ANÁLISIS DE LA MAQUINA DE REGLA JESS COMO HERRAMIENTA PARA EL DISEÑO DE SISTEMAS EXPERTOS ORIENTADOS A LA ENSEÑANZA

JHON ALBERTO FALCÓN ARELLANO

OMER MANUEL SALCEDO GALVÁN

UNIVERSIDAD TECNOLOGICA DE BOLIVAR

FACULTAD DE INGENIERIAS

PROGRAMA DE INGENIERIA DE SISTEMAS

CARTAGENA DE INDIAS, D. T. y C.

2009

ANÁLISIS DE LA MAQUINA DE REGLA JESS COMO HERRAMIENTA PARA EL DISEÑO DE SISTEMAS EXPERTOS ORIENTADOS A LA ENSEÑANZA

JHON ALBERTO FALCÓN ARELLANO

OMER MANUEL SALCEDO GALVÁN

PROYECTO DE GRADO PARA OPTAR AL TITULO DE

INGENIERO DE SISTEMAS

MOISES R. QUINTANA ALVAREZ

Licenciado en Matemáticas

Msc. En Informática Aplicada

DIRECTOR

UNIVERSIDAD TECNOLOGICA DE BOLIVAR

FACULTAD DE INGENIERIAS

PROGRAMA DE INGENIERIA DE SISTEMAS

CARTAGENA DE INDIAS, D. T. y C.

2009

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

_____

_____

_____

Firma Comité de Investigación

_____

Firma del Evaluador

_____

Firma del Evaluador

Cartagena de Indias D.T. y C., Bolívar, Colombia

A los _____ días de _____ de _____.

A Dios por regalarme la vida. Gracias por ser mi fiel amigo.

A mis Padres Oscar Salcedo y María Galván por brindarme su amor y apoyo incondicional en cada momento de mi vida, no tengo la vida misma para agradecerles.

A mis hermanos Eder y Weimer por brindarme el privilegio de la fraternidad y los buenos momentos que hemos pasados juntos.

A mi Esposa Kattia por ser un apoyo en momentos que sentía caer

A mi hijo que ha sido la mejor razón para seguir adelante.

Al profesor Moisés Quintana por ser piedra angular en mi proceso de formación. Gracias por su ayuda en el momento que mas lo necesitaba, de corazón.

Y especialmente a todas las personas y amigos que me enseñaron a nunca perder las esperanzas.

Omer Manuel Salcedo Galván.

Mi mayor agradecimiento es para Dios, sin el jamás hubiese podido llegar a conseguir cada una de las metas que hasta ahora he podido alcanzar. Él me ayudo a vencer obstáculos que muchas veces creí no poder sobrepasar, entre mas complejo era el obstáculo, mayor era su grandeza. De nuevo muchas gracias señor, por permitirme alcanzar esta nueva meta en mi vida.

De igual forma un agradecimiento especial a todas y cada una de las personas que he conocido a lo largo de mi vida, y que han sido parte importante para este logro, unas muy cercanas, otras no tanto, pero todos, absolutamente todos, con sus diferentes acciones, ya sea buenas, o no tan buenas, pero que me han aportado experiencia y conocimiento en el constante proceso de formación.

Jhon Alberto Falcón Arellano.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

INTRODUCCIÓN

En el campo de la informática educativa se ha buscado, desde sus orígenes aplicaciones o herramientas que en el momento de interactuar con el usuario final éstas tengan un comportamiento humano.

Ante esta problemática se han tratado de desarrollar Sistemas capaces de emular las actividades del ser humano para resolver problemas de distinta índole en la que, con la interacción de los usuarios, logre explicar sus razonamientos o base de conocimiento o adquirir nuevos conocimientos de acuerdo al contexto o escenario donde se desarrolla la actividad. Estos sistemas se conocen como Sistemas Expertos.

Estos sistemas permiten emular el comportamiento de un experto en un dominio concreto y en ocasiones son usados por estos. Su objetivo es buscar una mejor calidad y rapidez en sus respuestas dando así lugar a una mejora de la productividad del experto.

La meta de los sistemas experto es imitar las actividades realizadas por un humano para resolver problemas dentro de un escenario específico. Para ello estos sistemas puede utilizar para la resolución de problemas un conocimiento

declarativo (hechos sobre objetos, situaciones) o un conocimiento de control (información sobre el seguimiento de una acción) o ambos.

Los sistemas expertos basados en reglas son sistemas expertos que se caracterizan por que utilizan un conocimiento declarativo en donde éste aplica reglas heurísticas apoyadas generalmente en lógica ya sea proposicional, de predicados o difusa, para su evaluación y aplicación.

En el marco de estas aplicaciones hay un delimitado número de herramientas que permiten el desarrollo de estas aplicaciones de entre las cuales podemos mencionar, LIST, CLIPS, PROLOG, entre otras, muy buenas pero que, con el paso del tiempo y por estar ligado a una arquitectura específica han hecho que su uso se límite o que su potencial sea poco atractivo ante el surgimientos de nuevas arquitecturas y nuevos sistemas operativos, especialmente una gama de ellos que están sin explorar tras la popularización del software libre y las apariciones de nuevas plataformas tanto de hardware como software.

Ante este inconveniente de la diversidad de plataformas, aparece un lenguaje que permite que sus aplicaciones puedan ser ejecutadas en una máquina virtual, dando origen un conjunto de programas cuya ejecución sea independiente de la arquitectura de hardware y del sistema operativo. Ese lenguaje es JAVA que unido a su plataforma de maquina virtual han permitido cerrar un poco la brecha acerca de la compatibilidad y portabilidad de las aplicaciones ya que estas corren

en una plataforma que no se ata a las especificaciones de la arquitectura de hardware como ni las del sistemas operativo en el cual se ejecutan.

En la actualidad JAVA está incorporado en la mayoría de los sistemas operativos y probado en la mayoría de las arquitecturas de hardware, estando desde los electrodomésticos como su lavadora, pasando por sus herramientas de comunicación tales como su dispositivo móvil celular hasta los equipos de computo más poderosos en cuanto a su capacidad de almacenamiento y procesamiento, desde un sistema procesador de cinta, hasta su navegador Web.

Estas características que tiene el lenguaje de programación JAVA, fue la razón fundamental para que Ernest Friedman-Hill del instituto *Sandia National Laboratories* desarrollara una herramienta que tomara el legado de CLIPS unido con la fortaleza de la multiplataforma de JAVA naciendo JESS.

JESS es, por lo tanto, una maquina basada en reglas para la plataforma JAVA éste es un súper conjunto del lenguaje de programación basado en reglas CLIPS. Por ende JESS nos ofrece un lenguaje de programación basado en reglas unido con un potente lenguaje de programación como lo es JAVA en el cual se pueden desarrollar Java servlets, EJBs, applets y aplicaciones que usen una base de conocimiento utilizando reglas declarativas para obtener conclusiones y realizar inferencias.

A pesar de la gran capacidad de JESS como herramienta de diseño de sistemas expertos y el potencial que puede brindar su implementación en el campo de la enseñanza y otros campos similares, es poco el uso que se hecho de dicha herramienta en este aspecto, en parte debido a que no existen muchos textos o documentos investigativos que muestren las áreas en las cuales se podrían aprovechas dichas ventajas.

Bajo estas condiciones este proyecto busca en su meta definitiva valorar JESS como una alternativa viable para desarrollar sistemas que poseen estas características y, además, utilizarlos con el propósito de desarrollar productos de gran utilidad en el campo de la enseñanza.

1 PLANTEAMIENTO DEL PROBLEMA

El conocimiento se ha convertido en el bien más preciado para el hombre. Aun desde la antigüedad, las grandes civilizaciones y culturas atesoraban el conocimiento, inclusive por encima de bienes materiales como el oro y la plata.

El conocimiento era la característica diferenciadora por excelencia entre una cultura y otra, y en nuestra época esa realidad no ha cambiado en nada. Es así como una frase tan antigua como lo es “el conocimiento es poder” ha permanecido intacta hasta nuestros días. Pero, ¿como poder mantener ese conocimiento y más aun ampliarlo si no es a través de la enseñanza?.

Es aquí donde esas dos palabras “conocimiento” + “enseñanza” se unen para el avance de lo que hoy conocemos como ciencia.

Bajo estas premisas queda clara la importancia de ampliar el conocimiento existente ya que son los pilares fundamentales para continuar con el avance científico.

Pero en el mundo actual, en el que cada vez más la enseñanza se centra en el auto-aprendizaje y el cual es además marcadamente competitivo, a causa de que los mismos tutores están en un constante proceso de adquirir más conocimiento, se hace necesario el análisis y posterior implementación de herramientas

especializadas que permitan ampliar dicho conocimiento y redireccionarlo hacia los estudiantes por medio de una apropiado sistema de enseñanza, en el cual se aprende a medida que se enseña.

Esta ampliación del conocimiento lo podremos potencializar por medio del análisis y la implementación de herramientas que se conocen como sistemas expertos, y en su desarrollo, entran en el juego un conjunto de herramientas para la creación de los mismos, entre las que podemos mencionar JESS.

Teniendo en cuenta todo lo anteriormente mencionado, se decidió iniciar una investigación que permitiera analizar el motor de reglas JESS y determinar la forma como la implementación de una herramienta podría ayudar a potencializar los métodos de enseñanza actuales.

Son varias las aplicaciones que actualmente se realizan mediante la implementación de sistemas expertos, en las cuales su nivel de capacidad es tan bueno como el de un experto humano. De estas son pocas las que se enfocan en el aspecto de la enseñanza, sin embargo, el potencial de los sistemas expertos en este campo es enorme.

Es aquí donde da lugar el análisis de la máquina de reglas de JESS como herramienta para el diseño de sistemas expertos orientados a la enseñanza.

2 JUSTIFICACIÓN

Se toma la herramienta JESS como objeto de estudio debido a que ésta ofrece un conjunto de características que permiten obtener productos de software con las propiedades sobresaliente de los sistemas expertos y, que a su vez, puedan ser utilizados en cualquier sistema operativo y arquitectura de hardware que tenga JAVA instalado.

La meta es evaluar la factibilidad de esta herramienta para el desarrollo de aplicaciones que emulen un comportamiento humano. Si se logra este principio básico, entonces se pueden utilizar dicho potencial para desarrollar aplicaciones que ayuden al fortalecimiento pedagógico del saber humano dentro de un campo especifico de estudio, para ello puede valerse de la didáctica, utilizándose para probar demostraciones o ir mas allá a tal punto de obtener sistemas que nos apoyen en la toma de decisiones ante un evento en especial, o mejor aun; que sean, con la prudencia y delicadeza que amerita este caso, guías ante el marco del aprendizaje, en donde la vía virtual esta tomado mucha envergadura.

JESS nos ofrece un mundo nuevo en donde ahora las aplicaciones basadas en conocimientos tienen las ventanas abiertas hacia cualquier ambiente donde se puedan ejecutar apoyados en la multiplataforma, por lo tanto se obtendría una herramienta capaz de desarrollar sistemas expertos a tal punto que nos ayudarían desde los sistema de navegación de un avión, barco o submarino, hasta un juego de ajedrez dentro de un dispositivo móvil celular, y sobre todo nos ofrece una nueva gama de aplicaciones que nos ayuden en el proceso de aprendizaje en un ámbito determinado disponibles al alcance de nuestras manos.

Además de lo anterior, en nuestro entorno, como es en la Universidad Tecnológica de Bolívar, no se ha realizado un estudio con miras al a obtención de una serie de productos tales como un conjunto de aplicativos así como un documento que pueda servir como guía de apoyo y referencia a cualquier estudiante que quiera saber un poco acerca de los sistemas basados en conocimientos y su aplicabilidad dentro de nuestros sistemas virtuales de aprendizajes.

Por estas razones es que se da inicio a esta investigación.

3 OBJETIVOS

3.1 OBJETIVO GENERAL

Analizar la maquina de regla JESS con miras al diseño e implementación de Sistemas Expertos orientados a la Enseñanza.

3.2 OBJETIVOS ESPECIFICOS

1 Indagar sobre los principios que rigen la máquina de reglas de JESS.

2 Determinar los requerimientos, las características y las capacidades del motor de inferencias de JESS.

3 Diseñar un conjunto de aplicativos tomando como base el paradigma de la ingeniería del conocimiento, de tal forma que permita representar la dinámica de los proceso de enseñanza.

4 Valorar y mostrar a JESS como una alternativa viable para el desarrollo de herramientas de gran utilidad en el campo de la enseñanza.

5 Documentar las fortalezas de JESS como una de las herramientas más versátiles y completas en el desarrollo de sistemas expertos

4 ALCANCE

Esta investigación tiene como principal meta entregar un conjunto de productos de software realizados con la herramienta de diseño de sistemas expertos JESS así como su respectiva documentación a fin de mostrar su fortaleza especialmente en lo que respecta al área de la informática educativa.

Los productos a entregar con base a los objetivos planteados en este proyecto son:

1. Un conjunto de aplicativos desarrollados bajo el paradigma de la ingeniería de conocimientos específicamente en el diseño de sistemas expertos.

2. Un conjunto de documentos que explican a través de los paradigmas del proceso de ingeniería de software, el respectivo análisis y diseño de dichos aplicativos (documentación basados en el estándar UML en su versión 2.0)

3. Un documento que tiene por objeto mostrar la fortaleza de JESS para la realización de Sistemas Expertos, que pueden ser objeto de estudio e implementación para la informática educativa.

5 MARCO TEÓRICO

5.1 SISTEMA BASADOS EN CONOCIMIENTOS

5.1.1 Agente

De acuerdo a Russel y Norvig[1] un agente es todo aquello que puede considerarse que percibe su ambiente mediante sensores y que responde o actúa en tal ambiente por medio de efectores.

Por lo tanto un agente es cualquier sistema que esta interrelacionado con su entorno o espacio de aplicación y que reacciona a los estímulos que recibe de ese ambiente. Ante esta situación se puede dar cuenta que aplica en los sistemas computacionales pero de acuerdo a la esencia de la definición encaja perfecto en un somero grupo de sistemas computacionales, es decir que un agente es un sistema pero no todo sistema es agente.

Al mirar por ejemplo, un sistema generador de números aleatorios, se puede observar que éste sistema no recibe entradas de su entorno donde opera. Por ende al carecer de estructuras que le permitan obtener información de su entorno

[1]RUSSELL Stuart, NORVIG Peter. Inteligencia Artificial un enfoque moderno. México: Prentice Hall Hispano Americana, 1996. p 33

o ambiente entonces no puede responderle al entorno de la misma manera. Por lo tanto este sistema no cumple con los requisitos para que sea un agente.

En cambio si al mirar un sistema de diagnostico medico que recibe los síntomas de un paciente y con base a esos síntomas pueda diagnosticar una posible enfermedad y el procedimiento médico a seguir, este sistema si se ajusta a la definición de agente ya que recibe un estimulo del entorno (lista de síntomas de un paciente) reacciona a tal ambiente (retornando el posible diagnostico médico para el paciente).

Ahora conocido el concepto de agente entonces se mira a fondo el proceder de los éstos desde el punto de vista de su desempeño a su contexto particular. Este desempeño se puede tomar como medida de la racionalidad de un agente.

5.1.1.1 Agente Racional

Russel y Norvig afirman que un agente es racional cuando hace lo correcto sabiendo que lo correcto es aquello que le permite al agente obtener el mejor desempeño. La medida del desempeño es un criterio que sirve para definir que tan exitoso ha sido un agente desde el punto de vista cualitativo y cuantitativo[2].

Ante lo anterior se puede saber a cuando un agente es racional mirando el desempeño de éste al momento de realizar un conjunto de actividades que le

[2] Cf. RUSSELL Stuart, NORVIG Peter. Inteligencia Artificial un enfoque moderno. México: Prentice Hall Hispano Americana, 1996. p 33 - 34

permitan una interacción con el ambiente o contexto en el que se encuentra pero ¿será posible evaluar en un momento dado que tan racional pueda ser un agente?

Ante esta inquietud Russel y Norvig dicen:

El carácter de racionalidad de lo que se hace en un momento dado dependerá de cuatro factores:

1. De la medida con la que se evalúa el grado de éxito logrado
2. De todo lo que hasta ese momento haya percibido el agente. A esta historia perceptual completa se le denomina secuencia de percepciones.
3. Del conocimiento que posea el agente acerca del medio.
4. De las acciones que el agente pueda emprender.[3]

5.1.1.1.1 Agente Racional Ideal

Se puede definir que un agente racional es ideal cuando cumple a cabalidad y con un alto grado de perfección su desempeño llevándolo a su máximo nivel. La cual es muy acertada según Russel y Norvig que lo definen de la siguiente manera: “En todos los casos de posibles secuencias de percepciones, un agente racional deberá emprender todas aquellas acciones que favorezcan obtener el máximo de su medida de rendimiento, basándose en las evidencias aportadas por la secuencia de percepciones y en todo conocimiento incorporado en tal agente (,,,)

[3] RUSSELL Stuart, NORVIG Peter. Inteligencia Artificial un enfoque moderno. México: Prentice Hall Hispano Americana, 1996. p 35

El concepto de agente permite pensar en el como una herramienta para el análisis de sistemas como una caracterización absoluta que tajantemente divida al mundo en agentes y no agentes"[4].

De acuerdo a lo expuesto anteriormente se puede afirmar que la forma de actuar de un agente esta estrechamente ligada a la secuencia de percepciones en un momento especifico, por ende es posible generar un proceso de caracterización para dicho agente obteniendo un mapeo de secuencias de percepciones para acciones o lista de acciones y reacciones del agente (que en la mayoría de los caso es tediosa por su tamaño). Para los agentes ideales se definen los mapeos ideales como el conjunto de especificaciones de los tipos de acciones las cuales puede contemplar el agente como respuesta a una determinada secuencia de percepciones que constituye el diseño del agente en mención.[5]

Vale la pena aclarar que para realizar dicho mapeo no hay que hacer una lista completa de todas las entradas que recibe el agente y las posibles percepciones que realice éste. Para la realización del mapeo se puede utilizar métodos, procedimientos y algoritmos que, por su efectividad ayude sobremanera a optimizar el proceso de seguimiento del comportamiento para el agente mismo.

[4] Ibíd. p. 35.

[5] Cf. RUSSELL Stuart, NORVIG Peter. Inteligencia Artificial un enfoque moderno. México: Prentice Hall Hispano Americana, 1996. p 36.

Por ejemplo, se tiene un agente sencillo cuya finalidad es obtener un numero real el cual es la raíz de la ecuación *f(x)*, la secuencia de percepciones de este agente es la secuencia de números que actuarían como conjunto de entrada para la función *f(x)* un mapeo ideal se produce cuando la percepción es un número real negativo *x* tal que *f(x)=0* siendo $f(x)= x^3 - 2x^2 + 5$ con una precisión de 5 cifras decimales. Para la caracterización de esta secuencia de percepciones y acciones, no es necesario que el diseñador o analista del agente construya una tabla de entradas y salidas de la función $f(x)= x^3 - 2x^2 + 5$ para valores de $x < 0$; este último puede basarse en técnicas de búsquedas de raíces que le permitan obtener un análisis mas óptimo Para este ejemplo, una técnica que optimiza el proceso del mapeo ideal es el método iterativo de Newton Raphson.

El Método de Newton-Raphson es ampliamente utilizado para encontrar las raíces de la ecuación f(x)=0, ya que converge rápidamente, la contra es que uno debe conocer la derivada de f(x) y se necesita una aproximación inicial a la raíz. [6]

$$x_n = x_{n-1} - f(x_{n-1})/f'(x_{n-1})$$

Ahora aplicando el método para realizar el mapeo se plante el siguiente algoritmo cuyo procedimiento da una solución óptima al mapeo del agente en cuestión.

Algoritmo RaizFuncion

sea $f(x)= x^3 - 2x^2 + 5$,

[6] PROGRAMACIÓN GRAFICA. Método de Newton-Raphson [En línea]. <http://www.geocities.com/valcoey/newton_raphson.html> [con acceso 21 de Abril de 2008]

sea $f'(x)= 3x^2 – 4x$

sea x_0=1: real

sea z= x_0: real

sea k=0: entero

inicio

mientras ($|z – x_k| < 10^{-5}$) hacer

x_k=z;

z= x_k - $f(x_k)$/ $f'(x_k)$;

k=k+1;

fin mientras

devolver z;

fin

Obteniéndose el siguiente mapeo:

x	f(x)	f'(x)	z	ABS(z-x)	acción
1	4	-1	5		
5	80	55	3,54545455	4	sigo
3,54545455	24,42674681	23,5289256	2,50729636	1,45454545	sigo
2,50729636	8,189136311	8,83041963	1,57991842	1,03815819	sigo
1,57991842	3,951416639	1,16875298	-1,8009642	0,92737793	sigo
-1,8009642	-7,32832118	16,934273	-1,36821334	3,38088263	sigo
-1,36821334	-1,30532149	11,0888766	-1,25049885	0,43275086	sigo
-1,25049885	-0,08295904	9,69323752	-1,2419404	0,11771449	sigo
-1,2419404	-0,00042065	9,59500953	-1,24189656	0,00855845	sigo
-1,24189656	-1,1005E-08	9,59450749	-1,24189656	4,3841E-05	sigo
-1,24189656	0	9,59450747	-1,24189656	1,147E-09	hecho

Tabla 1: Mapeo del algoritmo de Newton - Rapson

Este ejemplo muestra la relación inherente entre el mapeo ideal y el diseño de un agente ideal. Como el desempeño del agente influye sobremanera al ambiente donde se relaciona, entonces es posible desarrollar e implementar una serie de agentes capaces de resolver conjunto variados de tareas a través de una serie de acciones en una ilimitada variedad de de ambientes.

