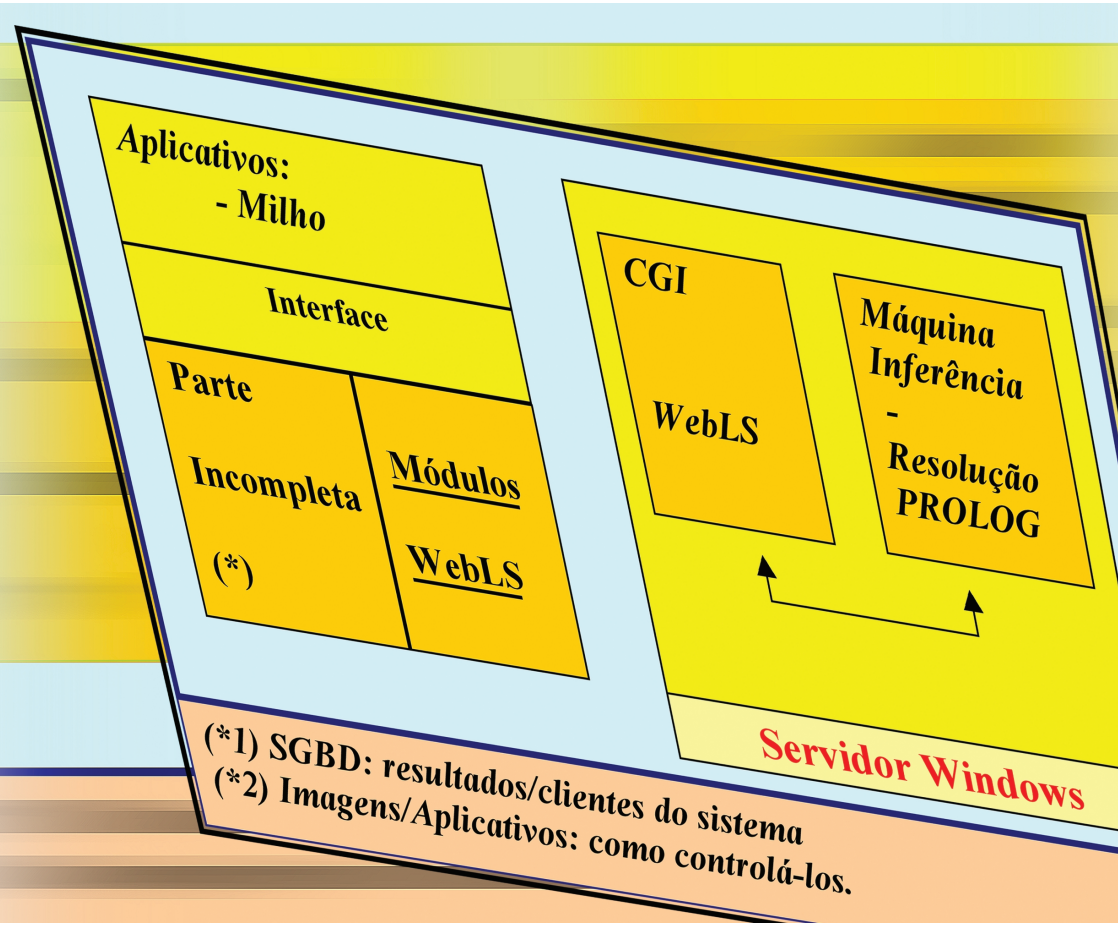
Ministério da Agricultura, Pecuária e Abastecimento

Documentos

Novembro, 2001 7

ISSN 1677-9274

Estudo de Expert System Shells para o Ambiente de Diagnose Remota



Embrapa

Embrapa

Novembro, 2001

Empresa Brasileira de Pesquisa Agropecuária
Embrapa Informática Agropecuária
Ministério da Agricultura, Pecuária e Abastecimento

Documentos 7

ISSN 1677-9274

Estudo de Expert System Shells para o Ambiente de Diagnose Remota

Maria Fernanda Moura
Sérgio Aparecido Braga da Cruz

Campinas, SP
2001

Embrapa Informática Agropecuária
Área de Comunicação e Negócios (ACN)
Av. Dr. André Tosello s/nº
Cidade Universitária "Zeferino Vaz" – Barão Geraldo
Caixa Postal 6041
13083-970 – Campinas, SP
Telefone/Fax: (19) 3789-5743
URL: http://www.cnptia.embrapa.br
Email: sac@cnptia.embrapa.br

Supervisor editorial: *Ivanilde Dispato*
Normalização bibliográfica: *Marcia Izabel Fugisawa Souza*
Capa: *Intermídia Publicações Científicas*
Editoração eletrônica: *Intermídia Publicações Científicas*

1ª edição

Moura, Maria Fernanda.
Estudo de expert system shells para o ambiente de diagnose remota / Maria Fernanda Moura e Sérgio Aparecido Braga da Cruz. — Campinas : Embrapa Informática Agropecuária, 2001.
22 p. : — (Documentos / Embrapa Informática Agropecuária ; 7)

ISSN 1677-9274

1. Sistemas especialistas. I. Cruz, Sérgio Aparecida Braga da. II. Título. III. Série.

CDD – 006.33 (21. ed.)

