UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

MOTOSYSTEM: PROTÓTIPO DE UM SISTEMA BASEADO EM CONHECIMENTO PARA DIAGNÓSTICO DE DEFEITOS MECÂNICOS EM MOTOCICLETAS

GUSTAVO LANA DA COSTA

BLUMENAU
2014

2014/1-05

GUSTAVO LANA DA COSTA

MOTOSYSTEM: PROTÓTIPO DE UM SISTEMA BASEADO EM CONHECIMENTO PARA DIAGNÓSTICO DE DEFEITOS MECÂNICOS EM MOTOCICLETAS

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Sistemas de Informação— Bacharelado.

Prof. Roberto Heinzle, Doutor - Orientador

BLUMENAU
2014

2014/1-05

MOTOSYSTEM: PROTÓTIPO DE UM SISTEMA BASEADO EM CONHECIMENTO PARA DIAGNÓSTICO DE DEFEITOS MECÂNICOS EM MOTOCICLETAS

Por

GUSTAVO LANA DA COSTA

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

	_____
Presidente:	Prof. Roberto Heinzle, Doutor – Orientador, FURB
	_____
Membro:	Prof. Aurélio Faustino Hoppe, Mestre – FURB
	_____
Membro:	Prof. Wilson Pedro Carli, Mestre – FURB

Blumenau, 02 de julho de 2014.

Dedico este trabalho a todos os amigos, especialmente aqueles que me ajudaram diretamente na realização deste.

AGRADECIMENTOS

A Deus principalmente, por tornar tudo isso possível.

À minha família, que sempre me apoiou.

À minha namorada, pela motivação.

Aos meus amigos, pelos empurrões e cobranças.

Ao meu orientador, professor Roberto Heinzle, por ter acreditado na conclusão deste trabalho.

Aos professores do Departamento de Sistemas e Computação da Universidade Regional de Blumenau por suas contribuições durante os semestres letivos.

Eu gosto do impossível porque lá a concorrência é menor.

Walt Disney

RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo de sistema baseado em conhecimento para diagnóstico de defeitos mecânicos em motocicletas. O protótipo possibilita ao usuário a gerência de uma oficina mecânica e a diminuição do tempo despendido para diagnosticar um defeito mecânico. O módulo de diagnóstico de defeitos, através de inteligência artificial, simula o trabalho de um especialista para identificar qual pode ser o defeito existente na motocicleta, conforme as informações inseridas pelo usuário. Este foi desenvolvido para a plataforma Java, utilizando a biblioteca Java *Expert System Shell* (JESS) para a criação do módulo de diagnóstico. Para armazenar as informações da base de conhecimentos, enquanto que para todos os cadastros do protótipo, foi utilizado o banco de dados MySQL. Através do desenvolvimento deste protótipo, o resultado alcançado foi um sistema baseado em conhecimento que apresenta ao usuário uma listagem de peças a serem verificadas, também possuindo funcionalidades administrativas, que permitem maior segurança de armazenamento das informações além de facilitarem sua respectiva manutenção.

Palavras-chave: Inteligência Artificial. Sistemas Baseados em Conhecimento. JESS.

ABSTRACT

The present study shows the development of a prototype of a knowledge-based system for diagnostic of motorcycles mechanical defects. The prototype makes the user be able to manager of a machine shop and reduction of spent time on mechanical defect diagnostic. The module for defect diagnostic, through the artificial intelligence, assumes the task of an expert to identify which can be the defect on the motorcycle, as per the informations inserted by user. This module was developed to a Java platform, using the Java Expert System Shell (JESS) library to create the diagnostic module. To store the information in the knowledge base, while for all entries of the prototype, the MySQL database was used. The results of this prototype development was a knowledge based system which presents to user a list of mechanical parts to be checked, also with management functionalities, which permits greater security for storage of information in addition to facilitating its maintenance.

Key-words: Artificial intelligence. Knowledge-based system. JESS.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

API - *Application Programming Interface*

BD - Banco de dados

CLIPS - *C Language Integrated Production System*

DER - Diagrama de Entidade e Relacionamento

EA - *Enterprise Architect*

FURB - Fundação Universidade Regional de Blumenau

IA - Inteligência artificial

IDE - *Integrated Drive Eletronics*

JESS - *Java Expert System Shell*

LISP - *List Processing Language*

NSBC - Núcleo do Sistema Baseado em Conhecimento

RF – Requisito Funcional

RNF – Requisito Não-Funcional

SBC - Sistema Baseado em Conhecimento

SE - Sistemas especialistas

SGBD - Sistema Gerenciador de Banco de Dados

SINTA - Sistemas Inteligentes Aplicados

UC - *Use Case*

SUMÁRIO

1 INTRODUÇÃO

No cenário mundial atual, as motocicletas estão conquistando o trânsito das cidades, e esta dominação é crescente. O crescimento se deve ao fato das motocicletas terem uma boa relação custo x benefício (HOLZ; LINDAU, 2009). Isso se justifica por conta das motocicletas serem mais rápidas, ou seja, mesmo com o congestionado trânsito das zonas urbanas este meio de transporte possui facilidade em se locomover. Holz e Lindau (2009) registram que a frota de motocicletas está crescendo consideravelmente no mundo, impulsionada por atrativos como incentivo para a aquisição, baixo custo de combustível, facilidade de estacionamento e circulação no trânsito das grandes metrópoles.

Junto do crescimento da frota de motocicletas, a demanda de manutenção também aumenta, e este aumento requer mais agilidade por parte das oficinas e seus respectivos profissionais. Uma forma de agilizar o atendimento realizado por parte de uma oficina mecânica seria a padronização dos métodos de diagnóstico dos defeitos mecânicos, pois cada mecânico possui uma forma única para identificar um defeito. Isso ocorre, porque ao longo do tempo, cada profissional adota métodos considerados por eles mesmos, mais apropriados para identificar o defeito. Isso é registrado por Soares (2010), que afirma “todos os mecânicos têm suas maneiras de identificar os defeitos pelo barulho do motor”. Com base na afirmação anterior é possível concluir que existem situações em que o mecânico acerta o defeito da motocicleta apenas utilizando a seu conhecimento e habilidade. Existem defeitos que um profissional qualificado consegue diagnosticar apenas através do som emitido pela motocicleta. Casos em que o motor produz sons como se estivesse arrastando um componente em outro pode ser um sinal do princípio de um defeito mecânico.

> Há quem diga que só pelo som, um mecânico experiente diz qual é o defeito e invariavelmente acerta o diagnóstico, porque em cada motor, cada peça móvel produz um ruído característico do movimento que ela faz e do atrito que produz. Se for suave, denota um deslizamento uniforme e perfeitamente normal, mas se for cheio de sons de arrasto ou como se estivéssemos amassando papel de alumínio, aí as coisas já se complicam. Outras peças que têm a função de produzir movimentos de vai-vem, deslocando outra peça, como uma válvula, por exemplo, ou uma biela. Nesses casos o ruído já apresenta outros sons que identificam batidas que indicam folgas que podem ser normais ou excessivas, de acordo com o nível e altura do som. (SOARES, 2010).

Outra situação bastante recorrente nas oficinas mecânicas é a rotatividade de mecânicos, conforme uma entrevista preliminar, realizada na oficina Moto Peças Kalarrari, estabelecida na cidade de Gaspar, estado de Santa Catarina, e documentada no Anexo A. Este

fato também afeta diretamente a agilidade do atendimento, pois nem todos os mecânicos possuem experiência completa em todos os módulos de uma motocicleta.

Analisando as causas das dificuldades encontradas em diminuir o tempo de atendimento dos clientes, foram identificados dois motivos principais, os quais serão o foco do protótipo deste trabalho. Os dois motivos citados são o tempo despendido para diagnosticar quais as peças da motocicleta que apresentam defeitos e a organização das informações administrativas do estabelecimento.

Levando-se em consideração os motivos citados anteriormente, o presente trabalho propõe, através de um protótipo *desktop*, uma solução para as questões de identificação de defeitos mecânicos de motocicletas e gerência administrativa do estabelecimento. O protótipo permitirá que o usuário digitalize todas as informações administrativas através de módulos de cadastros e gere relatórios com estas informações. Além disto, respondendo a poucas perguntas será possível obter um diagnóstico inicial do defeito presente nas motocicletas.

Assim, o protótipo desenvolvido no presente trabalho tem o intuito aumentar a agilidade de atendimento nas oficinas mecânicas, seja através de módulos de cadastros e relatórios ou com questionários que informarão o diagnóstico do defeito da motocicleta.

1.1 OBJETIVOS DO TRABALHO

O objetivo geral deste trabalho é apresentar um protótipo de um Sistema Baseado em Conhecimento (SBC) para administração de oficinas mecânicas, que auxiliará no diagnóstico dos defeitos mecânicos de motocicletas.

Os objetivos específicos desse trabalho são:

a) construir uma base de conhecimentos com a relação de defeitos mecânicos que podem ocorrer em motocicletas e sua respectiva solução;

b) desenvolver um módulo para o gerenciamento do setor administrativo das oficinas;

c) integrar o SBC a uma base de dados, para que seja possível a gerenciar o conhecimento existente na base de conhecimento.

1.2 ESTRUTURA DO TRABALHO

No primeiro capítulo tem-se a introdução ao tema principal deste trabalho com a apresentação da justificativa e dos objetivos.

No segundo capítulo apresenta-se a fundamentação teórica pesquisada sobre sistemas baseados em conhecimento, características de um sistema baseado em conhecimento, a estrutura de um sistema baseado em conhecimento, aquisição do conhecimento, regras de produção, a ferramenta JESS, a *shell* JESS, além de trabalhos correlatos.

O terceiro capítulo apresenta o desenvolvimento do protótipo iniciando-se com o levantamento de informações, tendo na seqüência a especificação, a implementação e, por fim, os resultados e discussões.

No quarto capítulo tem-se as conclusões deste trabalho bem como apresentam-se sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda os assuntos sistemas baseados em conhecimento, as características de um sistema baseado em conhecimento, a estrutura de um sistema baseado em conhecimento, a aquisição do conhecimento, as regras de produção, a ferramenta JESS, a *shell* JESS e o sistema atual, além de trabalhos correlatos.

2.1 SISTEMAS BASEADOS EM CONHECIMENTO

Segundo Rezende (2003, p. 13), os Sistemas Baseados em Conhecimento (SBC) têm sido utilizados tanto na área acadêmica quanto comercial por mais de 20 anos. Rich e Knight (1994) observam que no período das três últimas décadas, pesquisadores de IA passaram a considerar o conhecimento específico do domínio como um requisito indispensável na resolução de problemas complexos. Os SBC são fundamentados em um módulo explícito de conhecimento destinado a resolver problemas, sendo de grande avanço tecnológico na resolução computacional de problemas que antes só poderiam ser resolvidos por seres humanos. Os sistemas referidos anteriormente possuem como principais características uma base de conhecimento, onde é possível construir sentenças, modelando o problema a ser resolvido e um mecanismo de raciocínio capaz de realizar inferências sobre esta base de obter conclusões a partir deste conhecimento (REZENDE, 2003, p. 14).

Um dos primeiros sistemas capaz de resolver problemas que anteriormente eram realizados apenas por especialistas humanos foi o DENDRAL. Este programa faz parte da classe dos sistemas especialistas (SE) e Rezende (2003, p. 19), registra que “SBCs podem ser classificados como SE quando o desenvolvimento do mesmo é voltado para aplicações nas quais o conhecimento a ser manipulado restringe-se a um domínio específico e conta com um alto grau de especialização”.

Gomes (2010) registra que em 1969, na Universidade de Stanford, foi desenvolvido o DENDRAL, um programa em que era possível desenvolver soluções com capacidade de encontrar as estruturas moleculares orgânicas a partir da espectrometria de massa nas ligações químicas presentes em uma molécula desconhecida. A equipe responsável pela construção do

DENDRAL foi Edward Feigenbaum, Bruce Buchanan e Joshua Lederberg (LINDSAY et al., 1980).

O DENDRAL teve importância histórica para o desenvolvimento de programas inteligentes, porque representou o primeiro sistema bem-sucedido de conhecimento intensivo: sua habilidade derivava de um grande número de regras de propósito específico (RUSSELL; NORVIG, 2004). Conforme Rosa (2000) uma boa solução depende de uma boa representação. Na maioria das aplicações em IA, a escolha da representação é a fase mais difícil, pois as possibilidades são maiores e os critérios menos claros.

Ainda segundo Rosa (2000), são muitos os problemas para a elaboração de programas inteligentes, que usam conhecimento para realizar tarefas. Alguns desses problemas são:

a) como estruturar um sistema de representação que será capaz, de fazer todas as distinções importantes;

b) como manter-se reservado sobre os detalhes que não podem ser resolvidos;

c) como reconhecer, eficientemente, que conhecimento é relevante para o sistema, em uma situação particular;

d) como adquirir conhecimento dinamicamente, durante o tempo de vida do sistema;

e) como assimilar partes do conhecimento, na ordem em que elas são encontradas, ao invés de uma ordem específica de apresentação.

2.2 CARACTERÍSTICAS DE UM SISTEMA BASEADO EM CONHECIMENTO

Esta seção apresenta o que um sistema baseado em conhecimento deve ser capaz de realizar, segundo Rezende (2003, p. 16):

a) utilizando uma linguagem de fácil entendimento, questionar o usuário, para reunir as informações de que necessita;

b) a partir dessas informações e do conhecimento nele embutido, desenvolver uma linha de raciocínio para encontrar soluções satisfatórias. Para isso, o SBC necessita lidar com regras e informações incompletas, imprecisas e conflitantes;

c) caso seja questionado pelo usuário, explicar seu raciocínio, do porquê necessita de informações externas e de como chegou às suas conclusões. Para tanto o sistema deve memorizar as inferências realizadas durante o processo de raciocínio, ser capaz de interpretar esse processo e apresentá-lo de forma compreensível para o usuário do sistema;

d) conviver com seus erros, isto é, tal como um especialista humano, o SBC pode cometer erros, mas deve possuir um desempenho satisfatório que compense possíveis enganos. Em particular, as soluções apresentadas para problemas complexos devem ser equivalentes àquelas oferecidas pelo especialista humano, quando este existir.

As características citadas anteriormente definem funcionalidades que estão presentes em SBCs. Porém, elas não evidenciam as diferenças fundamentais entre um sistema baseado em conhecimento e um sistema convencional. É possível, embora não seja comum, construir um sistema convencional que também apresente tais características. Segundo Motta (1998), existe um modo mais operacional de se definir SBCs que auxilia a estabelecer essas diferenças. Nessa definição, os SBCs devem possuir as seguintes propriedades:

a) todo o conhecimento sobre o problema deve estar explicitamente representado na base de conhecimentos do sistema;

b) a base de conhecimentos deve ser usada por um agente capaz de interpretá-la. Esse agente é conhecido como o mecanismo ou motor de inferência;

c) os problemas resolvidos são aqueles sobre os quais não é conhecido um procedimento determinístico que garanta uma resolução efetiva. Esses sistemas usam conhecimento específico do domínio para contornar a formulação genérica do problema e ou ausência de conhecimento preciso e completo sobre o seu domínio.

Os dois primeiros itens dessa definição procuram tornar clara a diferença entre os sistemas convencionais e os baseados em conhecimento. Já o último item diferencia SBCs de sistemas nos quais há codificação explícita do conhecimento e a resolução de problemas se faz por meio de problemas determinísticos. Basicamente, sistemas baseados em conhecimento diferem dos sistemas convencionais em:

a) como são organizados;

b) como incorporam conhecimento;

c) como executam;

d) impressão que causam aos usuários com quais interagem.

Estas diferenças são evidenciadas no Quadro 1.

Quadro 1 - Diferenças entre sistemas convencionais e SBCs

Sistemas convencionais	Sistemas Baseados em Conhecimento
Estrutura de dados	Representação do conhecimento
Dados e relações entre dados	Conceitos, relações entre conceitos e regras
Tipicamente usa algoritmos determinísticos	Busca heurística
Conhecimento embutido no código do programa	Conhecimento representado explicitamente e separado do programa que o manipula e interpreta
Explicação do raciocínio é difícil	Podem e devem explicar seu raciocínio

Fonte: Motta (1998).

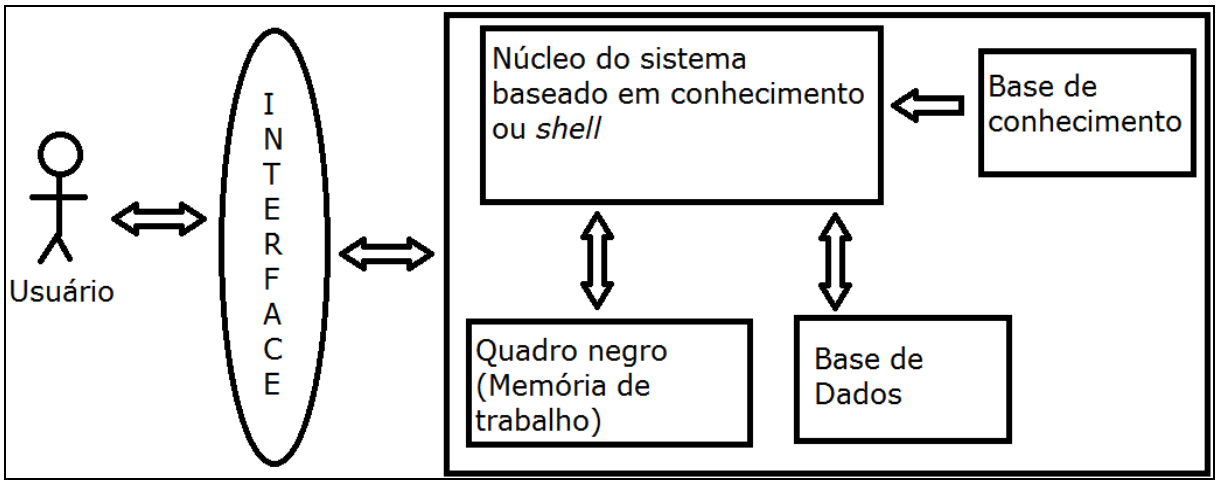
Como pode ser observado no Quadro 1, existem algumas diferenças entre um sistema convencional e um SBC. O primeiro apresenta seus dados todos estruturados, sempre os relacionando entre si. Utiliza algoritmos para determinar os resultados e o conhecimento lógico fica dentro do código do programa, isto é, o programa e o conhecimento estão juntos, além de dificilmente explicarem o raciocínio. Diferente do sistema anterior, um SBC faz a representação do conhecimento e trabalha com conceitos e regras. A determinação dos resultados é realizada através de uma busca heurística em uma base de conhecimento que não fica embutida no programa e é seu dever explicar o raciocínio utilizado para obter o resultado.

2.3 ESTRUTURA DE UM SISTEMA BASEADO EM CONHECIMENTO

Esta seção apresenta, em detalhes, a estrutura geral de um SBC, mostrando os principais módulos que o compõe. Cabe dizer que nem todos os sistemas baseados em conhecimento possuem a mesma estrutura. Contudo, a maioria apresenta uma estrutura geral

semelhante, na qual se destacam os módulos responsáveis pelo armazenamento da base de conhecimentos e pelo mecanismo de inferência (REZENDE, 2003, p. 22). A Figura 1 demonstra a estrutura básica de um SBC.

Figura 1 - Estrutura de um sistema baseado em conhecimento



Fonte: Rezende (2003, p. 23).

Conforme a Figura 1, o funcionamento básico da estrutura é o seguinte:

a) a interface é o canal de comunicação entre o usuário e o SBC;

b) o SBC, por sua vez, possui como módulo principal o Núcleo do Sistema Baseado em Conhecimento (NSBC), que irá fazer a comunicação com todos os outros módulos do SBC;

c) a base de conhecimentos armazena todo o conhecimento sobre o assunto, esta é acessada pelo NBSC para buscar as informações à serem inferidas;

d) o quadro negro (memória de trabalho) irá armazenar todas as conclusões intermediárias, repassadas pelo NSBC, utilizadas durante o processo de inferência, bem como, as respostas fornecidas pelo usuário;

e) a base de dados corresponde a um Banco de Dados (BD), que o sistema pode estar interagindo para obtenção ou armazenamento de dados e/ou informações.

2.3.1. Base de conhecimento

Quando o método de representação do conhecimento é realizado através de regras de produção, a base de conhecimento é o local onde estão armazenadas as regras e fatos de um SBC. Este conhecimento é passado ao sistema pelo especialista e armazenado de uma forma

própria que permitirá ao sistema fazer posteriormente o processo de inferência. Um novo fato pode modificar todo o processo de inferência de acordo com as regras existentes sobre ele que estão sendo aplicadas e também sobre os novos fatos gerados pela avaliação dessas regras (RIBEIRO, 1987). Quando o método de representação do conhecimento é outro, segundo Rezende (2003, p. 26), a base de conhecimentos armazena a descrição do conhecimento necessário para a resolução do problema a ser solucionado pelo sistema.

Segundo Heinzle (1995), no desenvolvimento de um sistema a fase mais complexa é a da construção da base de conhecimento, pois o conhecimento específico (especialista) não se encontra formalizado, tornando necessário um trabalho prévio para concluir tal tarefa. Essa afirmação também se aplica aos sistemas baseados em conhecimento, pois, uma base de conhecimento que não foi formalizada corretamente não fará o sistema obter uma resposta concisa sobre o problema. Sabbatini (1992), afirma que as sentenças que formam a base de conhecimentos são chamadas de regras de produção, que, por serem fixas, não permitem incertezas.

2.3.2. Núcleo do Sistema Baseado em Conhecimento

Segundo Rezende (2003, p. 23), o Núcleo do Sistema Baseado em Conhecimento (NSBC) é responsável pelas principais atividades do sistema, como:

a) controle na interação com o usuário ou com equipamentos externos;

b) processamento do conhecimento utilizando alguma linha de raciocínio;

c) justificativa ou explicação das conclusões obtidas a partir do raciocínio.

O módulo NSBC é composto basicamente por três submódulos interdependentes, cada qual com suas funções específicas:

a) motor de inferência;

b) módulo de justificação (sistema de justificação/explicações);

c) sistema de consulta (módulo coletor de dados).

O motor de inferência, também conhecido como máquina de inferência é a parte do NSBC responsável por manipular as informações armazenadas na base de conhecimento. Ribeiro (1987) afirma que esta é a parte que é responsável por buscar as regras ordenadas de uma maneira lógica, avaliá-las e ir direcionando o processo de inferência.

> O processo de inferência está diretamente associado com a estrutura utilizada para o armazenamento do conhecimento na base de conhecimentos. Entretanto, de forma geral, pode-se afirmar que o processo envolve um encadeamento lógico que permita tirar conclusões a partir do conhecimento existente. O motor de inferência é, portanto, o responsável pela ação repetitiva de buscar, analisar e gerar novos conhecimentos. (HEINZLE, 1995, p. 14).

