Expert System for Car Failure Diagnostics

RONNY MORÁN, GUSTAVO LONDA, RENATO ILLESCAS

GROUP NUMBER #2

A brief introduction expert systems are computer programs that replicate the knowledge and skills of human experts in a specific area or domain, and then solve problems of the assigned domain (as do human experts).

Expert systems have different forms of representing knowledge such as:

- Procedural representation
- Frame based
- Semantic Networks.

We choose Procedural representation that is represented as IF - THEN Rules.

Why Procedural representation?

The knowledge is represented as a set of instructions IF - THEN type of rules, this represent a very human friendly knowledge representation. Facts are connected through rules, and can have more than one condition.

Procedural representation (programs) express explicitly the interrelations between all fragments of knowledge but are difficult to modify, these systems have ease of use of Meta-Knowledge, which allows explicitly decompose the problem.

The components of the expert systems are:

- **Knowledge Base:** Has the knowledge to formulate, understand and solve specific problems. It is composed of two basic elements: special heuristics and rules that govern the use of knowledge to solve specific problems of a particular domain.

- **Facts Base: (Working Memory):** Base where the facts are on problem that has been subject a long analysis period and grows as new facts are discovered in cycles of inference.

- **Inference Engine:** Is the control structure of an expert system, the Inference Engine is an automatic mechanism that allows to answer queries from the information that is in the Knowledge Base based on rules.
  The inference engine conclude with the knowledge of the problem and its solution determines how the rules are applied.

  There are two inference strategies for the knowledge base based on rules, which are:

Backward chaining

The strategy of the expert system: finds the answer to the question, "Why was this done?"

Based on the past or what happened, the Inference Engine tracks the conditions that could be given in the past for this result. Strategy moves forward to find cause or reason. For example, the diagnosis of blood cancer in humans..

Forward chaining

It is a strategy of an expert system to answer the question, "What can happen next?"

The inference engine systematically executes the string of conditions, derivations to finally derive a result. Consider all the facts and rules and order them before you can conclude an answer.

Expert Systems Types

They use two elements: facts and rules which intertwine to form the premises

Facts: Known for particular situations, they are dynamic, ie they can be changed and stored in the working memory

Rules: are general relations between set of objects do not change unless the expert system incorporates learning elements, is stored in the knowledge base

- **User interfaces:** Is interaction between the Expert System and the end user, and is done through natural language.

SHELLS for JAVA

- CLIPS (C Language Integrated Production System)
- JESS (Java Expert System Shell)
- PROLOG (Programming in Logic)
- DROOLS (Business Rules Management System)

Below we describe some shells for java:

Clips

CLIPS is an expert system tool developed by the Division of Software Technology (STB) that provides the environment for the development and execution of expert systems, NASA/Lyndon B. Johnson Space Center. Since its launch in 1986, CLIPS has experienced improvements in its language to date and is used by thousands of programmers worldwide.

CLIPS is designed to facilitate the development of Software to model human knowledge or skill

The ways to represent knowledge in CLIPS are:

- *Rules*, are derived mainly from heuristic knowledge based on experience..
- *Deffunctions* and *generic functions*, Which are mainly procedural knowledge.
- *Object-oriented programming*, also primarily intended for procedural knowledge. The five generally accepted features of object-oriented programming are supported: classes,

message-handlers, abstraction, encapsulation, inheritance, and polymorphism. Rules may

pattern match on objects and facts.

A CLIPS program consists of rules, facts and objects. The inference engine is in charge of deciding which rules should be executed and for pattern recognition, also uses Rete algorithm.

CLIPS support algorithm forward chaining.

JESS

Using JESS, we can build expert systems in Java as it has the ability to reason, using knowledge provided by the expert through declarative rules. Jess uses an improved version of the Rete algorithm to process the rules and is a very efficient mechanism to solve the difficult problem of the match between the network of rete in several to several relationships (all against all nodes)

Each JESS rules engine contains a knowledge base called facts, this collection is working memory and is important because the rules can only react to additions, deletions and changes in memory. You can not write a rule of Jess that reacts to something else.