Un agente es autónomo cuando tiene presencia de algún conocimiento integrado, es decir que un sistema será autónomo en la medida de que su conducta este definida por su propia existencia. El grado de autonomía de un agente en la mayoría de los casos es guiado por el diseñador de éste. En donde la experiencia obtenida por el agente es un pilar fundamental en le grado de autonomía. Ya que si el agente carece de experiencias o en e caso promedio de que en su tiempo de vida halla adquirido poca experiencias, este, de acuerdo al diseño y las orientaciones asignadas por el diseñador del agente, tendría un comportamiento aleatorio, hasta que efectivamente el agente se valla dotando de inteligencia artificial con ciertos conocimientos iniciales y de capacidad de aprender.

5.1.1.2 Estructura de los agentes inteligentes

los agentes inteligentes se caracterizan porque todos tienen dos componentes fundamentales; el programa de agente en el cual es una función o algoritmo que permite implementar el mapeo de percepciones - acciones de un agente bajo estudio y la arquitectura la cual es un dispositivo de computo donde se ejecuta el

programa de agente. Vale la pena recordar que el programa elegido debe ser compatible con donde se va a ejecutar. Es decir si sabemos que se va a probar un agente en un sistema IBM PC, entonces el programa el cual va a probar dicho agente debe ser compatible para esa plataforma de hardware.

Russel y Norvig[7] clasifica los programas de agentes en 4 tipos:

- Agentes de reflejo simple.
- Agentes basados en metas.
- Agentes basados en utilidad.

5.1.1.2.1 Agente de Reflejo Simple.

Son agentes en la cual sus programas de agentes no consideran utilizar el recurso de tablas explicitas si no que utiliza fragmentos de la tabla en la cual se presenta con cierta frecuencia algunas asociaciones de entradas/salidas que el agente procesa. Es decir, el agente al realizar un proceso, como respuesta a una entrada y se establece alguna condición definida dentro del programa agente esto activa la conexión definida en el programa del agente y la acción correspondiente.

La siguiente figura muestra el esquema de una agente de reflejo simple:

[7] RUSSELL Stuart, NORVIG Peter. Inteligencia Artificial un enfoque moderno. México: Prentice Hall Hispano Americana, 1996. p 42.

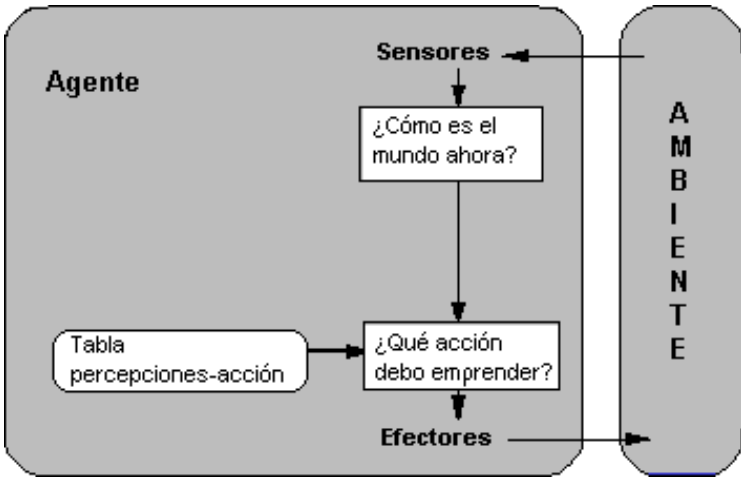

Ilustración 1: Diagrama esquematizado de una Agente de Reflejo Simple

Un agente reflejo tiene posibilidad de presentar un comportamiento más inteligente cuando es capaz de interpretar el estado del mundo en base a cierto conocimiento sobre:

Su estado interno: ¿Cómo me encuentro YO en este momento?

Cómo evoluciona el mundo: ¿Cuál es mi teoría del mundo (en particular de aquellas porciones de éste que no puedo percibir ahora, pero que sé que existen)?

El potencial efecto de sus acciones: ¿Qué ocurre en el mundo y en mi interior Cuando realizo una determinada acción?[8]

En este caso lo denominaremos un agente reflejo con un estado interno.

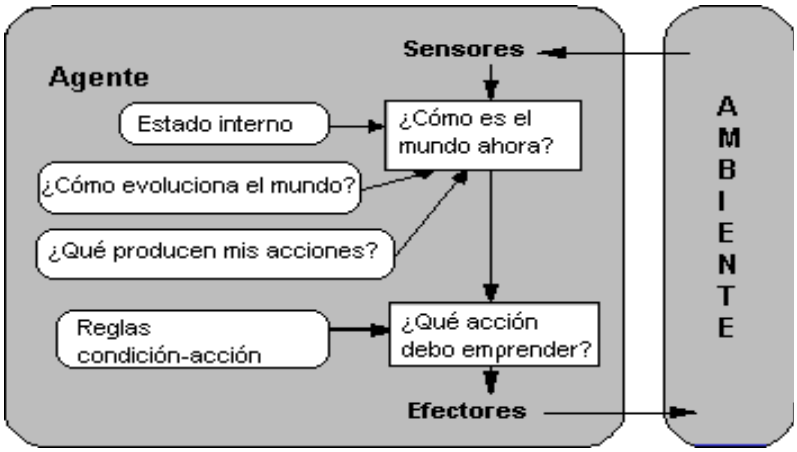

Ilustración 2: Un Agente de reflejo con estado interno

5.1.1.2.2 Agentes basados en metas.

En muchas ocasiones, la simple reacción o reflejo con mayor o menor información, puede ser insuficiente para lograr un comportamiento adecuado, siendo conveniente introducir el conocimiento del objetivo o meta que el agente debe lograr como elemento participante en el proceso de toma de decisiones de dicho agente, incrementando, de este modo, su grado de "inteligencia" al considerar

[8]HERRERA Mónica, PRIETO Jaqueline, LOPEZ Miguel, MARTINEZ Enrique, DE LAS CASAS flora. "Agentes inteligentes" [en Linea] <http://www.depi.itch.edu.mx/apacheco/expo/html/ai12/> [con acceso junio 18 de 2008]

durante dicho proceso las posibles condiciones futuras, tanto del mismo agente, como del mundo que lo rodea en caso de tomar una acción determinada.

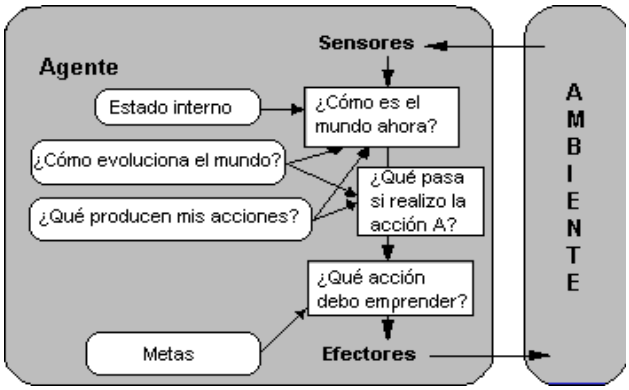

Ilustración 3: Agente con metas explicativas

5.1.1.2.3 Agentes basados en utilidad.

Un Agente Basado en utilidad o agente racional ideal es aquel que en todos los casos de posibles secuencias de percepciones, emprende todas aquellas acciones que favorezcan obtener el máximo de su medida de rendimiento, basándose en las evidencias aportadas por la secuencia de percepciones y en todo el conocimiento incorporado en tal agente.[Tipos-a]

La racionalidad está relacionada con:

- La medida con la que se evalúa el grado de éxito logrado.
- La secuencia de percepciones, es decir, todo lo que hasta ese momento el agente haya percibido.

- El conocimiento que el agente posea del mundo y de sí mismo.

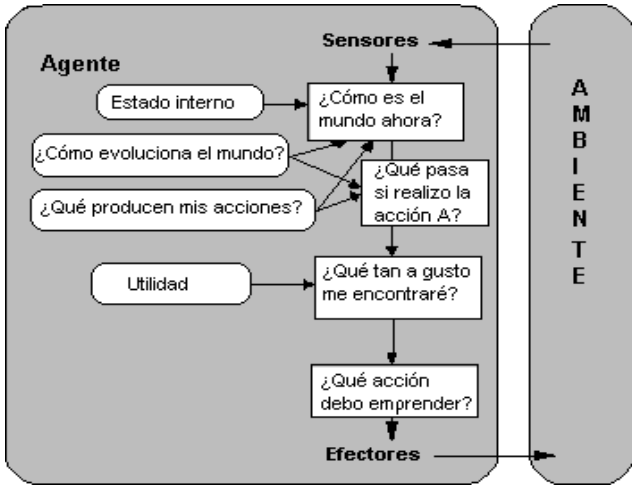

Ilustración 4: Agente basado en utilidad

5.2 SISTEMAS EXPERTOS

5.2.1 Definición

De acuerdo con Giarratano y Riley[9] un sistema experto es un programa que emula el comportamiento de expertos humanos en la resolución de problemas de diversos campos y su habilidad para tomar decisiones.

Según Stevens[10]:

[9] GIARRATANO Joseph, RILEY Gary. Sistemas expertos principios y programación. Editorial: Thomson Learning. Edición: 2000. p. 2

“Los sistemas expertos son máquinas que piensan y razonan como un experto lo haría en una cierta especialidad o campo. Por ejemplo, un sistema experto en diagnóstico médico requeriría como datos los síntomas del paciente, los resultados de análisis clínicos y otros hechos relevantes, y, utilizando estos, buscaría en una base de datos la información necesaria para poder identificar la correspondiente enfermedad. [. . .] Un Sistema Experto de verdad, no sólo realiza las funciones tradicionales de manejar grandes cantidades de datos, sino que también manipula esos datos de forma tal que el resultado sea inteligible y tenga significado para responder a preguntas incluso no completamente especificadas.”

Una definición formal[11] de los sistemas expertos, aceptada por muchos autores, es la aprobada por el Grupo Especialista en Sistemas Expertos de la Sociedad Británica de Ordenadores, que los define de la forma siguiente:

"Un sistema experto es visto como la incorporación en un ordenador de un componente basado en el conocimiento, que se obtiene a partir de la pericia (conocimiento técnico) de un experto, de tal forma que el sistema pueda ofrecer asesoramiento inteligente o tomar una decisión inteligente sobre una función del proceso. Una característica adicional deseable, que muchos considerarían

[10] CASTILLO Enrique, GUTIÉRREZ José Manuel, HADI Ali. Sistemas Expertos y Modelos de Redes Probabilísticas. Academia Española de Ingeniería. Disponible en linea <http://personales.unican.es/gutierjm/papers/BookCGH.pdf> [con acceso el 15-12-2008] p 14.

[11] SÁNCHEZ Antonio. Aplicación de los Sistemas Expertos en Contabilidad. Departamento de Contabilidad, Universidad de Valencia. Disponible en línea <http://ciberconta.unizar.es/Biblioteca/0002/Sanchez95.html> [con acceso el 12 -12-2008].

fundamental, es la capacidad del sistema, si se le solicita, de justificar su propia línea de razonamiento de un modo directamente inteligible para el interrogador...". (Connell, 1987, p. 221; Prado, 1991, p. 443).

Normalmente los expertos resuelven problemas de su especialización con base en la experiencia, la cual consta de conocimientos de hechos y soluciones de problemas, de tal forma que sean almacenables en un computador y procesables por un programa. Por tanto, los sistemas expertos son programas computacionales en los cuales se han reflejado conocimientos humanos. Pero debido a que el conocimiento humano es muy complejo, tenemos que restringir su campo de acción a ciertas "imitaciones técnicas"[12], y a campos de especialización muy determinados.

Los sistemas expertos hacen parte de una clase relativamente joven de software, el cual se origina como un subconjunto del campo de estudio de la llamada inteligencia artificial, por lo que los sistemas expertos son una rama de esta ultima, la cual hace un amplio uso del conocimiento especializado para resolver problemas de la misma forma que lo haría un especialista humano.

El conocimiento de un sistema experto puede obtenerse ya sea por la experiencia o por la consulta de los conocimientos que suelen estar disponibles en personas capacitadas y libros especializados. Este proceso se denomina ingeniería del

[12] NEBENDAHL Dieter. Sistemas Expertos: experiencia de la practica. Barcelona, ES. Editorial: Marcombo, 1988. p 1

conocimiento, y es la forma como el sistema experto adquiere la experiencia y el conocimiento que posee un experto humano.

Como cualquier nueva tecnología, existe aun mucho por aprender en lo referente a los sistemas expertos, debido a que se debe entender que se tratan de programas de computador, y por amplia que sea su base de conocimiento no poseen la creatividad, la imaginación y la conciencia, pero también es mucho el potencial y las ventajas que se pueden obtener con la implementación de los mismos en los distintos aspectos del desarrollo humano.

5.2.2 Características

Una característica principal de los sistemas expertos es la separación[13] entre el conocimiento (reglas y hechos) y su procesamiento, esto sumado a una interfaz de usuario y un componente explicativo detallado sobre cada uno de los razonamientos del experto.

Las reglas deterministas o reglas de inferencia constituyen la más sencilla de las metodologías utilizadas en sistemas expertos, son afirmaciones lógicas que relacionan dos o más objetos, y están constituidas por una premisa y una conclusión, ambas consisten en una expresión lógica como una o más afirmaciones, las cuales se conectan por medio de operadores lógicos “Y”, “O”, “NO”.

[13] NEBENDAHL Dieter. Sistemas Expertos: Introducción a la técnica y aplicación. Barcelona, ES. Editorial: Marcombo, 1988. p 33

Los hechos determinan las afirmaciones que sirven para representar conceptos, datos, objetos, etc. Y el conjunto de hechos que describen el problema es la base de hechos o memoria activa.

La base de conocimiento contiene el conjunto de reglas que definen el problema, y el motor de inferencia saca las conclusiones aplicando la lógica clásica a estas reglas.

Entre otras principales características de un sistema experto[14] podemos mencionar:

1. Alto desempeño: En el cual el sistema ha de tener la capacidad de responder a un nivel de competencia igual o superior al de un especialista humano. Lo cual equivale a que la calidad de la respuesta dada por el sistema experto debe ser muy alta.

2. Asimilación del conocimiento: El sistema debe ser capaz de aprender de los expertos humanos, y adquirir el conocimiento y la experiencia de estos.

3. Tiempo de respuesta adecuado: El sistema deberá responder en un tiempo igual o inferior al tiempo requerido por un experto humano para emitir una solución a un problema o a una decisión.

[14] GIARRATANO Joseph, RILEY Gary. Sistemas expertos principios y programación. Editorial: Thomson Learning. Edición: 2000. p. 8-9

4. Confiabilidad: El sistema debe ser totalmente confiable y no propenso a fallas, de lo contrario caería en desuso.

5. Comprensible: El sistema debe ser capaz de explicar los pasos de sus razonamiento mientras se ejecutan, de tal manera que sea totalmente comprensible, en lugar de ser solo una "caja negra", que produce respuestas que muchas veces no tienen forma de ser explicadas. Esto se resume en presentar sus conclusiones a los usuarios humanos de igual manera que un experto humano, el cual debe justificar, clarificar y explicar su modo de razonamiento e incluso instruir al interlocutor.

6. Flexibilidad: El sistema debe contar con un mecanismo eficiente para añadir, modificar y eliminar el conocimiento y la información que posee.

5.2.3 Diseño

Entre los componentes y elementos del diseño[15] de un sistema experto podemos mencionar:

- La base de conocimientos: Contiene el conjunto de reglas que permite representar el conocimiento del dominio del experto, así como las experiencias de los especialistas de un campo determinado. Normalmente los conocimientos son de tipo declarativos, por lo cual generalmente la base del conocimiento es una descripción de los conocimientos del experto.

[15] NEBENDAHL Dieter. Sistemas Expertos: introducción a la técnica y aplicación. Barcelona, ES. Editorial: Marcombo, 1988. p 33-39

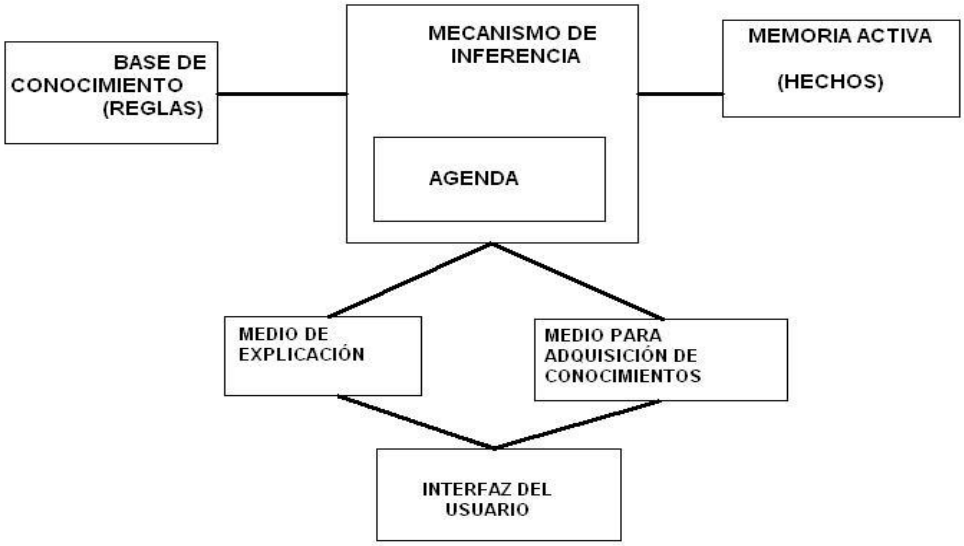

Ilustración 5: Estructura de un sistema experto basado en reglas

- El mecanismo de inferencia: El cual permite que el experto emule las estrategias de solución de uno o varios especialistas, esto lo hace seleccionando y ejecutando las reglas apropiadas, de tal forma que permita obtener las inferencias adecuadas para resolver el problema.

- Agenda: Contiene una lista con las prioridades asignadas a las reglas, la cual se origina dentro del mecanismo de inferencia y cuyos patrones satisfacen los hechos de la memoria activa.

- El componente explicativo: Este sirve para explicarle a un usuario la estrategia o estrategias de solución del problema y las bases sobre las cuales realizo el razonamiento para inferir tales estrategias.

- La interfaz de usuario: Este permite la comunicación entre el usuario y el sistema experto por medio de un lenguaje que resulte lo mas natural posible para la realización de consultas y la explicación de razonamientos y estrategias.

- El componente de adquisición: El cual ofrece una ayuda para la estructuración e implementación del conocimiento (hechos y reglas) dentro de la misma base del conocimiento, de tal forma que permita introducir conocimientos en el sistema.

Para construir un sistema experto existen metodologías de diseño específicas [GRE88, HAY83, IGN91]. A continuación se presenta un procedimiento general de diseño[16]:

[16] PACHECO Alberto. Metodología de Diseño de Sistemas Expertos. Instituto Tecnológico de Chihuahua. Disponible en línea <http://www.depi.itch.edu.mx/apacheco/ai/metodolo.htm > [con acceso 2009-05-04]

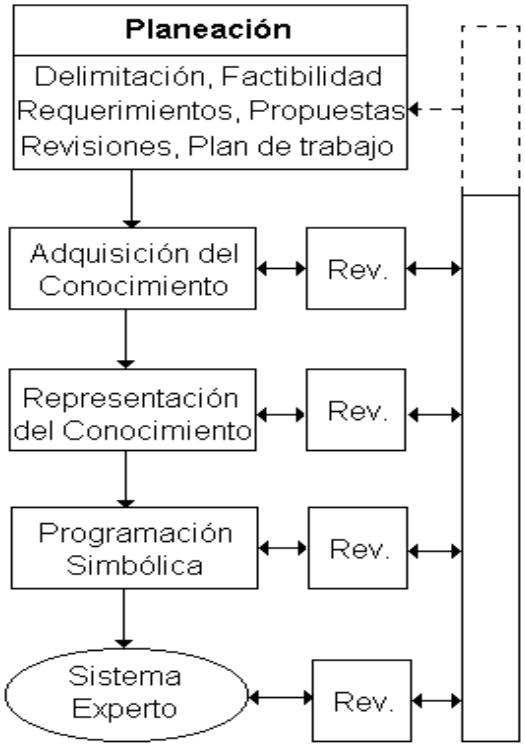

Ilustración 6: Etapas para la elaboración de un sistema experto

Un esquema más simple del proceso de diseño queda como sigue[17]:

[17] PACHECO Alberto. Metodología de Diseño de Sistemas Expertos. Instituto Tecnológico de Chihuahua. Disponible en línea <http://www.depi.itch.edu.mx/apacheco/ai/metodolo.htm > [con acceso 2009-05-04]

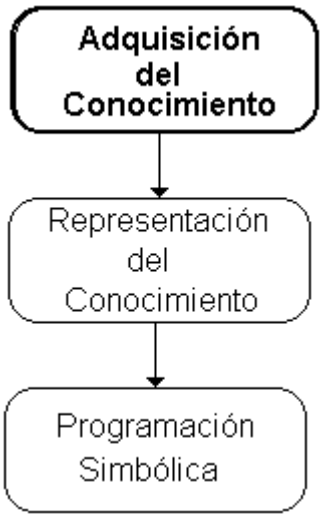

Ilustración 7: Etapas para el desarrollo de un Sistema Experto

5.2.4 Ventajas

Entre las numerosas ventajas[18] de implementar un sistema experto podemos mencionar:

1. Con la ayuda de un Sistema Experto, personas con poca experiencia pueden resolver problemas que requieren un conocimiento formal y especializado en un determinado campo o área.

2. Una vez que el sistema experto adquiere una base de conocimiento, este puede obtener conclusiones y resolver problemas de forma más rápida que los expertos humanos.

[18] GIARRATANO Joseph, RILEY Gary. Sistemas expertos principios y programación. Editorial: Thomson Learning. Edición: 2000. p. 4-5

3. El sistema experto actúa base a un conocimiento adquirido y no tiene sitio para la subjetividad, lo que resulta muy útil en situaciones complejas, donde la subjetividad humana puede llevar a conclusiones erróneas.

4. Una vez que el sistema experto finaliza su proceso de ingeniería del conocimiento, este tiene al menos, la misma competencia que un especialista humano.