Rabuske (1995) afirma que para a elaboração da máquina de inferência, em alguns casos, podem-se adotar *softwares* já existentes, enquanto em outros casos é necessário elaborá-los. Os dois casos possuem as suas vantagens e desvantagens. A utilização de um software pronto implica em sua aceitação por completo, portanto, devem-se considerar até mesmo itens deficientes para a finalidade do sistema. A opção pela elaboração de um software implicará em custos e tempo adicional, porém, esta segunda opção atenderá melhor as necessidades, pois é desenvolvida para permitir uma melhor adaptação do problema. Enquanto o sistema realiza a inferência das regras, o sistema de consulta é responsável por apresentá-las ao usuário de forma clara e concisa.

Conforme Alexandre (2000), geralmente o usuário é alguém que não participou da elaboração do sistema. Portanto, é natural que não conheça as estruturas do sistema e, que, provavelmente, não esteja familiarizado com as formas de representação do conhecimento utilizadas pelo sistema. Para que os usuários possam acessar com proveito e sem maiores dificuldades, é preciso munir o sistema de recursos para consulta.

O sistema de consulta é o módulo que possui recursos responsáveis por permitirem que os usuários acessem o SBC sem maiores dificuldades. Levando em consideração que os usuários não conhecem a estrutura do SBC, nem suas respectivas formas de representação do conhecimento, o sistema de consulta se faz necessário para o correto acesso ao sistema. Conforme Rabuske (1995), para que os usuários consigam utilizar o sistema com sucesso, é necessário que o mesmo possua recursos para consulta, que são módulos explícitos ou implícitos ao sistema. Durante o processo de consulta, caso o usuário necessite, ele pode solicitar ao sistema que o mesmo lhe explique como chegou àquela conclusão, essa é a responsabilidade do sistema de justificação.

O sistema de justificação, por sua vez, é o módulo que demonstra ao usuário como o SBC chegou à determinada conclusão. Para Heinzle (1995), o sistema de justificação tem a função de esclarecer o usuário a respeito de uma determinada conclusão apresentada ou explicar o motivo de uma pergunta está sendo feita. O módulo de justificação é na verdade um recurso de questionamentos, disponível ao usuário. Heinzle (1995) também afirma que em

muitos domínios de conhecimento, as pessoas não aceitam os resultados quando os mesmos não são devidamente justificados.

> Este módulo interage com o usuário esclarecendo-o de como o sistema chegou a determinada conclusão, ou porque está fazendo determinada pergunta. Utiliza diversos recursos e estruturas próprias para atender ao seu objetivo, mostrando que regra e que fatos foram usados na base de conhecimento, sempre que isso for solicitado por quem usa o sistema. (RIBEIRO, 1987).

Rezende (2003) afirma que o módulo de justificação é um requisito necessário nos SBCs. Este módulo é obrigatório em Sistemas Especialistas (SE) e SBCs, e segundo Rabuske (1995) ele deve possuir, capacidade de responder às seguintes perguntas:

a) como chegou a esta conclusão;

b) por que chegou a esta conclusão;

c) por que não chegou à outra conclusão?

Estes questionamentos são respondidos utilizando como base as respostas das perguntas já respondidas. Desta forma, conforme as perguntas são respondidas, o sistema de justificação poderá responder os questionamentos citados anteriormente.

2.3.3. Quadro negro (Memória de trabalho)

Conforme Rabuske (1995), o quadro-negro é a área de trabalho que o sistema utiliza durante o processo de inferência. Nesta área são armazenadas informações de apoio e suporte ao funcionamento do sistema quando este está racionando. Este lugar na memória é destinado para fazer avaliações das regras que são recuperadas da base de conhecimento para se chegar a uma solução. As informações são gravadas e apagadas de um processo de inferência até se chegar à solução desejada. Embora todos os SE usem o quadro-negro, nem todos o explicitam como o componente do sistema.

Segundo Ribeiro (1987), o quadro negro é formado por um conjunto de três módulos independentes, o primeiro trabalha diretamente com o interpretador de regras e com o justificador. O interpretador é responsável pela validação das condições da regra, relacionando variáveis a essas condições. O segundo é responsável pela escalonação das regras, isto é, qual regra será executada no instante seguinte. O terceiro módulo é responsável pelo armazenamento das possíveis soluções que servirão para formar as respostas para o problema.

2.3.4. Interface

Segundo Rezende (2003), a interface é responsável pela interação entre o SBC e o usuário. Este módulo proporciona a comunicação entre ambas às direções e realizando a intermediação entre a representação interna do sistema e a representação mental do usuário. Geralmente a linguagem de interface é mais abstrata do que a usada na representação do conhecimento do SBC e mais restrita do que as linguagens usadas pelo usuário no seu cotidiano. Quanto mais próxima essa linguagem for da linguagem que o usuário utiliza, mais fácil será utilizar o SBC, em contrapartida, será preciso disponibilizar mais esforço e investimento.

Princípios baseados em teorias cognitivas e semióticas têm sido propostos para projetos de interface, como resultado de pesquisas na área de Interação Humano Computador. Interfaces de SBCs são apropriadas para a aplicação e teste dessas teorias, uma vez que os tipos de interações podem ocorrer entre um usuário e um SBC são os mais variados possíveis, envolvendo desde a realização de questionamentos contextualizados até a apresentação de aplicações sobre as razões das perguntas e das conclusões (PREECE et al, 1994).

2.4 AQUISIÇÃO DO CONHECIMENTO

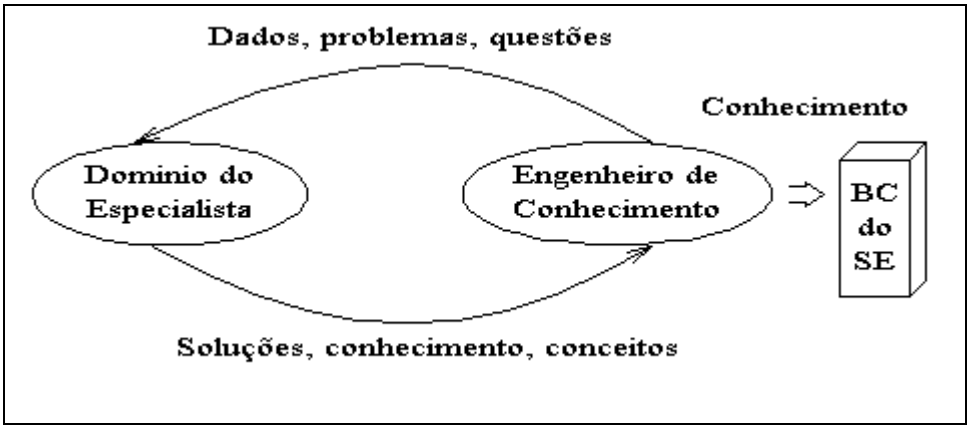
A aquisição do conhecimento é o processo de extração do conhecimento de um especialista e sua respectiva formalização para aplicá-lo em um sistema (GENARO, 1986). Segundo Pacheco (2003), é o trecho mais crítico no desenvolvimento de um sistema desta classe, porque envolve a extração, análise, interpretação para posterior representação do conhecimento. Para extrair o conhecimento, além do especialista, podem-se utilizar periódicos e livros sobre o assunto do sistema.

> Um ponto importante na aquisição do conhecimento é o tratamento de incoerências. Dependendo da forma como o conhecimento é adquirido pode haver erros de aquisição. Estes erros podem resultar da própria natureza do conhecimento, como em dados obtidos através de sensores sujeitos a ruído, ou podem ser gerados pela interface humana existente entre o mundo real e o sistema de representação. (BITTENCOURT, 2001).

O processo de aquisição do conhecimento funciona com o engenheiro do conhecimento repassando os problemas e questões ao especialista que lhe retorna com as

soluções e conceitos, criando assim, a base de conhecimentos, conforme demonstrado pela Figura 2.

Figura 2 - Processo de aquisição de conhecimento



Fonte: Teive (1997).

2.5 REGRAS DE PRODUÇÃO

Representação do conhecimento são os métodos adotados para modular o conhecimento de especialistas de alguma área, e deixá-lo disponível para acesso do usuário do sistema e pela máquina de inferência (LUCHTENBERG, 2000). Para Giarratano e Riley (1994), a representação do conhecimento é extremamente importante por duas razões: primeiro, *shells* para SEs e SBCs são desenvolvidos para certo tipo de representação como regras de produção ou lógica dos predicados. Segundo, o caminho pelo qual um sistema realiza a representação do conhecimento afeta o desenvolvimento, eficiência, velocidade e manutenção do sistema.

O termo "Regras de produção" é utilizado para descrever uma família de sistemas, que tem em comum o fato de serem constituídos de um conjunto de regras, que reúnem condições e ações. A condição é um padrão que determina a aplicabilidade da regra. A ação define o que será realizado quando a regra for aplicada. Segundo Laboratório de Inteligência Artificial (1995), as regras de produção são populares por possuírem as seguintes vantagens:

a) modularidade: cada regra pode ser considerada "pedaço" do conhecimento independente;

b) facilidade de edição: com independência, regras antigas podem ser modificadas e novas regras implementadas;

c) transparência do sistema: garante maior legibilidade da base de conhecimentos.

Geralmente, uma regra de produção segue o padrão apresentado no modelo exposto no Quadro 2, criado para este trabalho.

Quadro 2 - Modelo de regra de produção

```
Se <condição> ENTÃO <ação>
```

O SE de uma regra forma a condição, o "ENTÃO" define o conjunto de ações a serem realizadas se a condição for verdadeira. O Quadro 3 demonstra um exemplo de regra de produção para diagnosticar um possível defeito de uma motocicleta.

Quadro 3 - Exemplo de regra de produção

```
SE diagnóstico parcial = FALHA DO MOTOR DURANTE A ACELERAÇÃO
E tanque de gasolina = LIMPO
E carburador = SEM RESÍDUOS
ENTÃO diagnóstico = FILTRO DE AR COM IMPUREZAS
```

2.6 A FERRAMENTA JESS

Desenvolvida por Ernest Friedman-Hill na Sandia National Laboratories em Livermore, Canadá, a ferramenta *Java Expert System Shell*, popularmente conhecida como JESS, foi resultado de um projeto de pesquisa. No final do ano de 1995, a sua primeira versão foi finalizada. JESS foi originalmente inspirada na s*hell C Language Integrated Production System* (CLIPS), e cresceu com forte influência da linguagem Java (HERZBERG, 2002).

A linguagem utilizada no JESS também é compatível com o CLIPS, desta forma, é possível utilizar *scripts* criados para CLIPS e JESS. Assim como o CLIPS, JESS utiliza o algoritmo Rete para realizar o processamento das regras. Também foram adicionadas algumas capacidades como *backwards chaining* (encadeamento para trás) e a habilidade para

manipular e raciocinar diretamente sobre objetos Java. JESS permite criar e chamar métodos Java sem compilar código Java (HERZBERG, 2002).

2.6.1. O algoritmo RETE

Conforme Pio, Cancian e França (2007), este algoritmo é caracterizado por organizar as regras numa árvore, de modo que sejam divididas em padrões, assim, regras semelhantes percorrerão o mesmo ramo da árvore enquanto possuírem cláusulas iguais. Tal organização é mais onerosa em termos de memória, mas proporciona grandes melhorias na velocidade de verificação das regras. O motor de inferência JESS permite tanto o encadeamento regressivo quanto progressivo.

O algoritmo Rete é um método eficiente para a comparação de uma grande coleção de padrões com uma grande coleção de objetos. Ele encontra todos os objetos que corresponde a cada padrão. O algoritmo foi desenvolvido para uso em interpretadores de sistemas de produção, e que tem sido utilizado para sistemas contendo, desde algumas centenas a mais de um milhão de padrões e objetos (FORGY, 1982).

2.7 A SHELL JESS

Nesta seção serão demonstradas as características principais do JESS, conforme Herzberg (2002).

2.7.1 Tipos de variáveis

Em JESS existem tipos de variáveis para melhor utilizar a ferramenta. O átomo é a essência do conceito da linguagem JESS, e são como identificadores em outras linguagens. Um átomo JESS não deve começar com um número e pode conter letras, números e a seguinte pontuação: $*=+<>_?#.. Átomos são "*case sensitive*", ou seja, "FOO", "Foo" e "foo" são todos símbolos diferentes. Existem três símbolos que especiais para o JESS: "*nil*" que é similar ao valor nulo de Java e *true* e *false* , que são valores do tipo *boolean*.

Os números são utilizados pelo JESS através das funções Java "*java.lang.Double.parseInt"* e "*java.lang.Integer.parseDouble"* para analisar os números inteiros e ponto flutuante, respectivamente. Os seguintes números são válidos: 3,4,5.643, 6.0E4 e 1D. As variáveis do tipo *String* são denotadas usando aspas duplas ("). O caractere barra invertida (\), pode ser usado para inserir o caractere aspas (") dentro do texto, para que o *JESS* não interprete como o final da *string*.

Outro tipo de variável são as listas, que é uma das unidades fundamentais na sintaxe do JESS. A lista consiste em um conjunto de parênteses e zero ou mais átomos, números, strings ou outras listas. O primeiro elemento da lista é chamado de cabeça de lista. As seguintes listas são válidas: (+3 2), (a b c), ("*Hello, World*"), (), (*deftemplate foo (slot bar)*).

2.7.2 Funções

Como em *List Processing Language* (LISP*)*, todo código em JESS, como estruturas de controle, atribuições e chamadas de *procedures*, tomam a forma de uma chamada de função. Chamadas de função em JESS são simplesmente listas, que usam uma notação de prefixo. Uma lista cuja cabeça é um átimo como o nome de uma função existente, pode ser a chamada função. Por exemplo, uma expressão que usa a função (+) para somar os números 2 e 3, seria escrita (+ 2 3). Quando interpretado, o valor desta expressão será o número 5, e não uma lista que contém o único elemento 5.

JESS possui funções embutidas para a execução de cálculos matemáticos, manipulação de strings, controle de programas, fornecendo a *Application Programming Interface* (API) do Java. Uma das funções mais utilizadas é o *printout*, que serve para enviar o texto para a saída padrão do JESS, ou para um arquivo. No Quadro 4 podemos verificar um exemplo de função JESS deste trabalho.

Quadro 4 - Exemplo de função JESS

```
JESS> (printout t "Motocicleta: CG Titan 150 cc KS" crlf))
Motocicleta: CG Titan 150 cc KS
```

Outra função bastante usual é a *batch*, que serve para carregar um arquivo de código JESS. A função *batch* é chamada da forma exposta no Quadro 5.

Quadro 5 - Exemplo de carga de arquivo

```
batch exemplo.txt
```

Um exemplo de utilização prática da função *batch* seria para carregar as regras que a *shell* do protótipo possuirá.

2.7.3 Variáveis

As variáveis em JESS são átomos que começam com o caractere ponto de interrogação (?), sendo que este é parte do nome da variável. Uma variável normal pode se referir a um único átomo, número ou string. Uma variável cujo primeiro caractere é o cifrão ($), como por exemplo: $?X. É uma variável multivalorada, que pode se referir a um tipo especial de lista, que permite mais de um campo. A variável pode ser definida através da função *bind*. O Quadro 6 demonstra um exemplo de variável definida através da função *bind*.

Quadro 6 - Variável definida com a função *bind*

```
JESS> (bind ?x "Motocicleta")
"Motocicleta"
```

Estas listas com vários campos são geralmente criadas utilizando funções especiais como *create*$, e depois podem ser atribuídas a uma multivariável. O Quadro 7 apresenta um exemplo de lista que armazena peças de uma motocicleta, criada com a função especial *create$*.

Quadro 7 - Lista utilizando *create*$

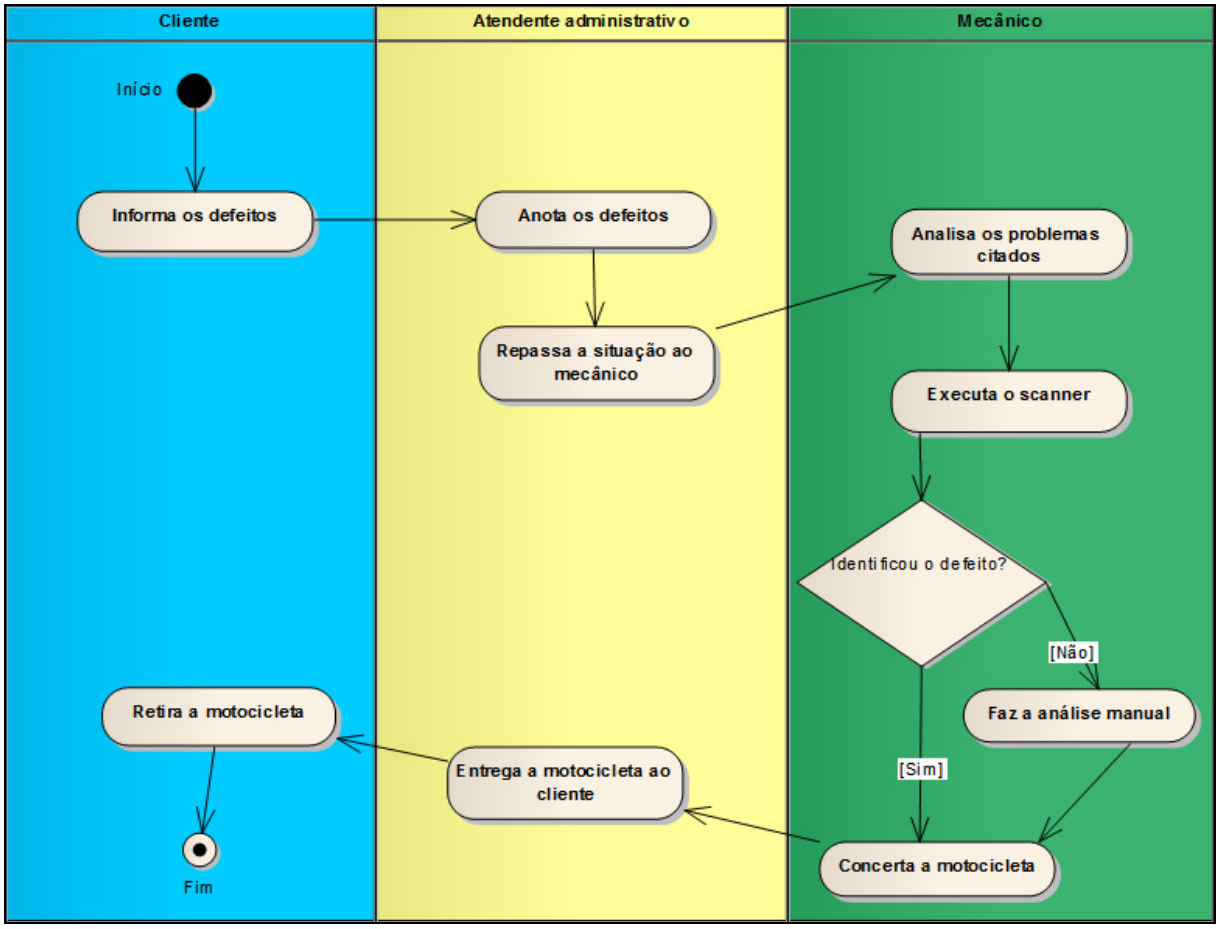
```
JESS> (bind $?motocicleta (create$ Motor "Carburador" "Pistão"))
(Motor "Carburador" "Pistão")
```

2.8 SISTEMA ATUAL

Para obter informações a respeito do processo de uma oficina mecânica, foram

realizadas conversas informais com três funcionários de três estabelecimentos distintos. Através destas conversas foi possível identificar que o processo de uma oficina mecânica começa a partir do momento que o cliente chega ao estabelecimento, após isso, um atendente administrativo faz o cadastro e questiona quais os defeitos existentes na motocicleta. Ao obter todos os defeitos mecânicos, o atendente repassa estas informações ao mecânico que analisará a motocicleta a fim de identificar as peças que devem ser trocadas. A Figura 3 apresenta um diagrama de atividades com o fluxo de funcionamento do processo.

Figura 3 - Diagrama de atividades atual



O diagrama acima demonstra o fluxo de uma oficina mecânica, onde o processo se inicia quando um cliente chega ao estabelecimento e informa os defeitos ao atendente. O atendente, por sua vez, anota dos defeitos da motocicleta e repassa estas informações ao mecânico, que analisará o caso e executará o *scanner*. Se o defeito for identificado, é iniciado o conserto da motocicleta, caso contrário será realizada uma análise manual para identificar o defeito. Por fim, depois de o mecânico realizar o conserto das peças, o atendente entrega a motocicleta ao cliente, que finaliza o ciclo retirando-a do estabelecimento.

Conforme a entrevista informal presente no Anexo A, atualmente uma das ferramentas

mais utilizadas na identificação de defeitos são *scanners* que fazem a leitura do motor e identificam defeitos hidráulicos como falhas e fissuras presentes no pistão e motor. Contudo, existem defeitos que esta ferramenta não consegue identificar. E são justamente estes defeitos que tornam o processo de diagnóstico moroso, pois, é necessário desmontar e testar, individualmente, várias partes da motocicleta. A Figura 4 apresenta um modelo de *scanner* existente hoje no mercado.

Figura 4 - *Scanner* MotoDiag

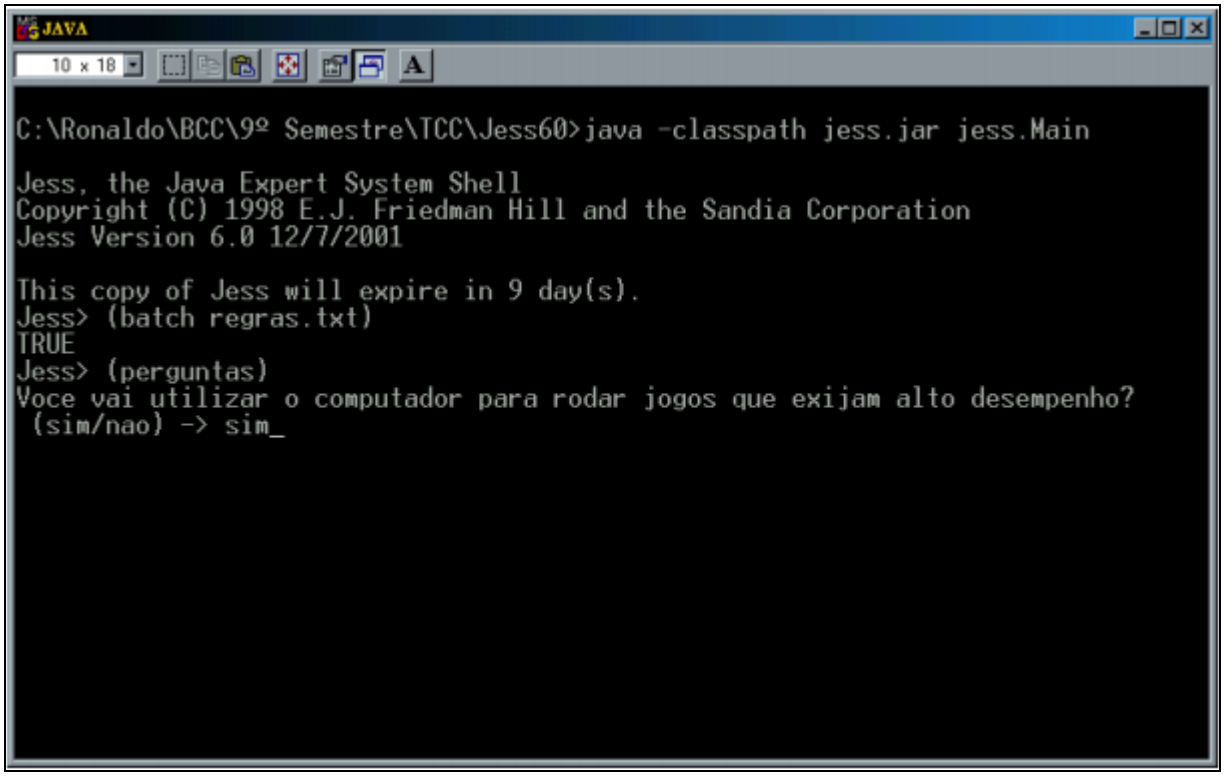
Fonte: Chiptronic (2013).

