UM SISTEMA ESPECIALISTA PARA AUXILIAR NA APRENDIZAGEM DA DISCIPLINA DE ARQUITETURA DE COMPUTADORES

Thiago Nascimento Pereira
Curso de Licenciatura em Informática – CEFET-MA
Av. Getúlio Vargas – Monte Castelo CEP 6503500 São Luís--MA
E-mail: thiagonasper@gmail.com

Omar Andres Carmona Cortes
Departamento Acadêmico de Informática – CEFET-MA
Av. Getúlio Vargas – Monte Castelo CEP 6503500 São Luís--MA
E-mail: omar@cefet-rna.br

RESUMO

O objetivo deste artigo é apresentar um sistema especialista (SE) para auxiliar no processo ensino-aprendizagem da disciplina de Arquitetura de Computadores do curso de Programação de Computadores do CEFET-MA. Basicamente, essa disciplina trata da montagem e manutenção de computadores. Nesse caso, o desafio da disciplina é capacitar o aluno a realizar manutenções seja qual for o problema apresentado por um computador. Em outras palavras, o desafio é realizar inferências dado um conjunto de fatores conhecidos e/ou desconhecidos.
Nesse contexto, possibilitar a maior quantidade possível de experiência dentro do laboratório é o objetivo do SE desenvolvido. A contribuição do SE esta no fato de que nem sempre é possível disponibilizar ao aluno uma quantidade suficiente de situações prática dada à falta intrínseca de material. Além disso, o sistema também possibilita ao aluno uma ferramenta para guiá-lo no processo de manutenção.
O SE foi desenvolvido em Java e fazendo chamadas ao Jess. O Jess foi utilizado para a construção da base de conhecimento, a qual foi desenvolvida abordando tanto o conhecimento sobre as diversas situações de problemas computacionais quanto as informações sobre os componentes do computador. Sendo assim, o sistema comporta-se como um tutor auxiliando o aluno na tarefa de manutenção. Além disso, o sistema apresenta informações sobre o funcionamento dos componentes envolvidos no problema identificado.

PALAVRAS-CHAVE: Sistema Especialista, Aprendizagem, Arquitetura de Computadores, Montagem e Manutenção.

1. INTRODUÇÃO

A informática, desde seu surgimento, tem tido o dom de encantar as pessoas com o poder e a eficiência de seus recursos. Com o desenvolvimento dos primeiros computadores surgiram idéias para sua utilização como meio de aprendizagem, através dos chamados Sistemas de Instrução Assistida por Computador (CAI).

A utilização de computadores na educação não é uma idéia nova, tanto que desde os anos cinqüenta já existiam projetos oriundos da área de educação (Barone, 2003). Esses primeiros projetos eram baseados no padrão tradicional de ensino, onde o conteúdo é previamente organizado e o aluno deve estudar uma seqüência pré-definida de assuntos, sendo que essa mesma estrutura foi inserida nos primeiros sistemas CAI. Com o passar do tempo foram identificadas algumas falhas pedagógicas nessa metodologia, especialmente em se tratando da escassez de recursos didáticos. A partir daí pesquisas foram iniciadas para fornecer novos recursos aos sistemas CAI. Nesse contexto, surgiu a idéia de se utilizar os recursos da Inteligência Artificial para melhorias dos sistemas educativos.

A inteligência artificial (IA) pode ser definida como a automação de atividades que associamos ao pensamento humano. Kurzwell (1990) define a IA como a arte de criar máquinas que executam funções que exigem inteligência quando utilizadas por pessoas. Dentre os muitos campos de pesquisa estudados na IA encontra-se os Sistemas Baseados em Conhecimento (SBC).

Os SBCs são utilizados basicamente na resolução de problemas que antes só poderiam ser resolvidos por seres humanos. Normalmente, uma boa indicação do uso desta tecnologia é a existência de um especialista humano com conhecimento capaz de solucionar problemas. Operacionalmente, o conhecimento que pode ser expresso por um SBC deve estar armazenado em uma base de conhecimentos (BC), sendo que a BC deve ser interpretada por um mecanismo independente de inferência, isto é, o mecanismo de inferência deve ser o mesmo para qualquer que seja a BC. Quando se aplica um conhecimento especializado na resolução de problemas difíceis do mundo real, diz-se que o SBC é um sistema especialista (SE) (Krishnamoorthy, 1996). Em outras palavras, SE é um SBC que soluciona um problema específico comumente solucionado por especialistas humanos. Sendo assim, um SE é um SBC cujo conhecimento restringe-se a um domínio específico (Rezende, 2003).