5. Es ideal cuando es muy elevado el volumen de datos que ha de considerarse para obtener una conclusión, lo cual podría ser una limitante para un experto humano.

6. Permite que la experiencia este disponible para cualquier hardware de computo adecuado.

7. Reduce enormemente el costo de poner la experiencia a disposición del usuario.

8. Pueden usarse en ambientes que podrían ser peligrosos para un ser humano.

9. Permite una permanencia de la experiencia. A diferencia de un especialista que podría retirarse, renunciar o incluso morir, el conocimiento de un sistema experto durara indefinidamente.

10. El conocimiento de varios especialistas puede estar disponible para trabajar de forma simultánea y conjuntamente en un problema, a cualquier hora del día o de la noche. Además el nivel de experiencia y conocimiento de varios sistemas expertos puede superar el del un solo especialista humano.

11. Ofrecen mayor confiabilidad al ofrecen una según opinión, incrementan la confianza en que un especialista humano a tomado la decisión correcta o dar el voto de calidad entre varios especialistas humanos que tienen opiniones en desacuerdo.

12. El sistema experto puede actuar como un tutor inteligente, dejando que un estudiante ejecute programas de ejemplo y el sistema explica el razonamiento del sistema.

13. Pueden usarse para acceder a bases de datos en forma inteligente.

5.2.5 Aplicaciones

Los sistemas expertos se pueden aplicar a casi todos los campos del conocimiento. Algunos se han diseñado como herramientas de investigación, mientras que otros satisfacen importantes funciones de negocios e industriales, así como áreas como medicina, economía, psicología, finanzas, derecho y prácticamente todas las ramas del conocimiento.

La aplicación de sistemas expertos será adecuada allí donde los expertos dispongan de un conocimiento complejo en un área estrechamente delimitada,

donde no existan algoritmos elaborados o donde los que existen no puedan dar solución a todos los problemas presentes ni existan teorías complejas.

Otro campo de aplicación es allí donde hay teorías pero resulta prácticamente imposible analizar todos los casos teóricamente posibles mediante algoritmos y en un espacio de tiempo razonable.

Cuando estas situaciones se presentan se hace necesario el conocimiento que un sistemas experto halla adquirido con base a la experiencia de uno o mas especialistas, de tal forma que pueda ofrecer una solución en un espacio de tiempo razonable.

La construcción de un sistema experto supone un alto costo de trabajo debido a la estructuración e implementación del conocimiento que el experto requiere, es por esto que la construcción del sistema será factible si y solo si el conocimiento y experiencia de este será validado durante un largo periodo de tiempo y será aprovechado por la mayor cantidad posible de personas.

Con base en lo anterior, se pueden aplicar los sistemas expertos a evitar fallos en labores y tareas rutinarias de alta complejidad, ampliar con mayor rapidez el conocimiento de los especialistas, diagnosticar fallas de forma más rápida y conseguir tareas de planificación más completas y consistentes.

La siguiente figura[19] muestra la relación esfuerzo complejidad con base en el procesamiento de conocimientos.

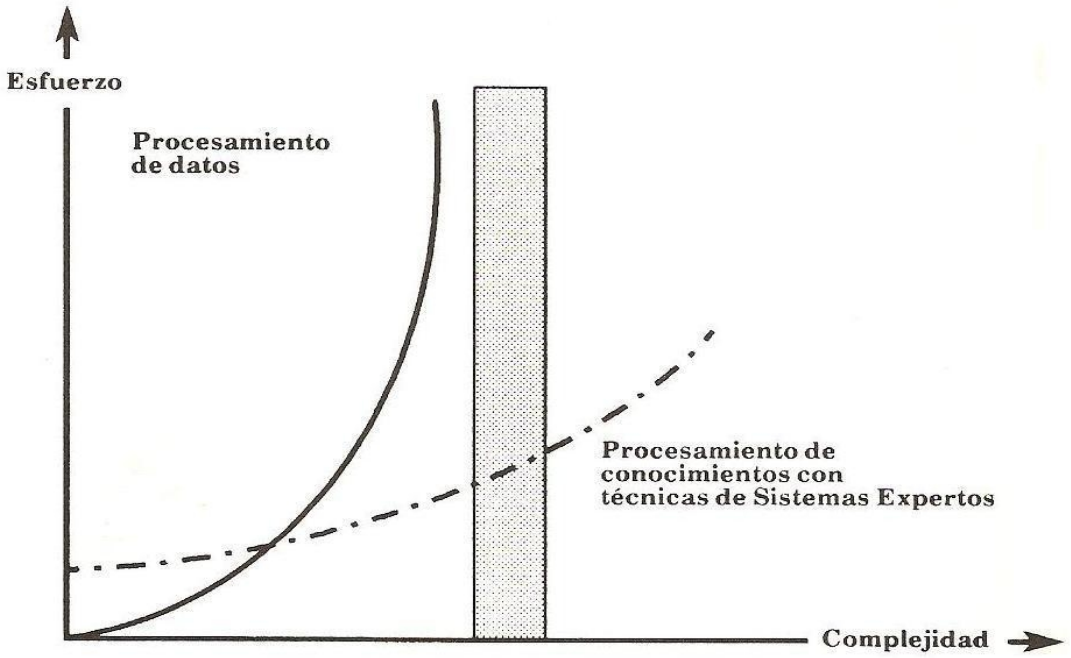

Ilustración 8: Procesamiento de conocimiento con sistemas expertos

Los sistemas expertos no se deben considerar como soluciones aisladas respecto a otros desarrollos de software.

Entre algunas aplicaciones de sistemas expertos podemos mencionar pioneros en este campo como lo son:

[19] NEBENDAHL Dieter. Sistemas Expertos: introducción a la técnica y aplicación. Barcelona, ES. Editorial: Marcombo, 1988. p. 28

MYCIN[20] [21], el cual es un sistema experto para la realización de diagnósticos médicos y la investigación de enfermedades infecciosas de la sangre, este se basa en solicitar una serie de datos a un paciente, información con la cual el sistema formula hipótesis, las cuales luego son evaluadas por el mismo sistemas con base en una serie de reglas y premisas dadas, ya sea sobre la base del conocimiento que el mismo sistema posee o también mediante la elaboración de nuevas preguntas para el paciente, las cuales le permiten al sistema verificar o rechazar la o las hipótesis planteadas.

XCON[22], es un sistema experto para configuraciones, desarrollado por la empresa *Digital Equipment Corporation*, el cual permite configurar redes de computadores, y debido a la gran cantidad de posibilidades y opciones de los diferentes productos que se consiguen en el mercado, la configuración completa y correcta puede tornarse en una tarea muy compleja, sin embargo, este sistema permite realizar configuraciones mucho más rápidas y mejor que las personas encargadas de esa labor.

[20] NEBENDAHL Dieter. Sistemas Expertos: introducción a la técnica y aplicación. Barcelona, ES. Editorial: Marcombo, 1988. p. 31

[21] GIARRATANO Joseph, RILEY Gary. Sistemas expertos principios y programación. Editorial: Thomson Learning. Edición: 2000. p. 509

[22] NEBENDAHL Dieter. Sistemas Expertos: introducción a la técnica y aplicación. Barcelona, ES. Editorial: Marcombo, 1988. p. 32

5.3 SISTEMAS BASADOS EN REGLAS

Son aplicaciones diseñadas para actuar como un experto humano en un dominio o conocimiento particular. Por lo tanto estas aplicaciones son sistemas expertos que poseen ciertas características que se describe a continuación.

Rodríguez y Díaz [23] definen la arquitectura de un sistema basado en reglas como se muestra se muestra en la siguiente figura:

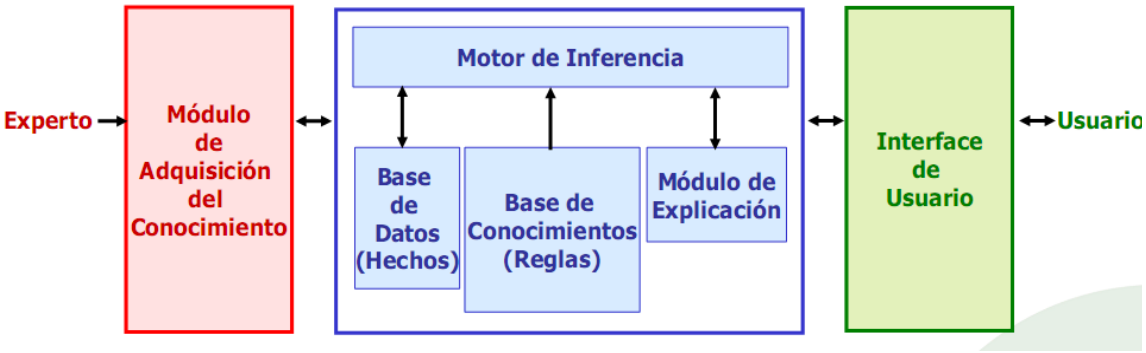

Ilustración 9: Componentes de un sistema basado en reglas

los componentes fundamentales de un sistema basado en conocimiento son una base de hechos que son los datos relevante a un problema concreto, una base de conocimientos de dominio del sistema (también conocida como base de reglas de producción) y una maquina deductiva (generalmente conocida como motor de inferencias).

[23]RODRÍGUEZ Camino, DÍAZ Irene. Apuntes del tema SISTEMAS BASADOS EN REGLAS del CURSO 2008-2009 de INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL. E.U.I.T. INFORMÁTICA DE GIJÓN [Transparencias en línea] Disponibles desde Internet en: <http://lear.inforg.uniovi.es/ia/Archivos/Apuntes%20y%20t/ReglasTeoria2008.pdf> [con acceso el 24-04-2009]. p 3

Tanto el conocimiento del dominio del sistema así como su base de hechos forman la estructura principal de un sistema basado en reglas.

5.3.1 Lenguajes de representación de conocimiento.

Nebendahl[24] define Los lenguajes de representación del conocimiento como herramientas auxiliares que permiten una representación clara y comprensible del mismo. Actualmente se hace una distinción entre el conocimiento y su procesamiento. Una de las divisiones nos permite diferenciar entre un conocimiento descriptivo (declarativo) y otro concluyente La parte declarativa y fácilmente representable del conocimiento se separa de las conclusiones.

Existen unos mecanismos de procesamiento que valoran el conocimiento y extraen conclusiones de la parte declarativa, que están a la disposición del técnico encargado del desarrollo de un sistema experto.

Los lenguajes de representación del conocimiento se describe mediante:

- La cantidad de formalismos disponibles en cada caso para la representación del conocimiento.
- El funcionamiento de los correspondientes mecanismos de valoración.

[24]NEBEBDAHL, Dieter. Sistemas Expertos Parte 2 experiencia de la práctica. Berlín y Munich: Siemens Aktiengesellshaft, 1991. p 7 - 8

5.3.1.1 Reglas

Nebendahl[25] afirma que el conocimiento práctico se formula a menudo, en lenguaje coloquial, en forma de un "si ... entonces". Por ende las reglas son una forma de representación del conocimiento de manera natural. Generalmente se expresan de la forma:

si ***antecedente*** *entonces* ***consecuente***[26]

Se define *antecedente* como una unión de de uno o varios atributos de un mismo dominio. Mientras que consecuente se define como los atributos que serán conocidos por el sistema.

El experto también capta su conocimiento en parte o en su totalidad en forma de reglas. La intensión de estas forma de representación de conocimiento basadas en reglas es reproducir esta formulación natural. La forma de funcionamiento de estos mecanismos de valoración se apoya aquí en el proceso pragmático del hombre a la hora de extraer sus conclusiones mediante reglas.

[25]NEBEBDAHL, Dieter. Sistemas Expertos Parte 2 experiencia de la práctica. Berlín y Munich: Siemens Aktiengesellshaft, 1991. p 7 - 8

[26]RODRÍGUEZ Camino, DÍAZ Irene. Apuntes del tema SISTEMAS BASADOS EN REGLAS del CURSO 2008-2009 de INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL. E.U.I.T. INFORMÁTICA DE GIJÓN [Transparencias en línea] Disponibles desde Internet en: <http://lear.inforg.uniovi.es/ia/Archivos/Apuntes%20y%20t/ReglasTeoria2008.pdf> [con acceso el 24-04-2009]. p 6

5.3.1.1.1 Representación del conocimiento mediante reglas.

Nebendahl[27] dice que para realizar una representación del conocimiento basado en reglas, debe fundamentarse en estos principios básicos.

la parte “si...” de una regla se definirá, en adelante, como premisa y la parte “entonces...” como la parte de las conclusiones. La parte de las conclusiones pueden contener conclusiones y acciones.

Un mecanismo de valoración de reglas recibe el nombre de mecanismo de inferencia.

Una regla sencilla es la siguiente:

“si Hecho 1 es valido, entonces también es valido hecho 2”.

Esta sencilla forma de regla es, sin embargo poco potente para aplicaciones reales. Solo mediante ampliaciones de esta forma básica se alcanzan formalismos útiles en la representación del conocimiento.

Una ampliación de las reglas consiste en permitir la anudación lógica de hechos y acciones en la premisa y en la parte de las conclusiones. Entonces una regla puede tener este aspecto:

si hecho 1 es valido o hecho 2 no es valido, entonces vale hecho 3 y además, hay que realizar acción 1.

[27]NEBEBDAHL, Dieter. Sistemas Expertos Parte 2 experiencia de la práctica. Berlín y Munich: Siemens Aktiengesellshaft, 1991. p 9 - 10

Otra ampliación del formalismo de reglas es la posibilidad de indicar valores de probabilidad (por ejemplo, para los hechos). Una regla podría ser:

si hecho 1 es valido o hecho 2 no es valido, entonces vale hecho 3 con mas de 50% de probabilidad.

Naturalmente, con esta representación está también implícitamente ligada la capacidad del mecanismo de inferencia de manipular correctamente las probabilidades.

Otra ampliación consiste en la posibilidad de utilizar variables en lugar de hechos. El mecanismo de inferencia entonces controla la forma en que se otorgan los valores a las variables.

5.3.1.1.2 Inferencia

Rodríguez y Díaz [28] definen inferencia como *la operación por la cual se obtiene conocimiento nuevo a partir de un conocimiento previo o existente*. Es decir es una deducción de un conocimiento a partir de otro. El proceso de inferencia puede ser visto por las siguientes formas lógicas: deducción, inducción y abducción.

Se define *deducción* como un método de razonamiento que parte de conceptos generales o principios universales para llegar a conclusiones particulares. En la

[28]RODRÍGUEZ Camino, DÍAZ Irene. Apuntes del tema SISTEMAS BASADOS EN REGLAS del CURSO 2008-2009 de INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL. E.U.I.T. INFORMÁTICA DE GIJÓN [Transparencias en línea] Disponibles desde Internet en: <http://lear.inforg.uniovi.es/ia/Archivos/Apuntes%20y%20t/ReglasTeoria2008.pdf> [con acceso el 24-04-2009]. P 7 - 10

cual corrobora que el nuevo conocimiento es cierto si en el proceso se inicia con conocimiento cierto. Esta es la fuerza de inferencia lógica y por tanto de la lógica.

Se define *inducción* como un método de raciocinio que consiste en alcanzar un principio que se deriva lógicamente de unos datos o hechos particulares. Éste es un mecanismo de aprendizaje automático en donde se igualan la correlación con la casualidad aunque éste método presenta problemas a lo que respecta la confiabilidad de la validez del conocimiento inferido.

Se define *abducción* como un proceso en el que se puede obtener un conjunto de hipótesis o explicaciones posibles a partir de un conocimiento base. Es decir que dado un conjunto de reglas y hechos observados, se producen un conjunto de explicaciones posibles que usando la deducción harían coherente el conocimiento de partida.

5.3.1.1.2.1 Mecanismo de inferencia para reglas.

De acuerdo Nebendahl[29] el conocimiento del experto estará configurado, en parte o en su totalidad, por una cantidad de reglas y hechos. La interrelación de las distintas reglas y sus efectos se controlan a través de mecanismos de inferencia correspondiente.

[29]NEBEBDAHL, Dieter. Sistemas Expertos Parte 2 experiencia de la práctica. Berlín y Munich: Siemens Aktiengesellshaft, 1991. p 10 - 12

Los sistemas basados en reglas utilizan la regla de inferencia simple llamada "*modus ponen*" expresión latina que traduce "*modo que afirmando afirma*" esta regla de inferencia se define así:

Sea P, Q proposiciones

Si P entonces Q.

Luego P.

Entonces, Q.

Desde el simbolismo de la lógica proposicional el '"*modus ponen*" se puede representar de la siguiente forma:

$$[p \wedge (p \rightarrow q)] \rightarrow q$$

Los sistemas basados en reglas automatizan los métodos de razonamiento usando técnicas de búsqueda con la comparación de patrones. Esta automatización de los métodos de razonamiento se conoce como "cadena de inferencias".

Los sistemas basados en regla aunque utilizan el "*modus ponen*" como regla base para el proceso de inferencia, estos, en la mayoría de los casos, no siempre lo utilizan como un método de deducción lógica ya que se puede aceptar

incertidumbres y, además, la monotonía no es constante ya que en algún momento un hecho derivado puede ser, retractado a futuro.

Los mecanismos de inferencia más utilizados se basan en dos principios que también pueden estar combinados.

1. valoración por encadenamiento hacia delante (accionado por datos)
2. valoración por encadenamiento hacia atrás (accionado por hipótesis y orientado a objetivo).

Estos dos principios básicos pueden describirse mediante una forma especialmente sencilla de reglas. Estas reglas son de tipo:

"Si hecho x es válido y hecho y es válido entonces vale también hecho z y además tiene que realizarse acción a"

5.3.1.1.2.1.1 Valoración por encadenamiento hacia delante

En esta valoración se parte de hechos conocidos (datos). Si las premisas se pueden concluir con estos hechos, entonces se realiza la parte de las conclusiones. Se dice también que las reglas "se disparan".

Si se cumplen todas la premisas de varias reglas a la vez, dependerá de la versión del mecanismo de inferencia cual de las reglas concurrente tiene que dispararse. Existen diferentes estrategias para solucionar este conflicto, propias de la forma de representación del conocimiento basadas en reglas. En la reglas con indicación

de probabilidad se podría disparar, por ejemplo, aquellas reglas cuyas conclusiones contenga el valor de probabilidad mas alto.

Generalmente para la resolución de conflictos se tiene varios criterios de selección, que en la mayoría de los casos estos criterios son estrategias de búsqueda. Rodríguez y Díaz [30] mencionan algunos criterios de resolución de gran utilidad:

- Mayor numero de premisas en el antecedente
- Mayor prioridad
- Búsqueda en profundidad
- Búsqueda en anchura.

El resultado de disparar estas reglas puede acarrear la creación de nuevos hechos. Asimismo, estos últimos pueden disparar, a su vez, nuevas reglas.

La valoración por encadenamiento hacia delante finaliza cuando ya no pueden dispararse mas reglas o cuando el mecanismo de inferencia encuentra otro criterio de finalización.

[30]RODRÍGUEZ Camino,DÍAZ Irene. Apuntes del tema SISTEMAS BASADOS EN REGLAS del CURSO 2008-2009 de INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL. E.U.I.T. INFORMÁTICA DE GIJÓN [Transparencias en línea] Disponibles desde Internet en: <http://lear.inforg.uniovi.es/ia/Archivos/Apuntes%20y%20t/ReglasTeoria2008.pdf> [con acceso el 24-04-2009]. P 18

La solución del problema buscada a través de la valorización de las reglas solo puede consistir, por ejemplo, en haber alcanzado un determinado objetivo o una ultima regla por disparar

podemos representar la valoración por encadenamiento hacia adelante como una función descrita a continuación[31]:

```
función encadenamiento_adelante
        mientras que (reglas a disparar y (no objetivo))
                asociar()
                resolver_conflictos()
                ejecutar()
        fin mientras
fin función
```

5.3.1.1.2.1.2 Valoración por encadenamiento hacia atrás

Nebendahl[32] afirma que en la valoración por encadenamiento hacia atrás se parte de la suposición de una conclusión (hipótesis). la elección de esta hipótesis la realiza el mecanismo de inferencia, para ello utiliza datos obtenidos generalmente mediante preguntas hechas al usuario. Esta valoración es muy útil en aplicaciones con muchos datos de partida disponibles de los que solo una pequeña parte son importantes para la solución del problema. Por su estrategia interactiva, es mas

[31] Ibíd P 19

[32]NEBEBDAHL, Dieter. Sistemas Expertos Parte 2 experiencia de la práctica. Berlín y Munich: Siemens Aktiengesellshaft, 1991. p 12 - 13

ventajoso que la valoración por encadenamiento hacia adelante debido a que el encadenamiento hacia atrás obtiene su conocimiento con una secuencia de preguntas hechas al usuario, las cuales son estrictamente necesarias. Esa interacción con el usuario la hace la diferencia.

Si la primera hipótesis no es confirmada directamente por los hechos ya existentes en la base de conocimientos, se procede de la forma siguiente:

Se buscan reglas que contengan esta hipótesis de arranque en la parte de las conclusiones.

Cuando se encuentren varias reglas hará faltan también estrategias de soluciones de conflicto. Estas estrategias de solución determinan en que secuencia deben analizarse las reglas afectadas.

Según al estrategia de solución de conflicto presente, se analizan las reglas que contenga la hipótesis como conclusión, buscando si se cumplen o pueden cumplirse sus premisas.

una premisa se considera concluida, cuando existen unos hechos que la confirman.

Si éste no fuera el caso para todos los hechos de una premisa, se tratarían cada uno de los hechos no confirmados, como hipótesis provisionales. Estas hipótesis

temporales se analizarán primero como las hipótesis de salida y dado el caso se confirmarán.