2.9 TRABALHOS CORRELATOS

Os trabalhos correlatos pesquisados são as monografias realizadas pelos alunos Ronaldo César Schork Júnior, Nívea Maria Pacheco e Elaine Starke, onde os trabalhos dos dois primeiros citados foram para a conclusão do curso de Ciência da Computação e o terceiro para a conclusão do curso de Sistemas de Informação, todos na Universidade Regional de Blumenau (FURB).

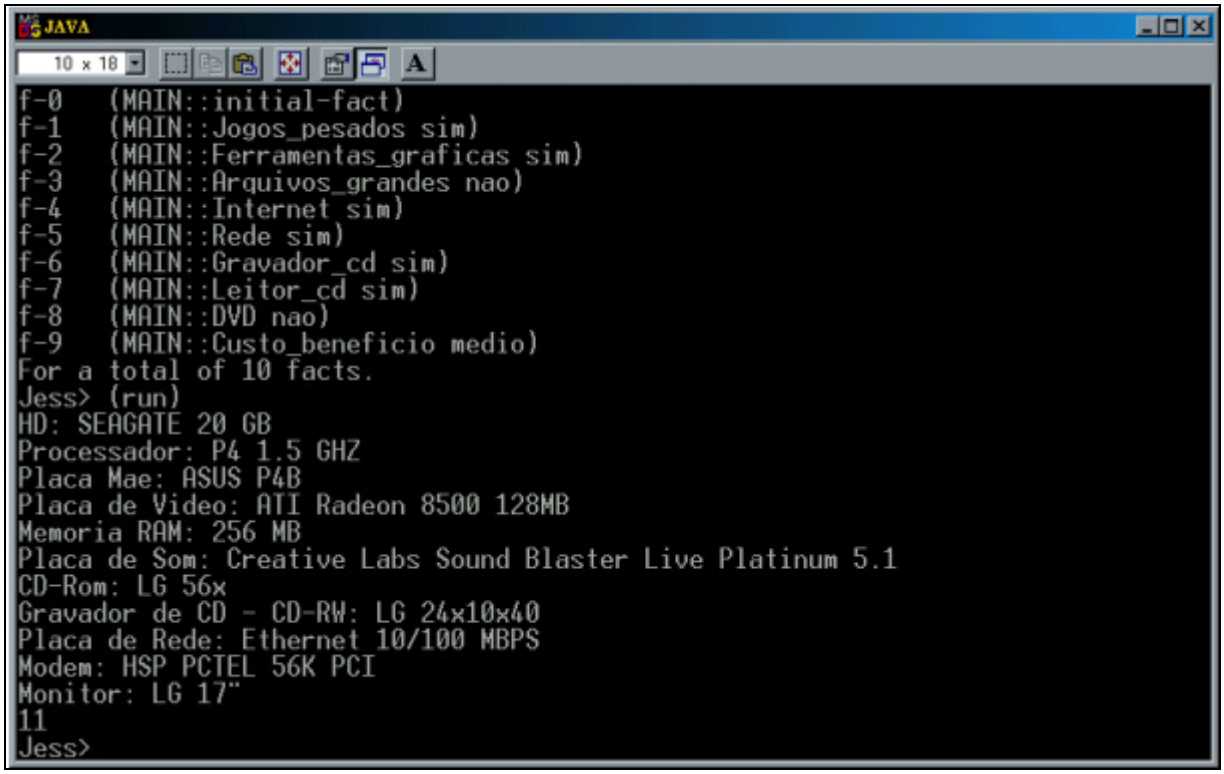
O trabalho de Schork Júnior (2002) foi desenvolver um protótipo de um sistema especialista para a seleção de microcomputadores, para auxiliar o usuário a escolher o computador que mais atende às suas necessidades. O protótipo desenvolvido no trabalho de Schork Júnior (2002) utilizou a ferramenta *shell* JESS, a qual também será utilizada neste trabalho. A Figura 5 e a Figura 6 demonstram algumas das principais telas do trabalho citado neste parágrafo.

Figura 5 - Tela principal – Trabalho Schork Júnior



Fonte: Schork Júnior (2002).

Figura 6 - Resultado na tela – Trabalho Schork Júnior

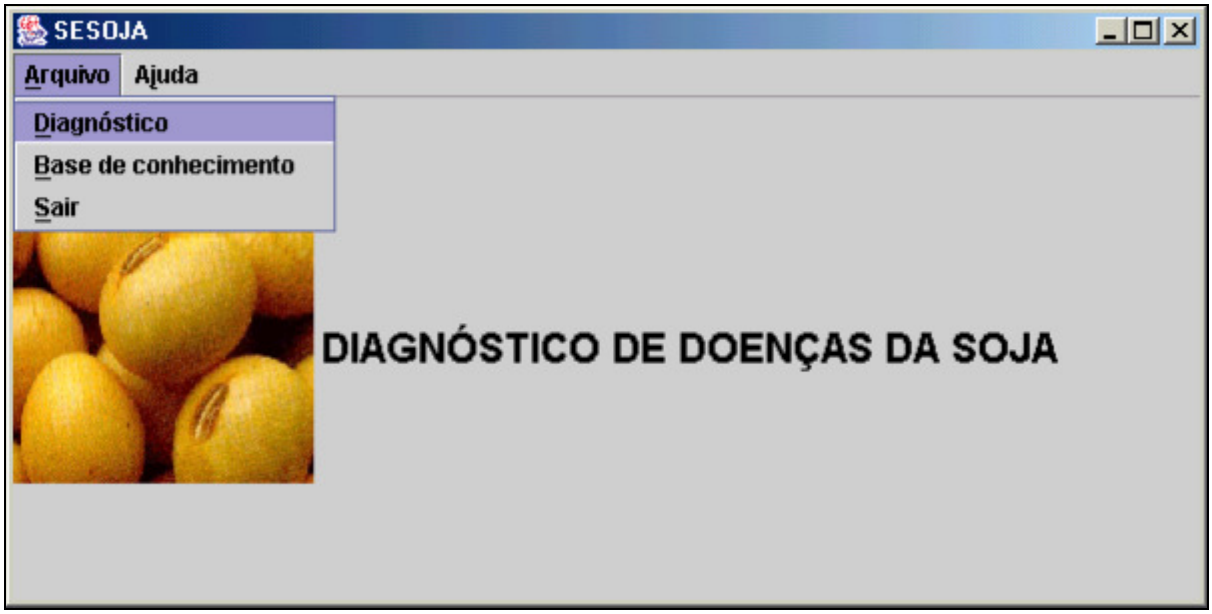


Fonte: Schork Júnior (2002).

Schork Júnior (2002) desenvolveu seu trabalho utilizando arquivos de texto para a criação das regras e fatos que serão analisados pela ferramenta JESS. Ao executar a ferramenta é criado um *prompt* para comandos, ou seja, um ambiente para a carga das regras através do comando *batch* e posterior consulta. Os resultados obtidos através deste trabalho possibilitam que o usuário identifique quais as configurações recomendadas para o seu computador, conforme as suas necessidades.

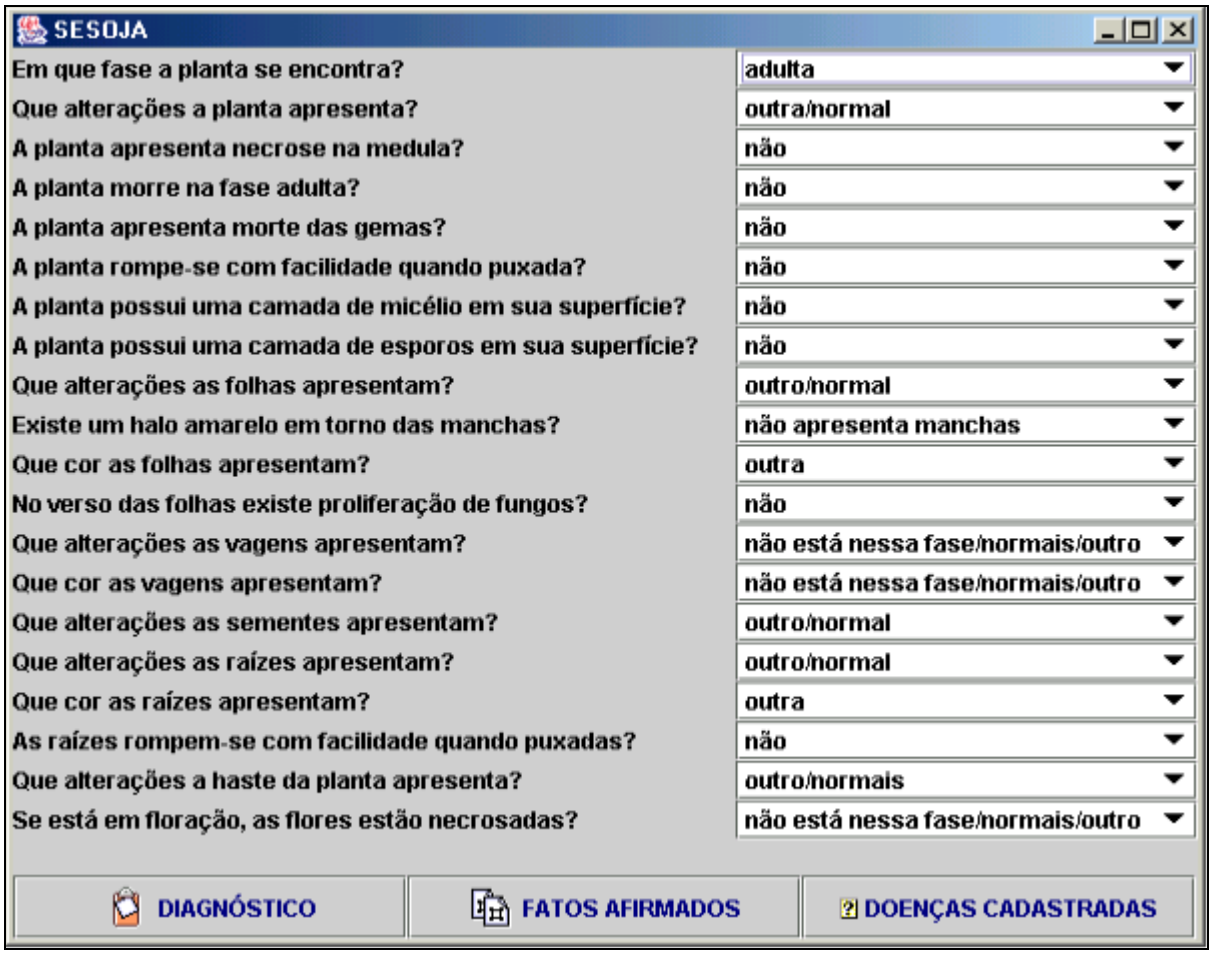
Pacheco (2003) criou um protótipo de um sistema especialista para auxílio na identificação de doenças da soja, a fim de servir de apoio ao agricultor. Como no trabalho de Schork Júnior (2002), a ferramenta utilizada foi a s*hell* JESS. Além desta ferramenta *shell*, Pacheco (2003) também utilizou a *Integrated Drive Eletronics* (IDE) JCreator LE para desenvolver o protótipo, que foi escrito em Java. A Figura 7 e a Figura 8 apresentam algumas das principais telas do trabalho de Pacheco (2003). A Figura 7 apresenta o menu principal do protótipo, enquanto a Figura 8 demonstra a tela de questionário, onde após responder as perguntas, será informado o diagnóstico da planta.

Figura 7 - Tela inicial do sistema – Trabalho Pacheco



Fonte: Pacheco (2003).

Figura 8 - Diagnóstico da planta – Trabalho Pacheco

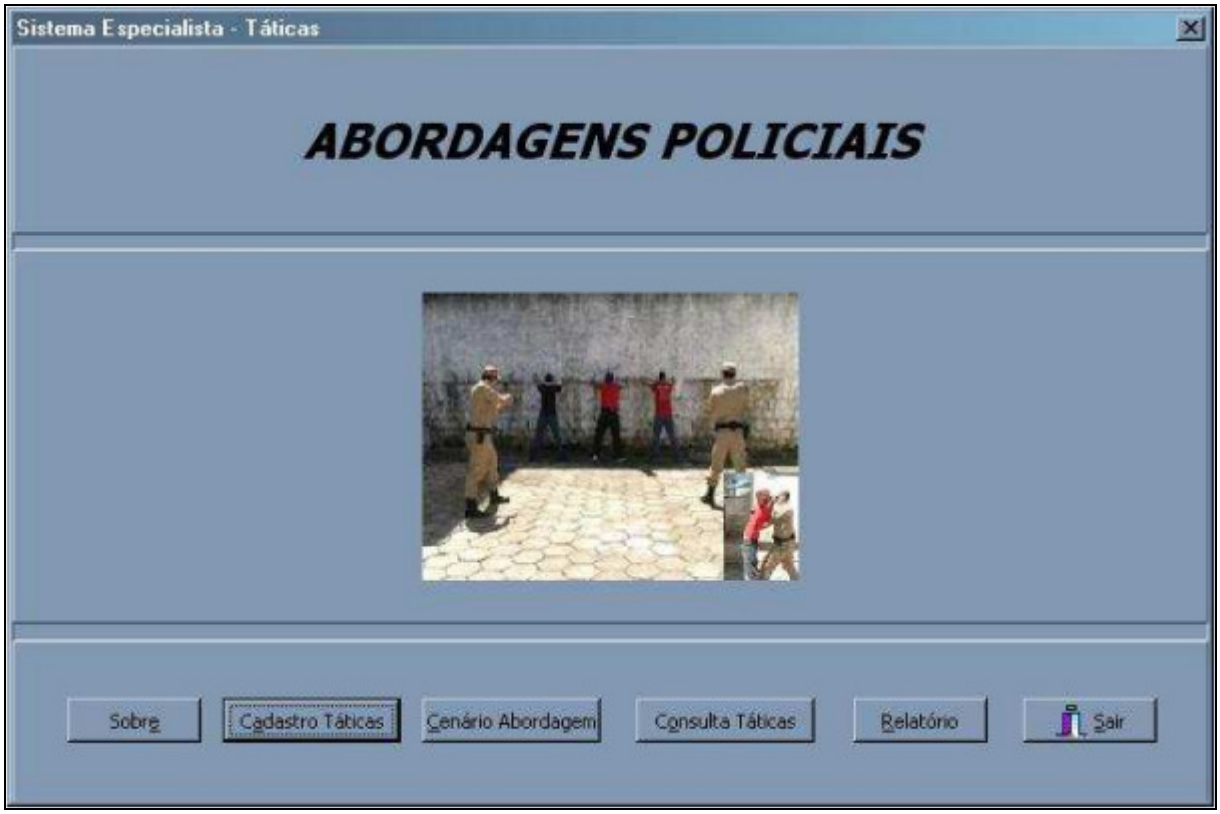


Fonte: Pacheco (2003).

Pacheco (2003) utilizou a IDE JCreator LE para o desenvolvimento de seu protótipo o que permitiu a apresentação de uma interface mais agradável aos usuários, pois as respostas dos questionamentos podem ser selecionadas em campos do tipo *lookup*. As regras, também armazenadas em arquivos de texto, foram todas carregadas (através do comando *batch*) automaticamente. Os resultados obtidos por Pacheco (2003) facilitaram, para usuários não especialistas, as formas de identificar qual a doença que a planta possui.

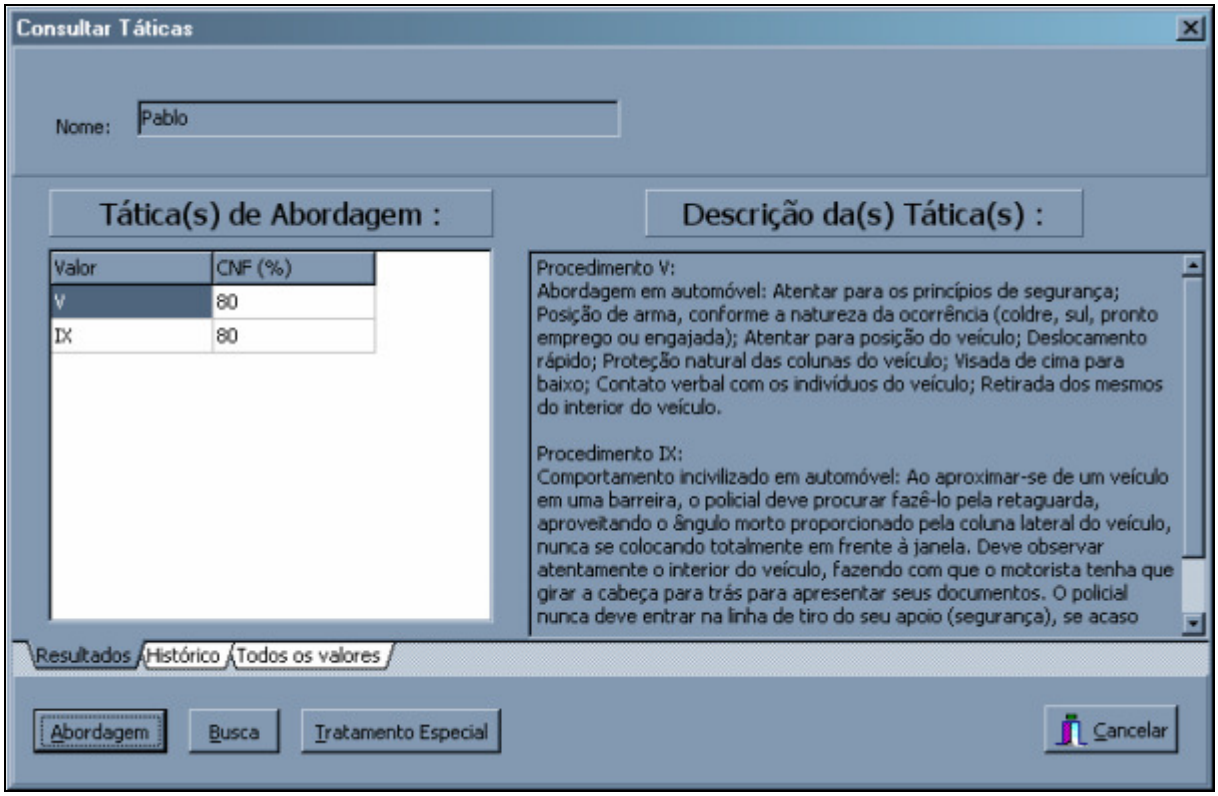
O trabalho de Starke (2007) foi desenvolver um sistema especialista de apoio à polícia militar de Santa Catarina na regional de Blumenau. Este SE tem a finalidade de definir qual a melhor tática de abordagem policial à determinada situação. Diferente dos dois primeiros trabalhos correlatos, este trabalho utilizar outras tecnologias e ferramentas. O ambiente de desenvolvimento escolhido foi o Delphi 3.0 e a *shell* para desenvolvimento do SE foi o *Expert* SINTA *Shell*, na versão 1.1. A Figura 9 e a Figura 10 demonstram as principais telas deste trabalho. Na Figura 9 pode-se visualizar a tela principal do trabalho de Starke (2007), enquanto que na Figura 10 é apresentada a tela de consultas de táticas de abordagem.

Figura 9 - Tela principal – Trabalho Starke



Fonte: Starke (2007).

Figura 10 - Consultar táticas – Trabalho Starke



Fonte: Starke (2007).

Starke (2007) utilizou, além de uma linguagem diferente dos trabalhos de Schork Júnior (2002) e Pacheco (2003), também adotou um banco de dados para o armazenamento de informações do cadastro de policiais. O banco de dados é o Paradox, que é inserido junto ao Delphi. Os resultados de Starke (2007) possibilitaram a modernização no sistema de consulta às táticas de abordagem no 10º Batalhão da Polícia Militar de Santa Catarina.

Ao fazer comparações entre os três trabalhos, pode-se observar que os três adotaram áreas da I.A. para produzirem seus trabalhos. Schork Júnior (2002) e Pacheco (2003) produziram sistemas especialistas, enquanto Starke (2007) desenvolveu um sistema baseado em conhecimento. O Quadro 8 demonstra um comparativo entre os trabalhos correlatos.

Quadro 8 - Comparativo trabalhos correlatos

Tecnologias e ferramentas	Schork Júnior (2002)	Pacheco (2003)	Starke (2007)
Área de aplicação	Seleção de computadores	Diagnóstico de doenças da soja	Seleção de táticas de abordagem
Área de conhecimento da IA	Sistemas especialistas	Sistemas especialistas	Sistemas especialistas
Ambiente/Linguagem	*JESS / Java*	*Jcreator LE / Java*	*Delphi*
Ferramenta *shell*	*JESS*	*JESS*	*Expert SINTA*
Banco de dados	Arquivos de texto	Arquivos de texto	Paradox

3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo são apresentadas as características do sistema desenvolvido tais como o levantamento de informações, a especificação de requisitos funcionais e não funcionais, o diagrama de caso de uso (UC) e o diagrama de entidade e relacionamento (DER). São descritas também as técnicas e ferramentas utilizadas no processo de implementação, a operacionalidade do aplicativo e os resultados obtidos.

3.1 LEVANTAMENTO DE INFORMAÇÕES

O levantamento de informações foi realizado através de visitas à oficina mecânica Moto Peças Kalarrari, estabelecida na cidade de Gaspar, estado de Santa Catarina, onde foi possível realizar uma entrevista com um dos mecânicos do estabelecimento. Com isso, foi possível identificar que o maior diferencial que uma oficina mecânica pode apresentar, além da qualidade nos serviços e peças, é a agilidade de atendimento. E as duas maiores dificuldades presentes no estabelecimento é o tempo de atendimento aos clientes e a forma de armazenamento de informações administrativas.