Nesse contexto vislumbrou-se a utilização de SE para detectar problemas em microcomputadores de acordo com informações previamente fornecidas. Devido à capacidade de um SE de explicar o raciocínio utilizado na solução de um problema, vislumbrou-se também a utilização do mesmo como auxiliar na disciplina de Arquitetura de Computadores do curso de Programação de Computadores do CEFET-MA, cujo principal objetivo é capacitar o aluno para realizar a montagem e manutenção em microcomputadores. Dessa forma, procurou-se associar o uso de do SE com a disciplina para que o aluno possa ter o máximo de experiências possíveis proporcionando ao aluno uma maior diversidade de cenários.

Para cumprir seu objetivo este artigo esta dividido da seguinte maneira: a Seção 2 faz uma breve introdução sobre a arquitetura de um SE; a Seção 3 apresenta como as tecnologias utilizadas são integradas; a Seção 4 mostra a arquitetura do sistema implementado; a Seção 5 discute alguns aspectos identificados no desenvolvimento do SE; finalmente, a Seção 6 apresenta as conclusões deste trabalho.

2. SISTEMAS ESPECIALISTAS

A arquitetura de um SE é formada pelos componentes exibidos na Figura 1. O mecanismo de Inferência é a parte do sistema responsável por processar as regras e fatos contidos na base de conhecimento. A base de conhecimento é a parte do sistema que contém todas as regras e fatos que modelam o conhecimento do especialista humano. Em outras palavras é o núcleo do sistema.

A máquina de inferência deve ser totalmente independente da base de conhecimento, ou seja, a mesma máquina de inferência deve ser capaz de utilizar diferentes bases de conhecimento de diferentes especialidades. Para enriquecer a base de conhecimento, faz-se uso de um sistema de aquisição de conhecimento que é projetado para transmitir o conhecimento do especialista humano para a base de conhecimento. Normalmente a passagem do conhecimento é feita de forma manual utilizando-se alguma forma de representação de conhecimento. Entretanto, pesquisas tentam encontrar formas para que essa aquisição possa também ser feita de forma automática.

Com toda estrutura básica funcionando, o SE precisa apenas de uma interface com o usuário final para que este possa fornecer ao sistema alguns fatos que são necessários para a inferência da base de conhecimento. Geralmente esta aquisição de fatos é feita através de uma entrevista com o usuário.

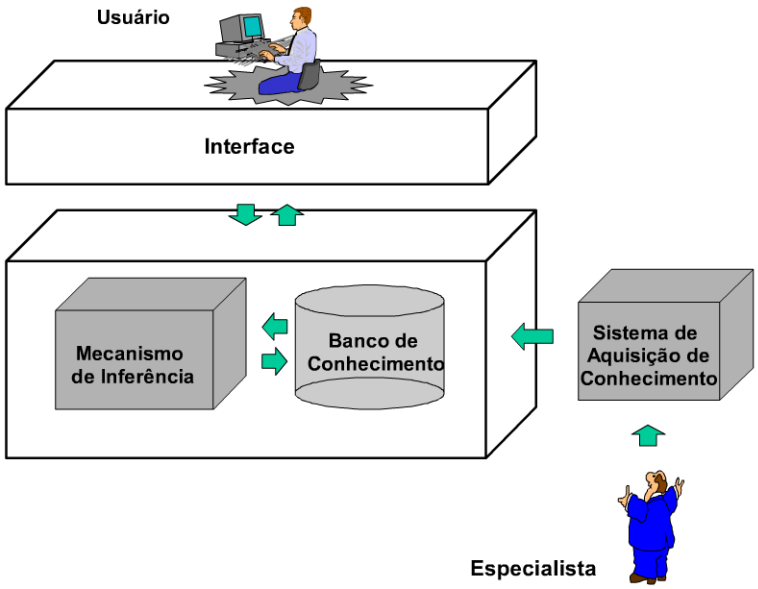

Figura 1 – Arquitetura de um sistema especialista.