Rodríguez y Díaz [33] describen el algoritmo por valoración por encadenamiento hacia atrás a continuación:

1. Se forma una pila inicial compuesta por todos los objetivos iniciales.
2. Considerar el primer objetivo de la pila. Localizar todas las reglas que lo satisfagan.
3. Examina las premisas de dichas reglas, en orden:
    1. Si todas las premisas se satisfacen. Ejecutamos las reglas y se derivan sus conclusiones. Si se derivó un valor para el objetivo actual entonces se elimina de la pila y se vuelve al paso 2.
    2. Si una premisa de una regla no se satisface (tiene un valor desconocido en la base de conocimientos), se mira a ver si existen reglas que concluyan un valor para ella. Si existen se inserta en el tope de la pila de objetivos y se vuelve al paso 2.

[33]RODRÍGUEZ Camino, DÍAZ Irene. Apuntes del tema SISTEMAS BASADOS EN REGLAS del CURSO 2008-2009 de INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL. E.U.I.T. INFORMÁTICA DE GIJÓN [Transparencias en línea] Disponibles desde Internet en: <http://lear.inforg.uniovi.es/ia/Archivos/Apuntes%20y%20t/ReglasTeoria2008.pdf> [con acceso el 24-04-2009]. P 23 - 25

3. Si por 3.2 no se encontró ninguna regla que concluya un valor para la premisa actual .Entonces se pregunta al usuario por dicho valor y se añade a la base de conocimientos.

    1. Si el valor satisface la premisa actual se continúa examinado el resto del antecedente.

    2. Sino se considera la siguiente regla que concluya un valor para el objetivo actual

4. Si se han examinado todas las reglas que concluyen un valor para el objetivo actual y todas fallaron entonces se marca el objetivo como indeterminado, se extrae de la pila y se vuelve al paso dos.

5. Si la pila esta vacía el proceso finaliza.

El pseudocódigo se muestra a continuación:

```
Función Encadenamiento_hacia_atrás
        Construir_pila_inicial();
        Para inicio_pila hasta fin
                Reglas_satisfacen_Objetivo [] = Buscar_Reglas();
                Para  Reglas_satisfacen_Objetivo_inicio hasta fin
                        Si(Premisa_Satisfacen(Reglas_satisfacen_Objetivo[i]))
                                Eliminar_Objetivo_pila ();
                        Sino
```

```
                    Si(Hay_reglas_que_permitan_derivarlo())
                        Inserta_pila();
                    Sino
                        bool conocido=Preguntar_Usuario();
                        Si(conocido)
                            Añadir_BC();
                        Sino
                            Considerar_siguiente_regla();
                        Fin Si
                    Fin Si
                Fin Si
            Fin Para
            Si (Todas_reglas_fallan)
                Objetivo_fallido();
                Eliminar_Pila();
            Fin Si
        Fin Para
Fin Función
```

Castillo, Gutiérrez y Hadi[34] comentan que el algoritmo de encadenamiento de reglas orientado a un objetivo (encadenamiento hacia adelante o encadenamiento hacia atrás) requiere del usuario seleccionar, en primer lugar, una variable o nodo objetivo; entonces el algoritmo navega a través de las reglas en búsqueda de una

[34]CASTILLO Enrique, GUTIÉRREZ José Manuel, HADI Ali. Sistemas Expertos y Modelos de Redes Probabilísticas. Academia Española de Ingeniería. p 14. disponible desde Internet en <http://personales.unican.es/gutierjm/papers/BookCGH.pdf> [con acceso el 24-04-2009] p 42- 47

conclusión para el nodo objetivo. Si no se obtiene ninguna conclusión con la información existente, entonces el algoritmo fuerza a preguntar al usuario en busca de nueva información sobre los elementos que son relevantes para obtener información sobre el objetivo.

Las estrategias de encadenamiento de reglas se utilizan en problemas en los que algunos hechos (por ejemplo, síntomas) se dan por conocidos y se buscan algunas Conclusiones (por ejemplo, enfermedades). Por el contrario, las estrategias de encadenamiento de reglas orientadas a un objetivo se utilizan en problemas en los que se dan algunos objetivos (enfermedades) y se buscan los hechos (síntomas) para que estas sean posibles.

5.3.1.2 Marcos

Según Nebendahl[35], el término marco (frame) se utiliza para varias formas de representación del conocimiento:

- los marcos se utilizan preferiblemente para la representación de conocimientos susceptibles de división en unidades, descriptibles mediante la indicación de cualidades.
- Los marcos son una especie de formulario, un “armazón de datos”, que puede contener las cualidades de una unidad de conocimiento.

[35]NEBEBDAHL, Dieter. Sistemas Expertos Parte 2 experiencia de la práctica. Berlín y Munich: Siemens Aktiengesellshaft, 1991. p 13 - 15

Un marco se divide en campos llamados descriptores (Slot). Un descriptor se define también como atributo. Un descriptor describe una cualidad de una unidad de conocimiento.

Un descriptor contiene, por su parte, las llamadas facetas (facets). El tipo y la cantidad de descriptores y de las facetas dependen de la variante del marco. En la mayoría de ellas hay, al menos, una faceta para su definición y una para su valor. Las demás se utilizan. Por ejemplo, por ejemplo para fijar las limitaciones para posibles valores del descriptor.

La siguiente figura muestra la estructura de un marco:

marco:			
nombre del descriptor	valor	ámbito de valores	
faceta del nombre	faceta del valor	faceta del ámbito	descriptor

Ilustración 10: Estructura de un marco

Ahora la siguiente figura representa una unidad de conocimiento mediante un marco con el nombre Persona:

marco:	persona	
nombre del descriptor	valor	ámbito de valores
nombre		texto
edad		numerico
lugar		texto
fecha nacimiento		aaaa/mm/dd
sexo		m,f

Ilustración 11: Ejemplo de un marco

Con un marco se describe una unidad de conocimiento. Para ello hay que definir las cualidades de esta unidad de conocimiento. Los descriptores pueden designarse por una faceta con el nombre del descriptor. La cantidad de descriptores de un marco puede elegirse de forma variable en la mayoría de lenguajes de representación de conocimiento orientados a marcos. La cantidad y la importancia de cada faceta, viene normalmente fijadas por el lenguaje de representación del conocimiento, que permite aplicar de forma más eficiente los mecanismos de valoración.

Los valores de las facetas pueden estar asignados previamente o estar vacíos. En el transcurso de la valoración del conocimiento, puede rellenarse las facetas vacías.

5.3.1.2.1 Herencia de propiedades

Como afirma Nebendahl[36] la mayoría de los lenguajes de representación de conocimiento orientados a marcos permiten estructurar jerarquías para las unidades de conocimientos representadas. Con los marcos se pueden crear relaciones de clase/subclase/instancia. Un marco especificado como clase, puede ceder contenidos de las facetas de sus descriptores, especialmente sus valores a subclases. Las instancias son los elementos más inferiores de la jerarquía. Los marcos que solo actúan como instancia solo pueden heredar.

[36]NEBEBDAHL, Deiter. Sistemas Expertos Parte 2 experiencia de la práctica. Berlín y Munich: Siemens aktiengesellshaft, 1991. p 15 - 16

Los mecanismos de herencia son una característica muy importante de las formas de representación del conocimiento orientado a marcos.

Según Rolston[37], un sistema de marcos se apoya altamente en el concepto de herencia en el mismo sentido que lo hacen las redes semánticas. Cualquier clase de objetos se pueden incluir en muchos y variados marcos que representan objetos a diferentes niveles de especificación. Por ejemplo la clase de Automóviles se puede incluir en marcos determinados OBJETO FÍSICO, VEHÍCULO o AUTOMÓVIL.

Un marco que representa una clase de objetos a cierto nivel dado de especificación puede incluir descriptores y valores de descriptores que se heredan de marcos que representan mayor nivel de abstracción. El uso de la opción por omisión y de los valores de los descriptores heredados posibilita el razonamiento eficiente debido a que tal utilización evita la necesidad de procesos costosos de razonamiento para re descubrir hechos anteriores en situaciones nuevas.

5.3.1.2.2 Razonamiento con marcos

Rolston[38] afirma que la representación del conocimiento con un sistema de marcos nos posibilita, hasta cierto punto, el razonamiento aun en el caso de que la

[37]ROLSTON David. Principios de Inteligencia Artificial y sistemas expertos. México: McGraw-hill interamericana, 1992. p 50

[38]ROLSTON David. Principios de Inteligencia Artificial y sistemas expertos. México: McGraw-hill interamericana, 1992. p 51-52

información disponible esté incompleta y además nos posibilita para inferir hechos rápidamente que no se observan explícitamente.

Para iniciar este proceso de razonamiento, se debe seleccionar en primer lugar un marco que represente la situación actual. En razón de que en la mayoría de los casos no se tiene un marco que exactamente se aplique en el contexto actual, entonces se debe iniciar con el más adecuado con base a la evidencia parcial disponible. Por ende se debe instanciar el marco seleccionado sobre la base de las condiciones especificas actuales.

En general este proceso de instanciación asocia un individuo particular con una clase. El proceso construye una instanciación, una descripción individual formada mediante la aplicación de las características individuales específicas a la descripción de clase genérica.

Una de las dificultades de la representación con marcos es en el establecimiento con precisión de los valores por omisión para un marco. Nunca existe un acuerdo exacto entre cualquier grupo de observadores en cuanto a las características típicas de cualquier objeto. La visión individual de cada sujeto de “típica” está matizada por la experiencia personal y por sus inclinaciones. Cuando se presenta la palabra “automóvil” una persona evoca una imagen de una camioneta y otra construye una imagen de un auto deportivo. Aunque parezca obvio para la mayoría de la gente, que debamos asumir que un árbol tiene hojas, para alguien

de una región desértica puede ser que un árbol tenga agujas en lugar de hojas. Adicionalmente a la situación anterior debemos enfrentar preguntas tales como "¿un modelo plástico de un automóvil es un automóvil?".

A pesar de estas dificultades, el esquema de representación por marcos continúa como un mecanismo poderoso para la representación del conocimientos y esta ganando terreno en cuanto a su uso, en forma creciente.

5.3.2 Diseño de sistemas expertos

Durkin[39] nos indica una serie de pasos o tareas que cuya meta es la construcción de sistemas expertos basados en reglas dependiendo del tipo de valoración de encadenamiento ya sea encadenamiento hacia delante o encadenamiento hacia atrás que un ingeniero de conocimiento debe realizar.

5.3.2.1 Diseño de Sistemas Expertos basados en reglas con encadenamiento hacia atrás

Durkin[40] afirma que antes de diseñar un sistema experto, lo primero que se debe hacer es entender el problema en forma general. Determinar los objetivos del sistema, para lo cual, se debe tomar en cuenta los aspectos que considera un experto y cómo trabaja con información confiable, para derivar recomendaciones.

[39]DURKIN John. Expert Systems: Design and Development. Cap 8 traducido por Lic. Arias Tapia Jhonny para el curso 2010029 - Sistemas Expertos. Universidad Mayor de San simón. Recurso Disponible en línea <http://www.cs.umss.edu.bo/doc/material/mat_gral_10/cap6_8_10.zip> [con acceso 27-04-2009]

[40] Ibíd p1

Luego de entender el problema, se debe comenzar a considerar los enfoques del diseño del sistema actual.

Una característica de cualquier sistema experto basado en reglas de encadenamiento hacia atrás o inducción es que el proceso de diseño es altamente iterativo. Primero se obtiene una pequeña cantidad de información del experto, codificándola y probándola. Los resultados sirven para descubrir deficiencias en el sistema y luego vienen nuevas sesiones con el experto. Este proceso cíclico continúa mientras el conocimiento del sistema va creciendo.

Este estilo de desarrollo cíclico es utilizado cuando se construye un sistema de encadenamiento hacia atrás. No obstante, la manera en que procederemos a través de este ciclo es única para este tipo de sistemas. Primero hay que determinar las mayores metas y la manera en que pueden ser establecidas (reglas meta). Luego se busca la manera de adquirir información que pueda sustentar estas reglas meta. Por supuesto que este proceso deja a otras reglas que trabajan con información primitiva. El desarrollo de un sistema de encadenamiento hacia atrás, se mueve desde lo abstracto a lo específico.

A continuación las mayores tareas que deberás realizar para desarrollar un sistema de encadenamiento hacia atrás:

1. Definir el problema
2. Definir las metas

3. Definir la regla meta

4. Expandir el sistema

5. Refinar el sistema

6. Diseñar la interfaz

7. Evaluar el sistema

5.3.2.1.1 Definición del problema

Para Durkin[41], el paso inicial en cualquier proyecto de sistemas experto, debe ser aprender, acerca del problema. Por ende se debe obtener esta información rápidamente. Para ello, los mejores recursos son los libros, documentos, reportes, los que ayudarán a tener una comprensión general del problema y también sus soluciones. Estos recursos son un buen punto de inicio, pero para la mayoría de los proyectos, se necesitará la ayuda de un experto real.

5.3.2.1.2 Diseño de las metas

La codificación de cualquier sistema de encadenamiento hacia atrás, comienza con la definición de las metas del sistema.

Se debería enfocar el diseño inicial del sistema a una pequeña pero representativa parte del problema, se debería tener una estructura del sistema que sea fácil de expandir para considerar más aspectos.

[41]DURKIN John. Expert Systems: Design and Development. Cap 8 traducido por Lic. Arias Tapia Jhonny para el curso 2010029 - Sistemas Expertos. Universidad Mayor de San simón. Recurso Disponible con acceso <http://www.cs.umss.edu.bo/doc/material/mat_gral_10/cap6_8_10.zip> [con acceso 27-04-2009] p 1-18

Escribiendo las declaración de las metas

Todo sistema de encadenamiento hacia atrás necesita de al menos una meta para comenzar.

Usando una declaración de una variable meta, nos solo se reduce el número de declaraciones, sino también que permite añadir otras metas después, sin necesidad de especificarlo en código, en una declaración de una meta. La forma de escribir la declaración de una variable meta depende del shell que escojas para desarrollar el sistema.

5.3.2.1.3 Diseñando las reglas meta

Cada meta de nuestro sistema debe tener al menos una regla (regla meta) que pueda ser concluida. Se debe diseñar una regla meta, de la misma forma que se hace con cualquier regla, es decir, buscar las condiciones necesarias que satisfagan la conclusión de la regla.

Tablas de decisión

Las tablas de decisión ofrecen una técnica de adquisición de problemas asociados con técnicas que evitan problemas normalmente asociados con técnicas de entrevistas.

Redes de inferencia

Además de la tabla de inferencia, en los estados iniciales de un proyecto se puede graficar una red de inferencia de reglas, por ser de gran ayuda. Una red de inferencia muestra las relaciones lógicas entre piezas de información que son representadas en las reglas.

Probando las reglas meta

Después haber codificado las reglas meta en tu sistema, entonces se procede a probar el sistema. En general, después de codificar cualquier conocimiento en un sistema experto, se debe probar el sistema inmediatamente.

Es muy importante hacer las pruebas necesarias al sistema en las primeras etapas de desarrollo, ya que el conjunto de reglas es relativamente pequeño, y puedes realizar todas las combinaciones de preguntas que sea posible. Se puede probar la correctitud y completitud del conocimiento del sistema, después, cuando el conocimiento va creciendo, se pierde esta habilidad. En este punto, se debe contar con técnicas alternativas para evaluar el sistema, tal como el uso de casos de prueba a partir de históricos del problema.

Orden de búsqueda

Otro aspecto muy importante a considerar en las primeras etapas de desarrollo del proyecto es la forma en que el sistema buscará el conocimiento. Al diseñar un

sistema experto, no solo se quiere que éste sea consistente con el del experto, también se espera que el enfoque de solución al problema que tome el sistema, sea el usado por el experto.

La mayoría de los shells de encadenamiento hacia atrás buscan reglas en el orden en que fueron ingresadas. En general, se debe entender como el shell que estas utilizando realiza la búsqueda de las reglas. Si se realiza en orden, se debe ajustar el orden de las reglas para adecuarlo a cualquier requerimiento de búsqueda. Algunos shells permiten asignar números "prioridad" a las reglas, lo que ayuda a ordenar la búsqueda. Se debe asignar números de prioridad altos a las reglas que sean consideradas primero.

Búsqueda exhaustiva versus búsqueda no exhaustiva

Se puede programar un sistema de encadenamiento hacia atrás que realice una búsqueda exhaustiva o no exhaustiva. Un búsqueda exhaustiva continua la búsqueda por todas las reglas para concluir en alguna meta, incluso aún si la meta ya fue establecida previamente por alguna un de las reglas; en cambio la técnica de búsqueda no exhaustiva detiene la búsqueda después de encontrar la regla que determina la meta. Para decidir cual de las técnicas aplicar, hay que estudiar la forma en que el experto resuelve el problema.

La mayoría de los shells ejecutan una búsqueda no exhaustiva. Otros sin embargo, requieren que uses alguna forma de comando "STOP" como una

conclusión añadida en cada regla. Este enfoque es adecuado para las reglas meta. Sin embargo, reglas que no son metas, este enfoque debe causar que finalice la sesión, previamente el sistema establece su regla primaria. Para habilitar una búsqueda exhaustiva en algún aspecto, la mayoría de los shells provee un comando que especifica el aspecto (asunto) que se permite que tenga mayor valor que otro.

5.3.2.1.4 Expandiendo el sistema

La manera principal en que se puede mejorar la inteligencia de un sistema experto es expandiendo su conocimiento. Hay dos opciones a tomar: ampliar o profundizar el conocimiento del sistema. Una forma de expandir el conocimiento del sistema es mediante entendimiento más amplio del problema. Es decir, enseñarle acerca de aspectos adicionales. Este tipo de expansión es bastante fácil y usualmente reservado para después en el desarrollo del proyecto, después de que el sistema ha demostrado ser exitoso. Otra forma de expandir el conocimiento del sistema mediante un entendimiento más profundo del problema. Es decir, enseñarle más acerca de asuntos conocidos. Por ejemplo, enseñarle como determinar las premisas presentes en una regla meta. Esta es la técnica de expansión más utilizada en las primeras etapas del proyecto.

En general, cuando se desarrolla un sistema basado en reglas se debe determinar que premisas pueden ser expandidas más profundamente. Se debe discutir con el

experto acerca de los aspectos que consideras deben ser expandidos, para hacer la tarea más rápida y fácil.

En este punto Durkin[42] llama la atención en un asunto que provoca problemas a los desarrolladores de sistemas expertos.

El problema con el "SINO":

Todo programador esta familiarizado con la sentencia si ... entonces ... sino, de la misma forma que un ingeniero de conocimientos, es por eso que cuando se codifican procedimientos, " si se puede reducir esta sentencias a una sola línea de código en ves de dos, lo hacen." Pero cuando se desarrolla un sistema experto, no se esta codificando procedimientos, se está codificando conocimiento. Incluso, al intentar comprimir dos piezas de conocimiento en una sola regla, puedes causar problemas. Estos problemas se encuentran frecuentemente con el usos de la declaración "sino."

Ejemplo:

```
si (Edad_Cliente<40) entonces
        Cliente ES joven
sino
        Cliente ES viejo
fin si
```

[42]DURKIN John. Expert Systems: Design and Development. Cap 8 traducido por Lic. Arias Tapia Jhonny para el curso 2010029 - Sistemas Expertos. Universidad Mayor de San Simón. Recurso Disponible en línea <http://www.cs.umss.edu.bo/doc/material/mat_gral_10/cap6_8_10.zip> [con acceso 27-04-2009] p 17

A simple vista no se percibe ningún problema, es una representación lógica provista por el experto, sin embargo esta regla causa problemas cuando el sistema necesita expansión. Por ejemplo asumamos que después necesitamos categorizar la edad:

```
si (Edad_Cliente<50  y Edad_Cliente>=40)  entonces
        Cliente ES Edad_mediana
fin si
```

Si encontramos una persona que tiene 45 años, el sistema debería concluir que es viejo y también de edad mediana, por lo tanto llagamos a conclusiones inconsistentes.

5.3.2.1.5 Refinando el sistema

Lo más aconsejable para manejar números en las reglas, es utilizar variables en vez de números constantes, ya que estos causan problemas a la hora de mantener el sistema. Las variables que utilices deben asumir valores durante la inicialización del programa. Este enfoque permitirá localizar fácilmente las variables que necesitan un reajuste.

Red de Seguridad Inteligente

En la mayoría de los programas de aplicación, los programadores realizan pruebas al código, introduciendo todas las posibles combinaciones de entradas

para verificar su operación. Es posible que la cantidad de datos forme parte de un conjunto limitado. Sin embargo al desarrollar un sistema experto, se encontrará una cantidad grande de combinaciones de datos de entrada, lo cual requiere una prueba exhaustiva del sistema. En esta situación, puede existir la posibilidad de que tu sistema no encuentre la recomendación final.

Para afrontar este problema, se necesita crear una "red de seguridad" al sistema. Una red de seguridad es simplemente un curso de acción que toma el sistema si éste falla en la búsqueda de una recomendación. Puede ser tan complejo como cargar otra base de conocimientos, para intentar corregir la situación, o tan simple como una declaración por defecto que se muestre en la pantallas para informarle al usuario acera de la falla.

Proveer señales

Un forma de acomodarse al usuario durante la consulta es mantenerlo siempre informado acerca de las cosas importantes y direcciones que ha tomado el sistema. Los usuarios están más cómodos con las consultas cuando se mantienen informados.

5.3.2.1.6 Diseño de la interfaz

Los usuarios de los sistemas expertos ven el sistema a través de la interfaz del mismo. La aceptación del sistema dependerá de que tan buena es la comodidad que le ofreces al usuario. Afortunadamente, la mayoría de los shells proveen

utilidades para el diseño de la interfaz. Permiten construir el diseño de la pantalla introductoria, pantallas para preguntas , y la pantalla final.

5.3.2.1.7 Evaluación del sistema

En este punto ha de tenerse terminado el prototipo de nuestro sistema. Todas las reglas han sido codificadas en el sistema y asumimos que la interfaz esta de acuerdo a las recomendaciones dadas en las secciones previas. Asumidos que el sistema pasó las pruebas exitosamente con cada expansión. El próximo paso es evaluar el sistema usando casos de prueba reales.

Repaso a la sesión

El primer punto a considerar es que el sistema recomienda lo mismo que un experto. Debes realizar muchos casos de prueba para comprobar que realmente va de cuerdo al experto.