Autores

Maria Fernanda Moura

M.Sc. em Engenharia Elétrica, Pesquisadora da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. e-mail fernanda@cnptia.embrapa.br

Sérgio Aparecido Braga da Cruz

M.Sc. em Engenharia Elétrica, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. e-mail sergio@cnptia.embrapa.br

Apresentação

O estudo apresentado neste documento faz parte do esforço da Embrapa Informática Agropecuária na disponibilização de serviços via Internet, como é o caso do Ambiente de Diagnose Remota. Nesse ambiente de diagnose remota, um sistema é invocado em um navegador e através de simples perguntas e respostas vai direcionando um diagnóstico.

O sistema de diagnose tem como base um sistema especialista, que é um programa projetado para modelar a habilidade de resolver problemas de um especialista. O modelo em si é descrito através de um conjunto de regras de decisão e à medida que fatos vão ocorrendo, ou vão sendo descobertos, o sistema toma novas decisões, até chegar a alguma conclusão. *Expert system shells* são ferramentas de software que combinam fatos e regras, de modos diversos, dependendo do tipo de resolução escolhida, em termos de eficiência e eficácia. Assim, este documento abrange o estudo de algumas dessas ferramentas e sua posterior utilização no ambiente de diagnose remota.

José Gilberto Jardine
Chefe-Geral

Sumário

Estudo de Expert System Shells para o Ambiente de Diagnose Remota

Maria Fernanda Moura
Sérgio Aparecido Braga da Cruz

Introdução

Este documento sintetiza os estudos realizados para a definição de uma ferramenta a ser utilizada no Ambiente de Diagnose Remota do Projeto Serviços Virtuais para Transferência de Tecnologia Agropecuária - SVTTA. A ferramenta deveria ser um *expert system shell* que possibilitasse a especificação de interfaces web para os sistemas especialistas que fossem desenvolvidos com ela, e, principalmente pudesse ser utilizada em ambiente UNIX. Os estudos culminaram com a decisão de implementar uma nova ferramenta, com base no *expert system shell* CLIPS e utilizando técnicas que flexibilizam a definição de interfaces com o usuário, como a adoção de XML/XSL.

O documento divide-se em apresentação dos principais conceitos utilizados no estudo; motivação para o estudo; resumo das ferramentas estudadas, com quadro comparativo; e, a solução proposta para o Ambiente de Diagnose Remota.

Conceitos

Esta parte do documento tem por objetivo estabelecer os conceitos e termos aqui utilizados, relativos ao domínio do problema (sistemas especialistas e diagnose remota) e relativos às soluções discutidas (mesmo

quando de baixo nível de implementação, desde que sejam apresentadas neste documento).

Sistemas Especialistas

Um sistema especialista é um programa projetado para modelar a habilidade de resolver problemas de um especialista, que segue o esquema apresentado na Fig. 1 (Durkin, 1984):

- ***Base de Conhecimento***: parte contendo o conhecimento do domínio, ou seja, as regras obtidas para o sistema;
- ***Memória de Trabalho***: parte contendo os fatos sobre o problema que são descobertos, concluídos, em uma sessão de execução;
- ***Máquina de Inferência:*** processo que combina os fatos da memória com a base de conhecimento para chegar às conclusões sobre o problema.

As principais características de um sistema especialista devem ser:

- ***Separar conhecimento do controle***: é importante para que se possa modificar o sistema simplesmente acrescentando ou subtraindo regras da base de conhecimento;
- ***Representar o conhecimento de um especialista:*** a especialidade deve ser focada e limitar-se a problemas solucionáveis; e
- ***Uso de heurísticas:*** para encontrar atalhos para as soluções e permitir soluções inexatas.