Jess support algorithm Forward and backward chaining.

Prolog

Prolog is a language based on non-procedural logic. This language involves constructing a knowledge base where these relationships are represented. Prolog uses an integrated installation to extract logical conclusions given by user inputs

It is based on statements by the provision of facts and rules to the database, in consultation by the operation through which the interpreter raises any questions. And determines whether the facts can be proved from the facts and rules of the database.

The inference method is based on resolution, uses the unification comparison process.

Prolog support algorithm backward chaining.

Shells Selected:

JESS of Sandia National Laboratories

The main advantage of JESS is its ease of integration with Java applications through an application interface accessible from Java itself. This is why jess is one of the main options when developing an expert system due to the independence of the Java language platform, it is flexible, JESS can be used in command line applications, GUI applications and servlets, therefore, to improve them with reasoning capabilities.

Jess have mechanisms that “simulates” backward chaining, using the do-backward-chaining function.

The rules engine of JESS uses the Rete algorithm (which constitutes a network of nodes, except for the root node (conditional pattern of a rule) and the root of a leaf (complete conditional part of the Rule) for a Coincidence with the rules with the knowledge base.Jess is actually faster compared to other shells especially in big problems, since pattern recognition allows a many-to-many comparison, where performance must predominate by the quality of the algorithm.
Rete is an algorithm that explicitly trades space for speed, so the use of memory is not insignificant, also some commands that allow sacrificing a little performance to decrease memory usage, compared to prolog whose pattern recognition is through unification or resolution

Why JESS?
The knowledge representation is write how IF - THEN rules (Easy, friendly representation) are in the rules.clp file in the project with the inference engine that “simulates” backward chaining and have ease of integration with Java Language (Because the members of the group have a high domain in this programming language).

In Jess to use backward chaining you can use the “do-backward-chaining” function after the facts deftemplate is defined.

Jess Syntax

Comment
; This is a comments

Simple patterns
(person (age ?a) (firstName ?f) (lastName ?l))

Lists
(+ 3 2) (a b c) ("Hello, World")

A simple loop
```
(while (> ?i 0)
        (printout t ?i crlf)
        (-- ?i))
```

Variables
(bind ?<var> "The value")

Global variables (or defglobals)
(defglobal [?<global-name> = <value>]+)

Deffunctions
(deffunction <function-name> [<doc-comment>]
(<parameter>*)
  <expr>* [<return-specifier>])
//You can define your own functions in the Jess rule language

defadvice
(defadvice before + (bind $?argv (create$ $?argv 1)))
//The defadvice construct lets you write some Jess code which will be executed before or after each time a given Jess function is called

Templates
```
(deftemplate template-name
   ["Documentation comment"]
   [(declare  (slot-specific TRUE | FALSE)
            (backchain-reactive TRUE | FALSE)
            (from-class class name)
            (include-variables TRUE | FALSE)
            (ordered TRUE | FALSE))]
   [extends template-name]
    (slot | multislot slot-name
      [(type ANY | INTEGER | FLOAT |
            NUMBER | SYMBOL | STRING |
            LEXEME | OBJECT | LONG)]
      [(default default value)]
      [(default-dynamic expression)])*)
```

// The deftemplate construct is the most general and most powerful way to create a template

Information of the next tablet taken from site Web JESS documentation

Defrule (defrule *rule-name* ["Documentation comment"] [(declare (salience *value*) (node-index-hash *value*) (auto-focus TRUE | FALSE) (no-loop TRUE | FALSE))] (*conditional element*)* => // take actions based on the contents of one or more facts,	**Unordered facts** (deftemplate <var> "description." (slot <var-attribute>) (slot <var-attribute>) (slot <var-attribute>(type DATA-TYPE)) (slot <var-attribute>(default description))) //Unordered facts offer this capability (although the fields are traditionally called *slots*.) **Ordered facts** (deftemplate <var> "description" (declare (ordered TRUE)))
Decisions and branching (if (> ?x 100) then (printout t "X is big" crlf) else (printout t "X is small" crlf)) (*function call*)*)	**Defquery** (defquery *query-name* ["Documentation comment"] [(declare (variables *variable*+) (node-index-hash *value*) (max-background-rules *value*))] (*conditional element*)*) //search it to find relationships between facts.