El siguiente punto a considerar es que la evaluación de nuestro sistema se acomoda al usuario a través de la sesión. La pantalla de introducción informa al usuario acerca de o que hace el sistema, y como lo hará. Realiza preguntas que pueden ser fáciles de encontrar en la pantalla de señales instruyendo al usuario sobre las operaciones del sistema. La información de la conclusión muestra una recomendación final con una justificación de ato nivel.

5.3.2.2 Diseño de sistemas con encadenamiento hacia adelante

A continuación Durkin[43] describe una metodología general de diseño para sistemas de encadenamiento hacia delante. La primera tarea cuando desarrollamos un sistema de encadenamiento hacia adelante basado en reglas es obtener una comprensión general del problema. Esta tarea implica definir el objetivo del sistema, los mayor problemas y las formas en que el experto trabaja con la información para derivar sus recomendaciones o conclusiones. Un sistema con encadenamiento hacia adelante comienza con datos del problema y reglas activadas para inferir nueva información. En un sistema de encadenamiento hacia adelante el motor de inferencia activa reglas cuyas premisas corresponden con la información contenida en la memoria de trabajo.

Las principales tareas para desarrollar un sistema con encadenamiento hacia adelante son:

1. Definir el problema
2. Definir los datos de entrada.
3. Definir estructura para el manejo de los datos.
4. Escribir el código inicial.
5. Realizar pruebas al sistema.
6. Diseñar la interfaz.
7. Expandir el sistema.

[43]DURKIN John. Expert Systems: Design and Development. Cap 8 traducido por Lic. Arias Tapia Jhonny para el curso 2010029 - Sistemas Expertos. Universidad Mayor de San simón. Recurso Disponible en línea <http://www.cs.umss.edu.bo/doc/material/mat_gral_10/cap6_8_10.zip> [con acceso 27-04-2009] p 21-24

8. Evaluar el sistema.

El problema de diagnostico es un problema muy común en la implementación de SE. Una de las razones es que este tipo de problemas es general mente más fácil de comprender que los problemas de diseño o planificación. Otra razón es que la solución de estos problemas tienen beneficios tangibles para las organizaciones. Muchos SE de diagnostico usan encadenamiento hacia adelante, esto debido a que tienen un número finito metas.

5.3.2.2.1 Definir el problema

El paso inicial para el diseño de un sistema con encadenamiento hacia adelante es obtener conocimiento respecto al problema, una forma de hacer esto es localizar un buen experto, un método alternativo es consultar un manual de referencia técnica, ya que sin duda estos manuales son escritos por los expertos.

5.3.2.2.2 Definir datos de entrada

Todos los sistemas de encadenamiento hacia adelante necesitan primero obtener datos iniciales, para poder funcionar, entonces es necesario escribir reglas cuya única tarea sea la obtención de información acerca del problema. Este tipo de reglas se denominan reglas de arranque, Cuando estas reglas se activan, realizan alguna consulta sobre alguna información del problema.

Después que el usuario selecciona el problema particular, el sistema dirige la resolución del problema hacia una área especifica. Esta simple pieza de información dirige al sistema para que considere problemas concernientes a la situación. El sistema continuará realizando preguntas para dirigir la resolución del problema hacia la conclusión más lógica.

5.3.2.2.3 Definir estructuras para el manejo de los datos

Teóricamente un sistema de encadenamiento hacia adelante funciona activando las reglas cuyas premisas correspondan con la memoria de trabajo.

En aplicaciones pequeñas esta forma de controlar la activación de las reglas puede ser adecuado, sin embargo en la mayoría de los casos es necesario incluir en cada regla una premisa que ayude a controlar cuando una regla debe ser activada.

5.3.2.2.4 Escribir código inicial

EL propósito de esta tarea es determinar si se ha capturado efectivamente el conocimiento del problema en una buena estructura de las reglas. Una buena estructura no es solo aquella que brinda resultados correctos sino también un patrón que seguir para el desarrollo de otras reglas. La tarea anterior es la que nos provee la estructura para estas reglas.

Una vez que hemos limitado nuestro problema a un problema específico, generalmente se necesitara una regla que cambie a otra tarea de verificación o seguir el razonamiento del SE.

5.3.2.2.5 Probar el sistema

Esta tarea verifica el conjunto de reglas. Se asume que “Tarea iniciada” ha sido inicializada en la memoria de trabajo. Esto causa la activación de la regla y el sistema pregunte Cuál es el problema El sistema entonces busca todas las reglas por la premisa que empareja, y continuar el proceso probando la interacción del SE y el usuario.

5.3.2.2.6 Diseñar la interfaz

Una vez que tenemos un conjunto de reglas trabajando adecuadamente el próximo paso es construir la interfaz del sistema. La interfaz es un componente muy importante del sistema. Deberíamos diseñar esta en paralelo con ella base de conocimientos y no después, ya que la forma en que se diseñe la base de conocimientos dependerá del diseño de la interfaz.

5.3.2.2.7 Expandir el sistema

Esta tarea expande el conocimiento del sistema. La expansión debe incluir también el diseño de pantallas de la interfaz y reglas que desplieguen las pantallas.

5.3.2.2.8 Evaluar el sistema

La tarea de evaluación se refiere a verificar el prototipo que tenemos del sistema con algún caso real de verificación. Normalmente esta verificación debería ser realizada por el experto, sin embargo si se uso, utilizando un manual de referencias técnicas, se puede trabajar solo con éste.

5.4 CLIPS

5.4.1 Definición

CLIPS (C Languaje Integrated Production System) es una herramienta de desarrollo para sistemas expertos creada en 1984 por el grupo ***Software Technology Branch***[44], en el centro espacial **Lyndon B. Johnson**[45] de la NASA,

CLIPS ofrece un entorno completo para el desarrollo de Sistemas Expertos.

Según **Giarratano y Riley**[46]:

[44]CLIPS Reference Manual Volume II Advanced Programming Guide. 2008. disponible en línea <http://clipsrules.sourceforge.net/documentation/v630/apg.htm> [con acceso 2009-05-04]

[45]Pagina web del instituto <http://www.nasa.gov/centers/johnson/home/index.html> [con acceso 2009-05-01]

[46] GIARRATANO Joseph, RILEY Gary. Sistemas expertos principios y programación. Editorial: Thomson Learning. Edición: 2000. p. 328

> "E*s un lenguaje de programación con paradigmas múltiples que proporcionan soporte para programación basada en reglas, orientada a objetos y por procedimientos*"

La versión estándar de CLIPS proporciona un entorno de desarrollo interactivo orientado a texto, incluyendo una herramienta de depuración, ayudas on-line y un editor integrado, aunque se han desarrollado interfaces visuales para plataformas Macintosh, Windows 3.x y el sistema X Window.

CLIPS fue diseñado para el desarrollo de un software que fuese capaz de modelar el conocimiento humano, y que ofreciera alta portabilidad, un bajo costo y una facilidad de integración.

5.4.2 Antecedentes

Anterior a la creación de CLIPS, los sistemas expertos se desarrollaban en base a **LISP**[47], el cual es el segundo lenguaje de programación más antiguo que aun se utiliza. Hasta antes del desarrollo de CLIPS, LISP era el lenguaje de programación dominante en el área de la inteligencia artificial, sin embargo, a pesar del desarrollo de docenas de prototipos de aplicaciones de sistemas expertos y de amplias demostraciones del potencial de los mismos, solo una pequeña parte de esas aplicaciones era aplicada de forma regular, esto podría deberse en gran medida a la utilización de LISP como lenguaje base para el desarrollo de sistemas

[47] RUSSELL Stuart, NORVIG Peter. Inteligencia Artificial un enfoque moderno. México: Prentice Hall Hispano Americana, 1996. p. 19

expertos, como consecuencia de **tres problemas que enfrentaba el lenguaje**[48], primero la escasa disponibilidad de LISP en una amplia variedad de maquinas convencionales para la época, el alto costo de desarrollo de las herramientas y de hardware requeridos, y por último la mala integración de LISP con otros lenguajes, debido a la gran dificultad para hacer aplicaciones de código embebido.

La sección de inteligencia artificial determino que lo mejor sería el uso de un lenguaje convencional como C, lo que podría eliminar la mayor parte de los problemas. Aunque en principio algunas herramientas desarrolladas en lenguaje C podían ser ejecutadas, aun el costo de estas era muy alto, y solo se limitaba a una pequeña cantidad de equipos de hardware, además la disponibilidad y los tiempos de respuesta eran desalentadores.

Para satisfacer las necesidades de tiempo y costo de forma oportuna y efectiva, se hizo evidente la necesidad de un sistema experto, por lo que, la sección de inteligencia artificial del centro espacial centro espacial Lyndon B. Johnson se decidió a desarrollar un sistema experto propio basado en la herramienta C.

La versión prototipo de CLIPS se desarrollo en la primavera de 1985, debido a su portabilidad, extensibilidad, capacidad y bajo costo, fue rápidamente aceptado por la industria, el gobierno y la academia.

[48]What is CLIPS? [en linea] <http://clipsrules.sourceforge.net/WhatIsCLIPS.html> [con acceso 2009-04-29]

CLIPS ha contribuido enormemente a mejorara la capacidad de ofrecer tecnología de sistemas expertos a todo el sector público y privado para una amplia gama de aplicaciones y entornos informáticos.

Actualmente CLIPS es una herramienta de software de dominio público.

5.4.3 Características

Entre otras características de CLIPS podemos mencionar:

1. Su integración con otros lenguajes de programación como C, C++, Ada, Java, entre otros.
2. Se puede llamar desde un lenguaje procedimental, realizando su función y devolviendo el control al programa que le llamó.
3. También se puede definir código procedimental como funciones externas llamadas desde CLIPS, y cuando el código externo finaliza su ejecución devuelve el control a CLIPS.
4. CLIPS distingue mayúsculas y minúsculas (case-sensitive), al igual que el lenguaje C.

Además de las características anteriores podemos mencionar que la CLIPS ofrece paradigmas heurísticos y procedimentales para la representación del conocimiento.

El conocimiento heurístico hace referencia a las reglas, las cuales permiten especificar un conjunto de acciones a realizar para una situación dada, es el creador del sistema experto quien crea estas reglas, que en conjunto son las resuelven los problemas.

El conocimiento procedimental hace referencia a las funciones y los objetos, este se expresa mediante funciones definidas por el creador del sistema experto o *deffunction*, funciones genéricas y programación orientada a objetos.

CLIPS permite el desarrollo de herramientas ya sea implementando solo reglas, solo funciones y objetos, o la combinación de ambos.

5.4.4 Estructura de los programas

La estructura de un programa en CLIPS consta de un Shell, el cual se encarga de realizar inferencias y razonamientos, proveyendo los elementos básicos de un sistema experto, como son la memoria global de datos o memoria de trabajo, la cual está conformada por entidades que corresponden o bien a hechos, o bien a instancias de una clase de objeto, es decir, o bien una lista de hechos o bien una lista de instancias; la base del conocimiento que contiene las reglas de la base de reglas; y el motor de inferencias, el cual controla la ejecución global de las reglas, de tal forma que decide que regla se ejecuta y cuando lo hace.

Por lo que un programa hecho en CLIPS puede consistir de hechos, reglas y objetos.

5.5 LA TECNOLOGIA JAVA

La tecnología Java [49] tiene su fundamento en su lenguaje de programación y su plataforma.

5.5.1 El lenguaje de programación Java

El lenguaje de programación Java es un lenguaje de alto nivel que es caracterizado por todas estas palabras:

Simple, Arquitectura neutral, Orientado a objetos, Portable, Distribuido, Alto Rendimiento, Multi Hilo, Robusto, Dinámico, Seguro

Cada una de las anteriores características se explican en mayor detalle en "The Java Language Environment"[50] , un paper escrito por James Gosling y Henry McGilton.

En el lenguaje de programación, todos los códigos fuentes inicialmente están escritos archivos de texto plano terminando con la extensión .java . Estos archivos son compilados para obtener los archivos .class por el javac compiler. Un archivo .class no contiene código nativo para algún procesador en especial; en lugar de esto contiene códigos en byte— que es interpretado por la maquina

[49]About the Java Technology [en línea] <http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html> [con acceso 2009/04/20]

[50]The Java Language Environment. 2006 [en línea] <http://java.sun.com/docs/white/langenv/> [con acceso 2009/04/20]

virtual *Java Virtual Machine*[51] (Java VM). Después la herramienta de ejecución del java corre la aplicación como una instancia de maquina virtual de java.

Como Java VM esta disponible en muchos sistemas operativos, entonces los archivos .class son capaces de ejecutarse en Microsoft Windows, en Solaris TM Operating System (Solaris OS), Linux, o Mac OS. Algunas maquinas virtuales, tales como "Java HotSpot virtual machine"[52], realizar pasos adicionales en tiempo de ejecución para dar a su solicitud de un aumento de rendimiento. Esto incluye diversas tareas tales como encontrar los cuellos de botella de rendimiento y recompilar (a código nativo) utilizan con frecuencia los artículos del código.

Ilustración 12: Un repaso del proceso del desarrollo de software en java

5.5.2 La plataforma Java

Una plataforma es el conjunto o entorno de hardware o software en el cual un programa se ejecuta. Las plataformas mas populares son Microsoft Windows, Linux, Solaris OS, y Mac OS. Estas puede describirse como una combinación del

[51] Los términos "Java Virtual Machine" y "JVM" significa una máquina virtual para la plataforma Java

[52] Java SE HotSpot at a Glance [en línea] <http://java.sun.com/javase/technologies/hotspot/index.jsp> [con acceso 2009/04/20]

sistema operativo y el hardware subyacente. La plataforma Java difiere de las demás plataformas en que, una misma aplicación puede correr en diferentes plataformas o componentes de hardware sin necesidad de recompilar en una plataforma especifica.

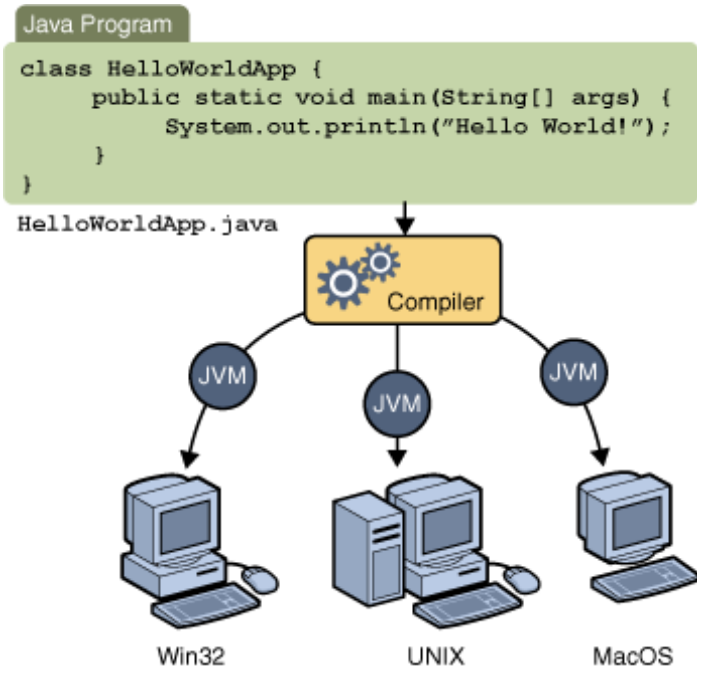

Ilustración 13: ejecución en múltiples plataformas.

La plataforma Java tiene dos componentes:

- la maquina virtual (Java Virtual Machine)
- la interfaz de programación de aplicaciones (Java Application Programming Interface - API)

La maquina virtual es la base para la plataforma Java y esta ha sido portada a varias plataformas basadas en un hardware especifico.

La API es una larga colección de componentes de software que ofrece al programador varias posibilidades. Estos componentes (que físicamente son archivos) se agrupan en las bibliotecas de las clases e interfaces; estas bibliotecas se conocen como paquetes.

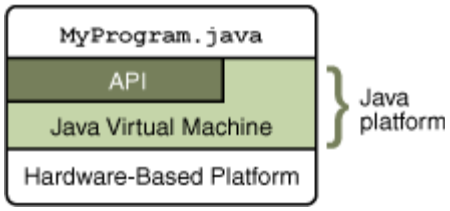

Ilustración 14: La API y la Máquina virtual de Java

Como es un entorno de plataforma independiente, la plataforma de java puede ser un poco mas lenta que la ejecución de una aplicación desarrollada bajo código nativo. Sin embargo, los avances en el compilador y las tecnologías de virtualización hacen que el rendimiento de las aplicaciones sean cercanas al de código nativo sin poner en peligro la portabilidad.

5.5.3 Aplicabilidad

El lenguaje de programación de java al ser un lenguaje de alto nivel en el que su desarrollo de las aplicaciones pueden ser empleado para propósito general y su

plataforma. Permite la que sus productos de software pueda tener las siguientes características [53]:

- Herramientas de desarrollo: son un conjunto de aplicaciones que permiten realizar las tareas necesarias para compilar, correr, monitorizar, depurar y documentar los programas o aplicaciones. Las principales herramientas que se utilizan son el compilador javac, el lanzador java, y la documentación de la herramienta javadoc.

- Interfaz de programación de aplicaciones (API): La API ofrece el núcleo de de la funcionalidad del Lenguaje de programación. Este ofrece una amplia gama de clases de utilidad lista para su uso en sus propias aplicaciones. Esta conjunto Que abarca todo, desde objetos básicos, a la creación de redes y la seguridad, a la generación de XML y acceso a bases de datos y mucho mas. El núcleo de API es muy grande, para obtener una visión general de lo que contiene, consulte la su documentación (Java SE Development Kit 6 (JDKTM 6) documentation[54])

- Tecnologías de despliegue: el conjunto de herramientas de aplicaciones de desarrollo de Java (JDK Java Development Kit) ofrece algunos

[53]What Can Java Technology Do? [en linea] <http://java.sun.com/docs/books/tutorial/getStarted/intro/cando.html> [con acceso 2009/04/20]

[54]JDK TM 6 Documentation [en línea] <http://java.sun.com/javase/6/docs/index.html> [con acceso 2009/04/20]

mecanismos que permiten que ponerlas al servicio del usuario final tales como Java Web start y los diferentes plungins.

- Herramientas para interfaces de usuario: las herramientas Java Swing y Java 2D permiten crear sofisticadas interfaces gráficas de usuarios (GUIs).

- Bibliotecas de integración: las bibliotecas de integración como Java IDL API, JDBCTM API, Java Naming and Directory InterfaceTM ("J.N.D.I.") API, Java RMI, y Java Remote Method Invocation over Internet Inter-ORB Protocol Technology (Java RMI-IIOP Technology) permiten acceder a bases de datos y a objetos remotos.

5.5.4 Ventajas

Las ventajas[55] mas significativas que tiene Java, en especial su lenguaje de programación, con respecto a otros lenguajes de programación, es que el; proceso de desarrollo hacen que sus aplicaciones tengan mejor rendimiento y requiere menos esfuerzo que en otros lenguajes. La tecnología de Java puede ser útil por los siguientes aspectos:

- Rápido aprendizaje: el lenguaje de programación de java es un poderoso lenguaje orientado a objeto, fácil de aprender, especialmente para programadores familiarizados con C o C++.

[55]How Will Java Technology Change My Life? [en línea] <http://java.sun.com/docs/books/tutorial/getStarted/intro/changemylife.html> [con acceso 2009/04/20]

- Menos escritura de código: comparaciones métricas de software (la cuenta de clases, la cuenta de métodos, y así sucesivamente) sugieren que un programa escrito en el lenguaje Java es cuatro veces mas pequeño que ese mismo programa escrito en C++.

- Mejor redacción del código: el lenguaje de programación de Java alienta buenas practicas de codificación, y la optimización de los recursos del sistema en lo que se refiere a la limpieza de objetos en memoria (garbage collection) que permiten un mejor desempeño de las aplicaciones. Su orientación a objetos, su arquitectura de componentes (JavaBeansTM), su amplio alcance, y fácilmente extensibles API le permiten reutilizar los componentes existentes, e introducir el código probado menos numero de *bugs*.

- Desarrollo de programas mas rápido: le lenguaje de programación de Java es mas simple que C++, y por pende en el desarrollo de aplicaciones , el tiempo de desarrollo podría ser hasta dos veces más rápido al escribir en él. Sus programas también requieren menos líneas de código.

- Independencia de la plataforma: se puede mantener un programa portable en las diferentes plataformas y no lo amarra al uso de librerías como ocurre en otros lenguajes.

- Escríbase una vez, ejecútelo donde quiera: ya que las aplicaciones escritas en el lenguaje de programación son compiladas dentro de una maquina virtual cuyo código no depende de una arquitectura y plataforma especifica, esto permite que el programa pueda correr en cualquier plataforma que tenga la maquina virtual de java instalada.

- Distribución de software fácilmente: con la aplicación Java Web Start, el usuario puede ejecutar cualquier aplicación con un solo clic. Una revisión automática de versiones al iniciar asegura que usuario este utilizando la ultima versión del programa. Si hay actualizaciones disponibles, entonces Java Web Start actualizará automáticamente la instalación.

5.6 JESS

5.6.1 Definición

JESS (Java Expert System Shell) es un motor de reglas de propósito general desarrollado con base en la plataforma JAVA. Para su uso, se especifica una lógica a seguir en forma de reglas, estas se pueden presentar en dos formatos, bien sea por el lenguaje de reglas de JESS (preferiblemente) o en XML.

JESS es un súper conjunto del lenguaje CLIPS, desarrollado por Enerst Friedman-Hill en Sandia National Labs.

JESS Ofrece una programación basada en reglas para la automatización de un sistema experto. Mas que una cuestión del paradigma de programación, donde un solo ciclo que se activa se ejecuta solo una vez; en el paradigma declarativo usado por JESS, continuamente aplica una colección de reglas a una colección de hechos atreves de un proceso llamado asociación de patrones. Las reglas pueden modificar la colección de hechos, o pueden ejecutar cualquier código en JAVA.