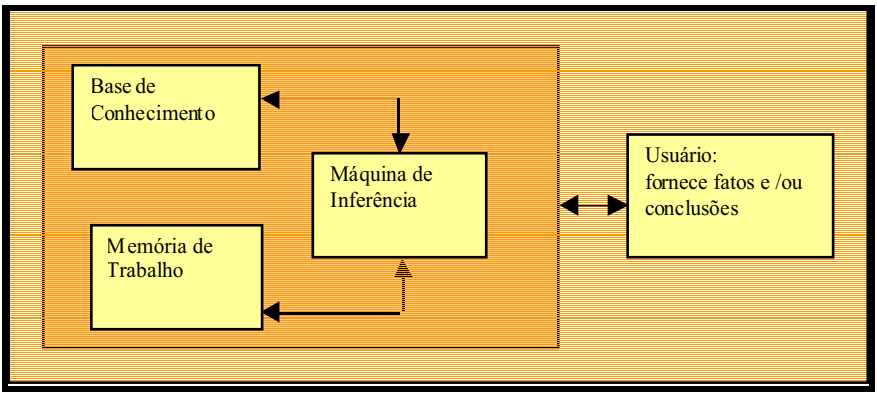


Fig. 1. Sistema Especialista.

Expert System Shell

Um *Expert System Shell* é uma ferramenta que embute uma máquina de inferência específica, assim como uma representação própria de conhecimento, e permite a construção de sistemas especialistas que utilizem a sua representação de conhecimento e máquina de inferência.

Diagnose Remota

O domínio do problema no âmbito do SVTTA refere-se, neste primeiro momento, à diagnose de doenças de plantas. Sistemas especialistas para diagnose devem ser desenvolvidos e utilizados remotamente, via Internet; assim como o Sistema de Diagnose de Doenças do Milho (Massruhá et al., 2000) – que pode ser consultado via página da Embrapa Informática Agropecuária (http://www.cnptia.embrapa.br).

Applets

A maioria dos programas JAVA são aplicações e *applets*. Aplicações são programas *stand alone* e *applets* são similares a aplicações, mas não rodam *stand alone*. Em lugar disso, os *applets* obedecem um conjunto de convenções que lhes permite rodar em um *browser* compatível com JAVA, configurados e chamados de dentro de um formulário HTML (maiores detalhes consulte: http://java.sun.com/docs/books/tutorial/getStarted/index.html).

ServLets

ServLets são módulos para rodar em um servidor orientados a requisições/respostas e que de certa forma permitem estender as funcionalidades do servidor. Por exemplo, os *ServLets* podem ser usados para tratar requisições de formulários HTML; ou seja, eles estão para o servidor como os *applets* para o cliente.

Para facilitar a implementação de *ServLets* há APIs (*Application Program Interfaces*) prontas para serem expandidas, sem que se necessite saber muito a respeito do seu funcionamento mais básico, que encapsulam os protocolos necessários. Eles substituem *cgis* e são facilmente desenvolvidos dado que tem-se uma classe JAVA para derivá-los e um simulador do servidor para testá-los. Maiores detalhes devem ser consultados nos seguintes endereços sob a Intranet da Embrapa Informática Agropecuária:

Servlet Tutorial: http://intranet.cnptia.embrapa.br/manual/jsdk/servlet_tutorial.html

JAVA Tutorial: http://intranet.cnptia.embrapa.br/manual/tutorialjava/

ServLets podem utilizar *cookies* e/ou *sessions* para guardar informações sobre um cliente:

- ***Cookies***: são uma forma de enviar informações para o cliente armazenar que mais tarde pode ser recuperada pelo servidor; implementadas como campos adicionais aos cabeçalhos das respostas HTTP. Como criá-los, atributos, envio, recuperação, classe para derivá-los, etc., pode ser estudado no JAVA Tutorial, *Lesson: Saving Client State*.
- ***Sessions***: é um mecanismo que os *ServLets* utilizam para manter o estado de uma série de requisições do mesmo cliente durante um certo período de tempo. Pode-se consultar como implementá-las no JAVA Tutorial, *Lesson: Saving Client State*.

Ambiente de Diagnose Remota e Motivação de Mudança

No âmbito do projeto SVTTA pretendia-se obter um ambiente de desenvolvimento de software que permitisse construir sistemas especialistas. Esse ambiente tem sido referido como Ambiente de Diagnose Remota porque contempla sistemas especialistas dedicados à diagnose de doenças de plantas.

Foi desenvolvida uma primeira versão desse ambiente com o uso de um *Expert System Shell* para Windows, denominado WebLS (AMZI, 2001) Fig. 2. A ferramenta WebLS é um CGI (*Common Gateway Interface*) sobre uma máquina de inferência Prolog que permite a construção dinâmica de páginas e formulários HTML (*Hyper Text Markup Language*), o que facilita a construção de sistemas especialistas para Web.