3. Integração das Tecnologias Utilizadas

3.1 Java

A tecnologia Java (Sun, 2006) foi escolhida para desenvolvimento deste SE devido à grande quantidade de recursos e vantagens que a mesma disponibiliza. Dentre alguns destes recursos pode-se citar a portabilidade entre diferentes plataformas. Dessa maneira, o SE poderá ser executado em qualquer Sistema Operacional que disponibilize uma máquina virtual Java.

Uma grande vantagem do Java é que, por ser uma linguagem orientada a objetos, permite que a aplicação possa ser dividida em camadas especializadas, através de um padrão de projeto chamado de MVC (*Model View Control* - Modelo Visualização Controle). Sendo assim, o sistema fica extensível de forma que cada camada possa evoluir sem causar danos ao sistema como um todo.

A adoção do padrão de projeto MVC traz uma importante vantagem para o SE, pois tendo a BC em uma camada independente, esta poderá ser estendida com novas regras de forma totalmente independente, ou seja, sem causar efeitos colaterais na aplicação. Torna possível também a adição de recursos áudio-visuais ao sistema alterando a camada de visualização, o que tornaria o SE bem mais atrativo para o aluno e facilitaria bastante o aprendizado.

Dessa forma, percebe-se que com a utilização da tecnologia Java o sistema fica bem estruturado, robusto e extensível.

3.2 *Java Web Start*

Visto que o laboratório de informática do CEFET-MA conta com uma rede de computadores bem estruturada. Optou-se pela utilização de um recurso do Java para *deployment* de aplicações via Web denominado *Java Web Start*. Com o *Java Web Start* é possível executar uma aplicação com um simples clique em uma página Web, ou mesmo executar o sistema diretamente em um arquivo com o formato jnlp. JNLP (*Java Network Launching Protocol* – Protocolo de Inicialização de Rede Java) é uma tecnologia que define o formato padrão do arquivo e as instruções de execução da aplicação.

Dessa forma, o usuário pode baixar e executar aplicações sem passar por procedimentos complicados de instalação. Caso a aplicação não esteja presente no computador do cliente, o Java Web Start automaticamente baixa do servidor

todos os arquivos necessários para a execução da aplicação. Caso a máquina do cliente já esteja com todos os arquivos da aplicação baixados, estes serão atualizados sempre que houver uma versão mais nova no servidor. Esta atualização ocorre somente quando a aplicação é executada. Assim é possível garantir que a máquina cliente esteja sempre com uma versão atualizada do sistema.

3.3 Jess

O Jess (JESS, 2006) é um Shell para sistemas especialistas totalmente escrito em Java. Foi desenvolvido pelo *Sandia National Laboratories*. A criação do Jess se deu a partir de um Shell já existente para a linguagem de programação C/C++, o CLIPS. O CLIPS foi desenvolvido pelo *Software Technology Branch* (STB).

O Jess possui a maior parte das funcionalidades do CLIPS, sendo um dos motivos que o tornou uma das ferramentas mais populares para construção de sistemas especialistas. Ambos fazem uso de uma linguagem que, à semelhança do LISP, emprega notação polonesa, com operador seguido de operandos. Apesar de funcionar sob uma Máquina Virtual Java, o Jess, em certos problemas consegue apresentar melhor desempenho que o próprio CLIPS. Isso ocorre pela adição de otimizações ao famoso algoritmo de resolução de conflitos RETE.

O Jess conta ainda com um ambiente de desenvolvimento com interface gráfica, que possibilita ao usuário desenvolver sistemas, *applets* e verificar a eficiência da rede de regras compilada. Possui ainda capacidade de encadeamento tanto para trás quanto para frente. Além disso, muitas outras facilidades foram implementadas pelos usuários e adicionadas ao pacote, como por exemplo, a manipulação de bancos de dados, entre outras.

3.4 Integração Jess x Java

A integração entre o Jess e o Java pode ser feita de três maneiras:

- Com todo o sistema escrito com scripts da linguagem Jess em um arquivo com a extensão CLP. Nesse tipo de desenvolvimento é possível fazer chamadas a classes, métodos e qualquer recurso do ambiente Java. Dessa forma é possível criar toda a interface gráfica do sistema dentro do arquivo de script do Jess que contem a base de conhecimento.

- Com o sistema desenvolvido em Java e a base de conhecimento em script Jess. A partir de uma classe é possível carregar e executar um script jess que contem a base de conhecimento. Dessa forma é possível armazenar o script Jess em qualquer meio de persistência, como banco de dados, sendo que o padrão do Jess é o arquivo com extensão CLP.

