Desarrollo de un Centro de Ayuda Inteligente mediante el uso de Tecnologías de Internet

Sonia Lilia Mestizo Gutiérrez
Dr. Alejandro Guerra Hernández
Dr. Ramón Parra Loera
Maestría en Inteligencia Artificial
Sebastián Camacho 5, Xalapa 91000, Veracruz, México

Contenido

1 Introducción **3**

2 Antecedentes **6**
2.1 Trabajo Relacionado 6
2.2 Sistemas Expertos 8
2.2.1 Arquitectura Clásica de un Sistema Experto 19
2.2.2 Sistemas Expertos y los Centros de Ayuda 23
2.3 Entornos de Desarrollo de Sistemas Expertos 25
2.4 Jess (Java Expert System Shell) 29
2.4.1 El Algoritmo *RETE* de Jess 30
2.4.2 Jess en el Web 36
2.5 Protégé 39
2.6 JessTab 40
2.7 Tecnología de Agentes 41

3 Descripción del Problema **43**
3.1 Sistema de Educación Distribuida "EMINUS" 43
3.1.1 Características tecnológicas 45
3.1.2 Características de comunicación y colaboración 45
3.1.3 Características pedagógicas 46
3.1.4 Definición del Problema 47

4 Desarrollo del Sistema Experto **49**
4.1 Introducción 49
4.2 Metodología de desarrollo 50
4.2.1 Funcionalidades del sistema experto 50
4.3 Adquisición del conocimiento 51
4.3.1 Uso de Protégé 55

4.4 Desarrollo del sistema de ayuda 56
4.4.1 JessTab 60
4.4.2 JESS 62
4.5 Pruebas 67

5 Conclusiones y Trabajo Futuro **70**
5.1 Conclusiones 70
5.2 Trabajo Futuro 71

A Código del Sistema Experto **74**

Capítulo 1

Introducción

En nuestros días, los Sistemas Expertos constituyen una subárea de la Inteligencia Artificial que ha evolucionado hasta alcanzar plena madurez.

Los Sistemas Expertos proveen un mecanismo robusto y flexible en la solución de problemas. Su uso está proliferando en muchos sectores de nuestra vida tecnológica y social que involucran al proceso de toma de decisiones y la solución de problemas.

El surgimiento de Internet, sin lugar a dudas, ha provocado una revolución tecnológica en el campo de la Informática. El Word Wide Web ofrece un gran potencial para el manejo de servicios basados en información incluyendo los servicios de aplicaciones inteligentes.

Actualmente, las aplicaciones basadas en Inteligencia Artificial se perfilan como uno de los principales dominios de aplicación en Internet. Ahora es mucho más fácil y práctico capturar conocimiento especializado y ponerlo disponible en Internet. La convergencia entre el Web e Inteligencia Artificial está creando una nueva vertiente de desarrollo que promete grandes beneficios.

Los Sistemas Expertos basados en Web son los tradicionales Sistemas Expertos basados principalmente en reglas y razonamiento basado en casos que han sido adaptados para usarse en Internet mediante arquitecturas cliente servidor e interfaces basadas en Web.

Por otra parte, la Tecnología de Agentes, particularmente los Sistemas Multiagentes se perfilan como un importante paradigma en el desarrollo de software y en el desarrollo de aplicaciones a nivel industrial.

Recientemente, el término de Sistemas Multiagentes (MAS) ha llegado a tener un significado más general, ahora se usa para referirse a todos los

tipos de sistemas compuestos de múltiples componentes semi autónomos. La investigación en MAS está interesada en el comportamiento de una colección de agentes autónomos pre-existentes cuya meta es resolver un problema dado.

Los Sistemas Multiagentes se conciben actualmente como sistemas computacionales en los cuales varios agentes semi autónomos (programas) interactúan entre sí ya sea para colaborar en la solución de un conjunto de problemas en la consecución de una serie de objetivos individuales o colectivos. Estos agentes informáticos pueden ser homogéneos o heterogéneos y pueden tener metas comunes o no, pero siempre involucrarán algún grado de comunicación entre ellos. Cada uno de estos agentes indivuales pueden o no tener comunicación directa con seres humanos a través de interfaces.

La próxima generación de la Tecnología Web se centra al desarrollo de portales que contengan conocimiento más que información.

El concepto de *Help Desk (Centro de Ayuda)* en línea está surgiendo con fuerza en el mundo digital, como un medio para compartir recursos y fuentes de información con el objetivo de solucionar problemas. El *Centro de Ayuda* permite resolver cualquier tipo de problema de forma ordenada, rápida y eficiente logrando una mayor productividad. De esta manera, se obtiene una significativa reducción en los costos de soporte.

Los Sistemas Expertos son aplicaciones prácticas de Inteligencia Artificial que contienen la experiencia y el conocimiento de uno o varios expertos en un determinado dominio.

En los *Centros de Ayuda*, un grupo de expertos humanos resuelve los problemas de los usuarios. Los roles de los especialistas están en función de su habilidad para resolver problemas y el grado de dificultad. Actualmente, es crucial, ofrecer el servicio de un *Centro de Ayuda* de alta calidad donde es fundamental la disponibilidad de expertos de alto nivel.

Con la ayuda de un sistema experto basado en web, se pueden resolver problemas que requieren un "conocimiento especializado". De esta forma, se incrementa el número de personas con acceso a un conocimiento experto.

En la solución de un problema, se requiere de un experto. Por ejemplo, imagine que su impresora láser no trabaja. El usuario puede efectuar una llamada para reportar su problema con un experto humano o consultar un sitio web para que le brinde soporte; en este caso, el contenido de dicho sitio es un listado de problemas comunes y soluciones. Por otra parte, puede consultar a un *Centro de Ayuda Inteligente* que inicia preguntando: ¿Cuál es la naturaleza de su falla?, ¿Cuál es el sistema operativo que está usando?, etc. como si realmente estuviera interactuando con el experto humano con

la ventaja de que estará disponible todo el tiempo, a diferencia del experto humano.

El objetivo central de este trabajo consiste en desarrollar un *Sistema Experto basado en Web* que brinde soporte técnico en línea, a los usuarios del Sistema de Educación Distribuida (EMINUS) de la Universidad Veracruzana. De esta manera, el servicio estará disponible las 24 horas del día, brindando atención a los problemas oportunamente y con calidad técnica.

Es importante destacar que la infraestructura del Sistema Experto propuesto, puede utilizarse para implementar cualquier centro de ayuda. Por ello, será utilizado para crear el *Centro de Ayuda Inteligente* del Portal de la Dirección General de Tecnología de Información de nuestra Universidad. De esta manera, se ratifica el uso de tecnología innovadora en nuestra Universidad.

Capítulo 2

Antecedentes

2.1 Trabajo Relacionado

Los *Centros de Ayuda* son sistemas que recuperan oportunamente, toda la información requerida para asistir a un usuario inexperto o un usuario avanzado en la solución de un determinado problema.

En un Centro de Ayuda es de vital importancia la actualización del conocimiento en dominios donde la información cambia rápidamente.

En los servicios convencionales de los Centros de Ayuda, grupos de expertos humanos con diferentes niveles de conocimiento y experiencia tratan de resolver los problemas de los usuarios. Los roles de los expertos, se determinan de acuerdo a su habilidad en la solución de problemas y el grado de dificultad del problema. En consecuencia, para brindar un servicio de alta calidad, es crucial la disponibilidad de expertos de alto nivel. Sin embargo, el número de expertos es limitado, por lo que se incrementa la demanda de sistemas de Centros de Ayuda de alta calidad.

En el contexto de los Centros de Ayuda, se utilizan varios paradigmas para brindar la solución a los problemas de los usuarios, de los cuales destacan:

1. *Recuperación de información y diagnóstico de problemas.* Consiste en la búsqueda de nueva información. *¿Qué es el WWW?* es un típico ejemplo de este tipo de esta búsqueda. También se refiere a la búsqueda de la solución de un determinado problema, como por ejemplo, un usuario puede solicitar al Centro de Ayuda la solución de un problema dado, por ejemplo: *Mi computadora no trabaja.*

2. *Sistemas Expertos.* El enfoque de los Sistemas Expertos constituye una

una viable solución, el cual se fundamentan en el área de diagnóstico basado en conocimiento.

Recuperación de información

La mayoría de los estudios de recuperación de información se centran en *cómo* encontrar información relevante de un amplio texto base. Un simple enfoque, consiste en recopilar documentos relacionados y proporcionar una máquina de búsqueda para dicha colección. Un buen número de los trabajos de investigación en esta área fallan en:

1. La representación del texto.
2. La representación de las preguntas del usuario.
3. El método de recuperación.

La representación del texto es uno de los clásicos temas en los estudios de recuperación de la información. El enfoque más simple extrae todas las palabras de los documentos excepto los pronombres y artículos y hace uso de métodos estadísticos para la encontrar presencia de palabras. El término frecuencia también se usa para proporcionar información adicional. [53].

La representación de las consultas del usuario se estudia para capturar apropiadamente dichas consultas. El enfoque más simple consiste en capturar las solicitudes mediante combinaciones de palabras clave. El tratamiento del lenguaje natural [16] y varias técnicas sofisticadas de interacción brindan mejores interfaces.

La función de recuperación [54] selecciona y clasifica los documentos. El método de clasificación es particularmente importante cuando se seleccionan una gran cantidad de documentos. Dado que la simple lógica booleana no brinda la clasificación, entonces se usan varios métodos estadísticos tales como el vecino k-más cercano que proporciona la clasificación de los documentos seleccionados.

Los métodos de recuperación de información son prácticos para construir típicos sistemas de *Help Desk*, cuya tarea primordial consiste en la búsqueda de nueva información. Sin embargo, este tipo de sistemas asume que el usuario puede especificar las palabras clave apropiadas para la búsqueda de documentos relacionados. Si el usuario no tiene la habilidad suficiente

para proveer las apropiadas palabras clave, el sistema puede fallar en la recuperación de los documentos relevantes o puede encontrar muchos documentos irrelevantes.

Sistemas Expertos

Para el desarrollo de *Sistemas Expertos* en el dominio de los Centros de Ayuda, se han utilizado los sistemas basados en reglas y sistemas basados en casos. El formalismo de representación del conocimiento basado en reglas es el más popular en la comunidad de desarrollo de sistemas expertos. El enfoque del razonamiento basado en casos (CBR)se ha usado frecuentemente para la construcción de los Centros de Ayuda [12], [13], [40], [55], [59]. Sin embargo, la mayoría de estos sistemas requieren de un mayor esfuerzo para mantener el caso base. Existen trabajos que usan el método *MCRDR - Multiple Classification Ripple Down Rules-*, para reducir el costo del mantenimiento del caso base y aumentar la velocidad del proceso de mantenimiento. En dicho método, el experto puede desarrollar y mantener el caso base sin la ayuda de los ingenieros del conocimiento [39], [15].

2.2 Sistemas Expertos

Los *Sistemas Expertos* son programas cuyo objetivo primordial es el de ejecutar tareas y/o resolver problemas en un dominio específico, mostrando un alto grado de desempeño, sólo comparable con el de los mejores expertos humanos. Razonan con reglas, funcionan con datos vagos e imprecisos, explican el porqué efectúan sus preguntas cuando están resolviendo un problema y justifican sus conclusiones [43].

Revisando un poco su historia, los primeros sistemas expertos aparecen a mediados de los 60's en la *Etapa de Invención* (1965-70) donde se presentaron básicamente, dos líneas de investigación:

1. Los métodos de búsqueda heurísticos.

2. Los métodos de deducción automática.

En la *Etapa de Prototipos* ubicada en el período de los años 1970-1977, se desarrollaron sistemas expertos que legaron formalismos de representación del conocimiento y de inferencia al desarrollo de otros sistemas expertos. Los

sistemas expertos más importantes de este período fueron *Mycin* y *Prospector*. Posteriormente, en la *Etapa de Experimentación* (1977-1981) comienzan a aparecer los *entornos de desarrollo* destacándose *Emycin* que está basado en *Mycin*.