Para essa primeira versão do ambiente foram desenvolvidos:

- alguns padrões de interface para os aplicativos Web;
- um modelo de dados, ainda incompleto, para armazenar: figuras utilizadas, resultados de consultas, base de clientes, etc.; e,
- um aplicativo para teste do ambiente – diagnose de doenças do milho.

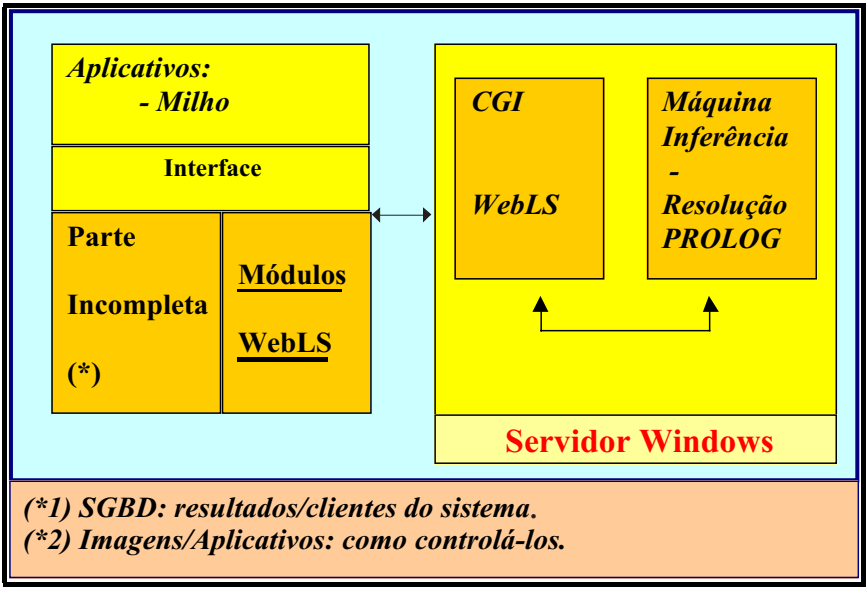


Fig. 2. Primeira Versão do Ambiente de Diagnose Remota.

Apesar de a solução citada ser satisfatória a principal motivação de mudança foi a escolha de uma ferramenta substituta ao WebLS para plataforma UNIX, que poderia ser um CGI ou um *Servlet* para UNIX com uma máquina de inferência acoplada. Ou seja, o objetivo básico era migrar o ambiente para plataforma UNIX. Porém, também seria interessante obter uma ferramenta que pudesse ser encapsulada em um aplicativo para CD – que pudesse ser desenvolvido em Object Pascal, C++ ou JAVA – preferencialmente JAVA para o caso de decidir-se utilizá-lo em *applets*.

Ferramentas Estudadas

Foram consideradas quatro ferramentas neste estudo, a saber:

- ***WebLS***: ferramenta em uso no ambiente, da AMZI (Lebano, OH, USA), da qual eram conhecidas as funcionalidades e limitações (AMZI, 2001);
- ***JESS***: *JAVA Expert System Shell*, da Sandia National Laboratories, Livermore, CA, USA (Jess ..., 2001);

- ***CLIPS*****:** *C Language Integrated Production System*, que é um *expert system shell* que tem sido desenvolvido e evoluído pela NASA, Johnson Space Center, desde 1985 (Riley, 2001);
- ***WebCLIPS*****:** CGI para o CLIPS, desenvolvido por Michael Giordano e distribuído sob licença GNU (Giordano, 2001).

WebLS

O WebLS, como já citado, é um *expert system shell* implementado como um CGI sobre uma máquina de inferência PROLOG; cuja plataforma exclusiva é Windows.

A grande vantagem do WebLS é a forma de especificar/implementar o sistema especialista. Sua linguagem de programação é modular e a sintaxe bastante simples, o que permite dividir o problema em componentes, isto é, representar a base de conhecimento de forma modular, separando regras de diferentes conteúdos/contextos. Além disso, a geração de formulários HTML dinamicamente e de forma bastante simplificada, utilizando as marcações HTML, facilita muito o desenvolvimento dos aplicativos (sistemas especialistas para acesso via Internet).

A documentação do WebLS é de muito boa qualidade, ampla e com vários exemplos de acordo com o tipo de sistema que se pretende desenvolver. Ainda, a linguagem simples e de fácil uso tem a intenção de poder ser utilizada por especialistas do domínio de conhecimento sem a intervenção de engenheiros de software.