Feito o levantamento de informações, foi identificado que o tempo de atendimento aos clientes é composto por duas partes: a primeira parte é o atendimento direto ao cliente e a segunda é a partir do momento que os mecânicos começam a procurar o defeito da motocicleta para consertá-la. Após identificar as duas situações descritas acima, foram definidos os requisitos funcionais e não funcionais e desenhada a modelagem. Após isto, foi desenvolvido um protótipo de um sistema baseado em conhecimento, que permite ao usuário gerenciar as informações administrativas da oficina e acessar a um módulo de consultas onde após responder algumas perguntas sobre os defeitos da motocicleta, será apresentada uma lista de peças a serem verificadas.

A partir do protótipo o usuário poderá, além de fazer diversos cadastros, emitir relatórios para melhorar o controle administrativo da oficina. Estes relatórios apresentam informações de clientes, motocicletas, contas a pagar e contas a receber. O módulo de consultas adota tecnologias de inteligência artificial, para diagnosticar quais as peças estão com defeito. Inicialmente o protótipo possui uma base de conhecimento com algumas regras,

porém, é permitido ao usuário desativá-las e/ou criar novas regras.

Para a construção do protótipo utilizou-se a linguagem Java e os seguintes recursos:

a) NetBeans IDE 7.4, como plataforma de desenvolvimento;

b) MySQL Server 5.6, como banco de dados para armazenamento dos dados no servidor;

c) MySQL Workbench 6.0 CE, para criação de tabelas e *functions* do banco de dados;

d) JESS 7, é a *shell* utilizada para a criação do SBC.

3.2 ESPECIFICAÇÃO

Esta seção apresenta além dos requisitos funcionais e não funcionais, os diagramas de caso de uso e o modelo de entidade relacionamento do módulo desenvolvido. Para gerar os diagramas foi utilizada a ferramenta Enterprise Architect (EA) e para o Diagrama de Entidade e Relacionamento (DER), utilizou-se a ferramenta DB Designer.

3.2.1. Requisitos do sistema

O Quadro 9 apresenta os requisitos funcionais previstos para o sistema e sua rastreabilidade, ou seja, vinculação com o(s) caso(s) de uso associado(s).

Quadro 9 - Requisitos Funcionais

Requisitos Funcionais	Caso de Uso	Atores
RF01: O protótipo deve permitir manter clientes.	UC01	Usuário/ Administrador
RF02: O protótipo deve permitir manter motocicletas.	UC02	Usuário/ Administrador
RF03: O protótipo deve permitir manter contas a pagar.	UC03	Usuário/ Administrador
RF04: O protótipo deve permitir manter contas a receber.	UC04	Usuário/ Administrador

RF05: O protótipo deve permitir que o usuário realize pesquisas no SBC para a identificação dos defeitos.	UC07	Usuário/ Administrador
RF06: O protótipo deve permitir que o usuário salve as pesquisas realizadas no SBC.	UC08	Usuário/ Administrador
RF07: O protótipo deve permitir a impressão de relatórios de contas a pagar.	UC05	Usuário/ Administrador
RF08: O protótipo deve permitir o cadastro de um histórico sobre a motocicleta do cliente.	UC06	Usuário/ Administrador
RF09: O protótipo deve permitir a impressão de relatórios de contas a receber.	UC09	Usuário/ Administrador
RF10: O protótipo deve permitir a impressão de relatórios de clientes atendidos no período.	UC10	Usuário/ Administrador
RF11: O protótipo deve permitir a impressão de relatórios de informações sobre as motocicletas.	UC11	Usuário/ Administrador
RF12: O protótipo deve permitir manter a base de conhecimentos do sistema baseado em conhecimento.	UC12	Administrador

O Quadro 10 lista os requisitos não funcionais previstos para o sistema.

Quadro 10 - Requisitos não funcionais

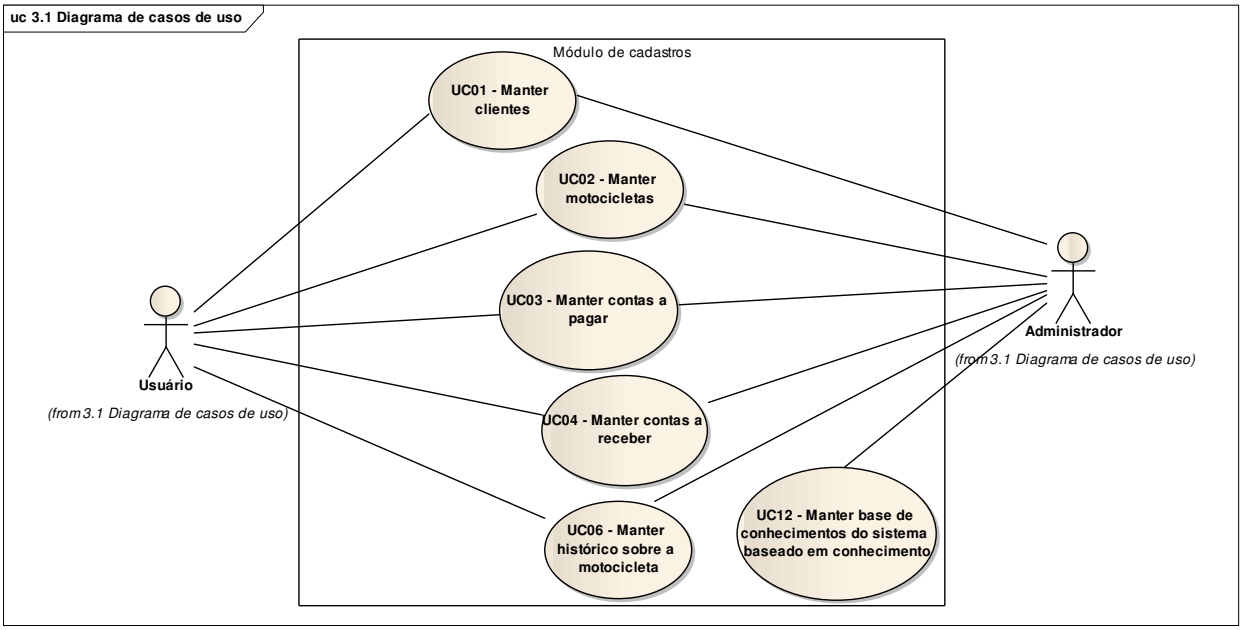
Requisitos Não Funcionais
RNF01: O protótipo deverá utilizar banco de dados MySQL Server 5.6.
RNF02: O ambiente de desenvolvimento será Java.
RNF03: A *shell* utilizada para a criação do SBC será JESS.
RNF04: A representação do conhecimento será realizada através de regras de produção.

3.2.2. Diagrama de caso de uso

Esta subseção apresenta os diagramas de caso de uso. Cada diagrama representa um módulo do protótipo. Para melhor entendimento do projeto, o detalhamento dos principais casos de uso (UC01, UC02, UC07 e UC12), encontra-se no Apêndice A. A Figura 11

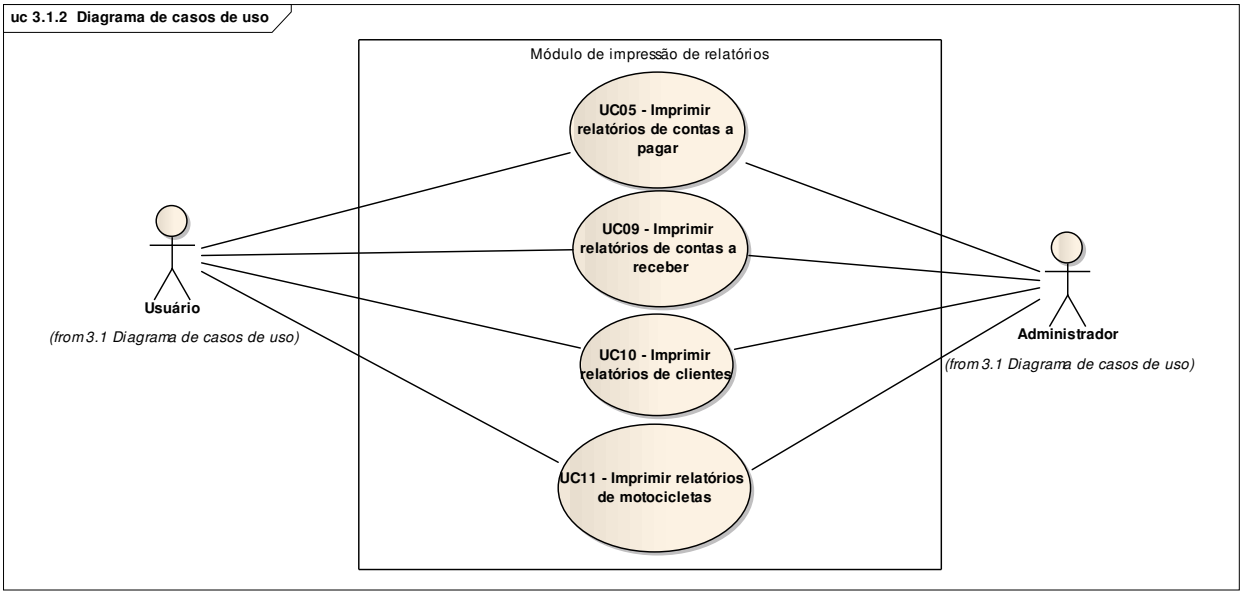
demonstra o diagrama de casos de uso módulo de cadastros.

Figura 11 - Diagrama de casos de uso – Módulo de cadastros



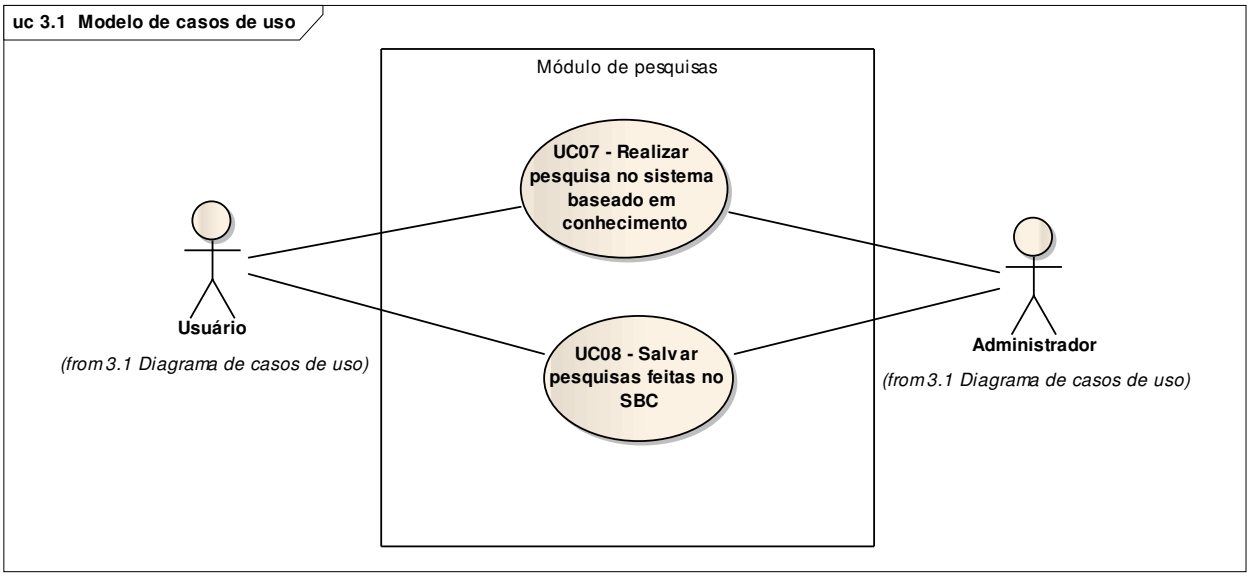
A Figura 12 apresenta o diagrama de casos de uso do módulo de impressão de relatórios.

Figura 12 - Diagrama de casos de uso – Módulo de impressão de relatórios



A Figura 13 apresenta o diagrama de casos de uso do módulo de pesquisas.

Figura 13 - Diagrama de casos de uso – Módulo de pesquisas



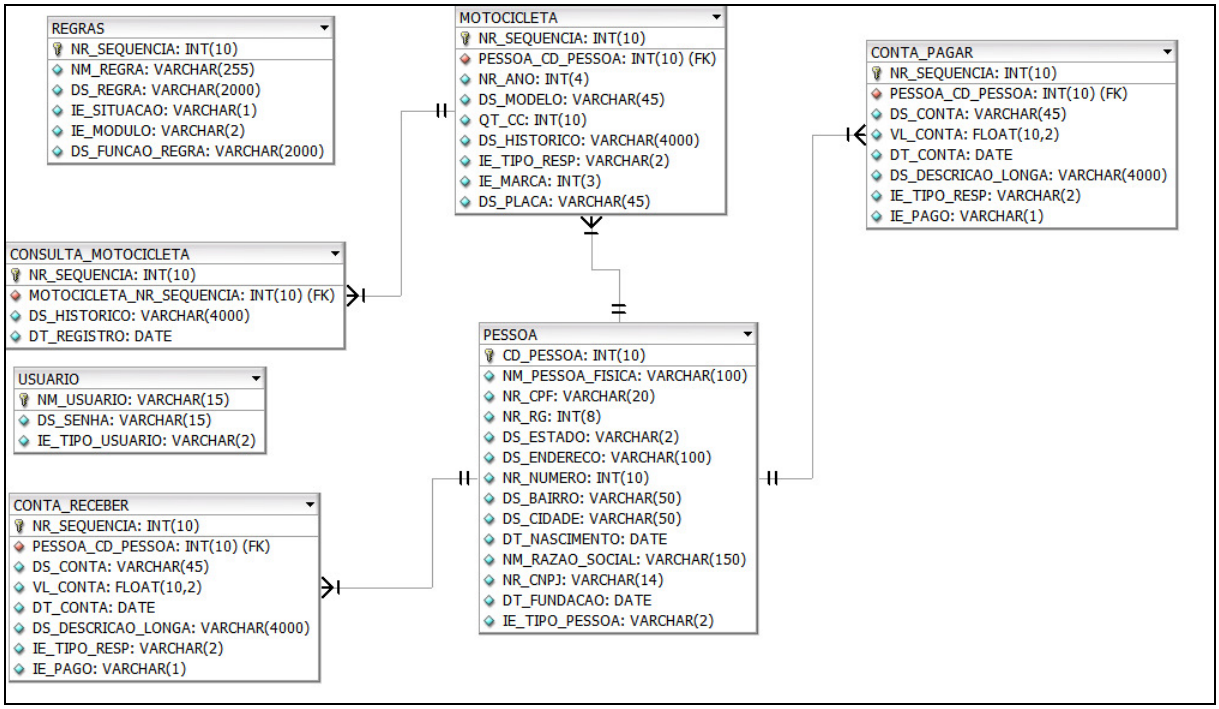
As funções dos casos de uso consistem em operações de três tipos, sendo elas operações de gerenciamento da área administrativa, consultas ao SBC e impressão de relatórios com informações administrativas.

3.2.3. Modelo Entidade Relacionamento

O modelo de entidade-relacionamento, conforme Sanches (2005), baseia-se em uma percepção de um mundo real, aonde se consiste uma coleção de objetos, denominados entidades, e o relacionamento entre os mesmos. Ainda segundo o autor, cada objeto se difere um do outro através de um conjunto específico de atributos que cada um possui.

Na Figura 14 apresenta-se o DER que representa as entidades que serão persistidas no banco de dados. O dicionário de dados é apresentado no Apêndice B.

Figura 14 - Diagrama de Entidade e Relacionamento



A função de cada entidade está descrita a seguir:

a) `regras`: responsável por armazenar as regras que serão utilizadas pelo SBC;

b) `pessoa`: responsável por armazenar os clientes;

c) `conta_pagar`: responsável por armazenar as contas a pagar;

d) `conta_receber`: responsável por armazenar as contas a receber;

e) `motocicleta`: responsável por armazenar as motocicletas;

f) `consulta_motocicleta`: responsável por armazenar as consultas realizadas nas motocicletas;

g) `usuario`: responsável por armazenar os usuários que poderão acessar o sistema.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1. Técnicas e ferramentas utilizadas

Para o desenvolvimento do protótipo foi utilizada a ferramenta NetBeans IDE 7.4, que permite ao desenvolvedor criar o leiaute das telas do sistema apenas arrastando os componentes com o mouse. A ferramenta NetBeans também nos permite, além da criação do leiaute das telas, o desenvolvimento das funcionalidades do protótipo. Tendo em vista que todos os módulos do aplicativo foram desenvolvidos com a linguagem Java, a Figura 15 apresenta um trecho para exemplo de sua utilização, o qual é utilizado para armazenar uma conta no banco de dados.

Figura 15 - Java

```
public void insertCP(String dsConta, Float vlConta, Date dtConta,
                     String dsDescLonga, int cdPfResp, int cdPjResp,
                     String ieTipoResp, boolean iePago) throws SQLException {
    try {
       // cria um preparedStatement

       if (ieTipoRespConta.getSelectedIndex() == 1) {
          PreparedStatement insert = getMenu().getLogin().getConexao().prepareStatement(""
                + "insert into conta_pagar (ds_conta,"
                + "                         vl_conta,"
                + "                         dt_conta,"
                + "                         ds_descricao_longa,"
                + "                         cd_pessoa_fisica_resp_cp,"
                + "                         ie_tipo_resp,"
                + "                         ie_pago) "
                + "                 values (?,?,?,?,?,?,?)");

          insert.setString(1, dsConta);
          insert.setFloat(2, vlConta);
          insert.setDate(3, dtConta);
          insert.setString(4, dsDescLonga);
          insert.setInt(5, cdPfResp);
          insert.setString(6, ieTipoResp);
          if (iePago) {
             insert.setString(7, "S");
          } else {
             insert.setString(7, "N");
          }

          insert.execute();
          insert.close();

          nrSeqConta.setText(obterMaxCodigoCP());
```

Os registros do protótipo serão todos armazenados em um Sistema Gerenciador de Banco de Dados (SGBD) chamado MySQL. Para isso, optou-se pela versão 5.6 do SGBD e foi utilizado o MySQL Workbench para gerenciá-lo. O MySQL Workbench é gerido pela Oracle Corporation e na Figura 17 é possível ver a ferramenta a tela de linhas de comando

aberta e com uma consulta realizada.

Para a criação da inteligência artificial do SBC, foi utilizada a *shell* JESS. Esta *shell*, por interagir com o Java, teve as suas bibliotecas importadas para dentro do projeto criado no NetBeans. Para o correto funcionamento dos módulos que utilizam inteligência artificial, a base de conhecimentos do protótipo deve possuir regras para o motor de inferência utilizá-las para diagnosticar os defeitos mecânicos. Cada estabelecimento será responsável pela criação e manutenção das suas regras, através do módulo de gerência de conhecimento, porém, o protótipo já possui uma carga inicial de regras (o conjunto completo de regras se encontra no Anexo C). A Figura 16 demonstra um exemplo de três regras do JESS, que serão utilizadas dentro do protótipo.

Figura 16 - Regra de produção JESS

```
(defrule regra-1
        (freio_baixo sim)
        (freio_pesado nao)
        =>
        (assert (freio_nao_funciona sim))
        (printout t "Manete/Oleo de freio/Pastilhas" crlf))

(defrule regra-2
        (freio_pesado sim)
        (freio_baixo nao)
        =>
        (assert (freio_nao_funciona sim))
        (printout t "Cabos/Patin de freio" crlf))

(defrule regra-3
        (freio_nao_funciona nao)
        =>
        (printout t "Cabos/Pastilhas/Oleo de freio/Manete" crlf))
```

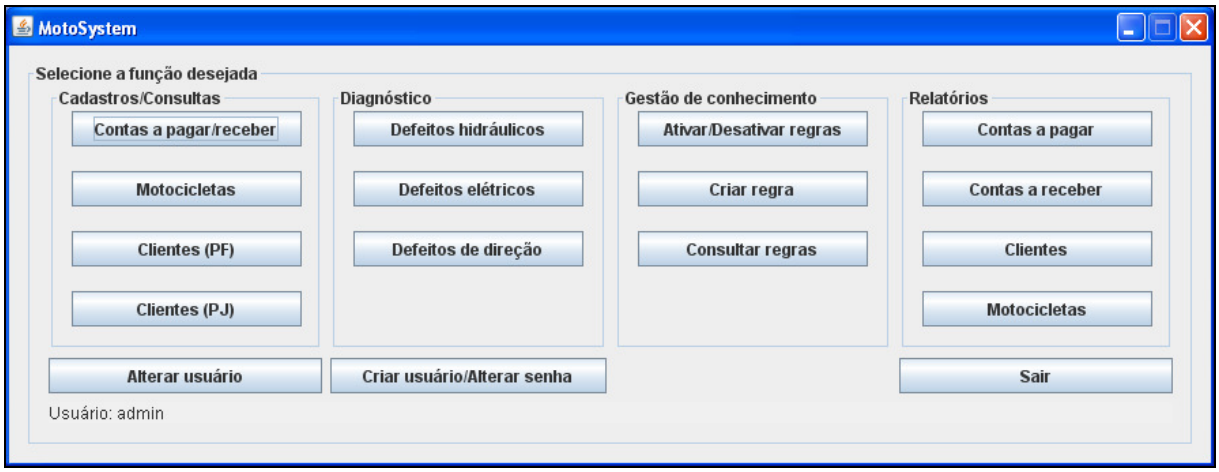
3.3.2. Operacionalidade da implementação

Nesta subseção apresentam-se as telas do protótipo denominado MotoSystem, como uma apresentação sobre suas funcionalidades, bem como trecho de códigos relevantes para o entendimento de algumas rotinas. A primeira ação a ser realizada é a autenticação através de um usuário e senha e ao sair da tela de autenticação é apresentado o menu principal (Figura 17), onde o usuário poderá selecionar qual função deseja realizar. Para selecionar alguma função, basta clicar sobre o botão da respectiva opção.

Para manter a tela organizada, as funções foram todas divididas em pequenos painéis, onde cada um deles representa um módulo do sistema, sendo eles, os cadastros, as consultas de defeitos, a gerência de conhecimento e os relatórios. Ainda nesta tela existe um pequeno campo onde é informado qual usuário está acessando o protótipo, uma opção para trocar de

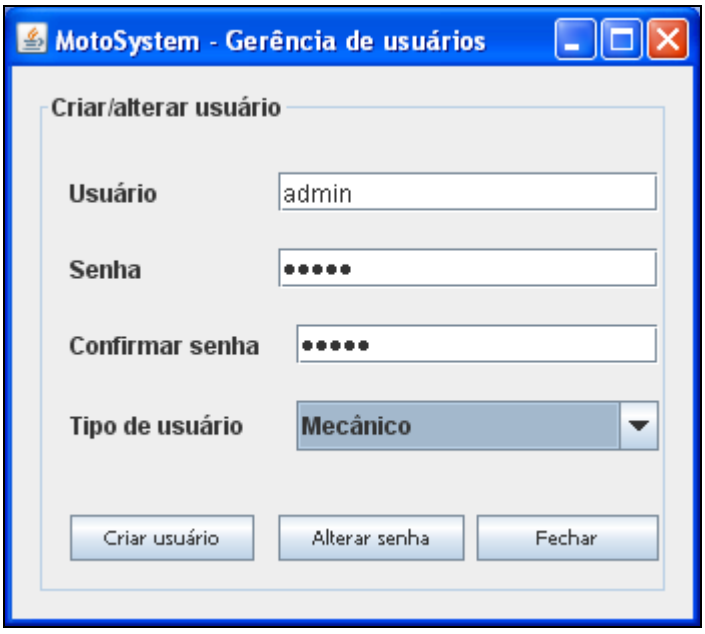
usuário e uma opção para cadastrar novos usuários.