A partir de 1981 a la fecha, en la *Etapa de Industrialización*, los sistemas expertos se difunden ampliamentem debido en gran medida al surgimiento de los lenguajes especializados y a los entornos de desarrollo. Por otra parte, surgen empresas dedicadas al desarrollo y comercialización de los sistemas expertos. Algunos de los sistemas expertos representativos de esta etapa son:

- *American Express*. Sistema para la autorización de tarjetas de crédito.
- *Atrex*. Creado por la compañía Toyota para el diagnóstico de fallas.
- *R1*. Desarrollado por la empresa Digital Equipment Corporation para configurar sistemas PDP y VAX [49].

Actualmente, las metodologías para el desarrollo de Sistemas Expertos se clasifican en once categorías [58]:

1. Sistemas basados en reglas y sus aplicaciones. Un Sistema Experto basado en reglas contiene la información del experto humano representada en forma de las clásicas reglas (IF-THEN). Las reglas se usan para ejecutar operaciones en los datos, efectuar inferencias y obtener una conclusión. Dichas inferencias brindan esencialmente una metodología de razonamiento sobre el conocimiento almacenado en la base de conocimiento y formulan conclusiones.

Las aplicaciones de los sistemas basados en reglas abarcan: análisis de transición de estados, tratamiento psiquiátrico, planeación, sistemas de consultoría, enseñanza, planeación de procesos automovilísticos, desarrollo de sistemas, validación y verificación de conocimiento, interpretación de histogramas de DNA, mantenimiento de bases de conocimiento, estrategias de programación, estimaciones de fraudes, adquisición de conocimiento, diagnóstico de fallas de sistemas de comunicación, nanotecnología, bioquímica, diagnóstico de fallas probabilísticas, agricultura, apicultura, diagnóstico agrícola, geociencia, control de sensores, sistemas tutoriales inteligentes, entre otras.

2. Sistemas basados en conocimiento. La definición más común de los Sistemas basados en conocimiento gira en torno a los humanos. Resalta

el hecho de que un Sistema basado en conocimiento tiene sus orígenes en el campo de la Inteligencia Artificial, la cual, intenta entender y representar el conocimiento humano en los sistemas computacionales. Usualmente se distinguen cuatro principales componentes: la base de conocimiento, la máquina de inferencia, la herramienta de ingeniería del conocimiento y la interfaz del usuario. Por otra parte, el término de Sistemas basados en conocimiento incluye a todas las aplicaciones de tecnología de información organizacional, tales como sistemas expertos, sistemas basados en conocimiento, *groupware* y sistemas de administración de bases de datos.

Algunas de las aplicaciones que han sido implementadas como sistemas basados en conocimiento son: tratamiento médico, planeación personal financiera, análisis de fallas en ingeniería, administración de la producción, soporte de decisiones, administración del conocimiento, representación del conocimiento, diseño del suministro de energía eléctrica, evaluación de construcciones, análisis financiero, administración y gestión de incidentes químicos, segmentación automática de tumores, negocios, predicción del clima, agricultura, diseño de la composición del acero, administración estratégica, protección del medio ambiente, tratamiento de aguas residuales, aprendizaje, control de procesos químicos, planeación de terapias, control de procesos de plantas, diseño de sistemas concurrentes, validación de casos, diseño de chips, planeación de producción de cultivos, robótica y diseño urbano.

3. Redes neuronales. Las redes neuronales son un paradigma de aprendizaje y procesamiento automático inspirado en la funcionamiento del sistema nervioso de los seres vivos. Consiste en simular las propiedades observadas en los sistemas neuronales biológicos a través de modelos matemáticos recreados mediante mecanismos artificiales (como un circuito integrado, un ordenador o un conjunto de válvulas). El objetivo consiste en conseguir que las máquinas den respuestas similares a las del cerebro humano que se caracterizan por su generalización y su robustez. Este concepto se usa para implementar software de simulación para procesamiento paralelo que involucra el procesamiento de elementos interconectados en arquitecturas de redes. Las neuronas artificiales reciben entradas que son análogas a los impulsos electromecánicos que las dendritas de las neuronas biológicas reciben de otras neuronas. La salida de la neurona artificial corresponde a las señales enviadas fuera de la neurona biológica sobre su axón. Estas señales artificiales pueden modificarse, similarmente a los cambios físicos que ocurren en

la sinapsis neural.

Algunas de las aplicaciones que han sido implementadas mediante redes neuronales son: diagnóstico de fallas, optimización de energía, toma de decisiones, sistemas de procesamiento de alarmas, mecanismos de inferencia, sistemas de diagnóstico, aprendizaje automático, control de procesos, proceso enseñanza aprendizaje, diseño de procesos de minería, sistemas robóticos, ajuste de parámetros, tratamiento de aguas residuales, ingeniería, control de procesos de mitigación, diagnóstico de señales acústicas, destilación de petróleo crudo y aplicaciones biomédicas.

4. Sistemas Expertos Difusos (*F*uzzy). Los sistemas expertos basados en lógica difusa tratan con la incertidumbre. La lógica difusa –borrosa o *fuzzy logic*– se caracteriza por querer cuantificar esta incertidumbre. Esta técnica hace uso de la teoría matemática de conjuntos difusos que simulan el proceso normal de razonamiento humano. En la lógica difusa, se usan modelos matemáticos para representar nociones subjetivas, como caliente/tibio/frío, para valores concretos que puedan ser manipulados por las computadoras.

La lógica difusa se adapta mejor al mundo real en el que vivimos, e incluso puede comprender y funcionar con nuestras expresiones, del tipo "hace mucho calor", "no es muy alto", "el ritmo del corazón está un poco acelerado", etc. La clave de esta adaptación al lenguaje, se basa en comprender los cuantificadores de nuestro lenguaje (en los ejemplos de arriba "mucho", "muy" y "un poco").

Este enfoque es ampliamente usado debido a que el proceso de toma de decisiones no siempre es exacto, es decir no siempre es verdadero o falso, sino que frecuentemente involucra áreas donde no se tiene la completa certeza de que ocurran determinados hechos, por ejemplo, se usa el término "puede ser". En este paradigma, también tiene un especial valor la variable del tiempo, ya que los sistemas de control pueden necesitar retroalimentarse en un espacio concreto de tiempo o pueden requerirse datos anteriores para hacer una evaluación media de la situación en un periodo de tiempo anterior.

Algunas aplicaciones implementadas como sistemas expertos difusos son: planeación en línea, diagnóstico de fallas de procesos químicos, planeación ecológica, sistemas de control, razonamiento incierto, integración de conocimiento, diagnóstico de fallas, clasificación de sistemas de energía, detección de fallas, evaluación de la demanda, tratamiento de aguas residuales, pre-

dicción del suministro de agua, clasificación de radiografías, procesamiento analítico en línea, selección de hoteles, análisis de frecuencia de inundaciones, sistemas de consulta médica, indexado del rendimiento, seguridad computacional, reconocimiento de gestos y diagnóstico médico.

5. Metodología orientada a objetos. La metodología orientada a objetos define los programas en términos de "clases de objetos". Los objetos son entidades que combinan estado (es decir, datos), comportamiento (esto es, procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar. De esta forma, un objeto contiene toda la información, (los denominados atributos) que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases (e incluso entre objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos). A su vez, dispone de mecanismos de interacción (los llamados métodos) que favorecen la comunicación entre objetos (de una misma clase o de distintas), y en consecuencia, el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan (ni deben separarse) información (datos) y procesamiento (métodos).

La metodología orientada a objetos combina objetos con procedimientos específicos que operan sobre los datos, donde el objeto combina datos y código del programa. Los programas envían un mensaje a un objeto que ejecuta un procedimiento que está incorporado en dicho objeto. El mismo mensaje puede enviarse a diferentes objetos, pero cada uno de ellos implementará el mensaje de manera diferente. Los datos de los objetos están encapsulados en otras partes del sistema, de manera que cada objeto forma un bloque independiente de software que puede usarse en sistemas diferentes sin necesidad de cambiar el código del programa.

A continuación se mencionan algunas de las aplicaciones implementadas con la metodología orientada a objetos: diagnóstico industrial, redes de información, mantenimiento de sistemas de energía, aprendizaje, ingeniería del conocimiento, programación sintáctica y representación del conocimiento.

6. Razonamiento basado en casos. La experiencia en la resolución de problemas es una cualidad que poseen los humanos y que se requiere

"capturar" para poder crear un modelo de comportamiento inteligente. La resolución de problemas es una manera de adaptar los conocimientos para aplicarlos adecuadamente. En Inteligencia Artificial, McCarthy anunciaba en 1958: "Nuestro último objetivo es el de construir programas que aprendan de su propia experiencia tan efectivamente como lo hacen los humanos".

El aprendizaje basado en casos consiste precisamente en aprender a partir de experiencias o casos pasados. El aprendizaje basado en casos también se conoce como razonamiento basado en casos por el hecho de que este tipo de aprendizaje no se concibe sin el proceso de razonamiento que conlleva la obtención de una nueva experiencia.

El Razonamiento basado en casos (CBR) es el proceso que se usa para solucionar nuevos problemas basándose en las soluciones de problemas anteriores.

En el CBR, las descripciones de experiencias pasadas de los especialistas humanos, son representadas en casos y almacenadas en una base de conocimiento para su posterior recuperación, los cuales se usan cuando el usuario encuentra un nuevo caso con parámetros similares. El sistema busca en los casos almacenados, casos con características similares al problema que se está tratando de resolver.

Los elementos básicos del razonamiento basado en casos son:

- Representación del conocimiento. El conocimiento se representa en forma de casos que describen experiencias concretas. Si fuera necesario, se pueden almacenar otros tipos de conocimiento sobre el dominio de aplicación, por ejemplo, casos abstractos y generalizados, tipos de datos, modelos de objetos usados como información.

- Medida de similitud: Los sistemas de razonamiento basado en casos poseen mecanismos y conocimiento para adaptar los casos recuperados completamente a la situación actual.

- Adaptabilidad: Las situaciones pasadas que son representadas como casos, difícilmente serán idénticas a las del problema actual. Los sistemas avanzados de razonamiento basado en casos tienen mecanismos y el conocimiento para adaptar los casos recuperados completamente y verificar si satisfacen las características de la situación presente.

- Aprendizaje: Para que un sistema se mantenga actualizado es necesario que sea capaz de recordar determinada situación que resolvió un

problema con éxito, como una característica de un nuevo caso. Las soluciones exitosas son etiquetadas y se almacenan con los otros casos en la base de conocimiento. Las soluciones no exitosas también son aprendidas junto con el caso base y con las explicaciones del por qué dichas soluciones no funcionaron.

El modelo más aceptado para el proceso de CBR, es el ciclo de CBR que engloba un ciclo de razonamiento continuo compuesto por cuatro tareas principales:

1. Recuperar los casos más similares de la librería de casos.
2. Reutilizar este caso para resolver el problema.
3. Revisar la solución propuesta.
4. Retener la experiencia representando el caso actual (o parte de esta experiencia) para la reutilización futura.

Algunas de la aplicaciones implementadas con CBR son: diseño de proceso de manufactura, administración del conocimiento, inspección ultrasónica, planeación médica, aplicaciones médicas, diagnóstico de fallas, "e-learning" y modelado del conocimiento.

7. Desarrollo de la arquitectura del sistema. La arquitectura de un sistema experto es similar al dibujo arquitectónico de una casa. Le brinda al usuario una idea general de cómo se verá e implementará el sistema. La arquitectura muestra las capacidades generales del sistema, las interfaces del usuario, las funciones del sistema, el flujo de datos, el sistema de administración, el DBMS, el protocolo necesario, el lenguaje de programación específico, etc. Una vez que se ha completado el diseño de la arquitectura del sistema y su implementación, los usuarios pueden manipular y controlar las funciones del sistema en la arquitectura del sistema.

Algunas de las aplicaciones implementadas mediante la arquitectura del sistema son: selección y evaluación de material, diseño asistido por computadora, diseño ergonómico, implementación del sistema ISO, ingeniería concurrente, aplicaciones militares, simuladores de entrenamiento, diseño de estructuras de retención de líquidos y configuración de transbordadores.