El motor de reglas de JESS utiliza el **algoritmo de Rete**[56] [57], el cual acelera en gran medida la correspondencia de los hechos que se añaden o se modifican con los respectivos antecedentes.

JESS puede ser utilizado para realizar aplicaciones en JAVA con pleno aprovechamiento del conocimiento por medio de reglas declarativas, con el fin de sacar conclusiones y hacer inferencias. Está licenciado para el uso comercial, pero está disponible de forma libre para el uso académico.

5.6.2 Antecedentes

La primera versión de JESS fue escrita a finales de 1995, mientras Enerst Friedman-Hill trabajaba en el departamento de computación científica del Sandia National Laboratories, en Livermore (California).

[56] JESS ®, the Rule Engine for the JavaTM Platform [en línea] <http://www.jessrules.com/ > [con acceso 2009-05-03]

[57] GIARRATANO Joseph, RILEY Gary. Sistemas expertos principios y programación. Editorial: Thomson Learning. Edición: 2000. p. 32

Enerst Friedman-Hill se encontraba escribiendo agentes de software que gestionaran dinámicamente la computación distribuida a través de redes, estos agentes de ejecutaban en cada máquina usando un método llamado “post and bid”, este método permitía decidir que maquina ejecutaría una determinada fracción de código, basado la capacidad de balanceo de carga de la máquina, sus “cerebros” motores de reglas de los sistemas de software que usando dichas reglas eran capaces de obtener conclusiones a partir de premisas. Ese proyecto dio lugar a otros similares, y pronto desarrollo un gran interés por los agentes que pueden viajar atreves de los nodos de una red de computadoras, manteniendo el estado de los mismos. Así nació la idea de un motor de reglas cuyo estado podría ser empaquetado, enviado a través de un cable y luego reconstituido. Para ese entonces estaba siendo liberado el lenguaje JAVA, y este fue visto como el vehículo perfecto para desarrollar dicho motor de reglas, dando lugar a lo que hoy conocemos como JESS (Java Expert System Shell).

A pesar de ser desarrollado sobre JAVA, la sintaxis de JESS es similar a la de LISP, y aunque en un principio podría parecer extraño es en realidad sencillo, es fácil de aprender y bien adaptado tanto a la definición de reglas como a la programación procedimental.

La inspiración original para desarrollar JESS fue CLIPS, sin embargo, sus implementaciones han sido muy diferentes. JESS es muy dinámico y centrado en JAVA, por lo que automáticamente obtiene acceso a todas las herramientas API

de JAVA para trabajos de red, gráficos, acceso a base de datos y muchas otras opciones con las cuales CLIPS no cuenta. A pesar de lo anterior, existe una gran similitud entre el lenguaje de reglas soportado por ambos sistemas, debido a que muchos de los conceptos fundamentales de JESS fueron derivados de CLIPS.

5.6.3 Características

Entre las principales características de JESS podemos mencionar:

1. La utilización de algoritmo de rete para el procesamiento de reglas.
2. Puede obtener mayor rapidez de procesamiento que CLIPS.
3. Incluye el encadenamiento hacia atrás (a partir de la versión 5.0).
4. Permite manipulación directa de objetos tipo JAVA (Incluso scripts), esto debido a que JESS está programado en JAVA y sus clases se usan como cualquier otra, por lo que se pueden añadir nuevas funciones a JESS programadas en JAVA al igual que nuevos paquetes de funciones.
5. Incluye lógica difusa (FuzzyJess a partir de la versión 5.0)
6. Posee 3 interfaces de trabajo: mediante líneas de comando por medio de la clase “Jess.Main”, mediante consola a través de la clase “Jess.Console” o mediante applets a través de la clase “Jess.ConsoleApplet”, esta ultima solo incorpora lo esencial de JESS para minimizar el tamaño de las paginas.

7. Cada motor de inferencia (objeto de la clase “Jess.Rete”) tiene asociado un espacio de almacenamiento de este tipo.

8. Desde JESS se puede crear una instancia de cualquier clase de JAVA y tener acceso a sus métodos y atributos públicos.

Además de las características anteriores podemos mencionar que al igual que CLIPS, JESS también ofrece paradigmas heurísticos y procedimentales para la representación del conocimiento.

5.6.4 Aplicaciones

JESS se ha aplicado en el desarrollo de una amplia gama de software comercial entre la que podemos mencionar:

1. Sistemas expertos que evalúan reclamaciones de seguros y aplicación de hipotecas.

2. Agentes que permiten predecir los precios de las acciones para comprar y vender valores.

3. Redes de detectores de intrusos y auditores de seguridad.

4. Asistentes de diseño que ayudan a los ingenieros mecánicos.

5. Conmutadores inteligentes de red para telecomunicaciones.

6. Servidores para ejecutar reglas de negocios.

7. Sitios inteligentes de comercio electrónico.

8. Juegos.

Las más recientes aplicaciones de los sistemas expertos incluyen el razonamiento como parte de los agentes inteligentes, como por ejemplo en los sistemas de planificación de recursos empresariales (ERP) y ordenes de validación para comercio electrónico.

6 MARCO METODOLÓGICO

Para conseguir las objetivos puestos en esta investigación se estimaron un conjunto tareas teniendo en cuenta los procesos de desarrollo de software en lo que respecta a la metodología de la ingeniería del conocimiento.

Se resaltan las siguientes actividades como las básicas dentro del proceso:

La Compilación de información relacionada herramientas para diseño e implementación de Sistemas Expertos, así como de JESS con el fin de mirar la fortaleza de estas herramienta.

El Análisis de las respectiva herramienta, especialmente su fortaleza en integración con otras herramientas de desarrollo (Entornos integrados de desarrollos (IDE), lenguajes de programación, bases de datos, en integración con otros componentes).

El diseño de sistemas cuyo producto final sean aplicativos que tenga como objetivo principal las ventajas y fortalezas que ofrece JESS.

La Abstracción de los requerimientos e identificación de componentes dentro del sistema (especificación de requerimientos, diseño de sistemas y modelado en UML)

Entrega preliminar de informe final.

Entrega definitiva del informe final y sustentación de la investigación.

7 APLICACIONES

A continuación se va a mostrar varios productos obtenidos por la investigación acerca de la herramienta para el desarrollo de sistemas experto que abarcan temas desde la lógica proposicional hasta el desarrollo de una maquina de inferencia para el apoyo de un tutor inteligente. Además de varios productos realizados por otras investigaciones en otros grupos de investigación de varias universidades e instituciones alrededor del hemisferio. Estas aplicaciones tienen gran utilidad en el campo de la enseñanza por su carácter pedagógico

7.1 El americano.

Una forma de probar la fortaleza que tiene JESS como maquina de inferencia es probarla con un caso de una inferencia de predicados.

Tenemos que:

Una persona nace en Cartagena de indias.

Cartagena de Indias es una ciudad de Colombia

Colombia es un país del continente americano.

Por ende todas las personas nacidas en Cartagena de indias son americanas.

Aquí se aplica la regla de inferencia para la lógica proposicional conocida como la regla de la cadena o ley del silogismo.

Siendo p, q y r proposiciones y el operador -> denota condicional (p->q se denota como "si p entonces q") podemos expresar la ley del silogismo mediante el siguiente esquema.

p->q

q->r

---

p->r

En JESS el planteamiento se resuelve de la siguiente forma:

```
; ejemplo de inferencia de predicados

; "Pepe es una persona"
; "Pepe vive en Cartagena"
; "Toda persona que viva en Cartagena es Colombiana"
; "Toda persona que viva en Colombia es Americana"

;introduciendo los hechos

; pepe es una persona

(assert (persona pepe))
; pepe nació en Cartagena

(assert (Cartagena pepe))
```

```
;definiendo las reglas


; regla que identifica que una persona nacida en Cartagena es colombiana

(defrule Colombiano

    ; x es una persona

    (persona ?x)
     ; x nació en Cartagena

    (Cartagena ?x)
   =>
    ; x es colombiano

    (assert (Colombiano ?x))
    (printout t ?x " Colombiano." crlf)


)

; regla que define que una persona nacida en Colombia es americana

(defrule Americano
    ;x es una persona

    (persona ?x)
    ; x es colombiano

    (Colombiano ?x)

   =>
     ;x es americano.

    (assert (Americano ?x))
    (printout t ?x " es Americano." crlf)

)
```

Este ejemplo al ser ejecutado dentro de la maquina de inferencia comprobara la regla de la cadena o ley del silogismo.

7.2 Una ventana hecha en JESS

Como JESS es un maquina de inferencia que combina un sistema basado en reglas con la flexibilidad que puede acceder a objetos de java. Por ende la hace una herramienta poderosa en el momento de la creación de sistemas experto y su fácil integración a aplicaciones de propósito generales en el las aplicaciones desarrolladas en el lenguaje de programación Java tiene mucha aceptación.

El ejemplo es muy sencillo.

Se tiene como escenario una ventana que tiene un botón que al presionarlo este muestra otro frame como respuesta al llamado del evento de haber presionado el botón.

El comportamiento del sistema se puede modelar con un diagrama de secuencias.

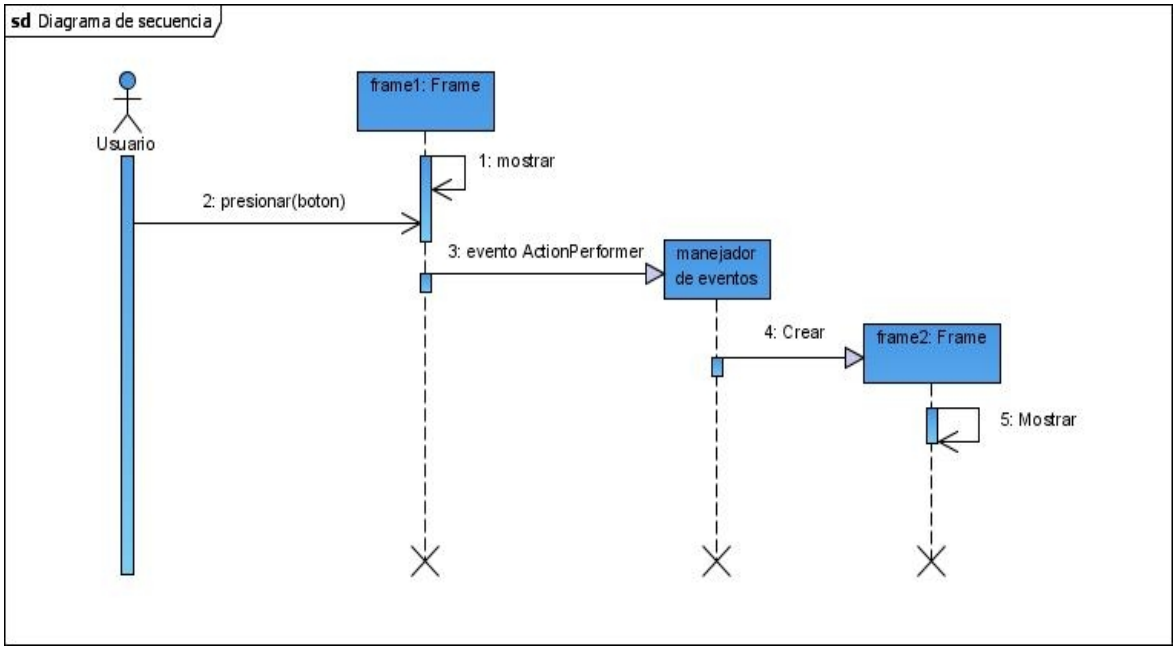

Ilustración 15: Diagrama de Secuencias que muestra el comportamiento de los eventos presentes en la ventana

Con JESS podemos crear interfaces gráficas de usuarios ya que este nos permite manipular objetos construido en java en el cual las clases del paquete AWT no le es indiferente. Ante esto podemos embeber objetos de java en el lenguaje de reglas del JESS.

Para agregar los paquetes de java en JESS se procede a agregar los paquetes como sentencias de JESS por ejemplo:

```
;; las importaciones van encerradas entre paréntesis
(import java.awt.*)
(import java.awt.event.*)
```

(import java.awt.event.WindowEvent)

para realizar una instancia o declaración de un objeto java dentro de JESS, la instancia de un objeto de java puede ser asignada a una variable local o global de JESS, dependiendo el alcance de este ultimo dentro del programa. Por ejemplo:

```
;; definiendo un frame dentro de una variable global
(defglobal ?*f* = (new Frame "ventana de Prueba"))
```

para acceder a métodos de un objeto de java dentro de JESS se debe tener en cuenta la siguiente estructura:

```
(obj metodo param1 param2 ... paramn)
```

Siendo

obj: el objeto o variable instanciada

metodo: metodo perteneciente al objeto

param: parámetros del metodo.

Por ejemplo:

```
;; ensamblando y mostrando la IGU
(?*f* setSize 200 60)
```

;; agregando u botón al frame

(?*f* add ?*b*) .

para modelar el comportamiento del sistema se tendrá en consideración:

En el momento de que el usuario presione el botón del frame entonces se genera un hecho.

El experto debe tener en consideración una regla que se active cuando se genere un hecho que identifique el evento generado por el usuario al momento que el usuario presionó el botón del frame.

La regla puede ser expresada de la siguiente forma:

Sea P(x, v): el usuario x presionó el botón titulado “presiona aquí” del frame v

Sea G(y, z): la ventana y muestra la ventana de bienvenida z

P(x, v) => G(v, z)

El modelo de este sencillo experto se puede separar en tres archivos

frame.clp: fuente que nos muestra el diseño, ensamblaje y la asignación de eventos

enventoboton.clp: fuente que expresa la regla P(x, v) => G(v, z)

respuesta.clp: fuente que nos muestra el diseño del frame que actúa como mensaje de respuesta por ejecución de la regla.

Los archivos se citan a continuación:

```
///frame.clp

(import java.awt.*)
(import java.awt.event.*)
(import java.awt.event.WindowEvent)

;; definiendo un frame dentro de una variable global
(defglobal ?*f* = (new Frame "ventana de Prueba"))

;; definiendo un botón dentro de una variable global
(defglobal ?*b* = (new Button "Presiona aquí"))

;; implementándose la interfaces acctionlistener
(?*b* addActionListener
  (
    implement ActionListener using
    (
       lambda (?name ?evt)
         (batch c:/Practicas/eventoBoton.clp)
         (assert (BotonPinchado botón))
```

```
        (run)
     )
  )
)

;; ensamblando y mostrando la IGU
(?*f* setSize 200 60);
(?*f* add ?*b*)

(set ?*f* visible TRUE)

;; agregando el evento WINDOW_CLOSING
(?*f* addWindowListener  (implement WindowListener using (lambda (?name ? event)
    (if (= (?event getID) (WindowEvent.WINDOW_CLOSING)) then
        (exit)))))

// eventoboton.clp

(import javax.swing.*)
(defglobal ?*msg* =  (new JOptionPane))

(defrule pinchado
          (BotonPinchado ?x)
          =>
```

```
        (printout t "llamado al botón pinchado" crlf)
        (batch c:/Practicas/Respuesta.clp)
)

//respuesta.clp
(import java.awt.*)
(import java.awt.event.*)
(import java.awt.event.WindowEvent)

;; definiendo un frame dentro de una variable global
(defglobal ?*f* = (new Frame "respuesta al hecho"))

;; definiendo un botón dentro de una variable global
(defglobal ?*l* = (new Label "has presionado el botón"))

;; ensamblando y mostrando la IGU
(?*f* setSize 200 60);
(?*f* add ?*l*)
(set ?*f* visible TRUE)
;; agregando el evento WINDOW_CLOSING
(?*f* addWindowListener  (implement WindowListener using (lambda (?name ? event)
   (if (= (?event getID) (WindowEvent.WINDOW_CLOSING)) then
      (exit)))))
(run)
```

7.3 Tomador de decisiones

A continuación se describe el análisis de un experto simple para un sistema evaluador o tutor; la función que cumple este experto es de actuar como componente para apoyar la toma de decisiones de un tutor.

Este sistema apoyara en la toma de decisiones el tutor en dos aspectos mencionados a continuación:

1. Retroalimentación a las preguntas.
2. Análisis de niveles.

Retroalimentación a las preguntas

El procedimiento es sencillo. para esta aplicación, el experto recibirá la información del estudiante (su identificador seguido de su opción de respuesta, objeto clave para la reacción del experto) y éste de acuerdo con el criterio de decisión generara una respuesta que va a ser de gran importancia para el tutor.

El procedimiento se describe a continuación:

1. El experto recibe la información del estudiante que es enviada por un componente o subsistema del tutor.

2. El experto analiza la información del estudiante y genera una respuesta de acuerdo al criterio de decisión.

3. El experto envía la respuesta al componente o subsistema que le envió la solicitud del análisis.

Con base a la descripción anterior; en este momento podemos mirar el experto como aplicación en la cual recibe unos parámetros de entrada (la cual son el identificador del estudiante y la respuesta de éste) y genera una salida el cual es la respuesta obtenida de acuerdo a los parámetros de entrada.

Vale la pena recordar que el experto es un sistema basado en conocimientos. Por ello su base para su funcionamiento radica en el uso de hechos y reglas que actúan con base a dichos hechos; entonces para que el experto realice su función este debe, de antemano, tener en cuenta cuales son los hecho que va a identificar de acuerdo con un patrón determinado y la reglas, las cuales va a reaccionan con base a una serie determinada de hechos. Esto último es lo que se conoce como criterio de decisión del experto. Para desarrollar el criterio de decisión a esta aplicación nos basaremos en los siguientes preceptos:

Parámetros de entrada:

Id del estudiante: es un número que identifica el estudiante dentro del sistema.

Id de la pregunta que el estudiante respondió.

La eficiencia para esa pregunta

Salida: un objeto de tipo respuesta con dos atributos:

id del estudiante.

respuesta: respuesta del experto obtenido después de que el éste aplicara su criterio de decisión. Esta se limita a tres valores:

0: ordena al tutor obtener una retro alimentación mala para esa pregunta.

1: ordena al tutor obtener una retro alimentación buena para esa pregunta.

2 : ordena al tutor obtener la mejor retro alimentación para dicha pregunta

Ahora bien los preceptos anteriores se pueden modelar en el campo de la ingeniería de conocimiento como un patrón de hechos la cual se describe a continuación.

Sea RespuestaEstudiante(x, p ,r) las respuesta r para la pregunta p dada por el estudiante x.

Sea EficienciaPregunta(p, v) el valor v que tiene la pregunta para el análisis del nivel.

Sea RangoEficienciaPregunta(p,min,max) el rango de eficiencia que el experto tomará al evaluar a un estudiante su desempeño realizado en la solución de la pregunta p

Sea Salida(x,p,s) la solución s del experto.

El hecho RespuestaEstudiante(x, p ,r) activará, dependiendo de los resultados obtenidos de las reglas decisión, el avance y los logros que el estudiante alcance con el tutor.

El hecho Salida(x,p,s) actuará como elemento condicional o patrón, dependiendo de la activación del hecho RespuestaEstudiante(x, p ,r), y mostrara a los usuarios el resultado del tomador de decisión.

Con base a los patrones ya identificados y declarados anteriormente, las reglas a las cuales reaccionará el experto se describen a continuación:

RespuestaEstudiante(x,p,r)^EficienciaPregunta(p,v)^RangoEficienciaPregunta(p,min,max) ^ v> max => salida(x,p,2).

RespuestaEstudiante(x,p,r)^EficienciaPregunta(p,v)^RangoEficienciaPregunta(p,min,max) ^ min< v< max => salida(x,p,1).

RespuestaEstudiante(x,p,r)^EficienciaPregunta(p,v)^RangoEficienciaPregunta(p,min,max) ^ v< min => salida(x,p,0).

Dada la definición de este mecanismo de reglas y hecho podemos diseñar el sistema experto como basado en reglas con encadenamiento hacia delante.

Tomando JESS como lenguaje declarativo para la definición del criterio del experto entonces los hechos como las reglas anteriormente citadas quedan descritas a continuación:

```
; plantillas para el análisis de las preguntas

(deftemplate RespuestaEstudiante
   "una plantilla que guardara la pregunta y al respuesta realizada al estudiante"
   (slot idUsuario (type INTEGER))
   (slot idPregunta (type INTEGER))
   (multislot Respuestas (type STRING))
)

(deftemplate EficienciaPregunta
   "la eficiencia de la pregunta"
   (slot idPregunta (type INTEGER))
   (slot valor (type FLOAT))
)

(deftemplate RangoEficienciaPregunta
```

```
    "limites de la eficiencia de la pregunta"
    (slot idPregunta (type INTEGER))
    (slot min (type FLOAT))
    (slot max (type FLOAT))
)
```

A continuación se definen las reglas de decisión para determinar el avance del estudiante, estas reciben como parámetros el ID del estudiante y la respuesta a la pregunta formulada por el tutor:

La regla de decisión retroalimentacion_excelente se aplicara cuando la respuesta des estudiante tenga un alto grado de eficiencia que supere los limites de rango de eficiencia estándar

```
(defrule retroalimentacion_excelente
        "obtiene la mejor retroalimentación para el estudiante"
      (EficienciaPregunta (idPregunta ?id) (valor ?v))
      (RangoEficienciaPregunta (idPregunta ?id) (min ?mn) (max ?mx))
      (EficienciaPregunta {valor > ?mx})
      =>
      (assert (salida 2 ))
)
```

La regla de decisión retroalimentacion_buena se aplicara cuando la respuesta del estudiante tenga un grado de eficiencia que este dentro de los límites del rango de la eficiencia estándar

```
(defrule retroalimentacion_buena
        "obtiene una retroalimentación buena"
      (EficienciaPregunta (idPregunta ?id) (valor ?v))
      (RangoEficienciaPregunta (idPregunta ?id) (min ?mn) (max ?mx))
      (EficienciaPregunta {valor <= ?mx && valor >= ?mn })
      =>
      (assert (salida 1 ))
)
```

La regla de decisión retroalimentacion_malo se aplicara cuando la respuesta del estudiante tenga un grado de eficiencia que esta por debajo del límite inferior del rango de la eficiencia estándar

```
(defrule retroalimentacion_malo
        "obtiene la mala retroalimentación para el estudiante"
      (EficienciaPregunta (idPregunta ?id) (valor ?v))
      (RangoEficienciaPregunta (idPregunta ?id) (min ?mn) (max ?mx))
      (EficienciaPregunta {valor < ?mn})
      =>
      (assert (salida 0 ))
)
```

Análisis de niveles

La segunda función que cumple la maquina de inferencia es brindar al tutor apoyo en la toma de decisión en lo que respecta a la selección del nivel de las pregunta que el tutor le de al estudiante en el momento de armar su cuestionario.