Information of the next tablet taken from site Web JESS documentation

A description of a prototype:

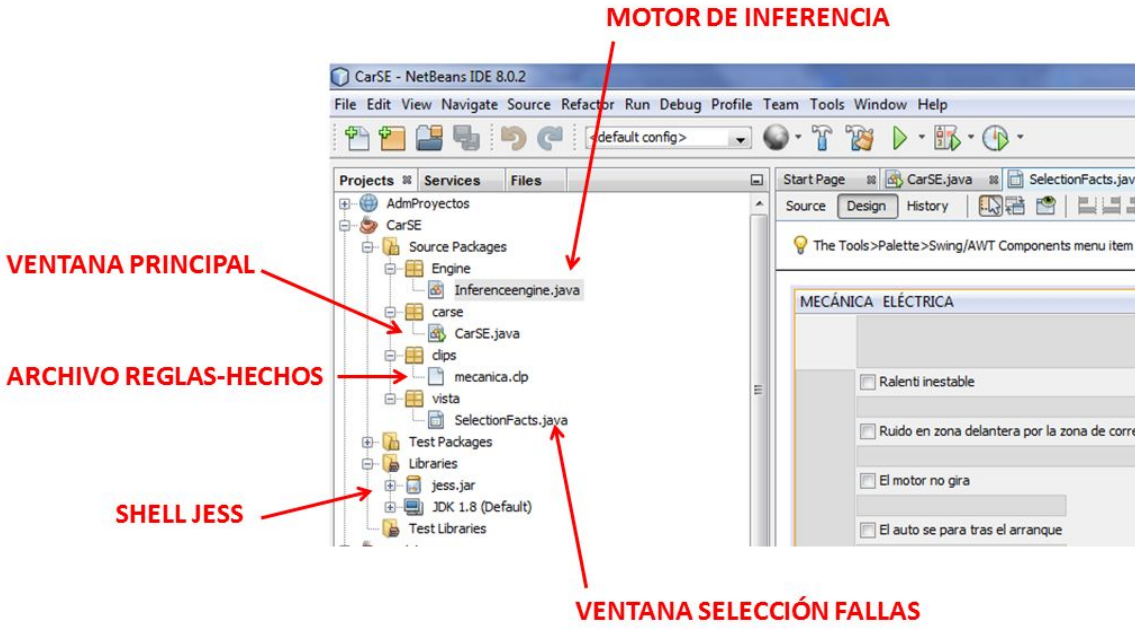
It's technically feasible to develop software using Procedural representation in JAVA environment along with JESS library (expert systems based on rules it has extensive documentation), knowledge base (we've experts in mechanics committed to contribute with knowledge) is wide and in fact is going to grow and this stored in a .clp file (based on CLIPS for easy access) so it will not allow before changes, also change the software code, that way it defragment the development components software for execution.

For the prototype the expert system as a desktop application, it used Netbeans IDE version 8.1 using the Java Programming Language, in which the Java expert system Shell library (JESS) was added where the class jess.Rete is the engine of Inference

The experts were identified by recommendation for the acquisition of knowledge, such as:
Ing. Cristian Arias - professor FIMCP ESPOL.
Ing. Sixto Alvarado - Supervision of truck maintenance at HOLCIM.
Also we consulted in mechanical workshops.