- O desenvolvimento pode ser feito completamente em Java, isto é, tanto o sistema quanto a base de conhecimento são implementados em Java. O desenvolvimento da base de conhecimento se dá através da API Java disponível no pacote do Jess. Dessa maneira as regras e os fatos são criadas e executadas através de classes e métodos.

Avaliadas as três opções, optou-se pela utilização da 2ª forma para o desenvolvimento deste projeto. As opções 1 e 3 tornam-se inviáveis por motivos similares. Na opção 1 tem-se uma grande complexidade para o desenvolvimento da interface gráfica, visto que a sintaxe da linguagem Jess não é amigável para tal. Na opção três a complexidade está no desenvolvimento da base de conhecimento, pois a criação das regras e fatos através de classes Java tornam a base de conhecimento muito rígida e de difícil extensão.

A opção 2 atende de forma satisfatória o desenvolvimento de todo o sistema, pois separa-se a base de conhecimento do restante da aplicação, porém permitindo a comunicação entre ambos. A comunicação entre a aplicação e a base de conhecimento se dá de forma trivial, pois o Jess disponibiliza uma API que permite que o ambiente Java realize chamadas ao ambiente do Jess e vice-versa.

Aliado a essas características, o Jess permite o uso de funções predefinidas e também a definição de funções próprias dentro do seu ambiente. Outro ponto importante deste ambiente é a possibilidade de importação de funções definidas dentro das classes Java. O acesso do ambiente Jess em um programa Java se da através da classe *jess.Rete*. A partir desta classe é possível executar qualquer comando no ambiente Jess como mostra o código da Figura 2.

```
public class BaseConhecimento extends Thread{

	private Rete engine = new Rete();


	@Override
	public void run() {
		engine.executeCommand("(batch bc/diagmicro.clp)");
		super.run();
	}

	public BaseConhecimento() throws JessException {
		engine.store("maquina", this);
	}

	public void fazerPergunta(String id,
				String pergunta,
				String [] respostasValidas)	throws
			JessException{
		idPerguntaAtual = id;
		Controlador.getInstance()
			.enviarPergunta(pergunta, respostasValidas);
	}

	public void assertResposta(String resposta)
					throws JessException{

		Fact sintoma = new Fact("resposta-usuario"
					, engine);
		sintoma.set(new Value(resposta, RU.ATOM), 0);
		engine.assertFact(sintoma);
	}
```

Figura 2 - Comunicação entre o ambiente Java e o ambiente Jess.

O código mostrado na Figura 3 exibe: a comunicação do ambiente Jess com o ambiente Java e a criação de funções próprias dentro do ambiente Jess através da palavra-chave *deffunction*. Dentro da função criada, ocorre uma chamada a um objeto através da função *fetch*, que permite acessar qualquer objeto armazenado na memória do Jess.

```
(defmodule ENTREVISTA)

(deffunction ENTREVISTA::perguntar
                          (?id ?texto $?respostas-validas)
    (bind ?maquina (fetch maquina))
    (?maquina fazerPergunta ?id ?texto ?respostas-validas)
)

(defrule ENTREVISTA::realiza-pergunta
    (declare (auto-focus TRUE))
    (MAIN::pergunta-atual ?id)
    (not (MAIN::resposta (id ?id)))
    (MAIN::pergunta (id ?id) (texto ?texto)
                    (respostas-validas $?respostas-validas))
     =>
    (perguntar ?id ?texto ?respostas-validas)
    (engine)  waitForActivations )
    (halt)
)

(defrule ENTREVISTA::verifica-resposta
    (declare (auto-focus TRUE))
    ?pergunta-atual <- (MAIN::pergunta-atual ?id)
    (not (MAIN::resposta (id ?id)))
       (ENTREVISTA::resposta-usuario ?valor)
       =>
    (assert (MAIN::resposta (id ?id) (valor ?valor)))
    (retract ?pergunta-atual)
    (halt)
)
```

Figura 3 - Chamada de funções Java dentro do ambiente Jess.

4. O SE PARA AUXÍLIO NA DISCIPLINA DE ARQUITETURA DE COMPUTADORES

A interface com o usuário foi desenvolvida para o ambiente *desktop*. Para isso foi utilizada a API padrão do Java para desenvolvimento de interface gráfica, o Swing. A escolha do ambiente *desktop* se deu pelo fato da interface gráfica considerada possuir uma riqueza de componentes disponíveis na própria API, sendo possível a futura adição de recursos áudio visuais, o que pode tornar o sistema bem mais atrativo para os alunos.