Figura 17 - Tela de menu principal



Para realizar a criação de um usuário no protótipo, o usuário deverá, a partir do menu principal, selecionar a opção “Criar/alterar usuário”, que será apresentada a tela para alterar ou criar novos usuários. A Figura 18 demonstra a tela citada.

Figura 18 - Gerência de usuários



A partir da tela de menu principal, o usuário poderá acessar o módulo do protótipo que será responsável por diagnosticar os defeitos mecânicos das motocicletas, atendendo ao requisito funcional RF05. A fim de manter os diagnósticos mais organizados e concisos, os defeitos foram separados em três grupos, sendo eles os hidráulicos (Figura 19), os elétricos (Figura 20) e os de direção (Figura 21).

Figura 19 - Diagnóstico hidráulico

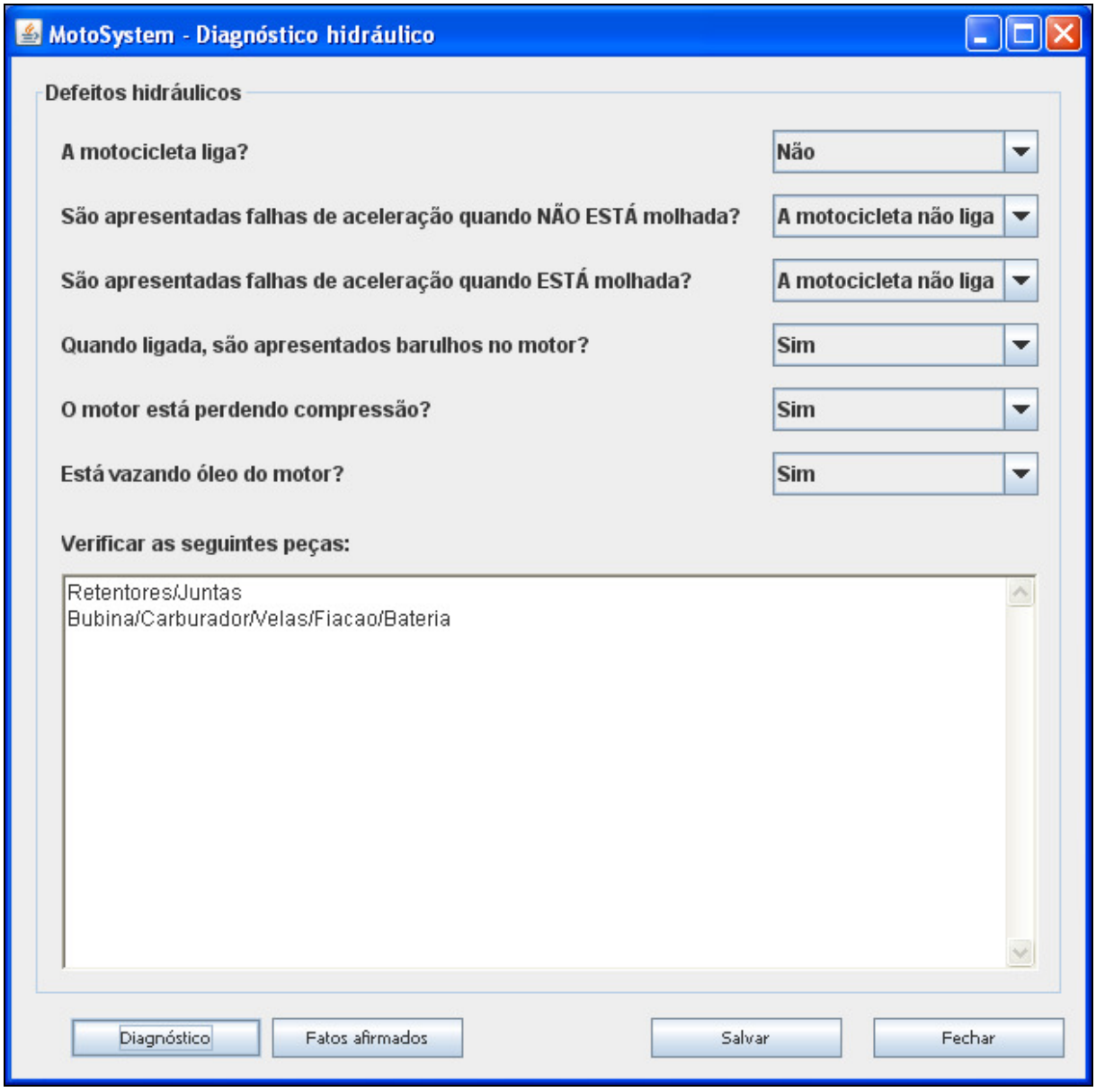


Figura 20 - Diagnóstico elétrico

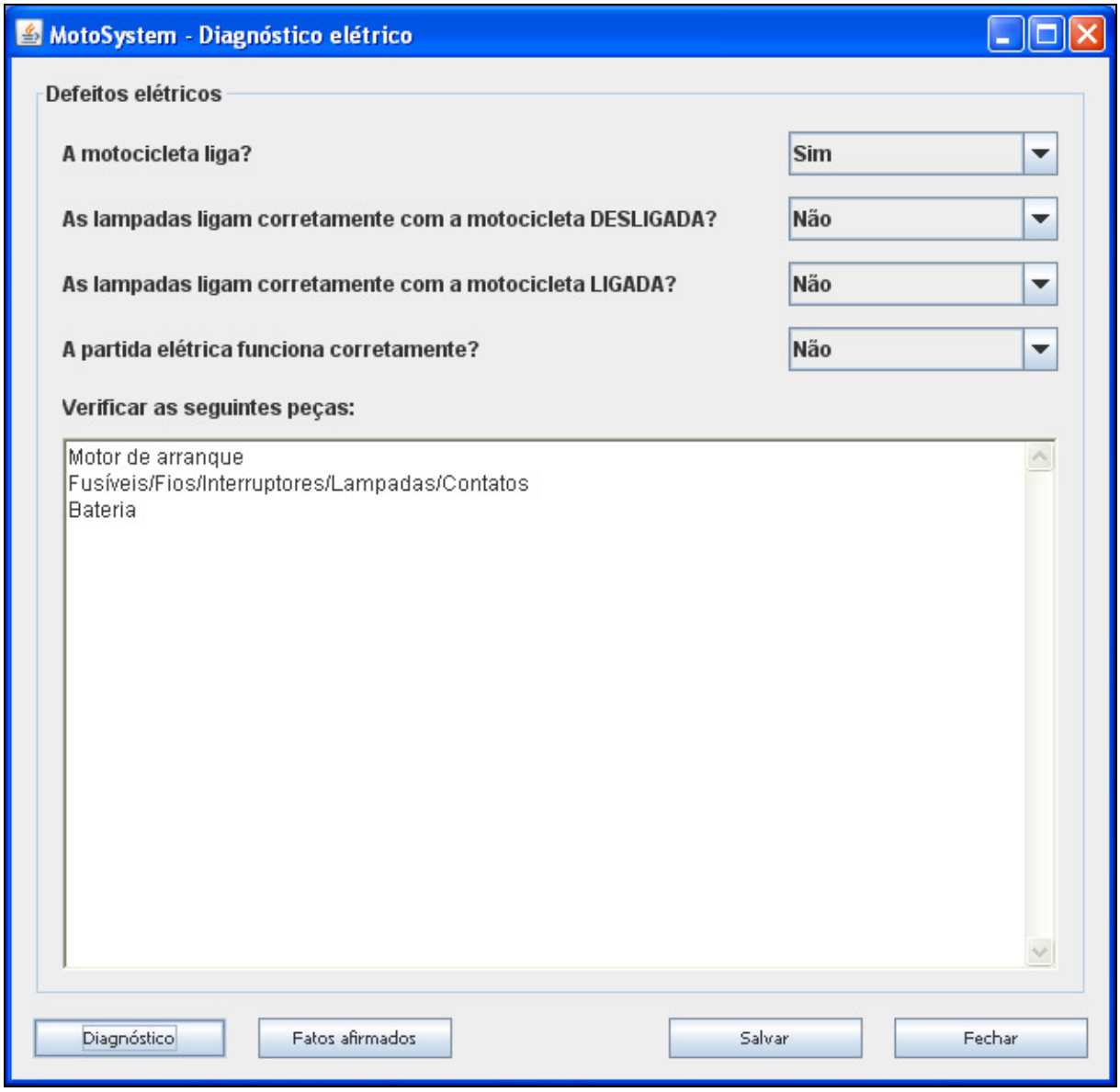
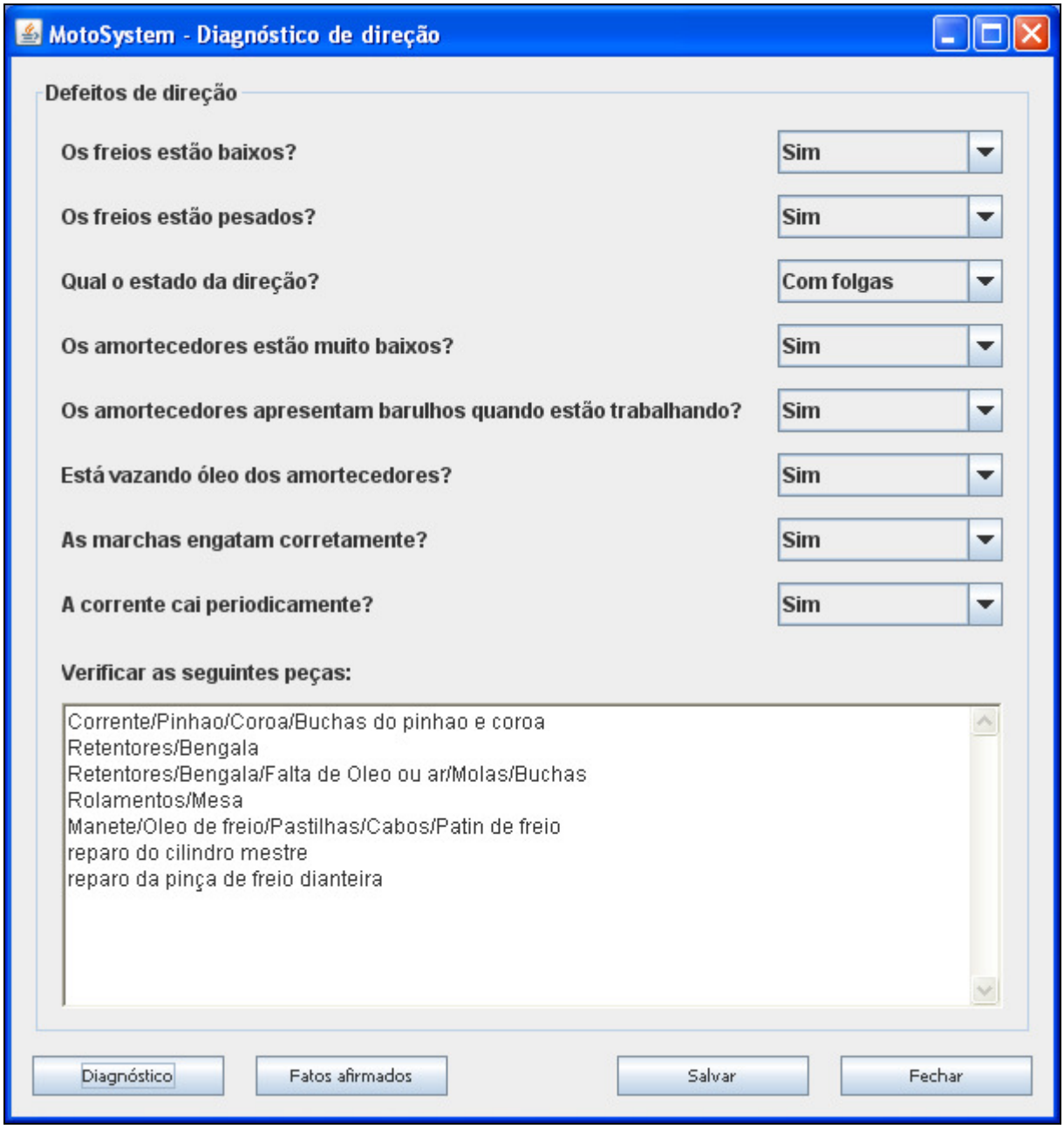


Figura 21 - Diagnóstico de direção



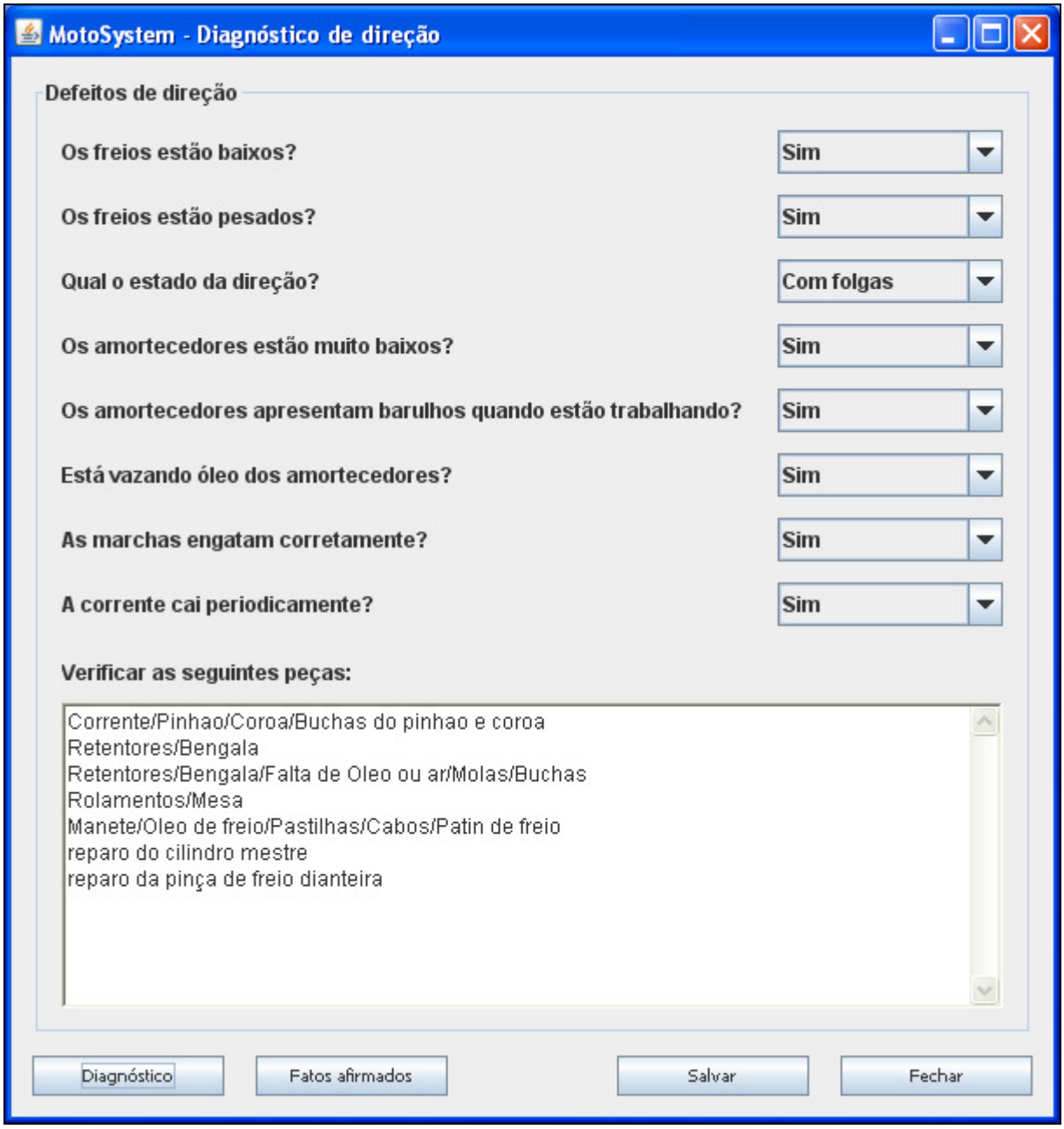
As regras da base de conhecimento estão todas inseridas dentro da tabela REGRAS, do SGBD, e ao acessar um determinado grupo de defeitos, o protótipo disponibiliza ao motor de inferência apenas as regras do respectivo grupo. A Figura 22 evidencia a forma como é realizada a carga das regras do grupo de defeitos elétricos.

Figura 22 - Carga defeitos elétricos

```
public Diagnostico_eletrica(Menu_principal_tela menu) throws JessException, SQLException {
    initComponents();
    rete = new Rete();
    this.menu = menu;
    gerarArquivoRegras();
    rete.batch("C:/Documents and Settings/Gustavo Costa/Meus documentos/NetBeansProjects/TCC_VM/regrasEletrica.txt");
}

public Menu_principal_tela getMenu() {
    return menu;
}

private void gerarArquivoRegras() throws SQLException {
    String regras = "";

    Statement select = getMenu().getLogin().getConexao().createStatement();
    String sql = "select ds_regra "
            + " from regras "
            + " where ie_modulo = 'DE'"
            + " and ie_situacao = 'A' ";

    select.execute(sql);
    while (select.getResultSet().next()) {
        regras += select.getResultSet().getString("ds_regra");
    }
    select.close();

    salvarArquivo(regras);
}
```

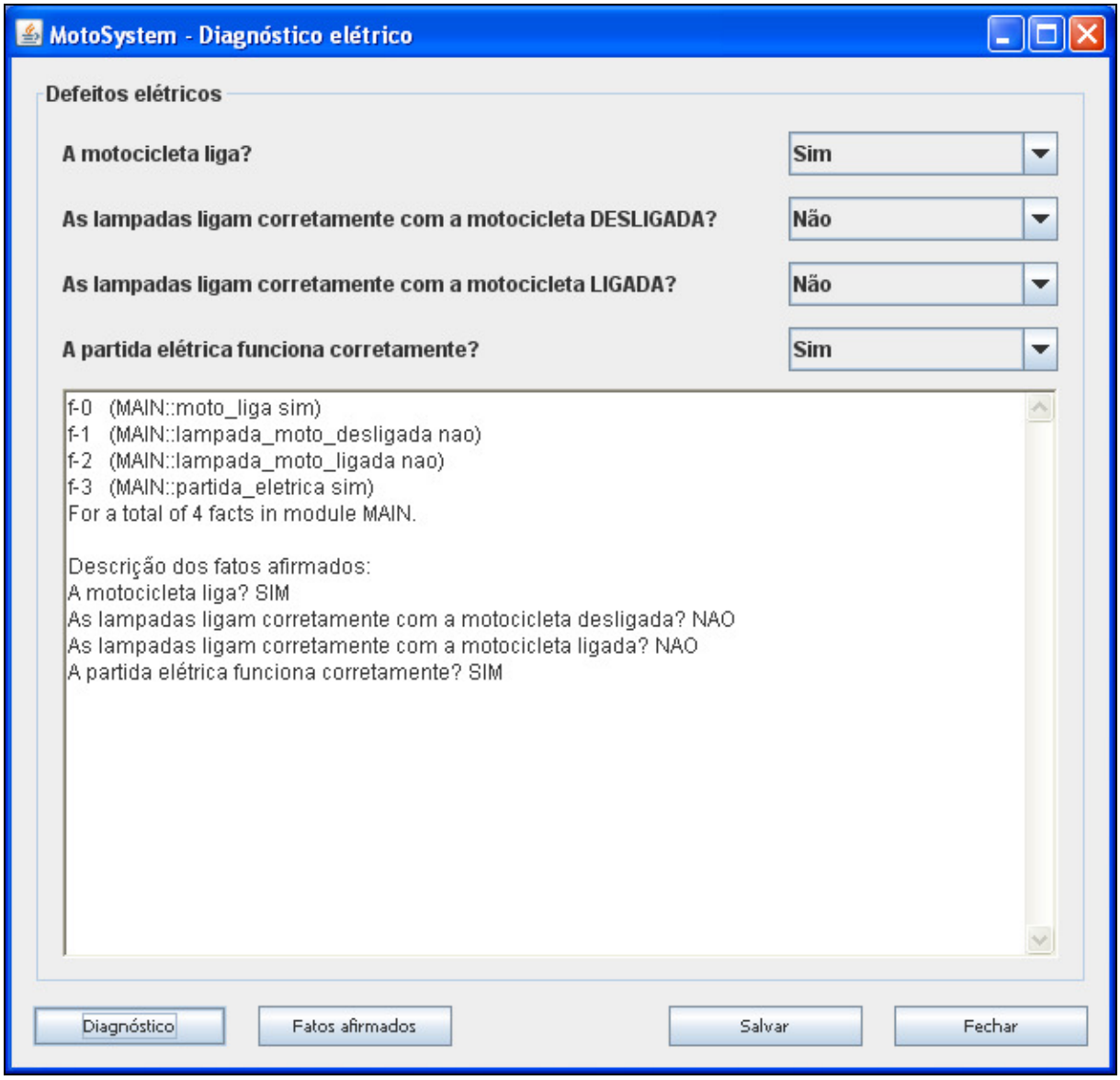
Observa-se que o JESS utiliza arquivos de texto (.txt) para carregar as regras, o protótipo executa um *select* no SGBD, insere essas informações em uma variável do tipo *String*, do Java, e cria um arquivo de texto com esta variável, que em seguida, será importado pela *shell*. Para realizar uma consulta para diagnóstico, basta acessar o módulo desejado, responder as perguntas com as opções existentes nas caixas de seleção e, por fim, acionar o botão Diagnóstico. A Figura 23 apresenta um exemplo deste processo.