8. Sistemas de agentes inteligentes. Actualmente, los agentes inteligentes y los sistemas multiagentes son temas ampliamente tratados debido a que representan una nueva manera de analizar, diseñar e implementar sistemas complejos de software. La *Teoría de Agentes* ofrece herramientas, técnicas y metáforas que son útiles en la conceptualización e implementación de muchos tipos de software. No obstante, existe una gran controversia en las definiciones de "agente", "sistema basado en agentes" y "sistema multiagente" dado que carecen de una definición universalmente aceptada.

Un *agente* es un sistema computacional, situado en algún ambiente, capaz de actuar autónomamente y dotado de flexibilidad para alcanzar sus objetivos. En esta definición resaltan tres conceptos clave: *situacionalidad*, *autonomía* y *flexibilidad*. *Situacionalidad* significa que el agente está sensando su ambiente. Ejemplos de ambiente donde los agentes están situados incluyen tanto al mundo físico como a Internet. La *autonomía* es un concepto difícil de definir de manera precisa, para nuestro contexto, autonomía es la medida en la que el sistema es capaz de actuar sin la intervención directa de humanos o de otros agentes; adicionalmente, debe tener control sobre sus propias acciones y su estado interno. En un sentido más estricto, *autonomía* significa que los agentes son capaces de aprender de su propia experiencia. La *flexibilidad* se presenta cuando el sistema es:

- Sensitivo o sensible. Los agentes deben percibir su ambiente y responder a los cambios que ocurran en él.
- Pro-activo. Los agentes simplemente no deben actuar en respuesta a su ambiente, sino que deben estar habilitados para exhibir *oportunismo*, conductas dirigidas a la meta y tomar la iniciativa cuando sea apropiado.
- Social. Los agentes deben estar habilitados para interactuar con otros agentes artificiales o humanos, cuando sea necesario, para resolver su propio problema o para ayudar a otros en sus actividades.

Naturalmente, algunos agentes tienen características adicionales y, para ciertos tipos de aplicaciones algunos atributos serán más importantes que otros. Sin embargo, la presencia de estos atributos distingue a los sistemas basados en agentes de los paradigmas de software relacionados tales como: sistemas orientados a objetos, sistemas distribuidos y sistemas expertos.

Un *sistema basado en agentes* puede contener uno o más agentes. Existen casos donde se necesita un sólo agente para solucionar un problema dado.

Un buen ejemplo lo constituye la clase de sistemas conocidos como asistentes expertos donde un agente actúa como una asistente experto para un usuario que intenta usar una computadora para efectuar alguna tarea.

Un *sistema multigente* está diseñado e implementado para la interacción entre agentes. Los sistemas multiagentes se utilizan para resolver problemas que tienen muchas maneras de solucionarse. Tales sistemas cuentan con los beneficios de los sistemas distribuidos y de la solución de problemas concurrentes con la ventaja adicional de incluir patrones sofisticados de interacción. Los principales tipos de interacciones son:

- Cooperación. Se refiere al trabajo conjunto para alcanzar una meta común.
- Coordinación. Es una actividad que trata de evitar interacciones adversas y explota las interacciones benéficas.
- Negociación. Su objetivo consiste en llegar a un contrato aceptable para las partes involucradas.

Algunas de las aplicaciones implementadas mediante agentes inteligentes son: sistemas tutoriales inteligentes, diseño y análisis de sistemas, mantenimiento de servicio electrónico, reglas de contaminación del carbono, representación del conocimiento, sistemas adaptativos, control de la contaminación del aire, diseo arquitectónico, agricultura, simulación industrial, ingeniería del conocimiento en la plataforma del WWW.

9. Metodología de bases de datos. Una base de datos o banco de datos es un conjunto de datos pertenecientes al mismo contexto y almacenados sistemáticamente para su uso posterior.

Un sistema de base de datos deductivas, es un sistema de base de datos pero con la diferencia de que permite hacer deducciones a través de inferencias. Se basa principalmente en reglas y hechos que son almacenados en la base de datos. También las bases de datos deductivas son llamadas base de datos lógicas, a raíz de que se basan en lógica matemática.

El Sistema de Gestión de Bases de Datos es un conjunto integrado de programas, procedimientos, lenguajes, etc., que suministra, tanto a usuarios no informáticos como a los analistas, programadores o al administrador, los medios necesarios para describir, recuperar y manipular los datos, manteniendo su integridad, confidencialidad y seguridad. Las funciones del sistema de gestión de bases de datos son las siguientes:

- *Data Warehouse* (almacén de datos), con herramientas para analizar toda la información acumulada a lo largo de años.

- *Data mining* (minería de datos), muy vinculado al anterior. El cual se define como una actividad de extracción cuyo objetivo es el de descubrir hechos contenidos en las bases de datos. La palabra "descubrimiento" está relacionada con el hecho de que mucha de la información valiosa es desconocida con anterioridad.

Un *Data Warehouse* es un almacén o repositorio de datos categorizados, que concentra un gran volumen de información de interés para toda una organización, la cual se distribuye por medio de diversas herramientas de consulta y de creación de informes orientadas a la toma de decisiones. El objetivo del *Data Warehouse (DW)* es agrupar los datos con el propósito de facilitar su posterior análisis, de forma que sean fáciles de acceder y, posteriormente, analizar dicha información.

Los *Data Warehouses* a menudo almacenan gran cantidad de información, la cual está a veces subdividida en pequeñas unidades lógicas. Periódicamente, se importan estos datos de otros sistemas de información dentro del Data Warehouse, para realizar sobre ellos un procesamiento posterior.

Data Mining es el proceso de descubrir patrones de información interesante y potencialmente útiles, inmersos en una gran base de datos en la que se interactúa constantemente. *Data Mining* es una combinación de procesos como:

- Extracción de datos.

- Limpieza de datos.

- Selección de características.

- Algoritmos.

- Análisis de resultados.

Las herramientas de Data Mining exploran gran cantidad de datos dentro de una BD grande, y mediante su análisis predicen posibles tendencias o comportamientos futuros, permitiendo al experto tomar decisiones de una forma rápida y utilizando un conocimiento que de otra forma no habría encontrado.

Algunas de las aplicaciones implementadas mediante dicha metodología son: sistemas de planeación de energía, planeación geográfica, sistemas de

información geográfica, interpretación de rocas sedimentarias, diagnóstico de medicina tradicional china y sistemas expertos médicos.

10. Modelado. La metodología del modelado ha llegado a ser una metodología de desarrollo de sistemas expertos para construir relaciones formales con el diseño de modelos lógicos en diferentes dominios de conocimiento. Además la tecnología de modelado brinda métodos cuantitativos para el análisis de datos y representar o adquirir conocimiento del experto con programación lógica inductiva o algoritmos de Inteligencia Artificial, ciencias cognitivas y otros campos de investigación que tengan plataformas más amplias para la implementación de tecnologías para el desarrollo de sistemas expertos.

Ejemplos de aplicaciones que han sido implementados mediante el modelado son: control de procesos, análisis médico, toma de decisiones, evaluación de software, validación de sistemas médicos, simulación y planeación de tareas de ensamblado, diseño de terminales de transporte, asignación de proyectos y clasificación de hiperplasia endometrial.

11. Ontologías. Una ontología es un vocabulario que se usa como concepto fundamental para describir las tareas de un dominio dado. Dicho vocabulario se usa como base para la comunicación entre los expertos del dominio y los ingenieros del conocimiento. Consecuentemente la ontología de un dominio específico se puede reusar.

Hemos de tener en cuenta que una "conceptualización" es en cierta medida una "simplificación de un mundo que se desea representar para algún propósito". Una ontología hace referencia a la formulación de un exhaustivo y riguroso esquema conceptual dentro de un dominio dado, con la finalidad de facilitar la comunicación y compartir la información entre diferentes sistemas.

A continuación, se mencionan algunas aplicaciones que se han implementado mediante el uso de ontologías: soporte a la toma de decisiones médicas, reuso del conocimiento, control preventivo, adquisión de conocimiento, heurísticas para el juego de ajedrez entre otras.

2.2.1 Arquitectura Clásica de un Sistema Experto

La arquitectura clásica de un sistema experto se muestra en la Figura 2.1 [62].

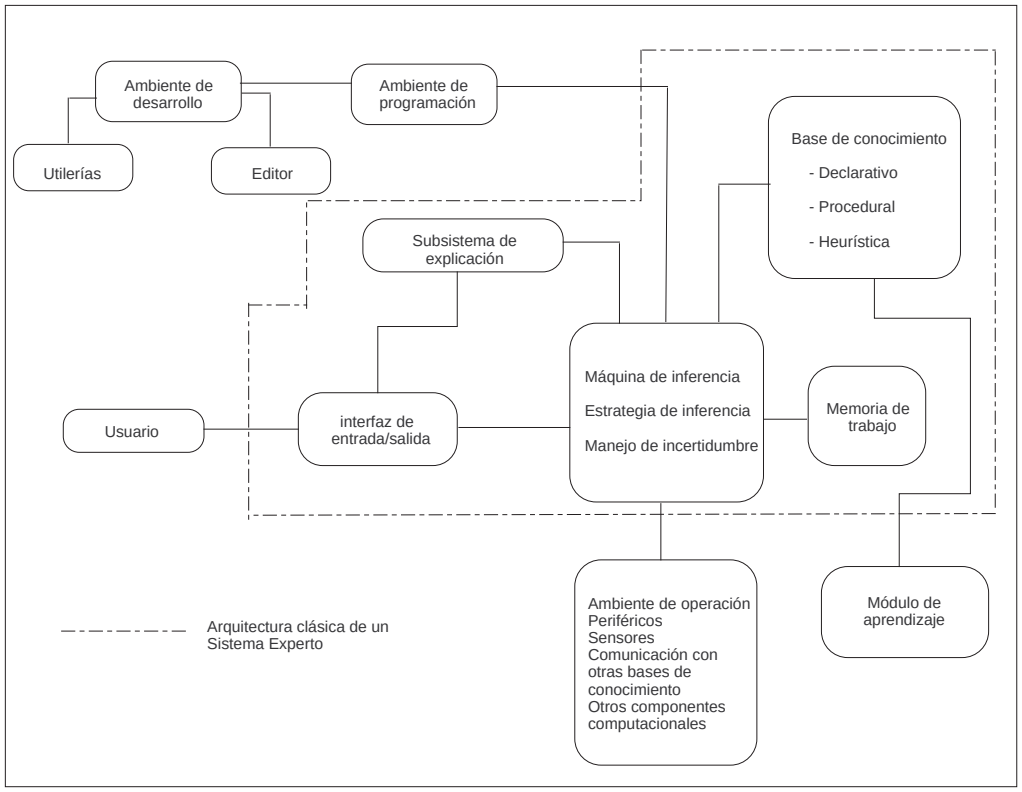

Figura 2.1: Arquitectura clásica de un Sistema Experto

La base de conocimiento, el mecanismo de inferencia, la memoria de trabajo y la interfaz entrada/salida son componentes que se encuentran en todo *Sistema Experto* basado en reglas, mientras que en algunos sistemas, el subsistema de explicación pudiera no estar presente y el módulo de aprendizaje sólo está presente en algunos [38].

La base de conocimiento

La base de conocimiento está formada por reglas, hechos e información acerca del dominio especializado de conocimientos. Dicho conocimiento lo utiliza el

mecanismo de inferencia para formular hipótesis.

Una *regla de producción* es una estructura de la forma:

IF < antedecente > THEN <consecuente>

Donde la parte izquierda es conocida como *antecedente* y se refiere a las condiciones o premisas correspondientes a una situación dada, mientras que la parte derecha -conocida como *consecuente*- contiene la conclusión, acción o consecuencia a realizar dado el caso de que las premisas o condiciones hayan sido satisfechas.

Las reglas se activan mediante datos (hechos)y el *intérprete de reglas* controla la activación y selección de las reglas en cada ciclo.