JESS

O JESS é uma extensão do CLIPS em JAVA; isto é, é um *expert system shell* que utiliza o CLIPS traduzido para JAVA e complementa as funcionalidades de um *expert system shell* para também ser um ambiente de desenvolvimento de aplicativos genéricos, incorporando componentes de interface gráfica do JAVA e permitindo a expansão do próprio JESS através da implementação de novas classes JAVA. O JESS é basicamente uma biblioteca, e embora o seu objetivo não seja gerar aplicativos Web exclusivamente, ele pode ser embutido em aplicativos Web ou mesmo utilizado como *applet* em uma página Web. Porém, como o JESS é

pesado em termos de execução, devido à complexidade da máquina de inferência, o ideal é rodá-lo na máquina servidora e não na cliente. Ele é compatível com JAVA 1.1 em diante e, conseqüentemente, não há restrições de plataforma.

Para desenvolver aplicativos em JESS deve-se conhecer a *JESS Language*, que é equivalente a *CLIPS Language* ampliada com novos componentes, e para isto há vasta documentação de excelente qualidade. No entanto a sintaxe do JESS não é tão simples, é muito semelhante a linguagem LISP, e dificilmente seria utilizada por um especialista do domínio sem conhecimentos de inteligência artificial.

O JESS pode ser utilizado em processamento em lote ou linha de comando a partir de sua chamada. Para o caso do lote desenvolve-se o aplicativo e invoca-se o mesmo em uma linha de comando do JESS. Também o mesmo aplicativo pode ser transformado em *applet*, porém, como já dito anteriormente, a execução desse *applet* pode ser muito lenta.

Como se trata de uma biblioteca JAVA pode-se embuti-lo em um *ServLet* ou em um CGI já desenvolvido em JAVA com pequena dificuldade, dado que sua documentação orienta como integrá-lo a outras ferramentas. Deve-se lembrar que toda essa expansibilidade do JESS acaba permitindo também que se possa implementar novas estratégias de resolução de conflitos de regras, o que pode ser muito interessante.

Quanto à máquina de inferência, o JESS utiliza o Algoritmo Rete, proposto em 1982 (Forgy, 1982) que é o mesmo utilizado pelo CLIPS e cuja complexidade é da ordem de O(RFP), onde: R= número de regras, P = média de padrões LHS (lado esquerdo das regras, combinações) e F = número de fatos na base de conhecimento. Porém, apesar da complexidade linear, o uso de memória é bastante grande, programas moderados chegam a ocupar 16 mega de *heap*.

CLIPS

O CLIPS também é um *expert system shell* que permite desenvolver além de sistemas especialistas também aplicativos genéricos. Uma de suas grandes vantagens é a portabilidade, desenvolvido em C e com essa preocupação, ele compila em várias plataformas, incluindo-se UNIX que é o objetivo deste estudo.

O CLIPS implementa o Algoritmo Rete, já comentado no item anterior, e é basicamente uma biblioteca, mas o seu ambiente de desenvolvimento permite uma grande flexibilidade e formas de interação como ilustrado na Fig. 3.

De acordo com a Fig. 3, o CLIPS é um interpretador de linhas de comando que pode executar arquivos em lote (aplicativos previamente desenvolvidos), ser utilizado diretamente (escrevendo-se as linhas de comando em seu console), permite exibir uma saída textual e ainda permite salvar diferentes arquivos: um programa que vem sendo desenvolvido diretamente em seu console (programa.clp) e/ou a base de fatos (memória de trabalho até aquele momento).

A sua linguagem é semelhante a LISP, não muito simples para um usuário sem conhecimentos de programação, porém muito flexível e expansível. Pode-se criar novos *templates*, *functions* e classes (ele possui elementos para programação orientada a objetos) na própria linguagem do CLIPS, em arquivos ".clp" que podem ser importados por aplicativos (programas com extensão ".clp"). A documentação do CLIPS é extensa, didática e de excelente qualidade, orientando não só a construção de sistemas especialistas mas também a expansão do CLIPS e sua integração com outras ferramentas.

Não há custo de distribuição, o *download* é gratuito a partir do *site* e deve apenas ser feito um registro de uso *online*.