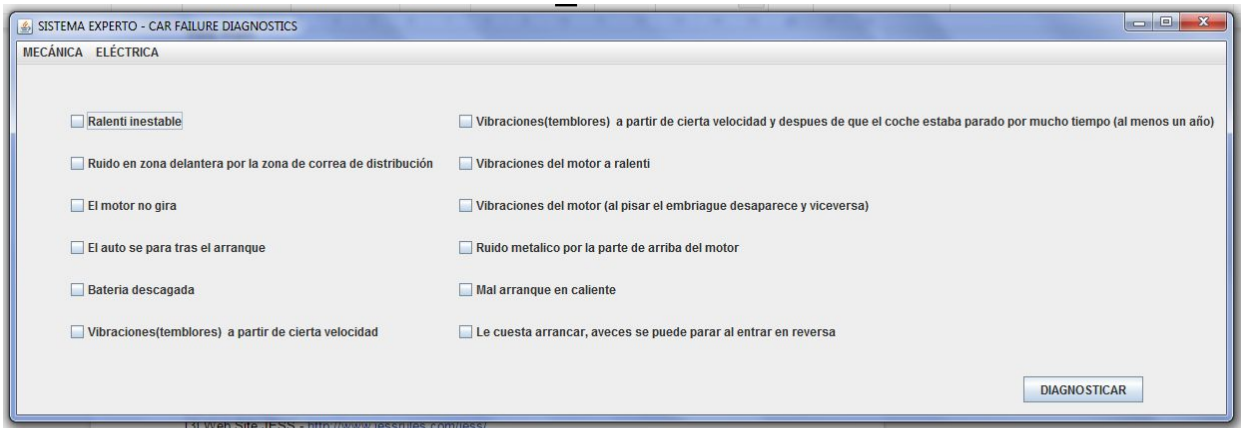
Space Work - Graphic

This is the basic structure of the software, which contains the components of the expert system.



Void main

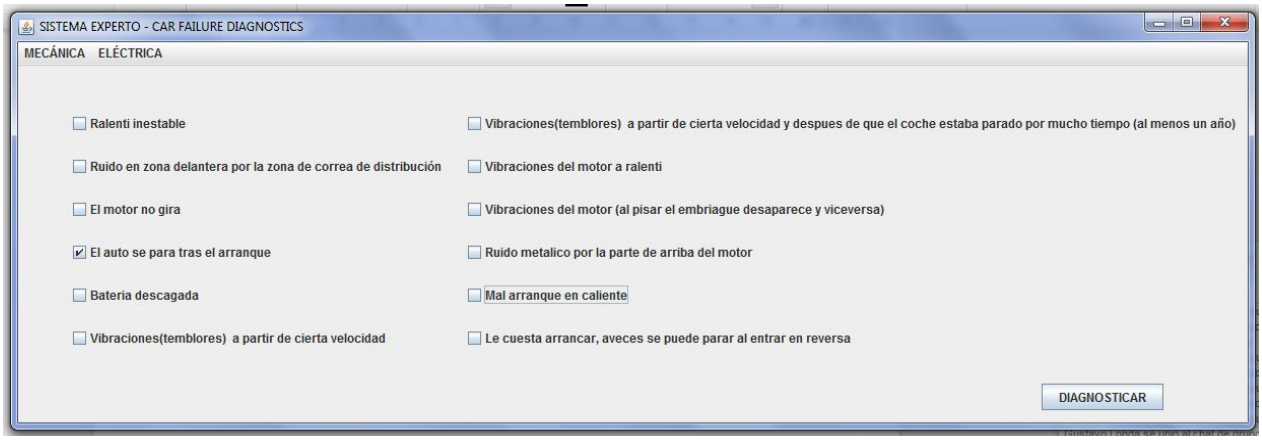
The main screen can select mechanical or electrical faults of the car.



Prototype - Expert System - Car Failure Diagnostics

Select(s) Faults

The fault-capture screen accepts the user-selected faults (working memory) of the user for further evaluation and diagnosis.



FILE MECANICA.CLP

In this file is where we going to put the rules.