La *memoria de trabajo* [1] se examina constantemente y se modifica a través de las reglas. El estado actual de una consulta en curso se almacena en la memoria de trabajo. La función primordial de la memoria de trabajo consiste en mantener los datos en la forma de vectores objeto-atributo-valor para que el intérprete active las reglas. Dada la presencia o ausencia de hechos y/o metas en la memoria de trabajo, se dispararán algunas reglas para satisfacer los patrones en sus premisas.

El intérprete de reglas se encarga de reconocer y ejecutar el conjunto de reglas de producción cuya premisa haya sido satisfecha. Hace uso del *ciclo de reconocimiento de acciones* que consta de tres pasos básicos:

1. Confrontar los patrones de las premisas de las reglas con los elementos almacenados en la memoria de trabajo.

2. En el caso de que pueda dispararse más de una regla, se elige una de ellas mediante un criterio predefinido, a este paso se le conoce como *resolución del conflicto*. Algunos ejemplos de criterios para determinar cuál acción ejecutar cuando exista una resolución del conflicto son:

   - Establecer orden en los datos.
   - Clasificar las reglas por prioridad de ejecución.
   - Elegir la regla que hace referencia a los elementos de la memoria de trabajo más recientemente actualizados.
   - Elegir la regla más específica.

[1] Es una estructura de datos central que almacena los hechos de manera dinámica.

- Aplicar meta-reglas [2].

3. Disparar la regla seleccionada y regresar al paso 1.

El control se lleva a cabo a través de un *encadenamiento hacia adelante* (conducido por los datos) o *encadenamiento hacia atrás* (conducido hacia la meta).

En el *encadenamiento hacia adelante* se parte de los hechos hacia la meta. Es decir, se toman los hechos iniciales del problema y se aplican las reglas correspondientes hasta llegar a la meta. En este tipo de razonamiento se consideran los hechos establecidos y se verifican con las reglas, al cumplirse el miembro izquierdo de la regla, se ejecuta la parte derecha. Este encadenamiento se basa en el *"modus ponens"*.

El *encadenamiento hacia atrás* está basado en el *"modus tollens"*. Se centra en la meta u objetivo que se localiza en la memoria de trabajo. Primero se encuentran las reglas que estén involucradas en la obtención de la meta, de esta manera se descompone el problema en una serie de submetas, así las premisas de las reglas que se aplican pasan a ser nuevos objetivos. El sistema trabaja hacia atrás desde la meta original hasta todas las submetas almacenadas en la memoria de trabajo. Aquí se consideran el miembro derecho de la regla junto con las metas posibles a alcanzar y sólo se verifican las reglas que concluyen esas metas.

También es posible utilizar un razonamiento mixto que es una combinación de encadenamiento hacia adelante y encadenamiento hacia atrás.

Algunas veces la base de conocimiento es de naturaleza estática, debido a que el conocimiento no se modifica ni se actualiza durante los procesos de solución de problemas, a menos que el Sistema Experto contenga un módulo de aprendizaje incorporado.

La cantidad y calidad del conocimiento almacenado en la base de conocimiento determinan el éxito de un Sistema Experto. Una buena base de conocimiento debe ser *exhaustiva* y *consistente*. El término *e*xhaustivo significa completitud, es decir, se deben incorporar todas las reglas y hechos necesarios para solucionar cualquier problema del dominio. Cuando la base de conocimiento no contiene reglas contradictorias, redundantes o innecesarias se dice que es *consistente*.

[2]Las meta-reglas son reglas que se aplican a reglas.

La máquina de inferencia

Uno de los elementos más importantes de la arquitectura de un Sistema Experto es la *máquina de inferencia*. Los procesos de inferencia en los Sistemas Expertos basados en reglas se llevan a cabo a través de encadenamientos de reglas: hacia adelante o hacia atrás o una combinación de ambos.

Para realizar las inferencias, se utiliza información dinámica y conocimiento estático. La información dinámica corresponde a los datos de entrada que fueron inicialmente aportados por el usuario y a las respuestas que dicho usuario proporciona a preguntas formuladas por el sistema. El conocimiento estático se encuentra almacenado en la base de conocimiento y lo utiliza el mecanismo de inferencia para formular hipótesis u obtener conclusiones acerca del caso o la situación actual.

Toda la información que aporta el usuario se almacena como hechos en la memoria de trabajo. Durante el proceso de inferencia, las reglas almacenadas en la base de conocimiento se comparan con el contenido de la memoria de trabajo. Cuando más de una regla logra satisfacer sus condiciones, se realiza un proceso de selección para determinar la regla que será ejecutada. Luego, su acción o consecuente pasa a formar parte de los hechos almacenados en la memoria de trabajo. Esto origina la formación de nuevas configuraciones de hechos, cuyas condiciones pueden ser satisfechas por otras reglas de la base de conocimiento. De esta forma, el proceso continúa recursivamente hasta que ya no sea posible encontrar nuevas reglas que se disparen.

La memoria de trabajo

La memoria de trabajo o base dinámica, almacena la información dinámica de manera temporal. Aquí se almacena toda la información brindada por el usuario al sistema (datos iniciales y respuestas a preguntas) en forma de hechos, así como las conclusiones de todas las reglas "disparadas" en el transcurso del proceso de inferencia.

El contenido de la memoria de trabajo se elimina cuando concluye o finaliza el proceso de solución de un particular. De esta manera, la memoria se encuentra vacía cuando inicie la solución de otro problema.

La interfaz de entrada/salida

La interfaz de entrada/salida permite la comunicación entre el usuario y el sistema. Mediante ésta, el usuario ingresa datos iniciales al sistema o

responde preguntas elaboradas por el sistema.

El subsistema de explicación

El subsistema de explicación brinda al usuario explicaciones del proceso de inferencia cuando sean solicitadas. Las explicaciones ofrecidas al usuario responden a las siguientes preguntas:

- ¿Cómo se alcanzó una hipótesis o conclusión?. El sistema muestra al usuario la cadena de reglas disparada durante el proceso de inferencia. De esta manera, el usuario puede apreciar la línea de razonamiento seguida por el sistema hasta llegar a la conclusión.

- ¿Por qué se requiere cierta información?. Consiste en explicar al usuario el papel que desempenã la información solicitada para obtener algún paso necesario dentro del proceso de razonamiento.

A través del subsistema de explicación el usuario puede conseguir muy de cerca el proceso de inferencia llevado a cabo durante la solución de un problema.

El módulo de aprendizaje

La función del módulo de aprendizaje es contribuir en la construcción y el refinamiento de la base de conocimiento. Un mecanismo de aprendizaje permite al sistema para adquirir nuevos conocimientos y para refinar el conocimiento existente en la base de conocimiento.

2.2.2 Sistemas Expertos y los Centros de Ayuda

Muchos sistemas expertos han sido desarrollados para el diagnóstico de problemas. **Las reglas de producción** son el formalismo más popular de representación del conocimiento. Los métodos para obtener reglas se clasifican en dos categorías:

1. Métodos automatizados (*"Machine Learning"*).

2. Métodos manuales (por ej. entrevistas).

Independientemente del método que se utilice, el enfoque basado en reglas construye una base de conocimiento que interpreta el problema y sugiere soluciones. Las reglas de una base de conocimiento constituyen una buena fuente de ayuda para el usuario. No obstante, existen diversos problemas en el desarrollo de sistemas expertos mediante el formalismo de reglas de producción:

- Es difícil construir la base de conocimiento.
- El mantenimiento de la base de conocimiento es una tarea complicada.
- La habilidad para actualizar el conocimiento.

Existen diversos enfoques para resolver dichos problemas, destacando el enfoque basado en el "nivel de conocimiento" de Newell [47].

A pesar de que se han usado ampliamente los sistemas basados en reglas para el desarrollo de centros de ayuda, también se utiliza frecuentemente, el enfoque del **razonamiento basado en casos (CBR)**. SMART [9], CASCADE [59] y CARET [55] son clásicos ejemplos de sistemas expertos en el dominio de los centros de ayuda que usan el enfoque del razonamiento basado en casos.

El razonamiento basado en casos construye sistemas expertos usando casos pasados para resolver nuevos problemas. Está basado en el siguiente supuesto: la experiencia real proviene de la experiencia del experto y de la memoria episódica, como un modelo apropiado para la experiencia.

El enfoque del razonamiento basado en casos, no encuentra reglas apropiadas en una base de conocimiento, sino que encuentra casos similares al caso base. El razonamiento basado en casos es apropiado cuando no existe conocimiento formalizado del dominio o es difícil para el experto humano expresar su experiencia en el formato de reglas.

La similaridad funcional entre el razonamiento basado en casos y los métodos de recuperación de información, radica en que ambos métodos realizan la misma tarea que consiste en la recuperación de los casos relevantes o documentos. Ambos métodos, mantienen un conjunto de casos o documentos, agregando los nuevos casos o documentos en la base de datos para su uso posterior. Mientras que, los estudios relacionados con la recuperación de la información se concentran en la recuperación de extensos documentos de las bases de datos, el razonamiento basado en casos logra representar el conocimiento para la solución de un problema humano mediante la representación del caso.

Sistemas Expertos basados en Web

Según [66], las metodologías existentes para el desarrollo de sistemas basados en reglas para Web se clasifican en cinco categorías, de acuerdo a las tecnologías aplicadas. La Tabla 2.1 muestra las cinco categorías y un resumen de sus características técnicas.

2.3 Entornos de Desarrollo de Sistemas Expertos

La adquisición y representación del conocimiento, así como la construcción de la base de conocimiento constituyen las tareas principales para el desarrollo de sistemas expertos. El gran número de reglas y hechos que requieren muchos de estos sistemas, se obtienen con la intervención de expertos en el dominio de la aplicación. La representación del conocimiento de los expertos en los sistemas es un proceso generalmente largo y tedioso.

Los *entornos de desarrollo o "shells"* permiten a un ingeniero del conocimiento o experto humano capturar y usar conocimiento sin necesidad de escribir gran código de soporte . Tienen incorporadas una o varias técnicas de inferencia (métodos de razonamiento del conocimiento) y un ambiente que mejora ampliamente el proceso de adquisición de conocimiento, facilitando de esta manera, la construcción de un sistema experto y reduciendo el tiempo de desarrollo con la consecuente disminución de los costos, ya que con el uso de un lenguaje tradicional como LISP o PROLOG se consume demasiado tiempo.

En nuestros días, existen en el mercado una gran variedad de *entornos de desarrollo* que soportan interfaces basadas en web, desarrollados para su total comercialización o como producto de los centros de investigación.

Los *"shells"* y los lenguajes para el desarrollo de aplicaciones basadas en web, usan mecanismos de razonamiento convencionales para la programación de sistemas expertos, tales como inferencia basada en reglas, inducción de reglas mediante árboles de decisión, razonamiento bayesiano, lógica difusa, los cuales se integran con tecnologías Web para la administración de la interfaz del usuario y los componentes cliente servidor de la aplicación. Los *entornos de desarrollo* difieren en el nivel de flexibilidad que brindan para el desarrollo de un sistema experto y para la representación del conocimiento. También difieren en el nivel de interoperabilidad de las bases de conocimiento

Localización de la máquina de inferencia	**Tipo de la máquina de inferencia**	**Características técnicas**
Servidor	Programa CGI	Usa el estándar CGI (Common Gateway Interface). El servidor web invoca al CGI mientras pasa los parámetros requeridos de acuerdo al estándar CGI.
	Script almacenado en el Servidor	La máquina de inferencia se desarrolla en ambientes tales como JSP, ASP y PHP.
Cliente	Servidor Web con un módulo de visualización externo incorporado	La máquina de inferencia está en el servidor web como un submódulo usando un API, como NSAPI.
	Applet de Java	Los *bytescodes* de la máquina de inferencia están localizados en el servidor, pero son transmitidos a un navegador web. La máquina virtual de Java del lado del cliente interpreta los *bytecodes* y los ejecuta.

Tabla 2.1: Categorías de Sistemas Expertos basados en Web.

y de los sistemas que los usuarios pueden desarrollar mediante el uso de dichos entornos de desarrollo de sistemas expertos. Típicos ejemplos de dichas herramientas son:

- *Cafe Rete*, fue desarrollado por Haley Enterprise Inc. en 1996. Es una librería de clases de Java que integra reglas de producción con aplicaciones de java, servlets, EJBs, etc.