A Figura 4 apresenta a arquitetura genérica do sistema utilizando o padrão MVC, onde o usuário utiliza a interface gráfica para interagir com o controlador. O controlador é o responsável por fazer a comunicação com o SE propriamente dito. A memória de trabalho é um espaço de memória onde o Jess armazena todos os fatos durante a execução do sistema. As regras que satisfazem a condição do seu lado esquerdo (antecedente) têm o seu lado direito executado (conseqüente) no caso do encadeamento para frente; e o contrário no encadeamento para trás. A execução o lado direito de uma regra implica na adição de novos fatos na memória de trabalho, que podem vir a ativar e executar novas regras de forma progressiva até que uma conclusão da situação seja encontrada.

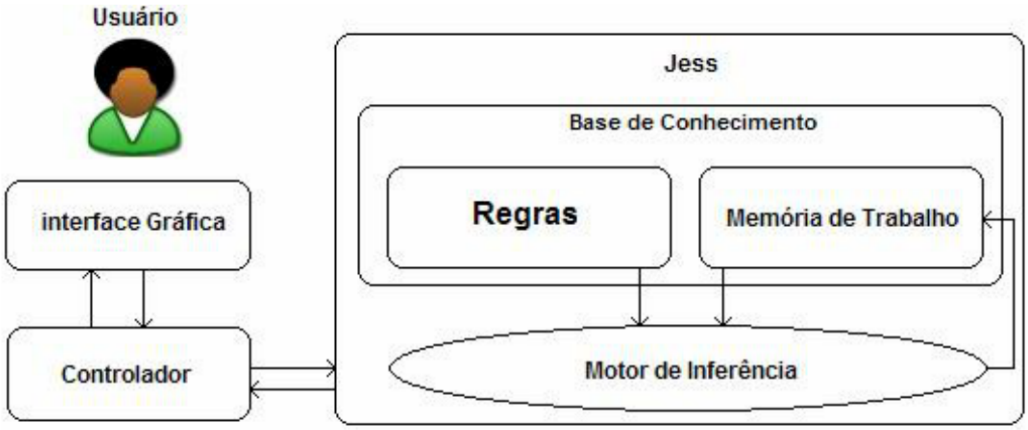

Figura 4. Arquitetura do SE com o padrão MVC.

A Figura 5 mostra o diagrama de classes do SE. Fazendo associações com a Figura 3, a classe *Mediador* e suas classes filhas fazem o papel da interface com o usuário; a classe *Controlador* faz a interação da aplicação com o Jess. A base de conhecimento é montada a partir de um arquivo CLP; e a classe RETE é a responsável por fazer a inferência, ou seja, unificar os padrões nas regras com a finalidade de determinar quais regras são satisfeitas, sendo que como já foi citado, pode-se utilizar tanto o encadeamento para frente quanto o encadeamento para trás.