Figura 23 - Diagnóstico finalizado



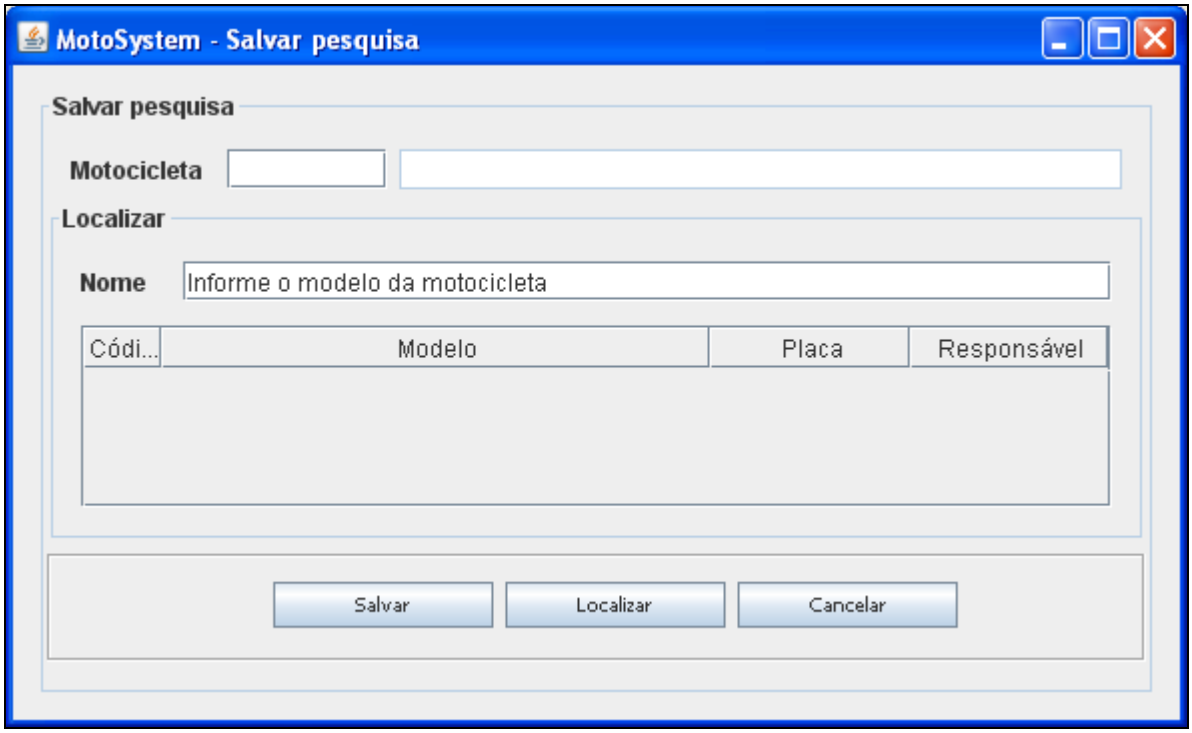
Depois que o diagnóstico é apresentado, o protótipo questiona o usuário se ele deseja visualizar os fatos afirmados sobre a consulta que acabara de ser realizada. Se o usuário optar por visualizar os fatos, o resultado será o descrito na Figura 24. Também é possível visualizar os fatos afirmados acionando o botão Fatos Afirmados, cujo resultado será o mesmo que a primeira forma.

Figura 24 - Fatos afirmados



O módulo de consultas ainda permite que os usuários salvem as consultas realizadas. Depois de efetuar todo o processo de diagnóstico, se for acionado o botão Salvar, o protótipo apresentará uma janela onde deverá ser informada a qual motocicleta a consulta se refere, conforme demonstrado na Figura 25, o qual faz o atendimento do requisito funciona RF06.

Figura 25 - Salvar consulta



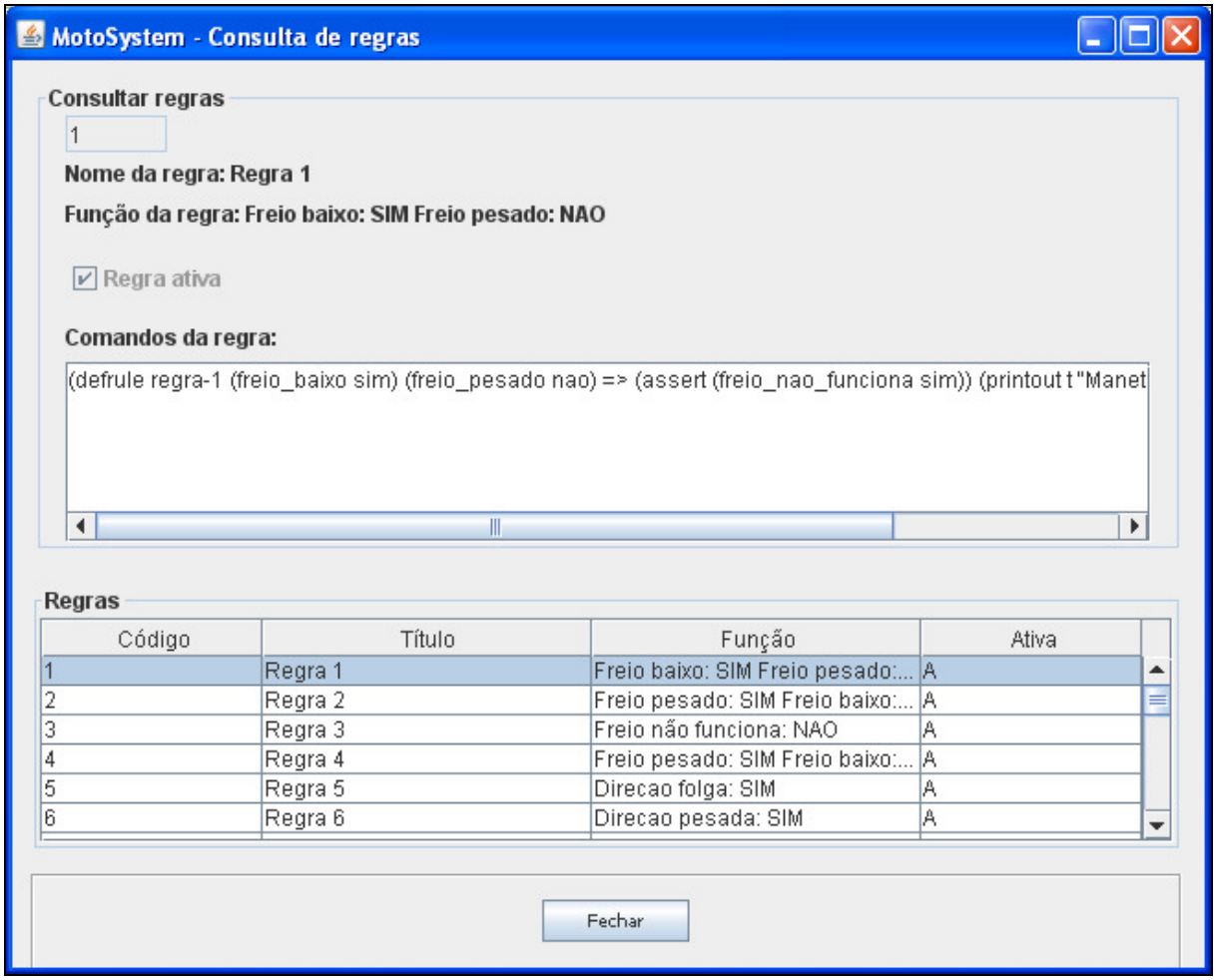
Com relação à técnica adotada para diagnosticar os defeitos mecânicos, o protótipo repassa todas as afirmações (*assert*) à *shell* através do comando "*executeCommand*", presente na classe Java do algoritmo *Rete*. Os comandos serão repassados conforme as respostas informadas nas caixas de seleção ao lado das perguntas. A Figura 26 apresenta um trecho do código fonte onde o protótipo repassa as informações para a *shell*.

Figura 26 - Iteração Java X JESS

```
private void asserts() throws JessException {

   fatosExplicativo = "\nDescrição dos fatos afirmados:";

   if (motoLiga.getSelectedIndex() != 0) {
      fatosExplicativo += "\nA motocicleta liga? ";
      if (motoLiga.getSelectedIndex() == 1) {
         rete.executeCommand("(assert(moto_liga sim))");
         fatosExplicativo += "SIM";
      } else if (motoLiga.getSelectedIndex() == 2) {
         rete.executeCommand("(assert(moto_liga nao))");
         fatosExplicativo += "NAO";
      }
   }

   if (lampadaDesligada.getSelectedIndex() != 0) {
      fatosExplicativo += "\nAs lampadas ligam corretamente com a motocicleta desligada? ";
      if (lampadaDesligada.getSelectedIndex() == 1) {
         rete.executeCommand("(assert(lampada_moto_desligada sim))");
         fatosExplicativo += "SIM";
      } else if (lampadaDesligada.getSelectedIndex() == 2) {
         rete.executeCommand("(assert(lampada_moto_desligada nao))");
         fatosExplicativo += "NAO";
      }
   }
```

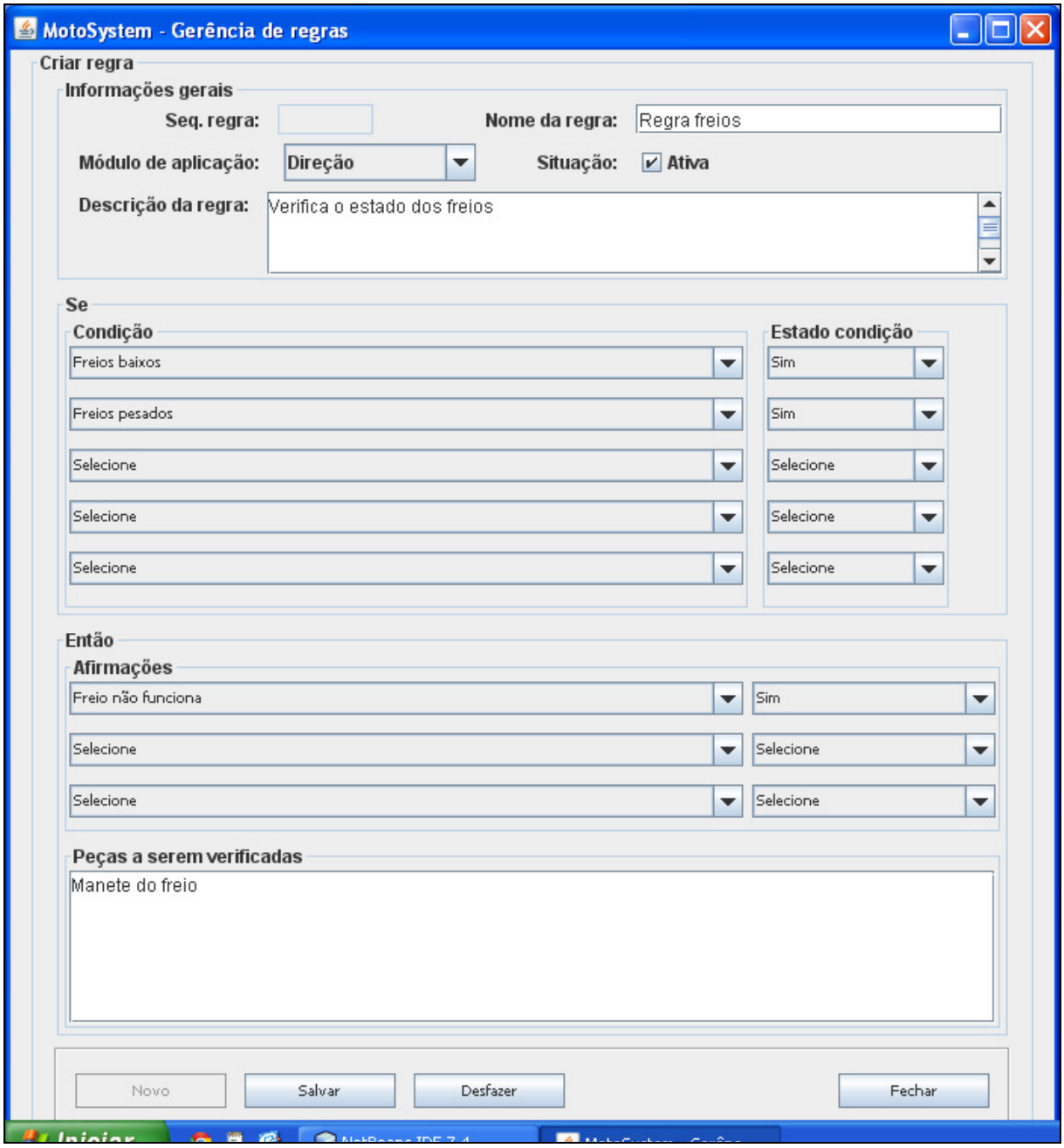
Um sistema baseado em conhecimento deve permitir alterações na sua base de conhecimentos, podendo ser para a adição de novas regras ou para a inativação de outras. Esta funcionalidade está implementada no módulo de “Gestão de conhecimento”, onde é possível que o usuário apenas consulte as regras existentes, desative-as ou crie novas regras, atendendo ao requisito funcional RF12. No menu principal, ao acionar a opção “Consultar regras”, será apresentada uma tela conforme a Figura 27 a qual possui uma única finalidade, que é demonstrar todas as regras cadastradas no protótipo.

Figura 27 - Consulta de regras



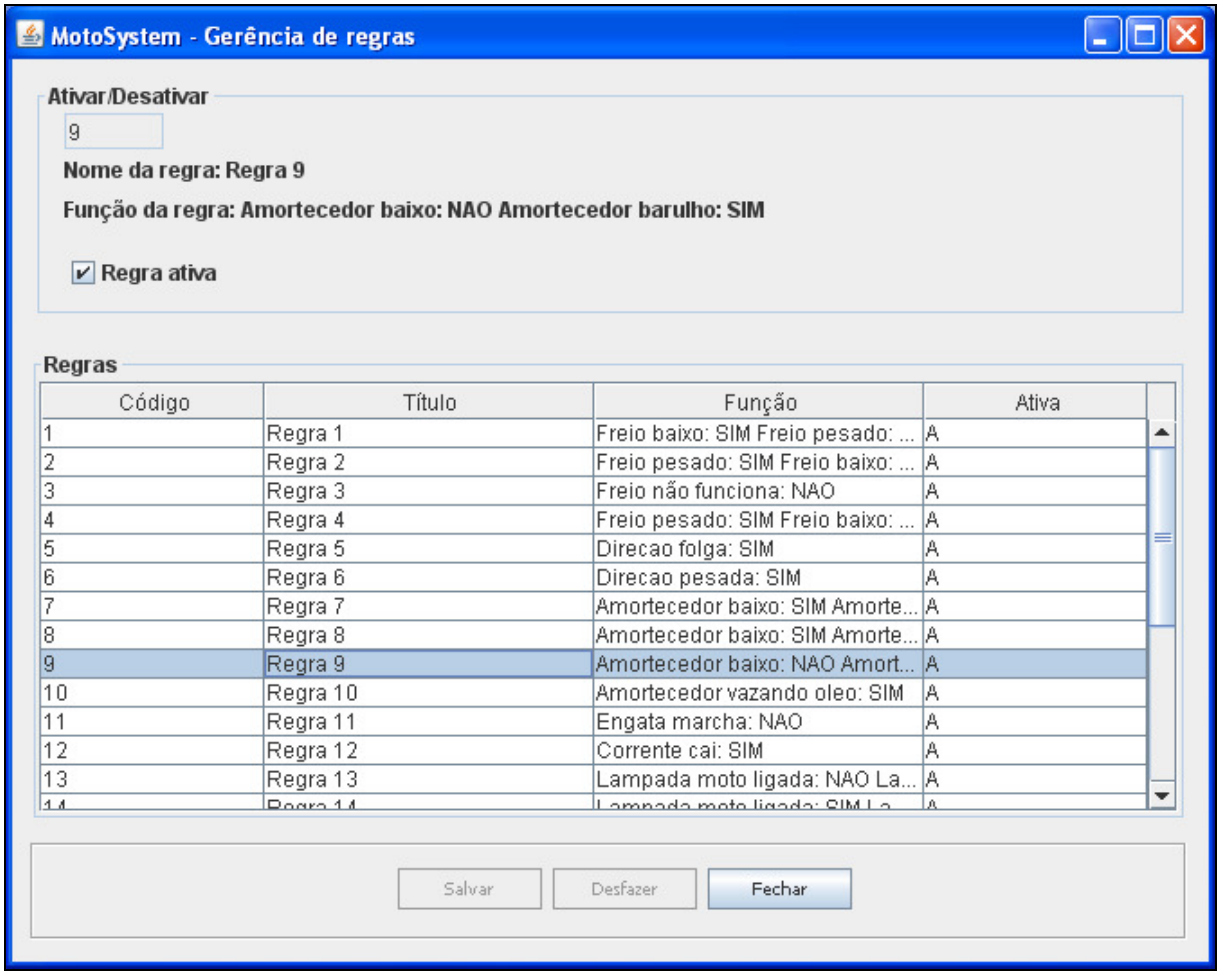
No menu principal, a partir do acionamento do botão “Criar regra”, a tela presente na Figura 28 será apresentada. Ao clicar em “Novo”, vários campos serão habilitados, os quais o usuário deverá preencher nova regra. Ao término do processo o botão “Salvar” deverá ser acionado para adicionar esta nova regra ao SGBD.

Figura 28 - Criar regra



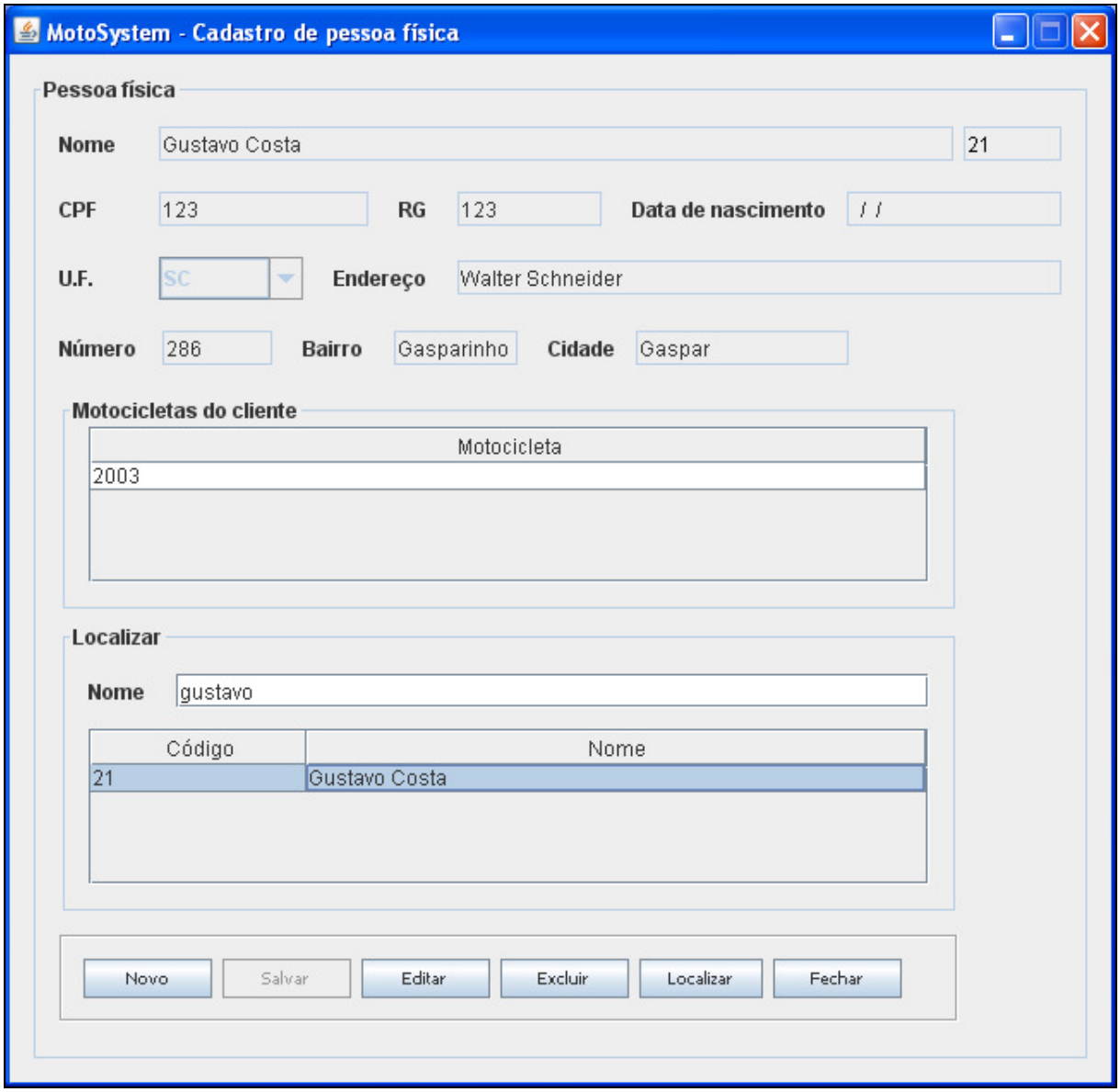
A última opção da gestão de conhecimento está no acionamento do botão "Ativar/Desativar regras", presente no menu principal. Ao acionar a opção citada será apresentada a tela da Figura 29. Onde o usuário poderá selecionar a regra que desejar e marcar ou desmarcar o *checkbox* "Regra ativa" para alterar o estado de uma regra.

Figura 29 - Ativar/Desativar regras



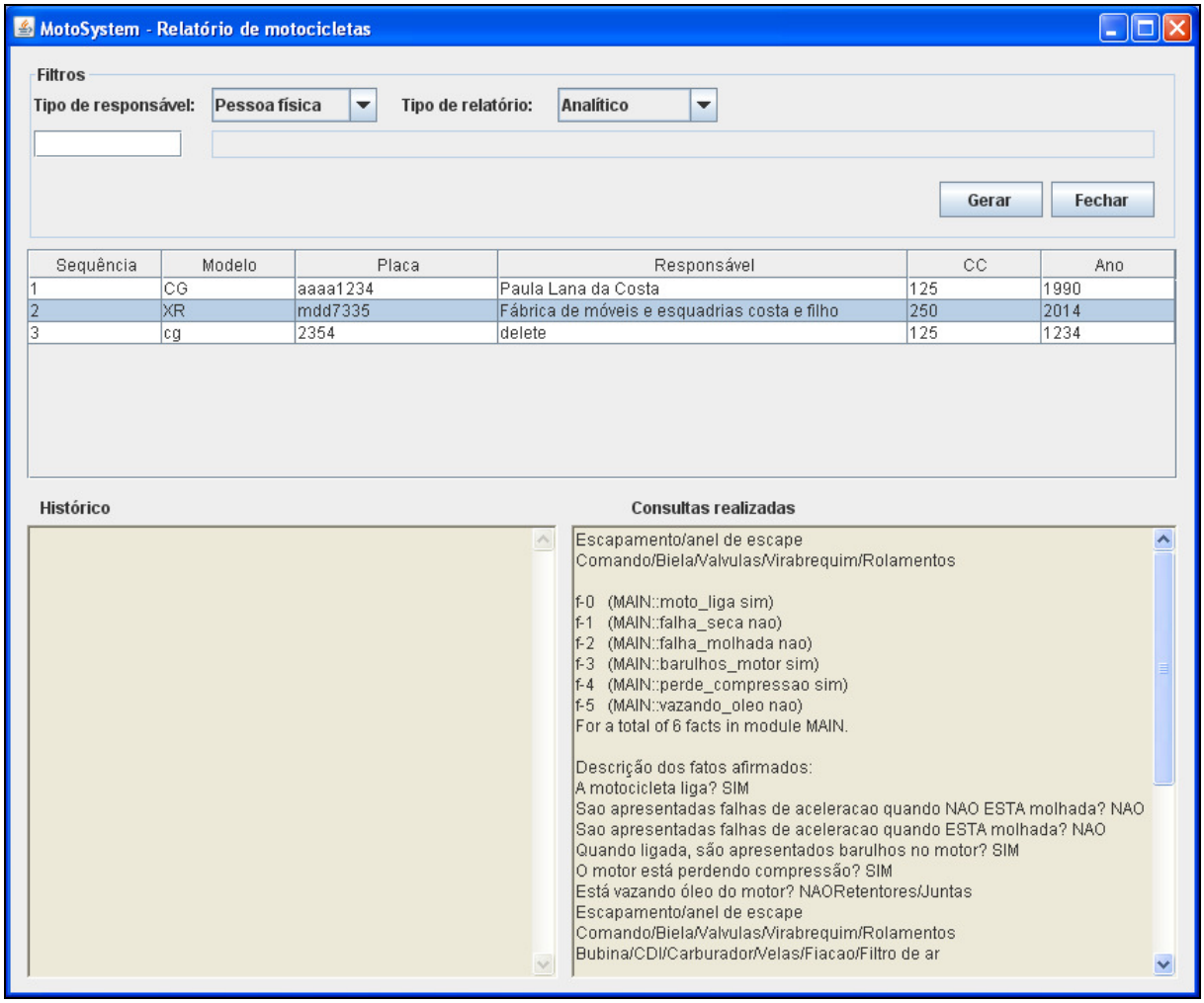
Também a partir da tela de menu principal, o usuário pode selecionar as funcionalidades do módulo administrativo, que é composto pelas telas de gerência de contas a pagar/receber, gerência de motocicletas e gerência de pessoas, sendo elas, físicas ou jurídicas. Os leiautes das funções de gerência seguem todos o mesmo padrão, possuindo os dados de cadastro na parte superior da tela, um localizador na parte inferior da tela e no rodapé existe um painel com as ações que o usuário poderá realizar. A Figura 30 apresenta a tela de gerência de pessoas físicas, onde é possível observar o leiaute citado.