- *CLIPS* desarrollado en la NASA, basado en reglas que incluye características de la metodología orientada a objetos y una interfaz directa a objetos de Java.[3]

- *Jess (Java Expert System Shell)*, fue desarrollado en *Sandia National Laboratories*, es un entorno de desarrollo de sistemas expertos que goza de gran popularidad y es de uso libre para el mundo académico.

  Jess es fácil de usar y brinda gran robustez para el desarrollo de sistemas expertos. Su mayor ventaja radica en su capacidad de integración con programas externos desarrollados en Java, mediante su bien definida API, la cual, permite controlar el razonamiento basado en reglas desde programas de Java. La máquina de inferencia de Jess está basada en el encadenamiento hacia adelante del algoritmo Rete [41], sin embargo, también soporta el encadenamiento hacia atrás. Otra importante ventaja de Jess es la interoperabilidad dado que hace posible el almacenamiento de las bases de conocimiento en formato XML. [4]

- *XPertRule* KBS Shell, ejecuta la inducción de reglas basada en árboles de decisión e incorpora lógica difusa, la interfaz al Web se hace a través de ASP (Active Server Pages).[5]

- *ExSys CORVID* incorpora razonamiento basado en reglas con lógica difusa. Está basado en programas CGI. Permite el desarrollo de aplicaciones locales o basadas en Web donde la interacción con el usuario emula una conversación con el experto humano para obtener las respuestas, aún en áreas donde existe una lógica probabilística compleja y muchos niveles de razonamiento.[6]

[3]http://www.ghg.net/clips/CLIPS.html
[4]http://herzberg.ca.sandia.gov/jess
[5]http://www.attar.com/
[6]http://www.exsys.com/

- *Blaze Advisor* (Fair Isaac Corporation). Está basado en programas CGI (*"Common Gateway Interface"*). Es un administrador de reglas, diseñado para la automatización de las decisiones operativas de las empresas, brindándoles agilidad, consistencia y precisión en las interacciones con sus clientes. [7]

- *ILOG's Business Rule Management System*, permite incorporar reglas de negocios en el Web. La tecnología empleada ofrece optimización y visualización de las reglas. Brinda flexibilidad para trabajar en Java, C, Visual Basic, Eclipse y Microsoft Visual Studio .NET. [8]

- *eXpertise2Go* provee el *"applet"* de java denominado *e2gLite* que carga la base de conocimiento del servidor y se ejecuta completamente en el navegador. Se distribuye gratuitamente para fines educativos.[9]

- *OPSJ* es el entorno de desarrollo más rápido de naturaleza comercial, basado en Java que permite un fácil desarrollo en ambientes de *Java 2 Enterprise Edition (J2EE).* [10]

- *JavaDON* es un entorno de desarrollo de sistemas expertos *open source* basado en OBOA (marco de trabajo para el desarrollo de sistemas inteligentes). La idea central del proyecto JavaDON fue brindarle al usuario una herramienta para la construcción de sistemas expetos, fácil de usar. Permite el desarrollo de complejas aplicaciones de sistemas expertos *stad alone* o basadas en Web. El esquema de representación de conocimiento de JavaDon, soporta el uso de elementos multimedia con las técnicas tradicionales tales como reglas y *frames*. Otra importante característica, es su capacidad de almacenamiento de las bases conocimiento en formato XML, por lo que es potencialmente fácil interoperar con otras bases de conocimiento en Internet [61].

Para la elección de un *entorno de desarrollo* es necesario tomar en cuenta la naturaleza del dominio del sistema experto a desarrollar y, determinar la estrategia apropiada que cubra las necesidades del sistema (técnicas de

[7] http://www.fairisaac.com/Fairisaac/Solutions/Enterprise+Decision+Management-/Business+rules/Blaze+Advisor/

[8] http://www.ilog.es/products/businessrules/products.cfm

[9] http://www.expertise2go.com/

[10] http://www.pst.com/opsj.htm

inferencia, formalismo para la representación del conocimiento, manejo de incertidumbre, etc.).

Dada la naturaleza del Sistema Experto basado en Web a implementar y a las múltiples facilidades para el desarrollo, así como su fácil integración con Java, se eligió a *Java Expert System Shell (Jess)* como herramienta de desarrollo.

2.4 Jess (Java Expert System Shell)

Jess (Java Expert System Shell) fue desarrollado como una herramienta para el soporte de la investigación en el marco de los sistemas inteligentes en Internet. El lenguaje Jess es una variante de CLIPS, el cual, posee una historia exitosa en el desarrollo de sistemas expertos. Jess soporta razonamiento basado hacia delante y hacia atrás e interfaces directas de objetos de Java, brindando todas las características de desarrollo del lenguaje de programación Java. Sin duda, es una herramienta ideal para agregar reglas de producción a sistemas basados en Java. Posee una buena documentación y una API bien documentada, así como un excelente soporte de la comunidad de desarrolladores. [20]

Jess ha sido usado para desarrollar un amplio rango de software comercial, basado en tecnología inteligente, tales como:

- Sistemas Expertos.
- Agentes que predicen precios y compran y venden seguros.
- Detectores de intrusos en redes y auditores de seguridad.
- Asistentes de diseño que ayudan a ingenieros mecánicos.
- Servidores que ejecutan reglas de negocios.
- Sitios inteligentes de comercio electrónico.
- Juegos.

En Jess se puede programar de dos diferentes maneras :

1. Como máquina de reglas. Un programa basado en reglas puede contener cientos o miles de reglas que Jess aplicará a los datos. Frecuentemente, las reglas representan el conocimiento heurístico de un experto humano en algún dominio y la base de conocimiento representa el estado del desarrollo de una situación. En este caso, las reglas se usan para constituir un sistema experto.

2. Jess también es un lenguaje de programación de propósito general que puede acceder a todas las clases y librerías de Java. En consecuencia, resulta muy fácil extender el lenguaje Jess con nuevos comandos escritos en Java o en el mismo Jess, de esta manera, se puede personalizar para aplicaciones específicas.

2.4.1 El Algoritmo *RETE* de Jess

La ejecución de Jess es rápida, debido a que el algoritmo usado en el *"pattern matching"* es muy eficiente y puede procesar grandes pilas de reglas y hechos en corto tiempo.

Dado que Jess hace uso intensivo de la memoria, su ejecución es sensible al comportamiento del *"garbage collector"* de java. El *"garbage collector"* es la parte de la Máquina Virtual de Java: JVM (*Java Virtual Machine*) quien es la responsable de encontrar y eliminar los objetos que no se usan.

La máquina de reglas de Jess usa una forma mejorada del **algoritmo Rete** [11] para el proceso de apareamiento (*"match"*) de las reglas con la memoria de trabajo. Charles Forgy's en su clásico artículo: *"Rete: A Fast Algorithm for the Many Pattern / Object Pattern Match Problem"* [41] describe al algoritmo Rete, el cual, se ha convertido en la base de rápidas herramientas de desarrollo de sistemas expertos tales como OPS5, su descendiente ART, CLIPS, Jess y otros. Cada sistema ha incrementado y refinado dicho algoritmo para mejorar el desempeño o flexibilidad.

OPS5 provee una estructura para la representación de hechos, relaciones y conocimiento incierto e incluye un compilador que elimina ampliamente las redundancias. Hace uso de un algoritmo de redes, precisamente del algoritmo *Rete*, para el procesamiento de reglas. Los cambios en los datos (*"tokens"*) fluyen a través de la red, provocando la activación de los nodos. En dichos nodos, se ejecutan las operaciones que a su vez crean nuevos *"tokens"* generando, de esta manera, un ciclo.

[11] Significa Red en Latín.

Las reglas se disparan en el ciclo de reconocimiento de acciones. En cada ciclo, se identifica el conjunto de reglas que podrían ejecutarse de acuerdo al estado global actual. La selección de una regla se lleva a cabo de acuerdo a la "estrategia de resolución del conflicto" que se emplee. Luego, se dispara la regla elegida, ejecutándose el consecuente y provocando una modificación del estado global. El ciclo termina cuando no haya reglas para disparar o tenga lugar un paro explícito. OPS5 ha servido para el desarrollo de muchas aplicaciones.

Jess usa el rápido y eficiente algoritmo RETE para el *"pattern matching"*. La fortaleza del algoritmo RETE radica en que usa un conjunto de *memoria* para almacenar la información del éxito o falla del *"pattern matching"* en ciclos previos. El algoritmo RETE construye una red de nodos del *"pattern matching"*. Jess hace uso de diferentes clases de nodos para representar las diferentes clases de las actividades del *"pattern matching"*. También hace uso de nodos especiales para el manejo de algunos elementos condicionales como *not* y *test*, así como comportamientos especiales en algunos nodos para manipular el encadenamiento hacia atrás.

A continuación, se hace una breve descripción de cómo se implementó el algoritmo *Rete* en Jess. Brevemente, el algoritmo elimina la ineficiencia de un simple *"pattern matching"*, recordando los resultados de las evaluaciones pasadas a través de iteraciones del ciclo de las reglas. En cada paso, únicamente se evalúan nuevos elementos de la memoria de trabajo o elementos eliminados con las reglas . Rete organiza el *"pattern matching"* para que solamente estos nuevos hechos sean evaluados contra el subconjunto de reglas que actualmente pueden hacer *match*.

Descripción del Algoritmo *RETE*

El Algoritmo *RETE* fue implementado para construir una red de nodos interconectados. Cada nodo representa una o más evaluaciones encontradas en el antecedente (LHS) de la regla. Un nodo tiene una o dos entradas y cualquier número de salidas. Los hechos que son agregados o eliminados de la memoria de trabajo son procesados por esta red de nodos. Los nodos de entrada están en el "tope" de la red y los nodos de salida están al final. Estos nodos juntos forman la *Red Rete*.

En el "tope" de la red, los nodos de entrada separan los hechos en categorías de acuerdo al contenido en su "cabeza". Dentro de la red, se efectúan finas discriminaciones y asociaciones entre los hechos, hasta que llegan al

fondo (bottom). En el fondo de la red, los nodos representan reglas individuales. Cuando un conjunto de hechos filtra todo hacia el fondo de la red, significa que han pasado todas las evaluaciones del antecedente de una regla en particular; este conjunto junto con la misma regla, pasan a formar un nuevo registro de activación o un comando para cancelar previamente, en el registro de activación existente.
[12]

La red está compuesta de dos amplias categorías:

1. Nodos de una entrada. Los cuales ejecutan evaluaciones de hechos individuales.

2. Nodos de dos entradas que efectúan evaluaciones a través de múltiples hechos.

A continuación se muestra un ejemplo del uso del algoritmo. Las siguientes reglas se compilan en la red como se ilustra en la Figura 2.2.

```
Jess> (deftemplate x (slot a))
TRUE
Jess> (deftemplate y (slot b))
TRUE
Jess> (deftemplate z (slot c))
TRUE
Jess> (defrule ejemplo1
      (x (a ?v1))
      (y (b ?v1))
      => )
TRUE
Jess> (defrule ejemplo2
      (x (a ?v2))
      (y (b ?v2))
      (z)
      => )
TRUE
```

[12]Un registro de activación es una asociación de una lista de hechos con una regla que ellos pueden activar.