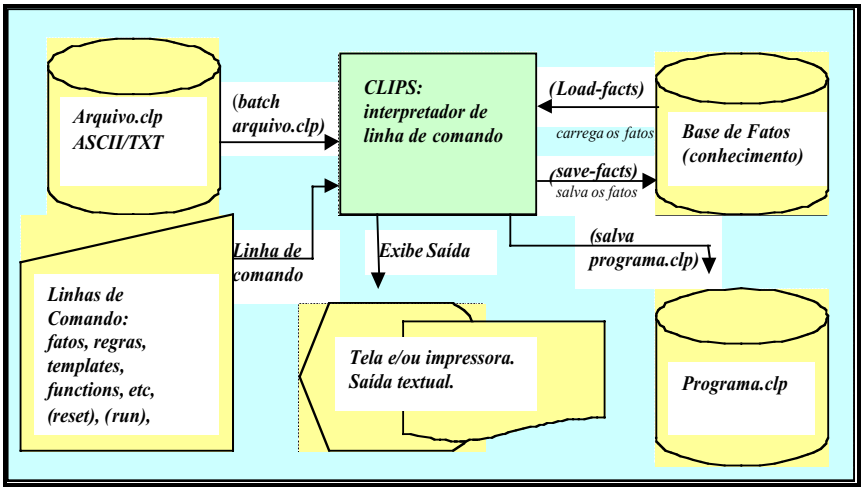


Fig. 3. Funcionamento do CLIPS.

WebCLIPS

Esta ferramenta implementa um *cgi* que encapsula o CLIPS, como ilustrado na Fig. 4. Assim, ela engloba todas as vantagens do CLIPS associadas à direta construção de aplicativos Web, pois permite gerar páginas e formulários HTML dinamicamente, trata os métodos GET e POST (da *query* enviada ao servidor), e permite salvar o *status* do cliente em *cookies*. Além disso, o WebCLIPS é implementado em C e há versões para plataforma UNIX e Windows. Sua distribuição é gratuita e ele se encontra sob licença GNU.

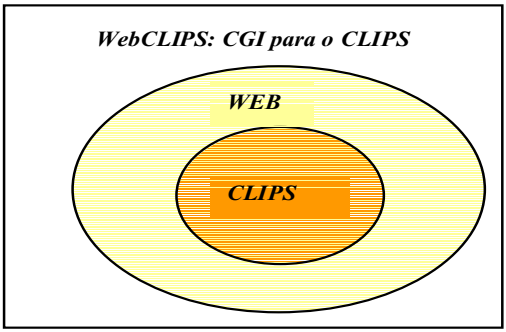


Fig. 4. WebCLIPS.

Uma importante característica do WebCLIPS é ilustrada na Fig. 5. A ferramenta não implementa páginas e formulários HTML propriamente ditos, ela utiliza as extensões de aplicativos CLIPS. O arquivo "wchtml.clp" nada mais é que um conjunto de definições de funções (*functions*) CLIPS que substituem o comando "*printout*" do CLIPS (saída padrão), trocando os resultados por especificações de *targets*/marcadores HTML, cobrindo completamente a versão 3.2. do HTML. O arquivo "wctmpl.clp" define *templates* que são as classes utilizadas em aplicativos WebCLIPS; cobrindo as definições de *Screens* e *queries* ODBC. As Screens são as unidades básicas de qualquer aplicativo WebCLIPS, cada *Screen* corresponde a uma página/formulário HTML, que não é fixo, pois dependendo das regras verdadeiras naquele momento as saídas, que serão os formulários HTML, podem estar em diferentes estados. Assim, um aplicativo WebCLIPS é um programa CLIPS ".clp" que importa esses *templates* e *functions*.

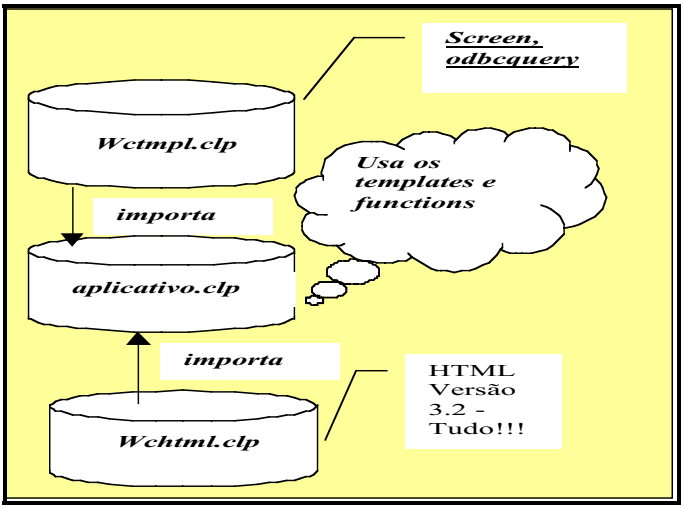


Fig. 5. Templates e Functions do WebCLIPS.