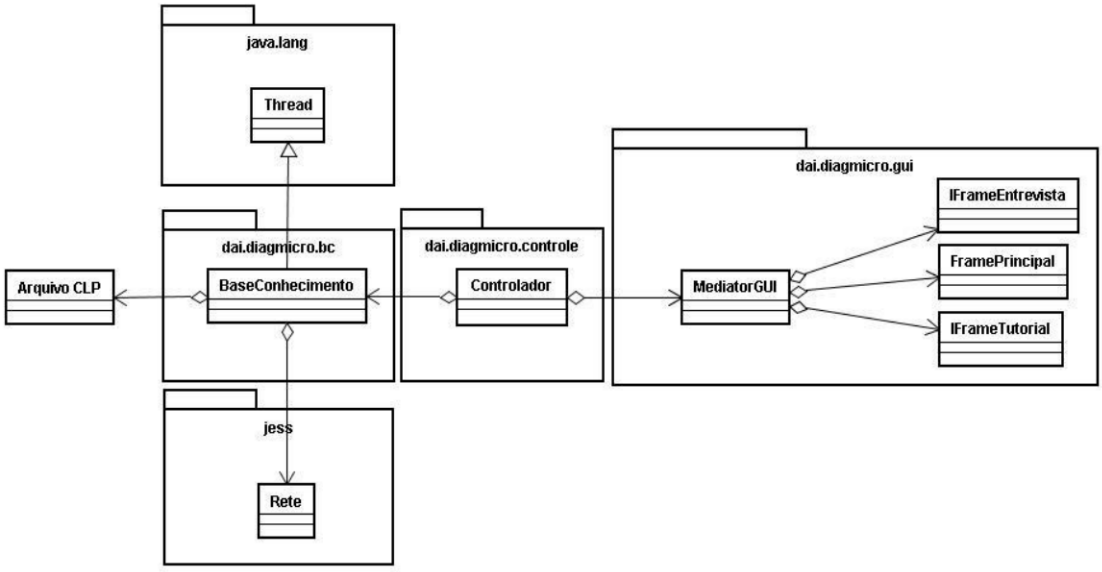

Figura 5 - Diagrama de classes.

Como já mencionado, a BC é o núcleo do sistema especialista. É nela que está toda a gama de informações a respeito dos sintomas de um determinado problema e toda a lógica de raciocínio realizado pelo SE. Um trecho da base de conhecimento utilizada na resolução dos problemas considerados neste trabalho é apresentado na Figura 6.

Para que o SE possa encontrar uma solução para um problema, algumas informações precisam ser fornecidas pelo usuário, o que deve ser feito através da interface gráfica. Estas informações são consideradas no SE como fatos, os quais são utilizados para a verificação das regras.

Durante a entrevista com o aluno, o sistema realiza chamadas ao Jess para saber qual a próxima informação necessária para confirmar alguma hipótese, então uma pergunta é enviada para o aluno. Ao coletar todos os fatos necessários e confirmar uma hipótese, o sistema retorna um resultado para a camada controladora informando quais componentes estão causando problemas com o micro, quais as soluções para o problema e um conteúdo informativo a respeito dos componentes envolvidos. Finalmente a camada controladora gera um tutorial e exibe para o aluno através da interface gráfica.

Para tornar a exibição da solução mais rica em informação, procurou-se armazenar na BC informações a respeito dos componentes do microcomputador. Dessa forma o sistema pode explicar a solução para o aluno e exibir informações que auxiliem no aprendizado dos conceitos de cada componente envolvido na situação apresentada.

O processo de interação com o aluno se dá em dois passos. No primeiro o sistema realiza uma série de perguntas. As respostas vão se transformando em fatos até que uma conclusão possa ser feita pela máquina de inferência. A Figura 7 apresenta um exemplo de interação e a inferência realizada pelo SE.

Na figura 8 o sistema exibe o resultado da inferência, onde constam informações sobre os componentes envolvidos no problema e os passos para que o aluno possa realizar a manutenção.

```
(deftemplate MAIN::pergunta
       (slot id)
       (slot texto)
       (multislot respostas-validas)
)
(deftemplate MAIN::resposta
       (slot id)
    (slot valor)
)
(do-backward-chaining resposta)

(deffacts MAIN::base-de-perguntas
    (MAIN::pergunta (id video) (texto "O computador exibe vídeo ?")
(respostas-validas sim nao))
    (MAIN::pergunta (id bip) (texto "O computador emite um bipe longo
repetidamente ?") (respostas-validas sim nao))
    (MAIN::pergunta (id led_power) (texto "O led verde do gabinete está acesso
?") (respostas-validas sim nao))
)

(defmodule HIPOTESES)

(defrule HIPOTESES::memoria
      (declare (auto-focus TRUE))
       (MAIN::resposta (id video) (valor nao))
       (MAIN::resposta (id bip) (valor sim))
       =>
       (assert problema-memoria)
       (halt)
)

(defrule HIPOTESES::fonte
    (declare (auto-focus TRUE))
       (MAIN::resposta (id video) (valor nao))
                    (MAIN::resposta (id bip) (valor nao))
       (MAIN::resposta (id led_power) (valor nao))
       =>
       (assert problema-fonte)
    (halt)
)

(defmodule TRIGGER)

(defrule TRIGGER::define-pergunta-atual
    (declare (auto-focus TRUE))
       (MAIN::need-resposta (id ?id))
       (not (MAIN::resposta (id ?id)))
    (not (MAIN::pergunta-atual ?))
       =>
    (assert (pergunta-atual ?id))
    (return)
)
```

Figura 6 – Trecho da base de conhecimento.