Figura 30 - Gerência de pessoas físicas



O último módulo do protótipo é o de geração de relatórios, sendo divididos em relatórios de clientes, contas a pagar, contas a receber e motocicletas. A Figura 31 apresenta o relatório de motocicletas, onde o usuário poderá selecionar vários filtros dentre eles, a forma de visualização do relatório (sintética e analítica). O relatório de motocicletas apresenta dois pequenos quadros que, quando selecionada uma motocicleta do relatório, demonstra o histórico e as consultas de diagnóstico vinculadas àquela motocicleta.

Figura 31 - Relatório de motocicletas



Para gerar todos os relatórios do protótipo, a interface obtém os registros do banco de dados, considerando os filtros informados pelo usuário, e apresenta estas informações em um *grid*. A Figura 32 demonstra o código utilizado para gerar o relatório contas a pagar.

Figura 32 - Código para geração de relatório

```
public ResultSet selectLocalizador() throws SQLException {
    ResultSet rs;
    Statement stmt = getMenu().getLogin().getConexao().createStatement();
    String sql = "";

    sql = "select           nr_sequencia, "
          + "               ds_conta, "
          + "               case "
          + "               when (ie_tipo_resp = 'PF') then obter_nome_pf(cd_pessoa_fisica_resp_cp) "
          + "               when (ie_tipo_resp = 'PJ') then obter_nome_pj(cd_pessoa_juridica_resp_cp) end nm_responsavel, "
          + "               dt_conta, "
          + "               vl_conta,"
          + "               ie_pago  "
          + "     from      conta_pagar "
          + "    where      1 = 1 ";
    if (!"  /  /    ".equals(dtInicial.getText()) && !"  /  /    ".equals(dtFinal.getText())) {
        dtInicialConv = converteData(dtInicial.getText());
        dtFinalConv = converteData(dtFinal.getText());
        sql += "and dt_conta between '" + dtInicialConv + "' and '" + dtFinalConv + "' ";
    }
    if (!"".equals(cdRespContaFiltro.getText()) && ieTipoRespContaRel.getSelectedIndex() != 0) {
        if (ieTipoRespContaRel.getSelectedIndex() == 1) {
            sql += " and ie_tipo_resp = 'PF' ";
            sql += " and cd_pessoa_fisica_resp_cp = " + cdRespContaFiltro.getText() + " ";
        } else if (ieTipoRespContaRel.getSelectedIndex() == 2) {
            sql += " and ie_tipo_resp = 'PJ' ";
            sql += " and cd_pessoa_juridica_resp_cp = " + cdRespContaFiltro.getText() + " ";
        }
    }
    if (ieSituacaoConta.getSelectedIndex() == 1) {
        sql += " and ie_pago = 'N' ";
    } else if (ieSituacaoConta.getSelectedIndex() == 2) {
        sql += " and ie_pago = 'S' ";
    }

    try {
        rs = stmt.executeQuery(sql);
```

3.4 RESULTADOS E DISCUSSÃO

Para o desenvolvimento do protótipo, foi utilizada a biblioteca JESS, que simula o conhecimento de um especialista por meio das regras existentes em sua base de conhecimentos. Uma dificuldade encontrada foi justamente na forma de utilizar esta biblioteca, pois a maioria da documentação encontrada não estava traduzida para o português, o que tomou muito tempo para poder entender e aprender a utilizar a ferramenta.

Depois de traduzir e estudar a documentação, foi realizado contato com os ex-alunos Ronaldo César Schork Júnior e Nívea Maria Pacheco e ambos enviaram uma cópia de seus respectivos trabalhos de conclusão de curso, onde foi possível analisar o código-fonte para entender como o JESS funciona. O restante do desenvolvimento do protótipo transcorreu sem maiores problemas, uma vez que a principal dificuldade encontrada foi à citada no parágrafo anterior. Sendo assim, a codificação transcorreu no cronograma planejado.

Para fazer a construção da base de conhecimentos inicial, o processo de aquisição e

formalização do conhecimento foi através da consultoria com o mecânico Artur Isensee. Depois que o mecânico repassou as informações adquiridas ao longo do seu tempo de trabalho, foram criados três grupos de defeitos, sendo eles os de direção, os hidráulicos e os elétricos. Com o objetivo de validar a confiabilidade do protótipo, foram realizadas algumas simulações de atendimento, onde o cliente informava os defeitos e o sistema apresentava o diagnóstico.

Os testes foram acompanhados por um mecânico, que foi responsável pela validação da base de conhecimentos. Em geral, os resultados apresentados foram positivos, pois em todos os casos as peças listadas realmente eram as que apresentavam defeitos. Porém foi ressaltado pelo mecânico responsável pela validação, que o protótipo, nos três casos de teste, deveria ter apresentado mais peças defeituosas, uma vez que os defeitos, geralmente são compostos por um grupo de peças com defeito.

A fim de avaliar os resultados do protótipo, foram feitas breves apresentações a funcionários de três estabelecimentos e após a apresentação foi respondido um questionário que abordava diversos itens, como estabilidade, confiabilidade e usabilidade. Em geral, o protótipo atende às necessidades básicas de uma oficina, sendo necessário criar mais divisões de grupos de diagnóstico de defeitos e adicionar mais filtros para os relatórios. Segundo as avaliações, outra funcionalidade a ser implementada seria a possibilidade de criar ordens de serviço, e poder vincular elas a motocicletas. Ao analisar a usabilidade do protótipo, as três avaliações o consideraram fácil de operar, intuitivo e com uma velocidade de funcionamento é rápida. O Quadro 11 apresenta um resumo das avaliações que se encontram disponíveis no Anexo B.

Quadro 11 - Resumo avaliações

Item avaliado/Avaliador	Usuário 1	Usuário 2	Usuário 3
Estabilidade	OK	OK	OK
Facilidade em utilizar (Muito fácil, Fácil, Difícil, Muito difícil)	Fácil	Fácil	Fácil
Os recursos atendem às necessidades?	Sim	Sim	Sim
O módulo de diagnóstico será uma ferramenta útil?	Sim	Sim	Sim
Uma versão *mobile* seria útil?	Sim	Sim	Sim
Módulo de cadastros (Nota de 0 a 10)	7	9	9
Módulo de consultas (Nota de 0 a 10)	8	10	9
Módulo de relatórios (Nota de 0 a 10)	8	10	9

Módulo de gerenciamento de conhecimento (Nota de 0 a 10)	10	10	9
Se utilizado no dia a dia do estabelecimento, vai diminuir o tempo de atendimento?	Sim	Sim	Sim

Desta forma, pode-se concluir que os objetivos do trabalho foram alcançados com sucesso, pois o protótipo contém o módulo de consultas para facilitar o diagnóstico de um defeito mecânico e os módulos para a gestão administrativa de uma oficina mecânica. Os mecânicos possuem a funcionalidade de diagnóstico de defeitos através de uma inteligência artificial embutida no protótipo e a área administrativa dispõe de diversos cadastros, como os de clientes, motocicletas, contas a pagar e contas a receber.

Ao realizar uma comparação com os trabalhos correlatos, este protótipo apresentou um diferencial no que diz respeito ao gerenciamento de conhecimento, pois o especialista pode criar novas regras e desativar outras no momento que desejar e apenas as regras ativas e do módulo desejado serão importadas pela *shell* no momento da sua utilização.

Ainda citando os trabalhos correlatos, o protótipo desenvolvido mantém maior similaridade com o trabalho de Schork Júnior (2002) no que diz respeito à *shell* utilizada, que foi a biblioteca JESS. A diferença é que o protótipo desenvolvido no presente trabalho não foi executado diretamente no *prompt* da *shell*, e sim por meio de uma *interface* Java Swing que facilita a iteração do usuário.

Com relação ao trabalho de Pacheco (2003), as técnicas adotadas foram semelhantes, uma vez que o trabalho atual e o correlato utilizaram a *shell* JESS juntamente a uma interface Java Swing. O diferencial do trabalho atual é a utilização de um banco de dados para armazenar e gerenciar as regras que serão importadas para a *shell* no momento de sua utilização.

O trabalho de Starke (2007) e o protótipo desenvolvido possuem relação na questão do modo de funcionamento, uma vez que foi utilizada uma *shell* juntamente a uma interface, para facilitar a utilização por parte do usuário e um banco de dados para armazenar informações. Além disto, um dos objetivos de Starke (2007) foi realizar "a formalização do conhecimento através de regras de produção", técnica também adotada no trabalho atual. Os dois trabalhos diferem quando são observadas as tecnologias de ambos. A linguagem de programação de Starke (2007) foi Delphi e seu banco de dados foi o Paradox enquanto este protótipo foi desenvolvido em Java juntamente ao banco de dados MySQL. As *shells* também são outra diferença, pois Starke (2007) utilizou o *Expert SINTA* e este JESS.

Para o desenvolvimento do protótipo, foi utilizada pelo menos uma característica de cada trabalho correlato, sendo elas:

a) a utilização de uma ferramenta *shell* para o desenvolvimento do módulo de consultas de defeitos;

b) o ambiente Java para a criação da interface de interação com o protótipo;

c) a integração com um banco de dados para o armazenamento de informações administrativas do protótipo.

4 CONCLUSÕES

O trabalho teve como principal objetivo criar um protótipo de um SBC que torne mais ágil o atendimento de uma oficina mecânica de motos aos seus clientes. Este objetivo foi atingido através de funcionalidades que permitem que o usuário realize a gerência administrativa de uma oficina mecânica e padronize os métodos de diagnóstico dos defeitos mecânicos das motocicletas.

A forma com que o usuário usa e contribui com o seu próprio conhecimento através da inserção de novas regras torna os diagnósticos cada vez mais concisos. Isso facilitará o seu dia a dia na hora de diagnosticar um defeito mecânico ou de atender um cliente, pois ele estará (ou será) cadastrado no sistema e suas informações estarão devidamente guardadas. Um dos objetivos específicos era a construção de uma base de conhecimentos de defeitos mecânicos existentes em motocicletas, e este foi atendido, pois a versão inicial contém uma base de conhecimento, que poderá ser incrementada ao longo do uso do sistema.

A escolha das ferramentas para o desenvolvimento do protótipo se adequou aos objetivos propostos. A plataforma escolhida é muito utilizada atualmente e a *shell* JESS é uma biblioteca dela, tornando a comunicação entre a interface e a *shell* mais ágil, pois dispensará integrações entre duas ferramentas de linguagens diferentes.

Ao fim deste trabalho chegou-se a conclusão que os resultados obtidos com o desenvolvimento deste protótipo foram satisfatórios, os requisitos propostos foram alcançados, e desta forma o objetivo principal do trabalho foi cumprido fornecendo ao usuário uma ferramenta única para administrar uma oficina e diagnosticar os defeitos mecânicos.

De um modo geral, este trabalho proporcionou um aprendizado no que diz respeito às ferramentas utilizadas para o desenvolvimento do protótipo e também como funcionam os processos de trabalho de uma oficina mecânica de motocicletas.

4.1 EXTENSÕES

O protótipo ainda apresenta limitações e pontos que podem ser melhorados. Sugere-se:

a) disponibilizar o aplicativo para plataformas *mobile*;
b) melhorar o sistema de aquisição de conhecimento do SBC;

c) simplificar o processo de criação de uma nova regra de produção;
d) criar funcionalidades para cadastro de peças e de ordens de serviço;
e) criar mais divisões de grupos de defeitos;
f) adicionar mais filtros aos relatórios.

REFERÊNCIAS

ALEXANDRE, Adriana Bombassaro. **Protótipo de um sistema especialista utilizando a ferramenta expert sinta shell para auxílio no setor de suporte de uma *software house***. 2000. 82 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

BITTENCOURT, Guilherme. **Inteligência artificial: Ferramentas e teorias**. 2.ed. Florianópolis: Ed. da UFSC, 2001.

CHIPTRONIC. **MotoDiag**: tabela de aplicação. Piraju, SP, 2013. Disponível em: <http://www.chiptronic.com.br/tabelas/baixar.php?arquivo=motodiag.pdf>. Acesso em: 22 maio 2014.

FORGY, Charles L. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. **Artificial Intelligence**. Holanda. v. 19. n. 3. p. 17-37, Setembro, 1982. Disponível em: <http://www.csl.sri.com/users/mwfong/Technical/RETE%20Match%20Algorithm%20-%20Forgy%20OCR.pdf>. Acesso em: 09 maio 2014.

GENARO, Sérgio. **Sistemas especialistas:** o conhecimento artificial. Rio de Janeiro: LTC - Livros técnicos e científicos S.A, 1986.

GIARRATANO, Joseph C; RILEY, Gary. **Expert systems:** principles and programming. 2d. ed. Boston: PWS Publishing Company, 1994.

GOMES, Dennis dos Santos. Inteligência Artificial: Conceitos e Aplicações. **Olhar Científico**. Faculdades Associadas de Ariquemes, Setor Grandes Áreas Ariquemes, Rondônia, v. 1, n. 2, p. 234-246, Agosto. 2010.

HEINZLE, Roberto. **Protótipo de uma ferramenta para criação de sistemas especialistas baseados em regras de produção**. 1995. 145 f. Dissertação (Mestrado em Engenharia de Produção) - Departamento de Engenharia de Produção e Sistemas, Universidade Federal de Santa Catarina, Florianópolis.

HERZBERG, Gerhard. **Jess, the Expert System Shell for the Java Platform.** Livermore, 2002. Disponível em: <http://herzberg.ca.sandia.gov/jess/>. Acesso em: 09 maio 2014.

HOLZ, Raquel da Fonseca; LINDAU, Luis Antonio. **Panorama internacional do uso e Operação de motocicletas**. 2009. Laboratório de Sistemas de Transportes, Universidade Federal do Rio Grande do Sul, Porto Alegre. Disponível em: <http://www.cbtu.gov.br/monografia/2009/trabalhos/artigos/gestao/3_320_AC.pdf>. Acesso em: 09 maio 2014.

LABORATÓRIO DE INTELIGÊNCIA ARTIFICIAL. **Expert SINTA**: uma ferramenta para criação de sistemas especialistas. Pernambuco, 1995. Disponível em: <http://www.ebah.com.br/content/ABAAAfFXUAB/manual-expert-sinta>. Acesso em: 09 maio 2014.

LINDSAY, Robert K. et al. **Applications of Artificial Intelligence for Organic Chemistry**: the Dendral project. USA: McGraw-Hill, 1980.

LUCHTENBERG, Jonas. **Protótipo de sistema especialista para área comercial utilizando a ferramenta SPIRIT**. 2000. 50f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

MOTTA, Paulo Roberto. **Transformação organizacional**: a teoria e a prática de inovar. Rio de Janeiro: Qualitymark Editora, 1998.

PACHECO, Nívea Maria. **Protótipo de um sistema especialista para auxiliar o diagnóstico de doenças da soja utilizando a ferramenta Jess**. 2003. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

PIO, Francesca V; CANCIAN, Maiara Heil; FRANÇA, Ricardo Bedin. **JESS – the Rule Engine for the JavaTM Platform**. 2007. Departamento de Automação e Sistemas – DAS, Universidade Federal de Santa Catarina, Florianópolis. Disponível em: <http://www.das.ufsc.br/~gb/pg-ia/Jess07/das6607-francesca-maiara-ricardo.pdf>. Acesso em: 09 maio 2014.

PREECE, Jenny et al. **Human-Computer Interaction**. USA: Addison - Wesley, 1994.

RABUSKE, Renato Antônio. **Inteligência artificial**. Florianópolis: Editora da UFSC, 1995.

REZENDE, Solange Oliveira. **Sistemas inteligentes**: fundamentos e aplicações. Barueri: Manole, 2003.

RIBEIRO, Horácio da Cunha e Souza. **Introdução aos sistemas especialistas**. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora S.A, 1987

RICH, Elaine; KNIGHT, Kevin. **Inteligência Artificial**. São Paulo: Makron Books do Brasil, 1994.

ROSA, João Luís Garcia. **Sistemas Baseados em Conhecimento**. Universidade de São Paulo, 2000. Disponível em: <http://www.icmc.usp.br/pessoas/joaoluis/sbc_jrosa.pdf>. Acesso em: 09 maio 2014.

RUSSEL, Stuart; NORVIG, Peter. **Inteligência Artificial**. 2. ed. Rio de Janeiro: Campus, 2004.

SABBATINI, Renato M. E. EXPERTMD: manual de uso. 4. ed. Campinas: Núcleo de Informática Biomédica/Unicamp, 1992.

SANCHES, André Rodrigo. **Fundamentos de armazenamento e manipulação de dados**. São Paulo, 2005. Disponível em:<http://www.ime.usp.br/~andrers/aulas/bd2005-1/aula6.html>. Acesso em: 22 maio 2014.

SCHORK JÚNIOR, Ronaldo César. **Protótipo de um sistema especialista para a seleção de microcomputadores utilizando a ferramenta Jess**. 2002. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SOARES, Renato. **Motos Dicas**. Defeitos na moto como identificar. [S.i.]. 2010. Disponível em <https://sites.google.com/site/motosdicas/news/defeitosnamotocomoidentificar>. Acesso em: 09 maio 2014.

STARKE, Elaine. **Sistema especialista em táticas de abordagens policiais aplicado à polícia militar de Santa Catarina na regional de Blumenau**. 2007. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

TEIVE, Raimundo C. Ghizoni. **Planejamento da expansão da transmissão de sistemas de energia elétrica utilizando sistemas especialistas**. 1997. Monografia (Pós Graduação em Engenharia de Produção), Universidade Federal de Santa Catarina.

APÊNDICE A – Descrição dos Casos de Uso

Este apêndice apresenta o Quadro 12, com a descrição dos casos de uso descritos na especificação deste trabalho.

Quadro 12 - Descrição dos casos de uso

UC01 Manter clientes
Permite ao usuário realizar o gerenciamento dos cadastros dos clientes.

Constraints
Pré-condição. Ter a chave de acesso validada no protótipo.
Pós-condição. Um cliente foi incluído no protótipo.
Pós-condição. Um cliente foi alterado no protótipo.
Pós-condição. Um cliente foi removido do protótipo.
Pós-condição. Cliente(s) foi(ram) consultado(s).

Cenários
Incluir cliente {Principal}.

1) Usuário abre o cadastro de clientes
2) Sistema apresenta tela para registro de clientes
3) Usuário insere os dados do cliente no sistema
4) Usuário salva o novo cadastro.

Alterar cadastro de cliente {Alternativo 1}.

1) Usuário abre o cadastro de clientes.
2) Sistema apresenta tela para registro de clientes.
3) Usuário abre o localizador de clientes.
4) Usuário localiza o cliente.
5) Usuário altera os dados do cliente.
6) Usuário salva o cadastro alterado.

Remover cadastro de cliente {Alternativo 2}.

1) Usuário abre o cadastro de clientes.
2) Sistema apresenta tela para registro de clientes.
3) Usuário abre o localizador de clientes.
4) Usuário localiza o cliente.
5) Usuário seleciona a opção “Excluir”.
6) Usuário confirma a exclusão do cadastro.

Cliente não localizado {Exceção}
No passo 4 do cenário alternativo "1", caso o cliente à ser localizado não esteja cadastrado no sistema, o sistema exibe a seguinte mensagem: "Cliente não localizado! Por favor, verifique os dados inseridos".

Cliente não pode ser removido {Exceção}
No passo 6 do cenário alternativo "2", se o cadastro à ser removido já possuir vínculo com qualquer outra tabela do sistema, será apresentada a seguinte mensagem: "Este cadastro já possui vínculo no sistema, não será possível remove-lo por segurança de histórico".

UC02 Manter motocicletas
Permite ao usuário o gerenciamento dos cadastros das motocicletas.

Constraints
Pré-condição. Ter a chave de acesso validada no protótipo.
Pré-condição. Possuir clientes cadastrados no protótipo.
Pós-condição. Uma motocicleta foi incluída no protótipo.
Pós-condição. Uma motocicleta foi alterada no protótipo.
Pós-condição. Uma motocicleta foi removida do protótipo.
Pós-condição. Motocicleta(s) foi(ram) consultado(s).

Cenários
Incluir motocicleta {Principal}.

1) Usuário abre o cadastro de motocicletas.
2) Sistema apresenta tela para registro de motocicletas.
3) Usuário insere os dados da motocicleta no sistema.
4) Usuário salva o novo cadastro.
5) Usuário seleciona qual cliente é dono da motocicleta cadastrada.

Alterar cadastro de motocicleta {Alternativo 1}.

1) Usuário abre o cadastro de motocicletas.
2) Sistema apresenta tela para registro de motocicletas.
3) Usuário abre o localizador de motocicletas.
4) Usuário localiza a motocicleta.
5) Usuário altera os dados da motocicleta.
6) Usuário salva o cadastro alterado.

Remover cadastro de motocicleta {Alternativo 2}.

1) Usuário abre o cadastro de clientes.
2) Sistema apresenta tela para registro de clientes.
3) Usuário abre o localizador de clientes.
4) Usuário localiza o cliente.
5) Usuário seleciona a opção "Excluir".

6) Usuário confirma a exclusão do cadastro.