El procedimiento que realiza la máquina de inferencia es muy parecido al que hace para el apoyo de la retroalimentación de las respuestas para cada pregunta. La diferencia es que este análisis se realiza al finalizar un cuestionario por parte del estudiante.

Los pasos que realiza el experto se describen a continuación:

1. el tutor le envía la información al estudiante a la maquina de inferencia relacionada con el nivel actual y la eficiencia obtenida para dicho nivel.

2. La maquina de inferencia evalúa la eficiencia obtenida por el estudiante de acuerdo a algunos parámetros establecidos previamente.

3. Realizada la evaluación entonces la maquina de inferencia le envía la decisión al tutor en la que le suministra la información acerca del nuevo nivel de preguntas o si ya esta capacitado para recibir una nueva lección o si es necesario reprobar el nivel.

Para desarrollar el criterio de decisión a esta aplicación nos basaremos en los siguientes preceptos

Parámetros de entrada:

id del estudiante: es un número que identifica el estudiante dentro del sistema.

Id de la nivel en el cual el estudiante fue evaluado.

La eficiencia para dicho nivel

Salida: un objeto de tipo respuesta con dos atributos:

id del estudiante.

nivel: nivel sugerido por la maquina de inferencia para el estudiante este puede ser un valor superior, igual o inferior al nivel actual.

Podemos modelar este procedimiento de acuerdo a los conceptos expresados por Durkin en lo que respecta al diseño de sistemas experto basados en reglas:

definición de los hechos:

Sea NivelEstudiante(x,n,ef) el hecho que define el nivel n con eficiencia e obtenida por el estudiante x.

Sea RangoEficienciaNivel(n,min,max,pms) el hecho que define los límites de la eficiencia de (máximos – max y mínimos – min) con permisividad pms para el nivel n.

Sea NuevoNivel(x,m) el nivel m propuesto por la maquina de inferencia para el estudiante x.

Sea esNuevoNivel(x,n) el hecho cuando en estudiante no ha pasado por ese nivel

El termino de permisividad es aplicado al estudiante cuando este ha reprobado el nivel en el cual esta siendo evaluado. Este concepto le da mayor flexibilidad al tutor en el momento de evaluar un estudiante que presente estas característica.

Con base a los patrones ya identificados y declarados anteriormente, las reglas a las cuales reaccionará el experto se describen a continuación:

NivelEstudiante(x,n,ef) ^ RangoEficienciaNivel(n,min,max,pms) ^ ef > max => NuevoNivel(x,n+1)

NivelEstudiante(x,n,ef) ^ RangoEficienciaNivel(n,min,max,pms) ^ (min <= ef <= max) => NuevoNivel(x,n)

NivelEstudiante(x,n,ef) ^ RangoEficienciaNivel(n,min,max,pms) ^ ef < min => NuevoNivel(x,n-1)

es deber del tutor buscar un nivel en el cual el cuestionario de preguntas no halla sido realizado para ese estudiante para ello es necesario que existan m niveles con preguntas diferentes o en el peor de los casos que las preguntas comunes sean mínimas .

En el caso que el estudiante halla pasado por ese nivel entonces el experto evaluará RangoEficienciaNivel(n,min,max,pms) sobre la permisividad en el caso de que el estudiante halla transitado por ese nivel es decir:

NivelEstudiante(x,n,ef) ^ (no esNuevoNivel(x,n)) => RangoEficienciaNivel(n,min-pms,max-pms,pms) con pms>0

El hecho esNuevoNivel(x,n) también puede definirse como una función que obtiene de un almacén de datos dicha información. Por conveniencia esto se puede aplicar en la definición del hecho RangoEficienciaNivel(n,min,max,pms).

Utilizando JESS como el lenguaje declarativo para el diseño de este sistema experto, la anteriores reglas son codificadas como se muestra a continuación:

```
;conjunto de reglas para el análisis de los niveles.
(defrule subir_nivel
   "sube el nivel de un estudiante"
      (NivelEstudiante (idEstudiante ?estudiante) (idNivel ?nivel) (Eficiencia ? eficencia))
    (RangoEficienciaNivel (idNivel ?nivel) (min ?min) (max ?max) (Permisividad ? permisividad))
   (RangoEficienciaNivel {max < ?eficencia})
   =>
   (assert (nivel (+ 1 ?nivel)))
)
```

```
(defrule mantener_nivel
   "mantiene el nivel de un estudiante"
      (NivelEstudiante (idEstudiante ?estudiante) (idNivel ?nivel) (Eficiencia ? eficencia))
    (RangoEficienciaNivel (idNivel ?nivel) (min ?min) (max ?max) (Permisividad ? permisividad))
   (RangoEficienciaNivel {max >= ?eficencia && min <= ?eficencia})
   =>
   (assert (nivel ?nivel))
)

(defrule bajar_nivel
   "baja el nivel de un estudiante"
      (NivelEstudiante (idEstudiante ?estudiante) (idNivel ?nivel) (Eficiencia ? eficencia))
    (RangoEficienciaNivel (idNivel ?nivel) (min ?min) (max ?max) (Permisividad ? permisividad))
   (RangoEficienciaNivel {min > ?eficencia})
   =>
   (assert (nivel (- 1 ?nivel)))
)
```

Diseño.

La maquina de inferencia al ser un componente integrable al sistema tutor entonces este se puede diseñar como un paquete que a su vez ofrece un conjunto

de clases que actúa como interfaces entre el experto y el tutor y a su ves tanto al motor de reglas JESS como acceso a la base de datos del sistema tutor en modo de solo lectura. En la documentación de este componente se puede observar en mayor detalle en la sección de anexos. Para ello se utilizó el lenguaje de programación java así como la poderosa API que ofrece JESS para dicho lenguaje.

implementación

Este sistema experto fue implementado con éxito como un componente de un sistema tutor inteligente un producto de la investigación realizada por Reyes y de la Ossa 58 en esta investigación se detalla dicha integración con el sistema tutor.

7.4 Acceso a objetos con conectividad a bases de datos

Una de las ventajas que tiene JESS es que, además de tener la posibilidad de acceder directamente a objetos de java embebiendo código Java dentro del código de reglas de JESS, este lenguaje declarativo presenta una API para Java, es decir con conjunto de clases, interfaces, paquetes y manejadores de excepciones, todo esto escrito en código fuente de java o compilado, esto nos permite integrar fácilmente la fortaleza de JESS sumado a la flexibilidad del lenguaje de programación JAVA.

Aprovechado esa facultad de integración que tiene JESS entonces este puede acceder objetos presentes en la API de java que permiten conectividad con

bases de datos, java soporta varios estándares de acceso a base de datos como ODBC (Open Database Connectivity), JDBC (Java Database Connectivity), SQLJ, entre otros. Estas API, generalmente las diferentes empresas u organizaciones responsables del desarrollo e implementación sus respectivos sistemas de gestión de base de datos, permiten desarrollar aplicaciones en lenguaje Java utilizando sus sistemas gestores de base de datos, es por eso que en primera instancia sean las empresas las que desarrollen dichas API.

MySQL[58] es un sistema gestor de base de datos que ha ganado muchos adeptos debido a su flexibilidad, rapidez y popularidad Esta a su ves ofrece un conjunto de librería que permite a las aplicaciones desarrollada en Java interactuar con su sistema gestor de base de datos valiéndose del estándar JDBC o el estándar ODBC, o cualquier otro estándar en donde el sistema gestor de base de datos tenga cierta compatibilidad.

La siguiente aplicación muestra la fortaleza que tiene la maquina de reglas JESS en la manipulación de objetos que tienen conectividad con un sistema gestor de base de datos. Para nuestro ejemplo la aplicación tendrá como objetivo acceder y manipular información de una base de datos realizada en MySQL de acuerdo a la toma de decisiones que tome el experto.

[58] MySQL Community Server. 2008. Versión 5.0 Uppsala – Suecia: MySQL AB [Programa informático] disponible en línea <http://dev.mysql.com/downloads/mysql/5.0.html#downloads> [con acceso 2009-04-05]. bajo licencia GPL

Se tiene un analizador de estados de edades, el objeto del experto es que, dada la información básica de una persona (tales como su nombres, apellidos, numero de identificación y edad), este los clasifica en mayor de edad o menor de edad y los almacena en su tabla correspondiente.

El procedimiento se describe a continuación:

1. El usuario introduce la información básica referente a la persona.
2. El sistema almacena la información en la tabla persona.
3. El sistema evalúa la información referente a la edad de la persona
    1. Si la persona tiene una edad mayor o igual a 18 años entonces el sistema procederá a almacenar el código de la persona en la tabla de nombre Mayor.
    2. Si la persona tiene una edad inferior a los 18 años entonces el sistema procederá a almacenar el código de la persona en la tabla de nombre Menor.

Bajo este escenario el sistema se puede modelar fácilmente desde el punto de vista de la ingeniería de software, pero para mostrar el poderío de JESS en la manipulación de objetos con conectividad a base de datos. Se decidió modelar el tomador de decisión (que en el marco de la programación estructurada solo sería una rutina condicional if … else … end if) bajo el paradigma de la ingeniería de

conocimiento. Luego entonces se deja a disposición de un experto que resuelva este caso.

Entonces bajo el paradigma de la ingeniería de conocimiento se puede detallado por Durkin, podemos definir los hechos y la reglas que actúan de acuerdo a la interacción con los hechos de la siguiente manera:

Sea p(id,e) el hecho que corresponde a una persona identificada en el sistema con el identificador id y tiene una edad e.

Sea mayor(id) el hecho que afirma que la persona con identificador id es mayor de edad.

Sea menor(id) el hecho que afirma que la persona con identificador id es menor de edad.

El conjunto de reglas se definen a continuación:

p(id,e) ^ e > =18 => mayor(id) con e>0

p(id,e) ^ e < 18 => menor(id) con e>0

la relación entre las clases y sus componentes y la interacción tanto con el usuario, la base de datos y la maquina de reglas se muestra en el siguiente diagrama de componentes:

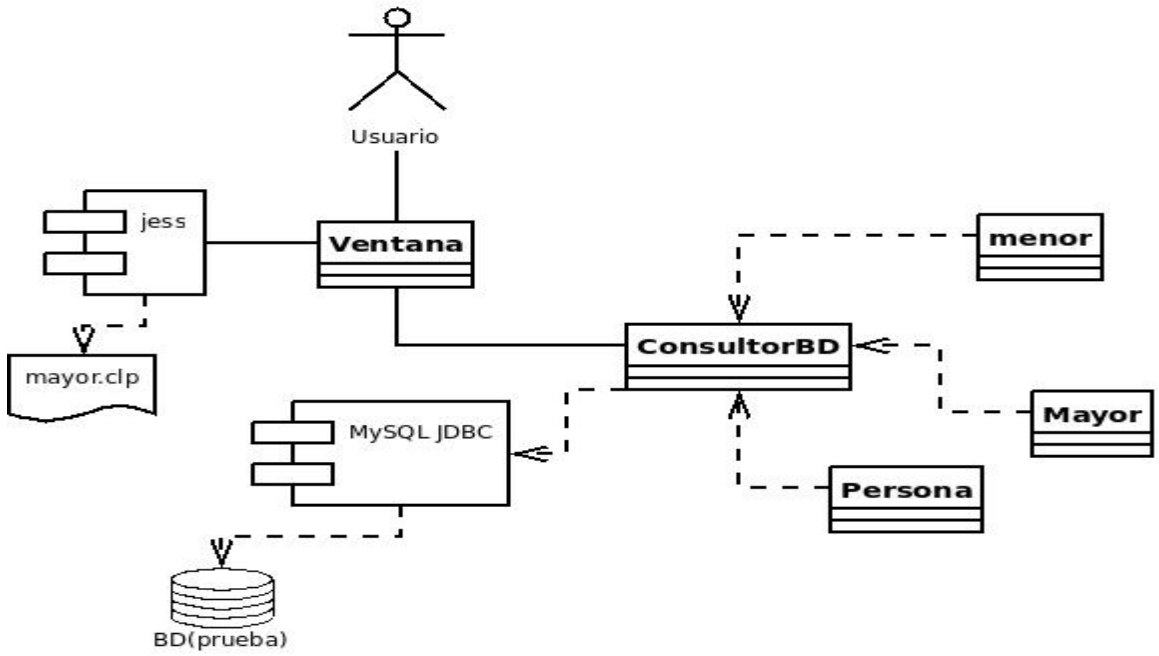

Ilustración 16: Diagrama de componentes del analizador de edades

De acuerdo a la ilustración mostrada, la clase ventana es donde confluyen los objetos relacionados tanto con la conectividad a la base de datos así como a la máquina de reglas.

La interfaz gráfica de usuario se muestra a continuación:

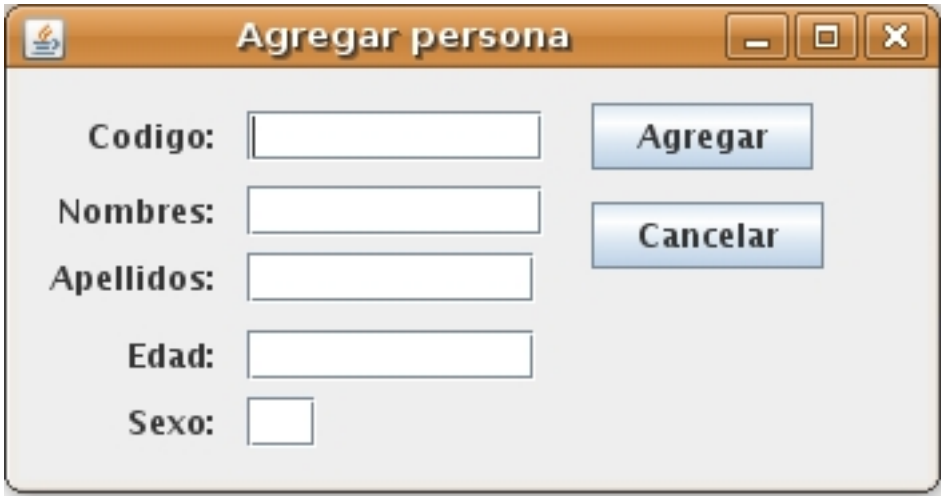

Ilustración 17: Interfaz gráfica del analizador de edades

En la documentación de este aplicativo se puede observar la información referente a cada uno de los objetos que la forman así como la relación de las clases y paquetes que interactúan con ella.

En el constructor de la clase Principal (ver documentación) inicializa tanto los objetos de acceso a base de datos como la carga del escenario principal de las reglas y las plantillas de hechos (marcos) a la máquina de reglas.

```
    Rete maquina;
    ConsultorBD cons;
public Principal() throws JessException {
        initComponents();
        cons= new ConsultorBD("root","adaluz02");
        maquina = new Rete();
```

```
        maquina.watchAll();
        maquina.batch("./interfaz/datos/mayor.clp");
    }
```

Aquí se esta utilizando la API de JESS para JAVA la clase principal de JESS es Rete. El método batch de la clase Rete permite cargar un archivo con código de JESS, el ese archivo es donde reposa las diferentes plantillas de hechos (marcos) y la definición de las reglas.

En el método donde se dispara el evento que reacciona en el momento de que el usuario presione el botón “aceptar” está presente el resto del código de la API JESS en la que permite agregar a la maquina la información del id de la persona y armar los respectivos hechos así como la ejecución de las reglas.

```
private void jbAgregarActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        int des = JOptionPane.showConfirmDialog(this, "¿Esta seguro?");
        if (des == JOptionPane.OK_OPTION) {
            try {
                interfaz.datos.Personas p = new Personas();
                p.setId(Long.parseLong(this.jtfCodigo.getText()));
                p.setApellido(this.jtfApellidos.getText());
                p.setEdad(Integer.parseInt(jtfEdad.getText()));
                p.setNombre(this.jtfNombre.getText());
```

```
            p.setSexo(this.jtfSexo.getText().toCharArray()[0]);
            cons.ejecutar(p.agregar());
            Responder(p.getId(), p.getEdad());
        } catch (JessException ex) {
         Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

el método Responder contiene la formación de los hechos y su introducción dentro de la máquina de reglas.

```
public void Responder(long id, int e) {
    try {
        maquina.store("id", id);
        Fact hecho = new Fact("Persona", maquina);
        hecho.setSlotValue("ID", new LongValue(id));
        hecho.setSlotValue("edad", new Value(e, RU.INTEGER));
        maquina.assertFact(hecho);
        maquina.run();
    } catch (JessException ex) {
        System.out.println(ex.getMessage());
        ex.printStackTrace();
    }
  }
```

}

En el método anteriormente descrito se muestra un conjunto de métodos que presenta la clase Rete para la realización de un hecho así como sus respectivos descriptores y la introducción en la maquina de reglas.

la clase Fact permite la realizar instanciaciones de hecho, que pueden ser agregados a la maquina de reglas a través del método assertFact de la clase Rete.

Para definir los valores de los respectivos descriptores de un hecho, la clase Fact tiene un metodo llamado setSlotValue.

Para hacer que la máquina de reglas evalué las reglas y los hechos presentes en la memoria de trabajo. La clase Rete presenta un método llamado Run que permite evaluar todas los hecho con el fin de ejecutar todas las reglas presente hasta que todas las reglas sean ejecutadas.

Ahora dentro del archivo mayor.clp reposan la definiciones de hechos y reglas de JESS que cuyo consecuentes presentes en las reglas permiten la manipulación de los objetos que presentan conectividad con la base de datos de MySQL.

```
;Se importa el paquete interfaz.datos
(import interfaz.datos.*)
; se definen los objetos que referenciarán las clases de java que presentan conectividad con la base de datos
(defglobal ?*per* =nill)
```

```
(defglobal ?*id* =nill)

(defglobal ?*adm* =nill)

; se instancia la clase ConsultorBD (clase responsable del acceso a la base de datos, utiliza en componente JDBC, esta es asignada a la variable global adm

(bind ?*adm* (new interfaz.datos.ConsultorBD root adaluz02))

; definición de la plantilla persona

(deftemplate Persona

        "una plantilla que guardara la edad de una persona"

        (slot ID (type LONG))

      (slot edad (type INTEGER))

)

;definición de la regla mayor de edad

(defrule Mayor_de_Edad

        "informa si una persona es mayor de edad"

        (Persona {edad >= 18})

        =>

        (assert ( Respuesta "la persona es mayor de edad"))

; captura del identificador de la persona enviada desde la API de JESS para el lenguaje JAVA, ver el método Responder presente en la clase Principal

      (bind ?*id* (fetch id))

      (printout t ?*id*)

; se instancia la clase Mayor cuya meta es representar la información que va a recibir la tabla con igual nombre de la base de dato (Clase Entidad), esta es asignada a la variable global per

      (bind ?*per* (new interfaz.datos.Mayor))

; asignación del id al objeto referenciado por per.

      (?*per* setId ?*id*)
```

; llamada a la ejecución de la sentencia SQL entregada por la clase menor para que el consultor de la base de datos ejecute la sentencias SQL a la base de datos.