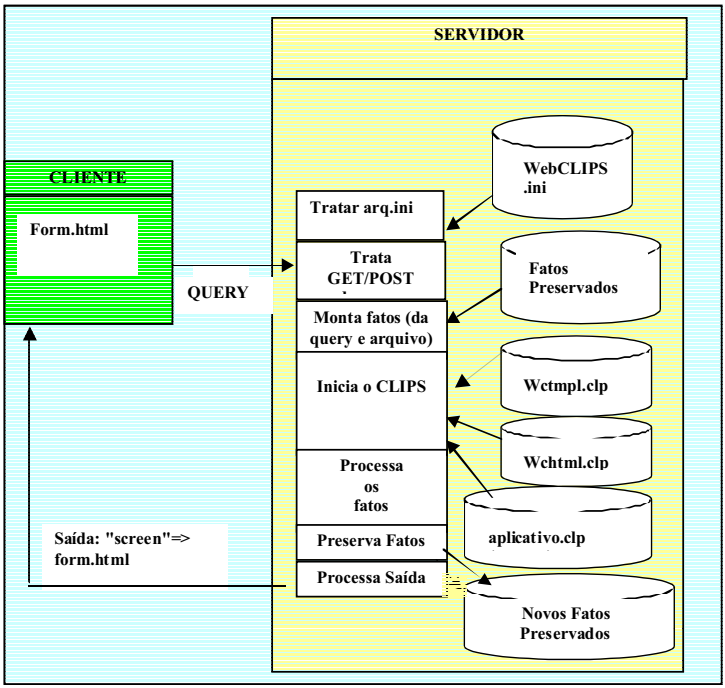


Fig . 6. Funcionamento do WebCLIPS.

O WebCLIPS funciona como ilustrado na Fig. 6. De modo que a passagem de fatos do formulário HMTL para o WebCLIPS é feita uma utilização de parâmetros *hidden*, sendo que as variáveis são exclusivamente denominadas "*fact*" e seus valores correspondem à definição do fato na linguagem do CLIPS. O WebCLIPS sempre inicia sua execução como se fosse a primeira vez que ele inicia a máquina de inferência para o processo cliente, assim, ele primeiro consulta um arquivo de configuração inicial, o ***webclips.ini***. Esse arquivo contém diretivas sobre onde encontrar os aplicativos (diretórios), onde escrever os resultados de saída (resultado de *debug*, erros, etc), onde gravar arquivos temporários (do tipo de base de fatos, etc.), onde se encontram os *templates* e *functions*, etc. A seguir ele trata o método GET ou POST, classificando a *query* e montando os novos fatos a partir dela. Então, ele checa se existe algum *cookie* associado a esse cliente, caso exista, significa que há fatos preservados para esse cliente, que devem estar armazenados no arquivo temporário cujo nome se encontra no *cookie*. O nome do *script*, ou aplicativo, faz parte também do arquivo webclips.ini, podendo ter sido indicado (através do comando *assert)* como novo fato na *query*, então o WebCLIPS analisa essa informação para decidir quem é o próximo *script* e mandar executá-lo com o novo conjunto de fatos formado, ou seja, com a nova memória de trabalho.

Quadros comparativos

Resumidamente nas Tabelas 1 e 2 a seguir estão colocadas as vantagens e desvantagens de cada ferramenta estudada.

Na Tabela 1, tem-se o uso de cada ferramenta, a característica da linguagem de construção dos aplicativos, a qualidade da documentação, tipo de perfil do usuário desenvolvedor de aplicativos e condições da licença de uso.

Na Tabela 2 tem-se o resumo de plataformas e linguagem de programação em que a ferramenta foi desenvolvida, sua capacidade de extensão funcional, interação com HTML, interação com sistemas gerenciadores de bancos de dados e qual a máquina de inferência utilizada.

Tabela 1. Uso, Linguagem de Desenvolvimento, Documentação e Licença.

Ferramenta	*Tipo /Uso*	*Linguagem Desenvolvimento*	*Qualidade da Documentação*	*Usuário Alvo*	*Licença*
JESS	Applets, servlets, aplicações	JESS *language*	Boa	Engenheiro do Conhecimento ou de Software	450 US $ Free se pesquisa
CLIPS	Biblioteca de funções	CLIPS (parecida com LISP)	Ótima	Eng. de Conhecimento ou Software	Free
WebCLIPS	CGI	CLIPS + targets HTML	Média	Eng. Conhecimento ou Software	GNU
WebLS	CGI	Própria (modular)	Ótima	Eng. conhecimento ou especialista do domínio	Free

Tabela 2. Plataformas, extensibilidade, interação com HTML e BD.