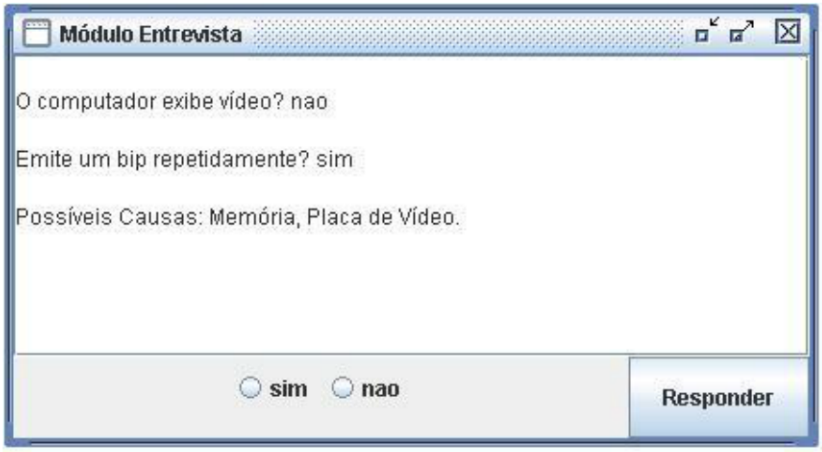

Figura 7 – Perguntas efetuadas ao aluno

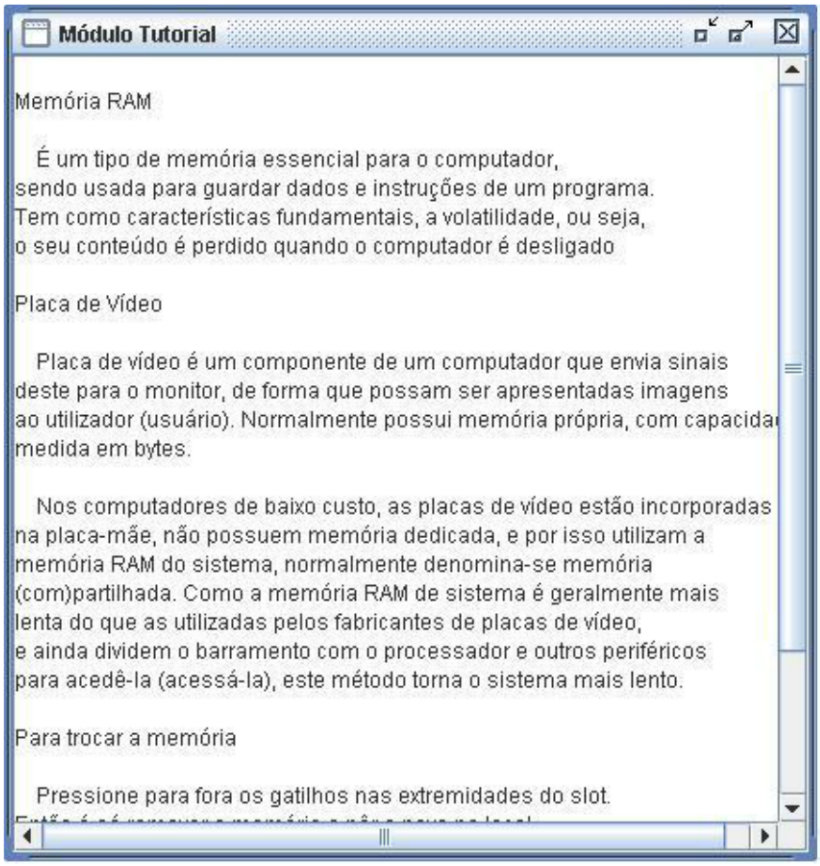

Figura 8 – Interface gráfica para exibição do resultado

5. DISCUSSÃO

O Jess utiliza o algoritmo RETE de inferência, o qual é considerado muito eficiente e empregado por linguagens baseadas em regras, tais como CLIPS, ART, OPS5, OPS83. O objetivo deste algoritmo é a unificação dos padrões nas regras, com a finalidade de determinar quais regras são satisfeitas.

Uma vantagem do algoritmo RETE é que esse algoritmo usa a similaridade estrutural presente nas regras. Esta similaridade refere-se ao fato de que regras podem ter padrões ou grupos de padrões similares. O RETE tira vantagem desta característica colocando os componentes comuns agrupados para que os mesmos não sejam considerados mais de uma vez, aumentando assim a eficiência.