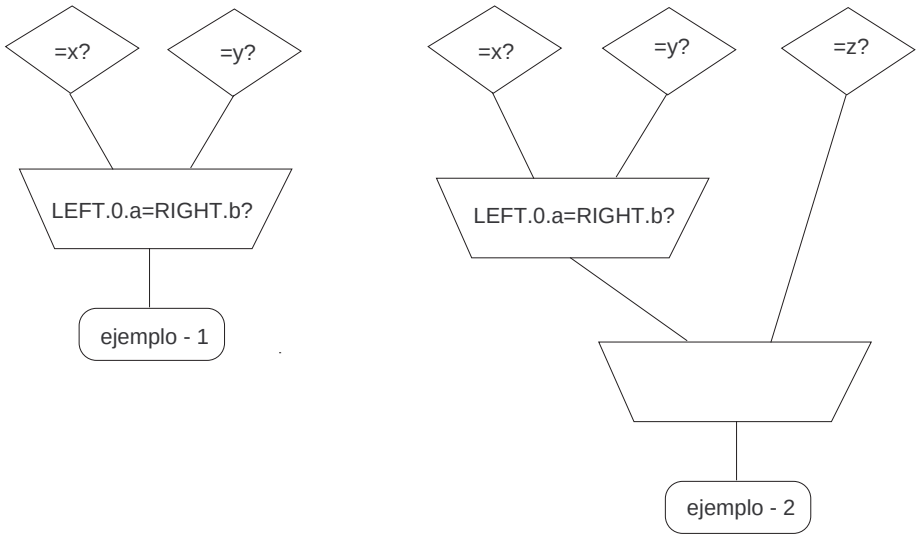

Figura 2.2: Una red RETE no optimizada para las reglas ejemplo1 y ejemplo2.

En este diagrama, cada caja representa a un nodo. Las entradas de los nodos se muestran en el tope y sus salidas hasta abajo. Los nodos en forma de diamante, marcados con *=q?* son los nodos de una entrada o nodos de patrones. Los nodos de patrones en este ejemplo, evalúan si la cabeza de un hecho es *q*. Los hechos que pasan dicha evaluación, son envíados a los nodos de salida y se ignoran los otros nodos.

Los nodos representados en forma de trapecio, son nodos de dos entradas o nodos *"join"* (unidos). Cada nodo *"join"* une los resultados del *"matching"* (apareamiento) de los primeros *n - 1* patrones (marcados en el diagrama con *LEFT*) con los patrones (*nth*) mostrados en el diagrama con la etiqueta *RIGHT*. Los nodos *join* recuerdan todos los hechos o grupos de hechos que llegan en cualquiera de sus dos entradas. La red se construye para que la entrada etiquetada con *LEFT* pueda recibir grupos de uno o más hechos y la entrada etiquetada con *RIGHT* sólo reciba un único hecho. Cada nodo *join* produce grupos de dos o más hechos como su entrada. Las llegadas de las dos entradas se mantienen en áreas separadas de memoria, tradicionalmente llamadas memorias *"alpha"* (etiquetada con *"LEFT"*) y *"beta"* (etiquetada con *"RIGHT"*).

La notación LEFT.p.q==RIGHT.R? indica una evaluación que com-

para los contenidos en el slot *q* en el hecho *p*th en un grupo de la memoria *LEFT* para el slot *r* en el hecho de la memoria *RIGHT*. Los nodos *"join"* producen una salida para cada par ordenado de un elemento de memoria *LEFT* y un elemento de memoria *RIGHT* que pasa las evaluaciones en este nodo.

Los nodos en forma ovalada en el fondo de la red, son los *nodos terminales* que representan reglas individuales. Tienen una entrada única pero no tienen salidas. Cuando reciben una entrada, construyen un registro de activación del *item* de entrada y la regla se representa y localiza en la agenda. Es importante destacar, que cualquier hecho que llegue hasta el tope de un nodo *join*, potencialmente podría contribuir a una activación, los cuales ya han pasado todas las evaluaciones que pueden aplicarse a hechos individuales.

Para la ejecución de la red, se presenta cada nuevo hecho de cada nodo en el tope de la red. El patrón de la red del ejemplo, elimina todos los hechos excepto los hechos *x,y.* y *z.* La red unida envía todos los pares {*x, y*} con *x.a == y.b* al nodo terminal del *ejemplo1* y para todos los triplos (dadas las mismas restricciones) para el nodo terminal del *ejemplo 2.* De esta manera, los nodos terminales conocen que registro de activación crear.

Que pasaría si después del procesamiento de los hechos iniciales, asertamos el hecho adicional: *(z (c 17))*? El hecho está representado para el nodo del patrón *=z?* y es enviado hacia abajo al nodo *join* que esté más abajo. La memoria *LEFT* del nodo *join* ya contiene todos los pares aceptables de los pares *x, y*, así que el triplo correcto *x y z* puede formarse sin repetir el *pattern matching* efectuado en el primer ciclo. Se creará una nueva activación para cada par *x y* pre calculado. La arquitectura Rete ayuda a evitar repetir cálculos en el tiempo.

Manejo del *"retract"*

El algoritmo RETE puede manejar el *"retract"* eliminando los registros de activación. Para lo anterior, no se deben enviar hechos a través de la red, sino que se deben enviar *"tokens"*. Un *"token"* es una asociación entre uno o más hechos y una etiqueta -*"tag"*- o comando. La etiqueta le dice a los nodos individuales como interpretar el *"token"*. Jess usa 4 diferentes *"etiquetas"*, las cuales están definidas como constantes en la clase JESS.RU:

1. ADD. Se usa para asertar hechos.

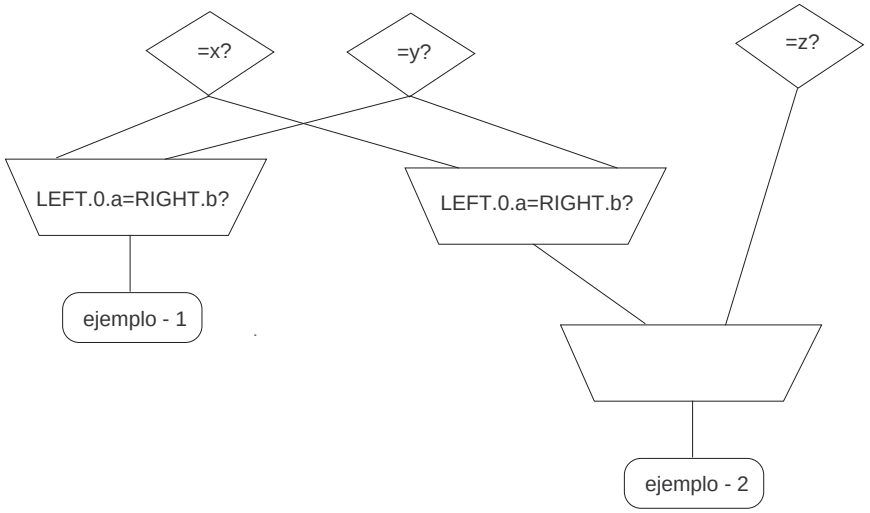

Figura 2.3: Una red RETE que comparte los nodos de patrones.

2. REMOVE. Se usar para efectuar retracciones. Si un *"token REMOVE"* llega a un nodo *"join"*, el nodo examina la memoria apropiada para encontrar un *"token"* con el que haga un *"match"*. Si lo encuentra, borra el *"token"*. Finalmente, si un nodo terminal recibe un *"token REMOVE"*, se localiza el registro de activación correspondiente y se elimina.

3. CLEAR. Permite limpiar la memoria de los nodos *"join"* y terminales.

4. UPDATE. Se usa cuando se ha agregado una nueva regla a una preexistente Red Rete y los nodos *"join"* que pertenezcan a esta nueva regla, tienen que ser poblados con hechos. La etiqueta *"UPDATE"* previene el almacenamiento de nodos duplicados en la memoria.

Optimizaciones del Algoritmo *RETE*

La primera optimización consiste en compartir nodos en la red de ancho de todo el tope a pesar de que sólo hay tres distintos. La red se puede modificar para compartir estos nodo entre las dos

Sin embargo, no son todas las redundancias en la red original. Observando la Figura 2.3 se puede observar que un nodo *"join"* está ejecutando exactamente la misma función. En los pares x, y de ambas reglas, como

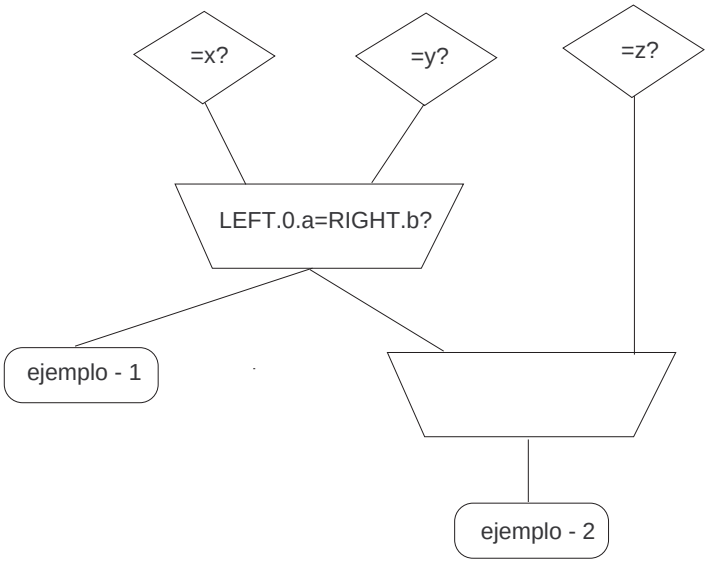

Figura 2.4: Una red RETE que comparte nodos de patrones y nodos *join*.

se ilustra en la Figura 2.4, el hecho de compartir los nodos *"join"* es una fructífera optimización. Puesto que la unión implica compartir hechos de ambos, las evaluaciones tienden a ejecutarse más veces que las evaluaciones de los nodos de los patrones.

2.4.2 Jess en el Web

En nuestros días, los Sistemas Expertos basados en Web, se están utilizando ampliamente como aplicaciones basadas en Web. La máquina de reglas de Java, puede ejecutarse en los clientes como un *"applet"*, o más comúnmente en un servidor (en un servidor de aplicaciones en J2EE).

Arquitecturas de Java para Web

Existen diversas maneras para el desarrollo de aplicaciones en el Web [13]. Para nuestros propósitos, el software basado en Web puede dividirse en dos amplias categorías:

[13]El término *Web* se refiere a aplicaciones que usan los protocolos HTTP y HTTPS.

1. Aplicaciones *"fat-client"*. Son más similares a una aplicación cliente servidor tradicional. El código se divide equitativamente entre la máquina cliente y el servidor. Las soluciones *"fat-client"* automáticamente descargan e instalan el software cliente en la máquina *"desktop"* o puede usar clientes fijos instalados mediante medios físicos o redes con descargas manuales. El software cliente puede poseer una elaborada interfaz gráfica, así como alguna fracción de la lógica del negocio de la aplicación. En Java existen dos maneras de construir soluciones *"fat-client"*: mediante un *"applet"* o usando un *"Java Web Start"*. Esta arquitectura posee las siguientes ventajas y desventajas:

   Ventajas:

   - Se requiere poca capacidad de procesamiento en el servidor porque el código se ejecuta en el cliente.
   - El programador tiene máximo control sobre la interfaz del usuario dado que puede escribirse usando todas las APIs de Java.

   Desventajas:

   - Resulta difícil escribir software complejo en el cliente que se ejecutará en un amplio rango de ambientes de escritorio.
   - Descargar el software del cliente puede ser intolerablemente lento para muchos usuarios.
   - La actualización del software se convierte en una tarea difícil para los usuarios.

2. Aplicaciones *"thin-client"*. En esta arquitectura, la mayoría del código específico de la aplicación se ejecuta en el servidor. Existen diversas maneras de escribir aplicaciones *"thin-client"* en Java. A continuación se describen las más importantes:

   "Servlets". Análogos a los *"applets"*. Son pequeños módulos que son invocados por un servidor Web u otro contenedor en respuesta a las solicitudes del navegador. Un servlet es una implementación de la interfaz javax.servlet.Servlet o más comúnmente una subclase de la clase javax.servlet.http.HttpServlet. Los *"servlets"* se ejecutan en un *"contenedor"* en un servidor. *Java 2 Enterprise Edition (J2EE)* es un estándar para el desarrollo de aplicaciones en un servidor, que incluye entre otras cosas, un contenedor de *"servlets"*.

"Java Server Pages (JSP)". Un JSP es una página HTML que contiene código de Java embebido. Un *servlet* es código Java que frecuentemente incluye sentencias que imprimen HTML. Los JSP's son compilados por un programa especial en el servidor web, generalmente en *"servlets"* y luego son ejecutados en respuesta a las solicitudes del navegador. Mientras que los *"servlets"* resultan ideales cuando un componente del lado del servidor necesita realizar una cantidad no trivial de trabajo y producir únicamente una pequeña cantidad de código HTML, los JSP's son perfectos cuando una gran página de código HTML necesita incorporar una pequeña cantidad de información procesada. Frecuentemente, los *Servlets* y los *JSP's* se usan juntos, el *JSP* suministra la interfaz del usuario y los *servlets* proveen la lógica.