Ferramenta	*Linguagem/ Plataforma*	*Extensão da Linguagem*	*Interação com HTML*	*Interação com Banco de Dados*	*Máquina de Inferência*
JESS	JAVA/qualquer	Altíssima	Zero/ Componentes	Zero/ Componentes	Algoritmo Rete
CLIPS	C/qualquer	Alta através de bibliotecas	Zero/ Bibliotecas	Zero/ Programar	Algoritmo Rete
WebCLIPS	C/ UNIX ou Windows	Alta	Alta/ Targets	Apenas para Windows	Algoritmo Rete
WebLS	Prolog/Windows	nenhuma	Alta/direta	Zero	Resolução Prolog

Solução Proposta e Conclusões

O estudo das ferramentas foi realizado em duas etapas. A primeira objetivava a escolha de uma das ferramentas estudadas para substituir o WebLS, de acordo com as funcionalidades oferecidas e as necessidades do projeto; esse primeiro estudo foi apresentado em reunião, junto a cenários de possíveis desenvolvimentos. Nessa reunião foi aventada a possibilidade de desenvolvimento também de um aplicativo em CD que embutisse o ambiente de diagnose. Assim, após essa reunião, decidiu-se estudar com maior detalhe o CLIPS e o WebCLIPS, de modo que se pensasse em utilizar as idéias do WebCLIPS para construir um ServLet e/ou um aplicativo em CD que permitisse embutir o CLIPS e atingir os novos objetivos no projeto.

Como resultado final dos estudos, decidiu-se pelo desenvolvimento de um servlet que encapsule o CLIPS, reutilizando as seguintes idéias do WebCLIPS:

- Definição de *templates*: os templates utilizados para definir *Screen* e *ODBCQuery* podem ser reutilizados na ferramenta do SVTTA, dado que a idéia de uso de Screens adapta-se perfeitamente às necessidades dos aplicativos do projeto;

- Definição de funções (*functions*): reutilizar a idéia do arquivo *wchtml.clp* trocando-o para o padrão XML/XSL (Harold, 1999). Embora o *wchtml.clp* contenha toda a versão 3.2 do HTML, o interesse é um padrão generalizado como XML, de modo que se possa gerar formulários HTML quando a resposta vem do ServLet ou componentes de interface gráfica JAVA, quando o aplicativo estiver em CD, bastando para isso modificar os formulários de estilo XSL;

- Passagem de fatos: o método de passagem de fatos do processo cliente para o servidor é um padrão interessante, será reusado o comando *"(assert (fact value))"*, como no WebCLIPS, em parâmetros *hidden*; e

- Preservação de fatos: os fatos serão preservados em arquivos temporários, porém não serão amarrados a *cookies* e sim a *sessions.*

Ainda, a arquitetura da nova ferramenta deve ser semelhante a do WebCLIPS, porém deverá prever o encapsulamento do CLIPS sob classes JAVA, para que as mesmas classes possam ser reusadas para o *servlet* e aplicativos para CD ou outros que vierem a surgir.

Referências Bibliográficas

AMZI. **WebLS product description**. Disponível em: <http://www.amzi.com/products/webls.htm>. Acesso em: 19 abr. 2001.

DURKIN, J. **Expert systems:** design and development. Englewood Cliffs: Prentice Hall, 1994. 800 p.

FORGY, C. L. Rete: a fast algorithm for the many pattern: many object pattern match problem. **Artificial Intelligence**, v. 19, p. 17-37, 1982.

GIORDANO, M. **WebCLIPS home page**. Disponível em: <http://www.monmouth.com/~km2580/wchome.htm>. Acesso em: 9 abr. 2001.

HAROLD, E. R. **XML bible**. Foster City: IDG Books Worlwide, 1999. 1015 p.

JESS: the expert system shell for the Java platform. Disponível em: <http://herzberg.ca.sandia.gov/jess>. Acesso em:17 out. 2001.

MASSRUHÁ, S. M. F. S.; CRUZ, S. A. B. da; SOUZA, E. de. Diagnose virtual: um sistema para diagnóstico de doenças do milho via Web. In: CONGRESSO DA SBI-AGRO, 2. = AGROSOFT 99, 1999, Campinas. **II Workshop da SBI-AGRO – II Congresso da SBI-AGRO – I Congresso da SBI-AGRO**: [anais]. Juiz de Fora: SBI-AGRO, [2000]. 1 CD-ROM – Agrosoft 99 – Trabalhos selecionados.

RILEY, G. **CLIPS**: a tool for building expert systems. Disponível em: <http://www.ghg.net/clips/CLIPS.html>. Acesso em: 19 abr. 2001.

Informática Agropecuária

MINISTÉRIO DA AGRICULTURA,
PECUÁRIA E ABASTECIMENTO

GOVERNO FEDERAL
Trabalhando em todo o Brasil