```
        (call ?*adm* ejecutar (call ?*per* agregar))
)
;definición de la regla mayor de edad
(defrule Menor_de_Edad
          "informa si una persona es menor de edad"
          (Persona {edad > 0 && edad < 18 })
          =>
          (assert ( Respuesta "la persona es menor de edad"))
        (bind ?*id* (fetch id))
        (printout t ?*id*)
        (bind ?*per* (new interfaz.datos.Menor))
        (?*per* setId ?*id*)
        (call ?*adm* ejecutar (call ?*per* agregar))
)
```

Una salida de un escenario evaluado por la maquina de reglas es la siguiente:

Escenario: la persona con id=2 llamada Adaluz Galván Gonzáles de sexo femenino y con 6 años de edad.

La salida revelada por el experto para este escenario es la siguiente:

MAIN::Mayor_de_Edad: +1+1+1+t

MAIN::Menor_de_Edad: =1+1+1+1+t

==> f-0 (MAIN::Persona (ID 2) (edad 6))

==> Activation: MAIN::Menor_de_Edad : f-0

FIRE 1 MAIN::Menor_de_Edad f-0

==> f-1 (MAIN::Respuesta "la persona es menor de edad")

<Java-Object:java.lang.Long>

7.5 Otras aplicaciones

En el marco de esta investigación, además de las anteriores aplicaciones, se coleccionaron un conjunto de programas basados en las investigaciones o apuntes desarrollados por otros grupos de investigación o instituciones alrededor del mundo, la cuales se citan a continuación:

El problema de las garrafas[59]: en la cual expresa el algoritmo de búsqueda en profundidad.

Varios ejemplos de JESS[60] desarrolladas a modo de ejemplo por la empresa desarrolladora de JESS Sandia National labs[61]. Estos archivos contienen una serie de bloque de códigos utilizado para probar diversas características de la maquina de inferencia. Algunos de ellos son programas clásicos (zebra.clp,

[59]PÉREZ Eva, PALACIOS Ernesto. CLIPS. Universidad de Don Bosco 2004. [recurso en línea] <http://www.cruzagr3.com/sistemasexpertos2004/files/Investigacion/grupo08.pdf>[con acceso 2005-05-04] p 26 -33

[60] Jess. 2006. Versión 7.0p1. Livermore: Sandia National Laboratories [paquete para java] disponible en linea <http://www.jessrules.com/jess/download.shtml> [con acceso 2009-05-04]. Bajo licencia académica.

[61] Pagina web del instituto[en línea] <http://www.sandia.gov/> [con acceso 2009-05-04]

wordgame.clp, fullmab.clp). Algunos de ellos son interactivos (sticks.clp, animal.clp). estos ejemplos dan una idea de lo que este sistema basado en reglas puede hacer.

Para ejecutar los programas animal.clp y animal.dat el archivo de datos debe estar en la carpeta donde reposan dichos programas

browse.clp es un ejemplo de un pequeño navegador Web.

console.html es una página Web que incluye el applet de la clase genérica Jess (jess.ConsoleApplet)

backchain.clp es un ejemplo muy simple de la valoración encadenamiento hacia atrás.

Fullmab.clp soluciona el problema de el mono y las bananas.

Dilemma.clp soluciona el clásico problema del granjero, el lobo y la oveja.

Sticks.clp nos muestra el juego de los palillos, pierde en el saque el ultimo palillo de la pila.

El grupo de Sistemas de Información Avanzados[62] (IAAA) de la Universidad de Zaragoza a través del curso INGENIERIA DE LOS SISTEMAS BASADOS EN EL CONOCIMIENTO[63], impartida por el docente José Ángel Banares[64], y *David*

[62]Grupo de Sistemas de Información Avanzados. Universidad de Zaragoza. Zaragoza - España [en línea] <http://iaaa.cps.unizar.es/showContent.do?cid=presentacion.ES>[con acceso 2009-05-04]

[63] Pagina del curso <http://iaaa.cps.unizar.es/docencia/ISBC.html> [con acceso 2009-05-04]

[64] Pagina web del docente <http://iaaa.cps.unizar.es/personal/jangelb/index.htm> [en línea 2009-05-04]

Portoles dieron solución al problema de los ascensores[65] propuesto por Deitel & Deitel[66], uno de los avances mas significativos fue el desarrollo de una librería que actúa como extensión de la misma máquina de inferencia.

A partir de la versión 5.0 JESS integra los conceptos de lógica difusa, el paquete *FuzzyJ ToolKit*[67] desarrollado por la organización *National Research Council of Canada's*[68], unido con la maquina de inferencia JESS permite desarrollar aplicaciones en la cual la toma de decisión dependa mucho del grado de certeza que el experto evalúe de un fenómeno dentro de un conjunto borroso de posiblidades. Un ejemplo de esto es un sistema controlador de temperatura de una ducha (Fuzzy Shower demo[69]).

[65] David Portoles. 2001-2002. Ascensores [programa informático en línea] Zaragoza: Universidad de Zaragoza. [en línea] <http://webdiis.unizar.es/asignaturas/ISBC/isbc/Ascensores/ascensor.zip> [con acceso 2009-05-04]

[66] DEITEL & DEITEL. Java: how to program. 3 edicion. Prentice hall, NewJersey. 1999 p 1256 -1270.

[67] FuzzyJ Toolkit. 2006. Versión 1.10a. Ottawa: National Research Council of Canada's [paquete para java con acceso] disponible en National Research Council of Canada's <http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJToolkit2.html> [con acceso 2009-05-04]. bajo licencia académica.

[68] Pagina web del Instituto <http://www.nrc-cnrc.gc.ca/> [con acceso 2009-05-04]

[69] Bob Orchard. 2006. FuzzyJess Shower Demo. Disponible en linea <http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyShowerJess.html> [con acceso 2009-05-04]

8 CONCLUSIONES

Tras analizar la herramienta para el diseño de sistemas experto JESS así como sus diferentes fortalezas tales como la implententación del algoritmo de reconocimiento de patrones de Rete, su integración en aplicaciones desarrolladas en java, su API para desarrollos en JAVA, la manipulación de objetos desarrollados en JAVA, la compatibilidad con el lenguaje CLIPS ya por ser un súper conjunto de este último, la integrabilidad que tiene para varios Entornos Integrados de Desarrollo tales como NetBeans o Eclipse incluyendo la manipulación de Objetos con conectividad a Base de Datos por ejemplos los objetos JDBC, soporte para lógica difusa, así como para realizar valoraciones a través del encadenamiento hacia delate o hacia atrás, integración con xml, entre otras se pueden obtener las siguientes conclusiones:

1. JESS es una herramienta que permite desarrollar sistemas expertos desde las demostraciones formales de la lógica proposicional hasta juegos en donde se apliquen los diferentes patrones de búsqueda, desde la anchura hasta la valoración por encadenamiento hacia delante o hacia atrás.

2. La integración con el lenguaje de programación JAVA, permite que dichos expertos desarrollados en JESS puedan ser ejecutados desde cualquier plataforma e incluso desde la Web, la única condición que se debe cumplir el que su navegador soporte el Plugins de java.

3. Como JESS permite desarrollar sistemas que ayuden en la toma de decisiones de agentes inteligentes, entonces este abre una gran posibilidad de desarrollar sistemas de apoyo a la toma de decisiones más complejos como por ejemplo, sistemas de control del tráfico aéreo, evaluador de sistemas críticos, en donde la toma de decisión de un fenómeno bajo estudio es fundamental, entre otros de relevante importancia.

4. Con base a la investigación realizada sobre esta herramienta se puede tomar esta investigación como punto de partida, desde el énfasis pedagógico, para el desarrollo de una guía de referencia para cualquier curso de sistemas basados en conocimiento.

5. JESS es una herramienta muy recomendable para el diseño de sistemas expertos, especialmente para cualquier proyecto donde dichas aplicaciones tenga un comportamiento humano.

6. Con JESS se pueden desarrollar herramientas que su objeto de estudio sea la pedagogía, desde juegos hasta verdaderos tutoriales o sistemas de ayuda tales como sistemas guías en donde orienten al estudiante acerca de

un tema o sesión específico. Para la muestra fue el tutor inteligente desarrollado por Reyes y de la Ossa[70] en donde éste actúa como guía y evaluador diseñado para la asignatura de métodos numéricos en la Universidad Tecnológica de Bolívar en donde el sistema desarrollado como máquina de inferencia para apoyar la toma de decisiones de dicho experto fue de gran importancia dentro del funcionamiento de éste.

7. La fortaleza de esta herramienta puede llevar a crear poderosos sistemas de diagnóstico o detección de problemas, muy útiles en el campo de la medicina.

8. Esta herramienta y su integración con objetos productos del lenguaje de programación JAVA puede abrir una gran brecha de nuevos desarrollos la unión de JESS con la herramienta JADE, la cual nos ofrece una plataforma para la creación de sistemas multi agentes, lo cual expande sobre manera el uso y fortaleza de esta herramienta con miras la obtención de sistemas basados en conocimiento de gran impacto en lo que respecta a su funcionalidad.

9. JESS por su diseño, permite que sus creaciones sean fácilmente integrables a cualquier aula virtual de aprendizaje.

[70]REYES leonardo, DE LA OSSA Andrés Diseño e implementación de un sistema tutor inteligente basado en web aplicado a la resolución de ecuaciones algebraicas no lineales utilizando métodos numéricos para ingeniería. Tesis de Pregrado. Universidad Tecnológica de Bolívar Cartagena de Indias. 2008

Podemos enumerar un sin número de ventajas que tiene JESS, pero su importancia va mas allá de lo útil, y de lo eficaz. JESS está marcando un hito en lo que respecta al desarrollo de sistemas experto y deja que su funcionalidad se apliquen a cualquier sistema en cualquier lugar, utilizando la tecnología de vanguardia, en la cual en sus diseños toma lo mejor de los paradigmas y modelos de la ingeniería de conocimiento como la ingeniería del software, siendo a su vez el legado más importante.

REFERENCIAS

Páginas Web

- PROGRAMACIÓN GRAFICA. Método de Newton-Raphson [En línea]. <http://www.geocities.com/valcoey/newton_raphson.html> [con acceso 21 de Abril de 2008]

- HERRERA Mónica, PRIETO Jacqueline, LOPEZ Miguel, MARTINEZ Enrique, DE LAS CASAS flora. “Agentes inteligentes” [en Línea] <http://www.depi.itch.edu.mx/apacheco/expo/html/ai12/> [con acceso junio 18 de 2008]

- SÁNCHEZ Antonio. Aplicación de los Sistemas Expertos en Contabilidad. Departamento de Contabilidad, Universidad de Valencia. Disponible en línea <http://ciberconta.unizar.es/Biblioteca/0002/Sanchez95.html> [con acceso el 12 -12-2008]

- PACHECO Alberto. Metodología de Diseño de Sistemas Expertos. Instituto Tecnológico de Chihuahua. Disponible en línea <http://www.depi.itch.edu.mx/apacheco/ai/metodolo.htm > [con acceso 2009-05-04]

- CLIPS Reference Manual Volume II Advanced Programming Guide. 2008. disponible en línea <http://clipsrules.sourceforge.net/documentation/v630/apg.htm> [con acceso 2009-05-04]

- What is CLIPS? Disponible en línea <http://clipsrules.sourceforge.net/WhatIsCLIPS.html> [con acceso 2009-04-29]

- About the Java Technology [en línea] <http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html> [con acceso 2009/04/20]

- The Java Language Environment. 2006 [en línea] <http://java.sun.com/docs/white/langenv/> [con acceso 2009/04/20]

- Java SE HotSpot at a Glance [en línea] <http://java.sun.com/javase/technologies/hotspot/index.jsp> [con acceso 2009/04/20]

- What Can Java Technology Do? [en línea] <http://java.sun.com/docs/books/tutorial/getStarted/intro/cando.html> [con acceso 2009/04/20]

- JDK TM 6 Documentation [en línea] <http://java.sun.com/javase/6/docs/index.html> [con acceso 2009/04/20]

- How Will Java Technology Change My Life? [en línea] <http://java.sun.com/docs/books/tutorial/getStarted/intro/changemylife.html> [con acceso 2009/04/20]

- JESS ®, the Rule Engine for the JavaTM Platform [en línea] <http://www.jessrules.com/ > [con acceso 2009-05-03]

- Grupo de Sistemas de Información Avanzados. Universidad de Zaragoza. Zaragoza - España [en línea] <http://iaaa.cps.unizar.es/showContent.do?cid=presentacion.ES> [con acceso 2009-05-04]

Apuntes de Cursos y transparencias

- RODRÍGUEZ Camino, DÍAZ Irene. Apuntes del tema SISTEMAS BASADOS EN REGLAS del CURSO 2008-2009 de INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL. E.U.I.T. INFORMÁTICA DE GIJÓN [Transparencias en línea] Disponibles desde Internet en: <http://lear.inforg.uniovi.es/ia/Archivos/Apuntes%20y%20t/ReglasTeoria2008.pdf> [con acceso el 24-04-2009].

- DURKIN John. Expert Systems: Design and Development. Cáp. 8 traducido por Lic. Arias Tapia Jhonny para el curso 2010029 - Sistemas Expertos. Universidad Mayor de San simón. Recurso Disponible con acceso <http://www.cs.umss.edu.bo/doc/material/mat_gral_10/cap6_8_10.zip> [con acceso 27-04-2009]

- PÉREZ Eva, PALACIOS Ernesto. CLIPS. Universidad de Don Bosco 2004. [recurso en línea] <http://www.cruzagr3.com/sistemasexpertos2004/files/Investigacion/grupo08.pdf> [con acceso 2005-05-04]

Aplicaciones

- MySQL Community Server. 2008. Versión 5.0 Uppsala – Suecia: MySQL AB [Programa informático] disponible en línea <http://dev.mysql.com/downloads/mysql/5.0.html#downloads> [con acceso 2009-04-05]. bajo licencia GPL

- Jess. 2006. Versión 7.0p1. Livermore: Sandia National Laboratories [paquete para java] disponible en línea <http://www.jessrules.com/jess/download.shtml> [con acceso 2009-05-04]. Bajo licencia académica.

- David Portoles. 2001-2002. Ascensores [programa informático en línea] Zaragoza: Universidad de Zaragoza. [en línea] <http://webdiis.unizar.es/asignaturas/ISBC/isbc/Ascensores/ascensor.zip> [con acceso 2009-05-04]

- FuzzyJ Toolkit. 2006. Versión 1.10a. Ottawa: National Research Council of Canada's [paquete para java con acceso] disponible en National Research Council of Canada's <http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJToolkit2.html> [con acceso 2009-05-04]. bajo licencia académica.

- Bob Orchard. 2006. FuzzyJess Shower Demo. Disponible en línea <http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyShowerJess.html> [con acceso 2009-05-04]

- NetBeans IDE. 2008. Versión 6.5. Santa Clara - EEUU: Sun Microsystems [Programa Informático] disponible en línea <http://www.netbeans.org/> [con acceso 2009-05-04] bajo licencia CDDL

- Eclipse IDE. 2008. Versión 3.4.1. Eclipse Foundation [Programa Informático] disponible en línea <http://www.eclipse.org/> [con acceso 2009-05-04] bajo Licencia Pública de Eclipse

- Dia. 2007. Versión 0.96.1. Free Software Foundation [Programa Informático] disponible en línea <http://live.gnome.org/Dia> [con acceso 2009-05-04] bajo Licencia Pública GNU

BIBLIOGRAFIA

- RUSSELL Stuart, NORVIG Peter. Inteligencia Artificial un enfoque moderno. ISBN: 0131038052. México: Prentice Hall Hispano Americana, 1996.

- GIARRATANO Joseph, RILEY Gary. Sistemas expertos principios y programación. ISBN: 0534950531. Thomson Learning. 2000.

- CASTILLO Enrique, GUTIÉRREZ José Manuel, HADI Ali. Sistemas Expertos y Modelos de Redes Probabilísticas. Academia Española de Ingeniería. Disponible en línea <http://personales.unican.es/gutierjm/papers/BookCGH.pdf> [con acceso el 15-12-2008].

- NEBENDAHL Dieter. Sistemas Expertos: experiencia de la practica. ISBN: 380094104X Barcelona, ES. Editorial: Marcombo, 1988.

- NEBENDAHL Dieter. Sistemas Expertos: Introducción a la técnica y aplicación. ISBN: 0471919667. Barcelona, ES. Editorial: Marcombo, 1988.

- ROLSTON David. Principios de Inteligencia Artificial y sistemas expertos. México: McGraw-hill interamericana, 1992.

- REYES Leonardo, DE LA OSSA Andrés Diseño e implementación de un sistema tutor inteligente basado en Web aplicado a la resolución de ecuaciones algebraicas no lineales utilizando métodos numéricos para ingeniería. Tesis de Pregrado. Universidad Tecnológica de Bolívar Cartagena de Indias. 2008
- DEITEL H. M. & DEITEL P. J. Java: how to program. 3 edicion. Prentice hall, ISBN: 0130125075. NewJersey. 1999.

ANEXOS

JESS LICENSE AGREEMENT

Jess, the Rule Engine for the Java Platform is Copyright (C) 2006 by the Sandia Corporation.

Jess software, owned by Sandia National Laboratories, will be made available upon request at no cost to U.S. Federal Government Agencies for their own internal use. Sandia will also provide Jess upon request to Universities, Academic Institutions, and other U.S. National Laboratories, for their own internal research and development, through a no cost, restricted R&D license. Any internal or commercial use of Jess requires a commercial license that can be negotiated as a running royalty or a fully paid up-front fee.

Our commercial license will grant your company, the LICENSEE, a nontransferable, nonexclusive right to use Jess Software to create derivative works by embedding Jess into your product(s) and to copy and distribute Jess software as embedded into your Product(s).

Note: Jess is not licensed under the GPL, the LPGL, the BSD license, or any other free software or open source license. Redistribution of the Jess source code under any free software or open source license is prohibited under this agreement.

For appropriate licenses, please contact Craig Smith (Telephone 925/294-3358); email: casmith@sandia.gov.

FUZZYJ TOOLKIT - END USER LICENSE AGREEMENT

IMPORTANT, PLEASE READ CAREFULLY:

This end user license agreement (the “Agreement”) is a legal agreement between you (the "User") and the National Research Council of Canada ("NRC") for the NRC software product FuzzyJ Toolkit, version 1.6 (the “Software”) as described in Exhibit A to this Agreement.

By installing, copying or otherwise using the Software, you agree to be bound by the terms and conditions of this Agreement. If you do not agree to the terms and conditions of this Agreement, do not install, copy or otherwise use the Software.

1.0 GRANT OF LIMITED LICENCE

1.1 In consideration of the User’s acceptance of the terms and conditions of this Agreement, NRC grants to the User, and User accepts, a free, non-exclusive, and limited license to use the Software solely for the User’s personal and non-commercial use, and only in the manner described under the heading "USAGE AND RESTRICTIONS". This license granted hereunder is NOT assignable, sub-licensable, or otherwise transferable by the User to any other party either directly or by operation of law.

2.0 USAGE AND RESTRICTIONS

2.1 The User may install the Software onto a single computer. One computer, in this agreement, means hardware that is controlled by a single operating system and which functions as a single computing entity. Networks, clusters, and other like configurations are

considered to be composed of multiple separate computers. However, the User may move the Software from a first computer to a second if it is completely removed from the first.

2.2 The User may only use the Software solely for his or her personal educational or non-commercial research purposes.

2.3 The User may make a single backup copy of the Software solely for backup or archival purposes.

2.4 The User must not attempt to disassemble, decompile or reverse engineer the Software in order to obtain its source code or make any translation, adaptation, arrangement or any other alteration of the Software.

2.5 NRC represents that the Software is proprietary to NRC, and that such Software comprises confidential information of NRC. Use of the Software is personal to the User and the User shall not give access to, or disclose, either in whole or in part to any other party without the prior written consent of NRC.

2.6 The User shall ensure that any copyright notice, trademark or other proprietary right notice placed by NRC on the Software remains in evidence and is reproduced on any copies of the Software, whether in whole or in part.

3.0 BENEFIT TO THE NATIONAL RESEARCH COUNCIL

3.1 All publications arising from use of the Software shall duly acknowledge such use in accordance with normal practices followed in scientific research publications.

3.2 Users are requested to inform NRC of noteworthy uses, suggestions for improvement and the general usefulness of the Software.

4.0 TITLE

4.1 The User acknowledges and agrees that all proprietary interest, right, title, copyright, or other intellectual property right in the Software and all copies thereof remain at all times with NRC.

5.0 NO MAINTENANCE OR SUPPORT

5.1 By downloading this Software, the user hereby understands and agrees that NRC shall be under no obligation whatsoever to provide maintenance or support for the Software, or to notify the User of bug fixes, patches, or upgrades to the Software (if any). If, in its sole discretion, NRC makes a Software bug fix, patch or upgrade available to the User and NRC does not separately enter into a written license agreement with the User relating to such bug fix, patch or upgrade, then it shall be deemed incorporated into the Software and subject to this Agreement.

6.0 LIMITED WARRANTY AND REMEDIES

6.1 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF: MERCHANTABILITY, FITNESS OR USEFULNESS FOR A PARTICULAR PURPOSE, THAT IT IS ERROR-FREE OR THAT ANY ERRORS WILL BE CORRECTED, AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL NRC BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWSOEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN AN ACTION OF CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OF THE SOFTWARE, EVEN IF NRC IS ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6.2 THE USER SHALL MAKE NO CLAIMS OF ANY KIND WHATSOEVER UPON OR AGAINST NRC, EITHER ON ITS OWN ACCOUNT OR ON BEHALF OF ANY THIRD PARTY, ARISING DIRECTLY OR INDIRECTLY OUT OF ITS USE OF THE SOFTWARE.

7.0 TERMINATION PROVISIONS

7.1 The User may terminate this license at any time, by destroying the Software, the Documentation and any back-up or archival copy of the Software.

7.2 This License shall terminate automatically upon any breach of any term of this Agreement by the User. Upon such termination the User shall destroy all copies of the Software in its possession and provide NRC with written certification of such destruction.

7.3 NRC and/or its licensors may pursue all available legal remedies to enforce this Agreement. The User agrees that NRC's licensors referenced in the Software are third-party beneficiaries of this Agreement, and may enforce this Agreement as it relates to their intellectual property.

7.4 The obligations of the User and NRC under Sections 4 (“TITLE”), 6 ("LIMITED WARRANTY AND REMEDIES"), 7 ("TERMINATION PROVISIONS") and 8 ("GENERAL PROVISIONS") hereof will survive termination of this Agreement and will continue in full force and effect thereafter.

8.0 GENERAL PROVISIONS

8.1 This Agreement constitutes the entire agreement between the User and NRC concerning the Software and it supersedes any prior agreement or representation.

8.2 This Agreement shall be governed by the laws of the Province of Ontario, and the federal laws of Canada applicable therein, without regard to its rules on the conflict of laws. The parties hereby irrevocably attorn to the non-exclusive jurisdiction of the courts of the Province of Ontario.

8.3 If any provision of this Agreement is determined to be invalid or unenforceable in whole or in part by any court of competent jurisdiction, such provision or part thereof shall be

deemed severed herefrom and the remaining part of such provision and all other provisions hereof shall continue in full force and effect.

8.4 The section headings in this Agreement are solely for convenience and will not be considered in its interpretation

8.5 The parties hereby confirm that it is their wish that this Agreement, as well as all other documents relating hereto, including all notices, have been and shall be drawn up in the English language.

8.6 If you have any questions concerning this license, or if you wish to acquire a commercial license to this Software, please contact George Forester, george.forester@nrc-cnrc.gc.ca, or write to: NRC Institute for Information Technology (NRC-IIT), Business Development Office, 1200 Montreal Road, Ottawa, Ontario, K1A 0R6, Canada.

Other Copyright Notices

* This Software includes an open source code called CUP Parser Generator Copyright Notice, License, and Disclaimer Copyright 1996-1999 by Scott Hudson, Frank Flannery, C. Scott Ananian. Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both the copyright notice and this permission notice and warranty disclaimer appear in supporting documentation, and that the names of the authors or their employers not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

The authors and their employers disclaim all warranties with regard to this software, including all implied warranties of merchantability and fitness. In no event shall the authors or their employers be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortuous action, arising out of or in connection with the use or performance of this software