A desvantagem do algoritmo RETE é o uso intensivo de memória: o uso de memória para simplesmente comparar todos os fatos com todas as regras não é significativo embora o número de iterações possa ser elevado. Já a quantidade de memória necessária para armazenar todo o ciclo de unificação e as alterações no conjunto de fatos é considerável.

De forma grosseira, pode-se comparar as regras com encadeamento para frente como várias sentenças de *if...then...else* em linguagens de programação. Dessa forma a parte *if* corresponde ao lado esquerdo de uma regra e o *then* ao lado direito (Figura 9). Porém, no Jess a execução das regras não é feita de forma estruturada como em uma linguagem de programação, mas em uma ordem determinada pelo sistema de execução do Jess, que utiliza o algoritmo RETE. Entretanto, esse tipo de encadeamento não é muito utilizado em SE.

```
(defrule HIPOTESES::memoria
       ;---------------------------Lado Esquerdo ou LHS
       (declare (auto-focus TRUE))
       (MAIN::resposta (id video) (valor nao))
       (MAIN::resposta (id bip) (valor sim))
       =>
       ;---------------------------Lado Direto ou RHS
       (assert problema-memoria)
       (halt)
)
```

Figura 9 – Exemplo de regra com encademanto para frente

O encadeamento para trás é o mais empregado no desenvolvimento de bases de conhecimento. Ele destaca-se por ser mais intuitivo para o desenvolvedor de SE. Nesse mecanismo, a máquina de inferência tenta executar o lado direito de alguma regra que possua o fato que está faltando no lado esquerdo de uma outra regra que está parcialmente satisfeita. Dessa forma, o Jess tenta completar o lado esquerdo da regra que está mais próxima de ser satisfeita por completo, podendo assim aumentar o desempenho do sistema no processo de inferência. Por esse motivo, este tipo de encadeamento foi adotado no desenvolvimento da base de conhecimento considerada neste trabalho. Para utilização do encadeamento para trás no Jess, é necessário fazer uma chamada a uma função específica que torna isto possível: *do-backward-chaining*.

6. CONCLUSÕES

Este trabalho apresentou o desenvolvimento de um SE para auxílio no processo de ensino-aprendizagem da disciplina de Arquitetura de Computadores do curso de Programação de Computadores do CEFET-MA.

O Sistema Especialista foi desenvolvido visando proporcionar ao aluno o máximo possível de experiência dentro do laboratório. A contribuição do SE está no fato de que nem sempre é possível disponibilizar ao aluno uma quantidade suficiente de situações práticas, dada à falta intrínseca de material. Além disso, o sistema serve como uma ferramenta para guiar o aluno no processo de manutenção.

Apesar de contribuir para o ensino-aprendizagem da disciplina de Arquitetura de Computadores, o SE desenvolvido neste projeto não chega a ser um Sistema Tutor Inteligente, pois não utiliza a IA para o processo de tutoria, mas somente no processo de identificação de problemas com microcomputadores. Porém é viável a evolução deste

sistema para um STI, visto que isto tornaria o processo de aprendizagem da disciplina de arquitetura de computadores mais eficiente e interessante para o aluno.

REFERÊNCIAS

BARONE, D., **Sociedades Artificiais: A Nova Fronteira da Inteligência nas Máquinas**, Porto Alegre: Bookman, 2003.

FRIEDMAN-HILL, Ernest. **Jess in Action: Rule Based Systems in Java**: Manning Publications Co, 2003.

JESS' HOME PAGE, http://www.jessrules.com/jess/download.shtml, visitada em 30/10/2006.

KRISNAMOORTHY, C. S. & Rajeev, S., **Artificial Intelligence and Expert Systems for Engineers**, CRC Press, 1996.

KURZWEIL, R.**The Age of intelligent machines**, MIT Press, Cambridge, Massachusetts.

REZENDE, S. O. & Pugliesi, J. B. & Varejão, F. M., **Sistemas Baseados em Conhecimento**. In: Sistemas Inteligentes: Fundamentos e Aplicações, Barueri: Manole, 2003.

SUN MICROSYSTEMS, http://www.sun.com, Visitada em 30/10/2006.

WENGER, E. & BROWN J. S. & GREENO, J. **Artificial Intelligence and Tutoring Systems, Computational and Cognitive Approaches to the Communication of Knowledge**. Morgan Kaufmann Publishers. Michael B. Morgan (editor). January 1987.