Motocicleta não localizada {Exceção}
No passo 4 do cenário alternativo "1", caso a motocicleta à ser localizada não esteja cadastrada no sistema, o sistema exibe a seguinte mensagem: "Motocicleta não localizada! Por favor, verifique os dados inseridos".

Motocicleta não pode ser removida {Exceção}
No passo 6 do cenário alternativo "2", se o cadastro à ser removido já possuir vínculo com qualquer outra tabela do sistema, será apresentada a seguinte mensagem: "Este cadastro já possui vínculo no sistema, não será possível remove-lo por segurança de histórico".

UC03 Manter contas a pagar
Permite ao usuário cadastrar contas a pagar, bem como pesquisar contas antigas e alterar dados de contas ainda em aberto. Além disso, será possível alterar o status das contas pendentes de pagamento.

UC04 Manter contas a receber
Permite ao usuário cadastrar contas a receber, bem como pesquisar contas antigas e alterar dados de contas ainda em aberto. Além disso, será possível alterar o status das contas pendentes de pagamento.

UC05 Imprimir relatórios de contas a pagar
Permite ao usuário imprimir relatórios de contas a pagar. Será possível filtrar o relatório por intervalo de datas e situação da conta ("Pago" ou "Pendente").

UC06 Manter histórico sobre a motocicleta
Permite ao usuário cadastrar e alimentar um histórico sobre uma motocicleta pré-cadastrada no sistema.

UC07 Realizar pesquisa no sistema baseado em conhecimento
Permite ao usuário acessar o módulo do sistema baseado em conhecimento e realizar uma pesquisa a fim de diagnosticar o defeito presente na motocicleta.

Constraints
Pré-condição. Ter a chave de acesso validada no protótipo.
Pré-condição. Possuir motocicletas cadastradas no protótipo.
Pós-condição. Diagnóstico do defeito da moto localizado.

Cenários
Realizar pesquisa {Principal}.

1) Usuário abre o módulo de diagnóstico do sistema.
2) Sistema apresenta tela para realização do diagnóstico do defeito.
3) O motor de inferência do SBC é ativado e começa a fazer a inferência dos questionamentos já cadastrados na base de conhecimentos.
4) Usuário informa as respostas das perguntas realizadas pelo sistema que são armazenadas no "quadro negro" do SBC.
5) Sistema apresenta o diagnóstico com base nas respostas inseridas.

Defeito não localizado {Alternativo 1}.
Ao término do cenário principal, se o sistema não possuir nenhuma diagnóstico para os questionamentos inseridos será apresentada a mensagem: “Não foi possível diagnosticar o defeito com base nas respostas inseridas. Deseja cadastrar uma nova regra?”. Se o usuário selecionar a opção “Sim”, o mesmo será direcionado à tela de cadastro de novas regras e procederá conforme o UC12.

UC08 Salvar pesquisas realizadas no sistema especialista
Permite ao usuário salvar a pesquisa realizada no UC07. Esta pesquisa será salva e atribuída a uma motocicleta previamente selecionada.

UC09 Imprimir relatórios de contas a receber
Permite ao usuário imprimir relatórios de contas a receber. Será possível filtrar o relatório por intervalo de datas e situação da conta (“Pago” ou “Pendente”).

UC10 Imprimir relatórios de clientes
Permite ao usuário imprimir relatórios de clientes. Será possível gerar o relatório de forma analítica e de forma sintética, buscando os dados do cliente.

UC11 Imprimir relatórios de motocicletas
Permite ao usuário imprimir relatórios de motocicletas. Será possível gerar o relatório de forma analítica e de forma sintética, buscando os dados da motocicleta e o respectivo histórico da motocicleta, que foi cadastrado no UC06.

UC12 Manter base de conhecimentos do sistema baseado em conhecimento
Permite ao usuário o gerenciamento de regras da base de conhecimentos do sistema baseado em conhecimento.

Constraints
Pré-condição. Ter a chave de acesso validada no protótipo.
Pós-condição. Uma nova regra foi incluída na base de conhecimentos.

Cenários
Incluir regra {Principal}.

1) O usuário abre o cadastro de novas regras.
2) O usuário escreve os questionamentos que o motor de inferência irá utilizar ao realizar uma pesquisa (Conforme UC07).
3) O usuário informa a resposta do questionamento (Sim, Não, Provavelmente sim, Provavelmente não e Parcialmente).
4) O usuário salva o registro da nova regra.

APÊNDICE B – Descrição do Dicionário de Dados

Este Apêndice apresenta a descrição das tabelas do banco de dados apresentadas na seção de especificação deste trabalho. Nos Quadros de 13 a 19 se apresenta o dicionário de dados das tabelas do protótipo. Os tipos de dados utilizados nos atributos são:

a) *int*: para variáveis numéricas inteiras;
b) *varchar*: armazena caracteres alfanuméricos;
c) *date*: armazena o tipo data(yyyy-mm-dd);
d) *float*: armazena variáveis numéricas com casas decimais.

Os tipos de chaves utilizadas nas tabelas são:

a) *primary key* (PK): identificador único do registro;
b) *foreign key* (FK): identificador do registro em outra tabela.

Quadro 13 - Tabela Pessoa

Pessoa – Armazena as pessoas físicas e jurídicas				
Campo	**Descrição**	**Tipo**	**Tamanho**	**Chave primária**
cd_pessoa	Código da pessoa	*int*	10	Sim
nm_pessoa_fisica	Nome da pessoa física	*Varchar*	100	Não
nr_cpf	CPF da pessoa física	*Varchar*	20	Não
nr_rg	RG da pessoa física	*int*	8	Não
ds_estado	Estado de residência	*Varchar*	2	Não
ds_endereco	Endereço de residência	*Varchar*	100	Não
nr_numero	Número do endereço	*int*	10	Não
ds_bairro	Bairro de residência	*Varchar*	50	Não
ds_cidade	Cidade de residência	*Varchar*	50	Não
dt_nascimento	Data de nascimento	*Date*	-	Não
nm_razao_social	Razão social	*Varchar*	150	Não
nr_cnpj	CNPJ da pessoa jurídica	*Varchar*	14	Não
dt_fundacao	Data de fundação	*Date*	-	Não
ie_tipo_pessoa	Tipo de pessoa	*Varchar*	2	Não

Quadro 14 - Tabela Motocicleta

Motocicleta – Armazena as motocicletas				
Campo	**Descrição**	**Tipo**	**Tamanho**	**Chave primária**
nr_sequencia	Código da motocicleta	*int*	10	Sim
nr_ano	Ano de fabricação	*int*	4	Não
ds_modelo	Modelo	*Varchar*	45	Não
qt_cc	Cilindradas	*int*	10	Não
ds_historico	Histórico	*Varchar*	4000	Não
ie_tipo_resp	Tipo de responsável	*Varchar*	2	Não
ie_marca	Marca	*int*	3	Não
ds_placa	Placa	*Varchar*	45	Não
ds_consultas	Diagnósticos realizados	*Varchar*	4000	Não
cd_pessoa_fisica	FK tabela Pessoa_fisica	*int*	10	Não
cd_pessoa_juridica	FK tabela Pessoa_juridica	*int*	10	Não

Quadro 15 - Tabela Consulta Motocicleta

Consulta_motocicleta – Armazena as consultas vinculadas às motocicletas				
Campo	**Descrição**	**Tipo**	**Tamanho**	**Chave primária**
nr_sequencia	Sequência da regra	*int*	10	Sim
dt_registro	Data do registro	*Date*	-	Não
ds_historico	Consulta realizada	*Varchar*	4000	Não
nr_seq_motocicleta	FK tabela Motocicleta	*int*	10	Não

Quadro 16 - Tabela Contas a receber

Conta_receber – Armazena as contas a receber				
Campo	**Descrição**	**Tipo**	**Tamanho**	**Chave primária**
nr_sequencia	Código da conta	*int*	10	Sim
ds_conta	Descrição da conta	*Varchar*	45	Não
vl_conta	Valor da conta	*Float*	10,2	Não

dt_conta	Data da conta	*Date*	-	Não
ds_descricao_longa	Descrição longa	*Varchar*	4000	Não
ie_tipo_resp	Tipo de responsável	*Varchar*	2	Não
ie_pago	Conta paga?	*Varchar*	1	Não
cd_pessoa_fisica_resp_cr	FK tabela Pessoa_fisica	*int*	10	Não
cd_pessoa_juridica_resp_cr	FK tabela Pessoa_juridica	*int*	10	Não

Quadro 17 - Tabela Contas a pagar

Conta_pagar – Armazena as contas a pagar				
Campo	**Descrição**	**Tipo**	**Tamanho**	**Chave primária**
nr_sequencia	Código da conta	*int*	10	Sim
ds_conta	Descrição da conta	*Varchar*	45	Não
vl_conta	Valor da conta	*Float*	10,2	Não
dt_conta	Data da conta	*Date*	-	Não
ds_descricao_longa	Descrição longa	*Varchar*	4000	Não
ie_tipo_resp	Tipo de responsável	*Varchar*	2	Não
ie_pago	Conta paga?	*Varchar*	1	Não
cd_pessoa_fisica_resp_cp	FK tabela Pessoa_fisica	*int*	10	Não
cd_pessoa_juridica_resp_cp	FK tabela Pessoa_juridica	*int*	10	Não

Quadro 18 - Tabela Regras

Regras – Armazena as regras da base de conhecimento				
Campo	**Descrição**	**Tipo**	**Tamanho**	**Chave primária**
nr_sequencia	Código da regra	*int*	10	Sim
nm_regra	Nome da regra	*Varchar*	255	Não
ds_regra	Comandos da regra	*Varchar*	2000	Não

ie_situacao	Regra ativa?	*Varchar*	1	Não
ie_modulo	Módulo de aplicação	*Varchar*	2	Não
ds_funcao_regra	Função da regra	*Varchar*	2000	Não

Quadro 19 - Tabela Usuário

Regras – Armazena as regras da base de conhecimento				
Campo	**Descrição**	**Tipo**	**Tamanho**	**Chave primária**
nm_usuario	Usuário	*Varchar*	15	Sim
ds_senha	Senha	*Varchar*	15	Não
ie_tipo_usuario	Tipo de usuário	*Varchar*	2	Não

ANEXO A – Entrevista

Neste anexo, apresenta-se o Quadro 20, que contém uma entrevista informal realizada com um dos mecânicos da oficina Moto Peças Kalarrari, estabelecida na cidade de Gaspar, estado de Santa Catarina.

Quadro 20 - Entrevista Mecânico

Gustavo Costa (Entrevistador): Há quanto tempo você trabalha como mecânico?
Mecânico 1 (Entrevistado): Sou mecânico há mais de 8 anos.

GC: Qual o principal "desafio" enfrentado pelo mecânico no momento da identificação do defeito?
Mecânico 1: Atualmente utilizamos um scanner para identificar o defeito da motocicleta, porém, esta ferramenta não consegue identificar todos os defeitos. Vários defeitos, principalmente as falhas na injeção eletrônica, são imperceptíveis a essa ferramenta. E quando o scanner não localiza o defeito, o processo de identificação se torna demorado demais, ou seja, ocorre a diminuição do rendimento do profissional. Além disso, temos alta rotatividade de funcionários, logo não conseguimos que todos os mecânicos possuam vasta experiência em todos os módulos da motocicleta.

GC: Qual o tempo médio para diagnosticar o defeito apresentado pela motocicleta?
Mecânico 1: Quando o scanner identifica o defeito, o tempo médio dura entre dez e quinze minutos. Este tempo é destinado à instalação do equipamento na motocicleta e leitura realizada pelo aparelho. Em casos que o scanner não consegue diagnosticar o defeito, este tempo médio sobe para 70 minutos.

O entrevistado ainda ressaltou que existem casos que o tempo médio é excedido em uma quantidade de tempo considerável. Um exemplo disso foi uma motocicleta que chegou à oficina no dia 06/05/2013, pela manhã, e o defeito foi identificado apenas no dia 07/05/2013, no período da tarde. O defeito presente na motocicleta era uma peça do sistema de aceleração, que apresentava falhas apenas quando o motor estava aquecido e a velocidade ultrapassasse 50 quilômetros por hora.

GC: Um software que auxiliasse o mecânico no diagnóstico do defeito seria útil para o estabelecimento?
Mecânico 1: Seria válido, pois os defeitos que o scanner não identificasse seriam apontados pelo sistema.

GC: A seu ver, quais os benefícios a utilização de um sistema da área administrativa traz para o estabelecimento?
Mecânico 1: Ocorreria melhora na organização da área administrativa e aumento na agilidade de atendimento, pois não seria necessário procurar o cadastro do cliente em uma ficha, basta localizar no computador.

GC: Sempre é guardado um histórico do defeito presente na motocicleta?
Mecânico 1: Sim, mas nem sempre alimentamos este histórico com novas informações.

GC: O histórico das motocicletas é digital ou através de arquivos de papel?
Mecânico 1: Nada é gravado no computador. Temos uma pasta para cada cliente e nela guardamos informações sobre a motocicleta.

ANEXO B – Questionários de avaliação

Este anexo apresenta os Quadros 21, 22 e 23, que contém os questionários de avaliação de mecânicos que avaliaram o protótipo.

Quadro 21 - Avaliação Usuário 1

Função: Gerente/Mecânico

Tempo de experiência: 19 anos

1. O software é estável?

(X) Sim ( ) Não

Em caso negativo, informe quais:

2. Considerando a facilidade em utilizar o software, como você o avalia?

( ) Muito fácil (X) Fácil ( ) Difícil ( ) Muito difícil

Observações:

3. Os recursos disponíveis no software atendem às necessidades do estabelecimento?

(X) Sim ( ) Não

Observações: Necessita aprimorar os cadastros de diagnósticos, dividindo-os em grupos mais específicos de serviços (Motor, freio, etc).

4. O módulo de diagnóstico de defeitos será uma ferramenta útil ao processo dos mecânicos?

(X) Sim ( ) Não

5. Este mesmo software, porém, em uma versão para dispositivos móveis seria útil ao estabelecimento?

(X) Sim ( ) Não

6. Os relatórios apresentados pelo sistema são concisos?

(X) Sim ( ) Não

7. Qual a avaliação geral para o módulo de cadastros?

(Nota de 0 a 10) __7__

Observações: Melhorar os cadastros de diagnósticos e implementar mais funcionalidades para o módulo administrativo, como a vinculação de ordens de serviço a uma motocicleta (e a seu respectivo diagnóstico).

8. Qual a avaliação geral para o módulo de consultas?

(Nota de 0 a 10) __8__

Observações: Necessita aprimorar os cadastros de diagnósticos, dividindo-os em grupos mais específicos de serviços (Motor, freio, etc).

9. Qual a avaliação geral para o módulo de relatórios?

(Nota de 0 a 10) __8__

Observações: Necessita aprimorar os filtros de informações.

10. Qual a avaliação geral para o módulo de gerenciamento de conhecimento?

(Nota de 0 a 10) __10__

Observações: Atende ao propósito.

11. Caso este software seja utilizado no dia a dia do estabelecimento, o tempo de atendimento aos clientes diminuiria?

(X) Sim ( ) Não

Quadro 22 - Avaliação Usuário 2

Função: Proprietário/mecânico

Tempo de experiência: 17 anos

1. O software é estável?

(X) Sim ( ) Não

Em caso negativo, informe quais:

2. Considerando a facilidade em utilizar o software, como você o avalia?

( ) Muito fácil (X) Fácil ( ) Difícil ( ) Muito difícil

Observações:

3. Os recursos disponíveis no software atendem às necessidades do estabelecimento?

(X) Sim ( ) Não

Observações:

4. O módulo de diagnóstico de defeitos será uma ferramenta útil ao processo dos mecânicos?

(X) Sim ( ) Não

Observações: Deve ser verificada a necessidade de cada oficina para fazer a respectiva modelagem do sistema.

5. Este mesmo software, porém, em uma versão para dispositivos móveis seria útil ao estabelecimento?

(X) Sim ( ) Não

Observações: Para a identificação de falhas intermitentes (falhas de estrada).

6. Os relatórios apresentados pelo sistema são concisos?

(X) Sim ( ) Não

Observações:

7. Qual a avaliação geral para o módulo de cadastros?

(Nota de 0 a 10) __9__

Observações:

8. Qual a avaliação geral para o módulo de consultas?

(Nota de 0 a 10) _10_

Observações:

9. Qual a avaliação geral para o módulo de relatórios?

(Nota de 0 a 10) __10__

Observações:

10. Qual a avaliação geral para o módulo de gerenciamento de conhecimento?

(Nota de 0 a 10) __10__

Observações:

11. Caso este software seja utilizado no dia a dia do estabelecimento, o tempo de atendimento aos clientes diminuiria?

(X) Sim ( ) Não

Em caso negativo, justifique:

Quadro 23 - Avaliação Usuário 3

Função: Mecânico

Tempo de experiência: 4 anos

1. O software é estável?

(X) Sim ( ) Não

Em caso negativo, informe quais:

2. Considerando a facilidade em utilizar o software, como você o avalia?

( ) Muito fácil (X) Fácil ( ) Difícil ( ) Muito difícil

Observações:

3. Os recursos disponíveis no software atendem às necessidades do estabelecimento?

(X) Sim ( ) Não

Observações:

4. O módulo de diagnóstico de defeitos será uma ferramenta útil ao processo dos mecânicos?

(X) Sim ( ) Não

Observações:

5. Este mesmo software, porém, em uma versão para dispositivos móveis seria útil ao estabelecimento?

(X) Sim ( ) Não

Observações:

6. Os relatórios apresentados pelo sistema são concisos?

(X) Sim ( ) Não

7. Qual a avaliação geral para o módulo de cadastros?

(Nota de 0 a 10) __9__

Observações:

8. Qual a avaliação geral para o módulo de consultas?

(Nota de 0 a 10) __9__

Observações:

9. Qual a avaliação geral para o módulo de relatórios?

(Nota de 0 a 10) __9__

Observações:

10. Qual a avaliação geral para o módulo de gerenciamento de conhecimento?

(Nota de 0 a 10) __9__

Observações:

11. Caso este software seja utilizado no dia a dia do estabelecimento, o tempo de atendimento aos clientes diminuiria?

(X) Sim ( ) Não

Em caso negativo, justifique:

ANEXO C – Regras de produção – Carga inicial

Este anexo apresenta os Quadros 24, 25 e 26 que contém as regras de produção presentes na carga inicial do protótipo.

Quadro 24 - Regras - Defeitos de direção

```
(defrule regra-1
      (freio_baixo sim)
      (freio_pesado nao)
      =>
      (assert (freio_nao_funciona sim))
      (printout t "Manete/Oleo de freio/Pastilhas" crlf))

(defrule regra-2
      (freio_pesado sim)
      (freio_baixo nao)
      =>
      (assert (freio_nao_funciona sim))
      (printout t "Cabos/Patin de freio" crlf))

(defrule regra-3
      (freio_nao_funciona nao)
      =>
      (printout t "Cabos/Pastilhas/Oleo de freio/Manete" crlf))

(defrule regra-4
      (freio_baixo sim)
      (freio_pesado sim)
      =>
      (assert (freio_nao_funciona sim))
      (printout t "Manete/Oleo de freio/Pastilhas/Cabos/Patin de freio"
crlf))

(defrule regra-5
      (direcao_folga sim)
      =>
      (assert (direcao_pesada nao))
      (printout t "Rolamentos/Mesa" crlf))

(defrule regra-6
      (direcao_pesada sim)
      =>
      (assert (direcao_folga nao))
      (printout t "Rolamentos/Mesa" crlf))

(defrule regra-7
      (amortecedor_baixo sim)
      (amortecedor_barulho sim)
      =>
      (assert (amortecedor_vazando sim))
      (printout t "Retentores/Bengala/Falta de Oleo ou ar/Molas/Buchas"
crlf))
```

```
(defrule regra-8
      (amortecedor_baixo sim)
      (amortecedor_barulho nao)
      =>
      (assert (amortecedor_vazando sim))
      (printout t "Retentores/Bengala/Falta de Oleo ou ar/Molas" crlf))

(defrule regra-9
      (amortecedor_baixo nao)
      (amortecedor_barulho sim)
      =>
      (assert (amortecedor_vazando nao))
      (printout t "Retentores/Buchas/Molas" crlf))

(defrule regra-10
      (amortecedor_vazando sim)
      =>
      (printout t "Retentores/Bengala" crlf))

(defrule regra-11
      (engata_marcha nao)
      =>
      (printout t "Disco de embreagem/Separadores/Garfo de marchas" crlf))

(defrule regra-12
      (corrente_cai sim)
      =>
      (printout t "Corrente/Pinhao/Coroa/Buchas do pinhao e coroa" crlf))
```

Quadro 25 - Regras - Defeitos elétricos

```
(defrule regra-13
      (lampada_moto_ligada nao)
      (lampada_moto_desligada nao)
      =>
      (printout t "Fusíveis/Fios/Interruptores/Lampadas/Contatos" crlf))

(defrule regra-14
      (lampada_moto_ligada sim)
      (lampada_moto_desligada nao)
      (moto_liga sim)
      =>
      (printout t "Fusiveis/Fios/Interruptores/Lampadas/Contatos" crlf))

(defrule regra-15
      (moto_liga nao)
      (lampada_moto_ligada moto_nao_liga)
      (lampada_moto_desligada nao)
      =>
      (printout t "Bateria" crlf))

(defrule regra-16
      (moto_liga sim)
      (lampada_moto_desligada nao)
      =>
      (printout t "Bateria" crlf))
```

```
(defrule regra-17
      (moto_liga nao)
      =>
      (printout t "Bubina/CDI/Velas/Fiacao" crlf))

(defrule regra-18
      (partida_eletrica nao)
      =>
      (printout t "Motor de arranque" crlf))
```

Quadro 26 - Regras - Defeitos hidráulicos

```
(defrule regra-19
      (moto_liga sim)
      (falha_molhada sim)
      (falha_seca sim)
      =>
      (printout t "Bubina/CDI/Carburador/Velas/Fiacao/Filtro de ar" crlf))

(defrule regra-20
      (moto_liga sim)
      (falha_molhada sim)
      (falha_seca nao)
      =>
      (printout t "Bubina/CDI/Velas/Fiacao" crlf))

(defrule regra-21
      (moto_liga sim)
      (falha_molhada nao)
      (falha_seca sim)
      =>
      (printout t "Carburador/Velas/Fiacao/Filtro de ar" crlf))

(defrule regra-22
      (moto_liga nao)
      =>
      (printout t "Bubina/Carburador/Velas/Fiacao/Bateria" crlf))

(defrule regra-23
      (moto_liga sim)
      (barulhos_motor sim)
      =>
      (printout t "Comando/Biela/Valvulas/Virabrequim/Rolamentos" crlf))

(defrule regra-24
      (moto_liga sim)
      (perde_compressao sim)
      =>
      (printout t "Escapamento/anel de escape" crlf))

(defrule regra-25
      (vazando_oleo sim)
      =>
      (printout t "Retentores/Juntas" crlf))
```