"Web Services". Los *web services* están diseñados para proporcionar una interfaz que otro software puede usar. El término *web service* generalmente se refiere a una aplicación que puede enviar comandos usando XML basado en mensajes de SOAP (*Simple Object Access Protocol*). Los *"Web Services"* se pueden usar para construir aplicaciones basadas en web con interfaces regulares pero también se pueden usar como componentes de vastos sistemas de software. El desarrollo de una aplicación como *"web service"* permite la interacción de gente y compañías alrededor del mundo. Por ejemplo, si la compañía A desarrolla una aplicación de configuración de pedidos como un *"web service"*, luego el departamento de compras de la compañía B puede desarrollar software para que automáticamente compre los productos de la compañía A. JavaSoft ofrece el *Java Web Services Development Pack (JWSDP)* y lo agrega al J2EE como una plataforma para el desarrollo de *web services*.

Las arquitecturas *"thin client"* poseen las siguientes ventajas y desventajas:

Ventajas:

- Se requiere poco procesamiento por parte del cliente, dado que el código se ejecuta en el servidor.
- Únicamente se necesitan unos cuantos requerimientos en el cliente, por lo que la compatibilidad no resulta un problema.
- Las actualizaciones se efectúan fácilmente, puesto que tienen lugar en el servidor.

Desventajas:

- Se requieren servidores robustos.

2.5 Protégé

Protégé es el resultado de varios proyectos de inteligencia artificial y de modelado del conocimiento del grupo de Informática Médica de la Universidad de Stanford [28].

Protégé es una de las herramientas *"open source"* más exitosas para el modelado del conocimiento.

Mediante el uso de Protégé, los ingenieros del conocimiento y los expertos del dominio construyen modelos conceptuales y bases de conocimiento para accederlos a través de un API de Java. Los modelos resultantes pueden usarse para la implementación de sistemas de apoyo a la toma de decisiones, captura de requerimientos de software, bases de datos, generación de clases de Java y diagramas UML , acceso a la semántica web y para compartir y reusar modelos del dominio.

En la fase inicial de desarrollo de un sistema experto, es necesario obtener el conocimiento de los expertos y capturarlo en alguna clase de modelo del dominio.

Protégé se puede utilizar para el modelado de clases, edición de instancias y procesamiento e intercambio de modelos:

- Modelado de clases. Protégé posee una interfaz gráfica que modela clases (conceptos del dominio) y sus atributos y relaciones.

- Edición de instancias. De estas clases, Protégé automáticamente genera formas interactivas que permiten a los ingenieros del conocimiento o a los propios expertos validar las instancias.

- Procesamiento del modelo. Protégé posee una librería que ayuda a definir las semánticas, formular preguntas y definir el comportamiento lógico.

- Intercambio de modelos. Los modelos resultantes (clases e instancias) se pueden cargar y almacenar en varios formatos, incluyendo XML, UML y RDF.

Desde la perspectiva del programador, una de las características más atractivas de Protégé, es la existencia de un API *open source* que permite conectarse con componentes de Java y acceder a los modelos del dominio de la aplicación. En consecuencia, se pueden desarrollar sistemas rápidamente.

2.6 JessTab

JessTab es un *plug in* para Protégé que permite el uso conjunto de Jess y Protégé [21]. En la Figura 2.5 se muestra la historia de JessTab.

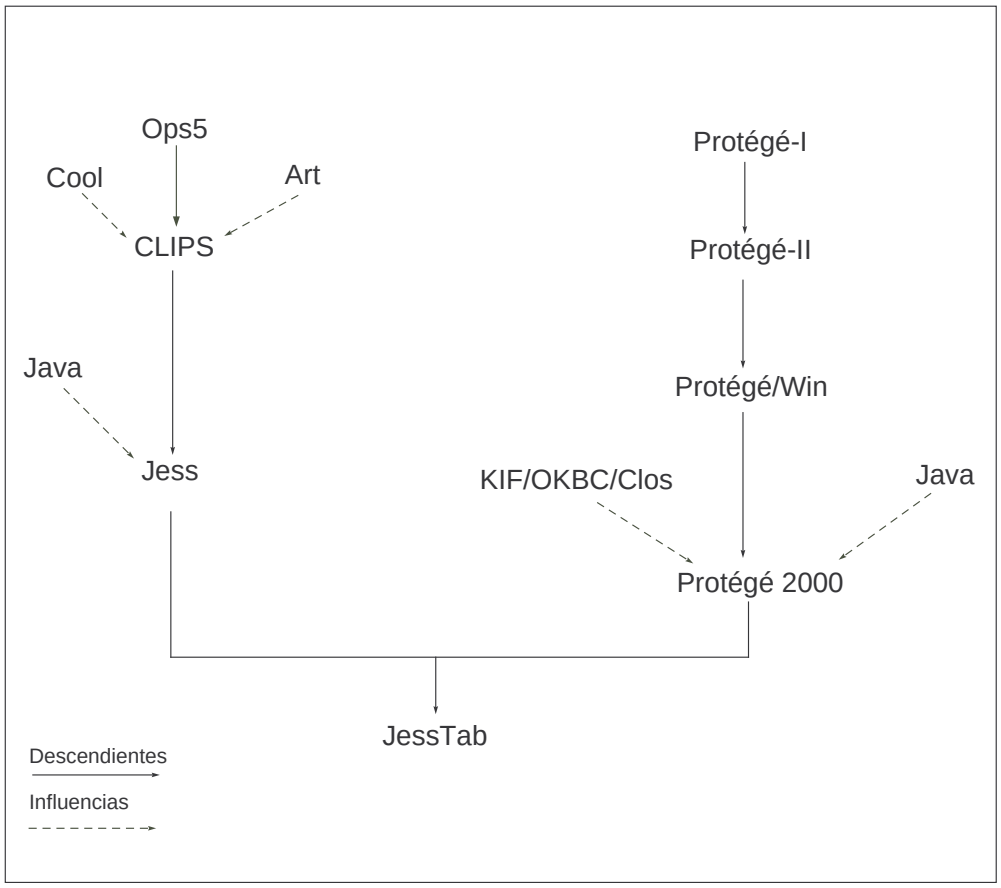

Figura 2.5: Historia de JessTab

Como se ilustra en la figura anterior, JessTab es el puente entre Protégé-

2000 y Jess. JessTab integra una consola de Jess en Protégé y un conjunto de extensiones para Jess que permiten mapear las bases de conocimiento a hechos de Jess y el manejo de las bases de conocimiento de Protégé. Mediante el uso de JessTab es posible crear programas que usen las bases de conocimiento de Protégé. Por ejemplo, los patrones de reglas de Jess se pueden aparear con instancias de Protégé. Dado que está basado en Java, es posible implementar sistemas en Jess, Java y Protégé. La mayoría de las funciones de Protégé son distintas de las correspondientes funciones de Java para evitar colisiones.

La integración Protégé–Jess permite desarrollar las bases de conocimiento en Protégé y ejecutar el solucionador de problemas en Jess usando las bases de conocimiento de Protégé para ejecutar sus tareas. Jess puede manipular las bases de conocimiento de Protégé (por ejemplo, instanciación de clases y cambio de valores de los *slots*). Las reglas de Jess pueden aparearse con las instancias de las bases de conocimiento de Protégé durante la adquisición del conocimiento y ejecutar acciones tales como la modificación de la base de conocimiento.

Mediante JessTab es posible usar Jess como elemento de ejecución para métodos reusables de solución de problemas que aprovechen las bases de conocimiento de Protégé. El código de Jess puede implementar los mapeos requeridos para integrar ontologías del dominio. En resumen, Jess actúa como la contraparte de ejecución de Protégé. Mientras tanto, Protégé puede actuar como la contraparte de la base de conocimiento gráfica de Jess.

JessTab brinda editores para los constructores de Jess, tales como reglas y funciones. Dichos editores se pueden usar para dar un vistazo y redefinir las definiciones de Jess. Adicionamente, se puede instruir a JessTab para almacenar definiciones de Jess con la base de conocimiento de Protégé. Alternativamente se puede almacenar un programa de Jess en un archivo por separado y se puede cargar como un archivo de inicio cuando se cargue la base de conocimiento.

2.7 Tecnología de Agentes

El paradigma de agentes ha sido objeto de estudio en los últimos años, destacando el desarrollo de aplicaciones en diferentes dominios tales como: mercados electrónicos, e-turismo, aplicaciones web, ambientes inteligentes y en análisis de sistemas complejos.

Actualmente, la tecnología de agentes, particularmente los Sistemas Multiagentes (*Multi-Agent System -MAS-*) desempeñan un rol importante dentro del desarrollo de software y en la industria. Hasta hace algunos años, la investigación de la comunidad de sistemas multiagentes se centraba en el desarrollo de conceptos, arquitecturas, técnicas de interacción y enfoques generales para el análisis y especificación de sistemas multiagentes.

En la actualidad el trabajo sobre multiagentes se ha incrementado notablemente. Cientos de grupos tanto de universidades como de laboratorios industriales están trabajando en proyectos puramente académicos como prácticos. Los primeros sistemas de aplicación que utilizan este tipo de técnicas ya están en operación.

En el contexto de los Sistemas Multiagentes, una vertiente de desarrollo e investigación consiste en el uso de las metodologías de los Sistemas Multiagentes para brindar asistencia a los usuarios en el proceso de minería de datos. El modelo *Belief-Desire-Intention -BDI-* ofrece un amplio nivel de abstracción dentro de este ámbito. El proceso de minería de datos es un proceso complejo que requiere usuarios capacitados en aprendizaje automático, estadísticas y en el dominio de la aplicación. Los *MAS* están compuestos por agentes especializados en minería de datos. Sin embargo, en algunos enfoques se ha observado que no poseen el nivel de abstracción requerido en el razonamiento del proceso de minería de datos y sus participantes. Una agencia conceptualiza e implementa a los agentes como sistemas intencionales tal y como lo efectúa el modelo *BDI*. En el enfoque *BDI*, los agentes ejecutan un razonamiento práctico usando planes para formar intenciones para satisfacer sus deseos. Este modelo brinda una comunicación más efectiva entre los participantes dentro del proceso de minería de datos.

Mediante la a metodología denominada *Prometheus*, se puede elaborar el diseño de un *MAS* formado por agentes racionales que ayuden a los expertos humanos en el proceso de minería de datos. El *Prometheus Design Tool* se usa para obtener asistencia en el proceso de diseño y para generar una especificación de un *MAS* de fácil implementación [32].

Capítulo 3

Descripción del Problema

3.1 Sistema de Educación Distribuida "EMINUS"

Dentro del marco de la sociedad del conocimiento que persigue la generación y distribución del mismo, mediante la colaboración entre individuos, se persigue finalizar con la individualidad y con el aislamiento de conocimientos.

El presente que habitamos está conformado por la dinámica de las sociedades postindustriales, en las que predomina la generación, aplicación y distribución social del conocimiento, lo que se consigue gracias a la interacción eficaz de las políticas de investigación, desarrollo e innovación y distribución, generados a través de las tecnologías del entorno virtual (telecomunicaciones y redes informáticas) y de la educación de las sociedades en una alfabetización científico tecnológica crítica.

En el contexto de esta sociedad, a partir del uso de las tecnologías de la información, es posible hacer extensivo este modelo a organizaciones que se configuran en modo de red, por lo cual, la horizontalidad de los procesos de generación de conocimiento, la heterogeneidad de los nodos y la riqueza simbólico-conceptual emanada de su vocación interdisciplinaria, multiplican la eficacia de todo este ciclo.

Dado el avance tecnológico, las instituciones enfrentan el reto de proveer nuevas y mejores oportunidades en la educación. En respuesta a tal demanda, se están desarrollando ambientes de aprendizaje basados en las Tecnologías de Información y la Comunicación (TIC's) y fundamentados en la Tecnología Educativa (TE).

