Verlag, 1996. (Lecture Notes in Computer Science, v. 1177).

[NEW 88] NEWELL, A. Putting It Ali Together. In: KLHR, D.; K. (Eds.). **Complex Information Processing:** the Impact of Herbert Simon. Hillsdale, NJ: Lawrence Erlbaum, 1998.

[NIS 95] NISSEN, Mark. **Telecommunications and Distributed Processing.** [S.Ls.n.], 1995.

[NUN 97] NUNES, Ivônio Barros. **Noções de educação à distância.**
Disponível por WWW em http://www.ibase.org.br/~ined/ivoniol.html (1997).

[NWA96] NWANA, Hyacinth S. **Software Agents:** an Overview^ [S.l.]: Cambridge University Press, 1996.

[ORA 99] ORACLE. **OracIe8i JDBC Developer's Guide and Reference.** Disponível por WWW em http://www.oraclexom/java/documentatior^jdbc/html/overvw.htm (1999).

[PER 87] PERRY, W. **A short guide to distante education.** Cambridge: International Extension College, 1987.

[PER 98] PEREIRA, A. S.;. D'AMICO, C. B.; GEYER, C. F. R. Gerenciamento do conhecimento no Ambiente AME-A. In: CONGRESSO INTERNACIONAL DE EDUCAÇÃO À DISTÂNCIA, 5., 1998. **Anais...** São Paulo: [s.n.], 1998. [QUA 97] QUARESMA, J.; LOPES, J.G. A logic programming framework for the abduction of events in a dialoue system. In: WORKSHOP OF THE AUTOMATEF REASONING, AISB, 1997, London, England, **Proceedings...** [S.I.: s.n.], 1997.

[RAO 95] RAO, A.; GEORGEFF, M. BDI agents: From theory to practice. In: CONFERENCE ON MULTIAGENT SYSTEMS, ICMAS, 1., 1995. San Francisco. **Proceedings...** [S.I.: s.n.], 1995.

[RAO 96] RAO, A., GEORGEFF, M. Modeling and design of multi-agent systems. In: ECAI WORKSHOP (AT AL), 1996, Budapest, Hungary. **Proceedings...** Budapest: Springer-Verlag, 1996. [ROD95] RODRIGUES, A.; RARPER, J., Capitão. Implementing Intelligent Agents for Spatial Information. In: THE JOIN EUROPEAN CONFERENCE ON GEOGRAPHICAL INFORMATION, 1995. **Proceedings...** [SI: s.n.], 1995.

[RUS95] RUSSEL, S.; NORVIG, P. **Artificial Inteiligence a Modern Approach.** [S.l.]: Prentice-Hall, 1995. [SAN 96] SANTOS, Antônio Mello. Educação à Distância. **Tecnologia Educacional,** [S.L], v.24, n.128, jan./fev. 1996.

[SHI94] SCHNEIDER, Daniel. Teaching & Learning with the Web. In: INTERNATIONAL CONFERENCE ON THE WORLD - WIBE - WEB, 1., 1994, Gênova. **Proceedings...** Disponível por WWW em http://tecfa.unige.ch/edu-comp/edu-ws94/contrib/schneider/schneide.fm.html

[SHO 91] SHOHAM, Y. AGENT-O: A simple agent language and its interpreter. In: CONFERENCE ON ARTIFICIAL INTELLIGENCE, 9., 1991, Anaheim **Proceedings...** CA:MIT Press, 1991. p. 704-709.

[SHO 93] SHOHAM, Y. Agent-oriented programming. **Artificial Inteiligence,** [S.l.]v. 60, n.l,p. 51-92, 1993.

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

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

[SIL 96] SILVEIRA, R.A. **Eletrotutor II:** um Tutor na Web. Porto Alegre PPGC/UFRGS, 1996.

[SIL 98] SILVEIRA,R. A. **Ambientes Inteligentes de Ensino-Aprendizagem.** Porto Alegre: PPGC da UFRGS, 1998. Exame de Qualificação no. EQ-19.

[SIL 99] SILVEIRA, R. **Modelagem orientada a agentes aplicada a ambientes distribuídos de ensino.** Porto Alegre: PPGC da UFRGS, 1999.

[SUN 98] SUN MICROSYSTEMS. **Java Remote Method Invocation - distributed computing for Java.** Disponível por WWW em http://java.sun.com/marketing/collateral/javarmi.html (1998). [TH0 94] THOMAS, S. R. **The PLACA Agent Programming Language.** Berlin: Springer-Verlag, 1994. Lecture Notes in Artificial Intelligence. [THO 97] THOMAS, M. D. **Programando em Java para a Internet.** São Paulo: Makron Books, 1997.

[VIC 89] VICCARI, Rosa M. **Um tutor inteligente para a programação em lógica:** idealização, projeto e desenvolvimento. Coimbra: Universidade de Coimbra, 1989. Tese de Doutorado.

[WHI99] WHITE, Alan R.; McCLEAN, Phillip E.; SLATOR, Brian M. The Virtual Cell: An Interactive, Virtual Environment for Cell Biology. In: CONFERENCE ON EDUCATIONAL MEDIA, HYPERMEDIA and TELECOMMUNICATIONS, (ED-MEDIA 99), 1999, Seattle, WA. **Proceedings...** Disponível por WWW em http://www.ndsu.nodak.edu/wwwic/abstracts/ed-media.htm Ambiente Virtual Cell disponível por WWW em http://www.ndsu.nodak.edu/instruct/mcclean/vc/. [W00 94] WOOLDRIDGE, M. Jennings; NICHOLAS R. **Intelligent Agents: Theory and Practice.** Disponível por WWW em http://www.doc.mmu.ac.uk/STAFF/mike/ker95-html.html (1994). [W00 95] WOOLDRIDGE, M. Jennings; NICHOLAS R. **Intelligente Agents:** Theories, Architectures, and Languages. Heidelberg, Germany: Springer-Verlag, 1995. (Lecture Notes in Artificial Intelligence, v. 890).