The click event of the "Diagnostics" button, creates the working memory and executes the class of the inference engine reads the file "mecanica.clp" and validates the rules that are triggered and performing pattern recognition by the RETE algorithm method.

```
;;;======================================================
;;;
;;;      Expert System for Car Failure Diagnostics
;;;      Grupo#2
;;;
;;;======================================================

;;******************
;;* DEFTEMPLATES   *
;;******************

(deftemplate mecanica
  (slot fallas-mecanicas)

(deftemplate electrica
  (slot fallas-electicas)

(do-backward-chaining mecanica)
(do-backward-chaining electrica)
```

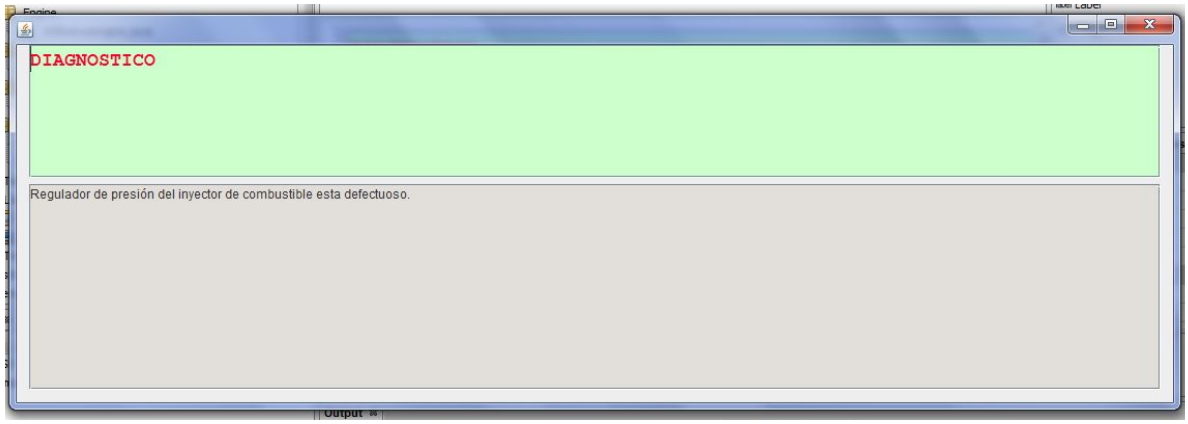
.

.

.

Show Results - Graphic example

Finally the Result window, shows the diagnosis found after the faults selected by the user



References:

[1] http://www.emse.fr/~picard/cours/ai/chapter-jess.pdf
[2] CLIPS User's Guide http://clipsrules.sourceforge.net/documentation/v630/ug.pdf
[3] Web Site JESS - http://www.jessrules.com/jess/
[4] Prolog http://groups.engin.umd.umich.edu/CIS/course.des/cis400/prolog/prolog.html
[5] http://www.cse.unsw.edu.au/~billw/cs9414/notes/prolog/intro.html
[6] http://capacitacionjava.blogspot.com/2010_09_01_archive.html
[7] http://users.ecs.soton.ac.uk/phl/ctit/ho2/node1.html
[8] http://haleyai.com/pdf/BackwardChaining.pdf
[9] https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_expert_systems.htm
[10] http://www.dsi.fceia.unr.edu.ar/downloads/IIA/presentaciones/SEarquitectura2007.pdf
[11] https://www.cs.us.es/cursos/ia2-2008/trabajos/propuesta-fjmm.html
[12] https://www.infor.uva.es/~calonso/MUI-TIC/HerramientasJess.pdf
[13] http://www.sc.ehu.es/jiwhehum2/prolog/Temario/Tema1.pdf
[14] http://www.jessrules.com/doc/61/language.html
[15] http://www.jessrules.com/jess/docs/71/constructs.html
[16] http://www.cs.upc.edu/~luigi/II/IA-2007-fall/3b-inferencia-en-agentes-basados-en-conocimiento-(es).pdf