Actualmente, la *Universidad Veracruzana* fortalece las estrategias en la adquisición de competencias para la formación de por vida al desarrollar un sistema que permite organizar, aplicar e integrar diferentes ambientes flexibles de aprendizaje.

Figura 3.1: Interfaz de Eminus

El Sistema de Educación Distribuida *Eminus*, permite organizar, aplicar e integrar diferentes ambientes flexibles de aprendizaje para ampliar la cobertura de educación y facilitar con el apoyo de las TIC's y la TE, los procesos de Enseñanza-Aprendizaje, comunicación y colaboración, para la formación integral de los seres humanos. La interfaz de Eminus se presenta en la Figura 3.1

El Sistema *Eminus* es un Sistema de Administración de Ambientes Flexibles de Aprendizaje, el cual sirve para presentar cursos en línea para distribuirse en Internet o redes internas. Permite la comunicación en forma sincrónica y asincrónica ya que utiliza la Tecnología de la Información y la Comunicación para aprovechar la facilidad de distribución de materiales formativos y herramientas de comunicación. Esto permite crear un entorno completo para el aprendizaje ayudando a la vez a mejorar los niveles educativos sin límites de tiempo y de distancia, permitiendo a cada estudiante tomar el control de su aprendizaje y formación de manera independiente y colaborativa. Con este sistema se redefine la docencia de manera más pla-

centera, útil y eficiente con énfasis en la comunicación, la colaboración y la distribución de materiales de enseñanza y aprendizaje.

3.1.1 Características tecnológicas

El sistema Eminus cuenta con diferentes características que le permiten ser una herramienta versátil para la organización de la información, la comunicación y la promoción del aprendizaje, mediante herramientas tecnológicas, de comunicación y de facilitación de materiales tanto para el docente como para el estudiante.

Su estructura se organiza en los siguientes módulos:

- Administración de usuarios. Aquí se pueden determinar distintos tipos de usuarios como: administrador, maestro, asistente, alumno, se asignan usuarios a los cursos y se realiza el seguimiento de los alumnos.
- Administración de cursos. Este módulo presenta categorías por áreas, cursos, módulos (unidades) y materiales y se organizan los materiales, bibliografías, actividades por curso, formas de evaluación, etc.
- Catálogos. En este módulo se contemplan todos los cursos, maestros, alumnos, descripciones de los cursos, etc.

3.1.2 Características de comunicación y colaboración

Eminus contiene la tecnología más avanzada en ambientes electrónicos para el trabajo colaborativo en línea, y cuenta con aplicaciones de administración y comunicación como:

- Salón Virtual de clases con sesiones en línea.
- Salón de tutorías en línea.
- Videoconferencias.
- Salas de chat.
- Foro.
- E-mail.

- Pizarra electrónica.
- Soporta material electrónico textual, gráfico y multimedia para su uso y consulta en múltiples formatos.
- Contiene herramientas de organización como calendarios y avisos.
- Cuenta con herramientas básicas como calculadora.

Todas sus herramientas de organización y comunicación promueven actividades centradas en el alumno, en la tarea o en el grupo. El facilitador puede planificar las actividades y organizar los materiales de acuerdo a las características de aprendizaje de su población objetivo.

3.1.3 Características pedagógicas

Eminus permite desarrollar el aprendizaje colaborativo y significativo, ya que promueve la interacción social dentro y fuera del aula y acerca al estudiante a sus compañeros y a los diferentes contenidos del conocimiento.

Por ser un sistema de soporte de ambientes flexibles de aprendizaje, se adapta a las condiciones, necesidades y características del sistema educativo donde se aplique. El sistema EMINUS puede integrarse en diferentes modalidades educativas como:

Presencial. En Modalidad Presencial se puede utilizar para facilitar los contenidos, promover el análisis, facilitar la comunicación, etc. Funciona como un apoyo tanto para docentes como para alumnos para dar un seguimiento estructurado al curso.

Aprendizaje distribuido. En la Modalidad de Aprendizaje Distribuido, funciona como una excelente herramienta para las sesiones en línea y como apoyo para las sesiones presenciales. Permite que el alumno ingrese al campus en línea las veces que lo desee, accede a los contenidos o interactúa con sus compañeros desde cualquier lugar que se encuentre y así en las sesiones presenciales se enfoca a procesos de aplicación de conocimientos y no solamente de recepción de los mismos.

Educación a Distancia. En Educación a Distancia es una plataforma ideal para la operatividad de cualquier programa académico. Sus herramientas de organización y comunicación permiten que tanto académicos como alumnos se encuentres comunicados de forma sincrónica y asincrónica. De igual forma los lugares de tutorías y debates permiten que la distancia transaccional disminuya considerablemente permitiendo al usuario aprender eficazmente.

Educación Continua. En Educación Continua se presenta como una herramienta fundamental para proporcionar a los estudiantes las oportunidades de acceder a los contenidos del curso las veces que lo considere necesario.

Educación Abierta. En Educación Abierta se promueve la comunicación, el acceso, la interacción y el acercamiento del alumno al conocimiento, ya que al estar en línea puede acceder al ambiente de aprendizaje de acuerdo a sus necesidades. Esto proporciona a todo el sistema educativo una flexibilidad y bajos costos para la educación de las personas fundamentada en la adquisición de competencias a un ritmo personalizado.

3.1.4 Definición del Problema

Los sistemas de administración de ambientes flexibles de aprendizaje no sólo proporcionan nuevas oportunidades para aprender a distancia, en colaboración y durante toda la vida, también llevan consigo un conjunto de acciones y carencias que pueden frustrar o desmotivar al estudiante en línea [36]. Esta frustración puede afectar negativamente al aprendizaje del estudiante o incluso motivar su abandono y, además, puede repercutir en el docente. Algunos de los principales elementos o acciones que originan frustración, desilusión o agobio en el estudiante en línea son los siguientes [23]:

- Desconocimiento de los canales de ayuda. Una carencia típica del estudiante en línea consiste en no saber dónde o cómo pedir ayuda, ya sea en relación con asuntos administrativos, de contenidos o del funcionamiento de su formación. Este desconocimiento puede derivar en no poder resolver problemas a tiempo. Por lo tanto, el estudiante debe saber desde el principio qué canales de ayuda tiene a su alcance, dónde se encuentran y cómo ha de utilizarlos.

- Ayuda técnica deficiente. Esta carencia es una constante en la literatura sobre motivación y frustración: las dificultades técnicas son un elemento clave en la frustración y la desmotivación del estudiante en línea, y en muchas ocasiones se convierten en un obstáculo insalvable. Se observa que un buen servicio de ayuda informática es fundamental para la buena marcha del aprendizaje y la formación. Cualquier estudiante y docente saben que las dificultades técnicas influyen poderosamente tanto en el grado de aprendizaje como en la satisfacción.

Por consiguiente, es muy importante dedicar presupuesto y personal para proporcionar un eficiente servicio de ayuda técnica al estudiante, estableciendo los canales adecuados -y dándolos a conocer- de petición de ayuda y de resolución de problemas.

En respuesta a tal demanda, la Universidad Veracruzana propone el desarrollo de un Centro de Ayuda para el sistema de administración de ambientes flexibles de aprendizaje: Eminus.

En Eminus, resulta crucial y de suma importancia el soporte técnico para los usuarios del sistema (estudiantes y facilitadores). Por lo que se requiere un Centro de Ayuda donde se brinde asesoría técnica y solución a los problemas oportunamente durante las 24 horas del día con la calidad técnica de expertos de alto nivel. Para proveer un servicio de alta calidad, la disponibilidad de un alto nivel de expertos es de gran importancia. Sin embargo, el número y la disponibilidad de expertos de alto nivel es limitado y la necesidad de un Centro de Ayuda se incrementa día a día. La metodología de los Sistemas Expertos basados en Web constituyen una viable solución.

El objetivo de este trabajo radica en desarrollar un *Sistema experto basado en Web*, que brinde a los usuarios una asistencia rápida, oportuna y de calidad, a usuarios con poco conocimiento del dominio o usuarios avanzados que requieren información más especializada.

Capítulo 4

Desarrollo del Sistema Experto

4.1 Introducción

Los sistemas de ayuda en línea son herramientas que resultan útiles para auxiliar a los usuarios a resolver dudas o problemas de carácter técnico que se le plantean en el uso de algún servicio o sistema. La complejidad de un sistema de este tipo depende de la complejidad del servicio o sistema al cual apoya, cuando es muy complejo las variantes en cuanto a causas de fallas se vuelven inmensas que sólo es posible implementar un sistema eficaz utilizando técnicas de inteligencia artificial (reglas de producción, lógica, razonamiento basado en casos, redes neuronales, etc.).

En este trabajo, se presenta un sistema de ayuda para un proceso complejo. Dicho sistema será implementado en EMINUS que actualmente es el Sistema de Administración de Ambientes Flexibles de Aprendizaje de la Universidad Veracruzana. En este sistema existen dos grandes niveles de funcionamiento que requieren para sus usuarios: contar con un sistema de ayuda, uno, a nivel educativo en la creación, administración y acceso de cursos y el otro, en la parte técnica de utilización de la red, los recursos y servicios informáticos que ofrece la plataforma, el trabajo que nos ocupa se centra en brindar una solución al primer aspecto. De esta manera, se garantiza el éxito en el uso de la plataforma evitando la frustración y desmotivación del estudiante en línea. En este trabajo se presentará un sistema de ayuda para la plataforma EMINUS que utiliza un sistema experto basado en los sistemas *open source*, JESS (Java Expert System Shell) como herramienta para la construcción del sistema experto, Protégé como editor de ontologías

y de la base de conocimiento y JessTab como herramienta para conjuntar JESS y Protégé. Hay que mencionar que el sistema de ayuda realizado, es general y podría ser utilizado para otros campos de aplicación. En el caso de EMINUS se hizo una interfaz que se integra a la plataforma.

4.2 Metodología de desarrollo

Los sistemas expertos se construyen de manera progresiva, haciendo correcciones y efectuando adiciones a la base de conocimiento. Existe una metodología ampliamente utilizada para estos fines denominada: "prototipos sucesivos", la cual se utilizó en el desarrollo de este proyecto. Esta técnica permite la construcción rápida de modelos de trabajo o emulaciones de los componentes importantes del sistema, donde los usuarios emiten sus opiniones del uso del sistema, dando lugar a los refinamientos.

Cuando se considera que se ha obtenido una visión general del problema, se procede a modelar conceptualmente al sistema: se selecciona un formalismo para la representación del conocimiento, se determina la estrategia de búsqueda y se procede a efectuar el diseño de la interfaz del usuario. A partir del modelo conceptual, se inicia la construcción de un prototipo. Dicho prototipo será capaz de resolver problemas en un área pequeña del dominio. Una vez que se implemente el prototipo, se procederá a efectuar una evaluación para la corrección de las deficiencias del sistema. Después, se procederá al desarrollo de una nueva versión. Este proceso se repite hasta que el sistema tenga un desempeño satisfactorio. Finalmente, en la fase de desarrollo de un sistema experto, el programa nunca debe considerarse como terminado ya que, es muy factible que en cualquier momento se necesiten agregar nuevas reglas o modificarlas.

Las características conceptuales del sistema experto propuesto son: encadenamiento hacia atrás como mecanismo de inferencia y las reglas de producción como formalismo para la representación del conocimiento.

4.2.1 Funcionalidades del sistema experto

A continuación, se presentan las principales funcionalidades del sistema de ayuda inteligente:

- Proporcionar soporte técnico en línea a los usuarios del Sistema EMINUS para garantizar el éxito en el uso de la plataforma evitando la

frustración y desmotivación del estudiante en línea.

- Brindar a los usuarios de EMINUS (alumno, facilitador) asistencia rápida, oportuna y de calidad.
- Mejorar la productividad al ofrecer solución a los problemas de forma rápida y ordenada.
- Permitir la reducción de los costos de soporte técnico.

4.3 Adquisición del conocimiento

Como se ilustra en la Figura 4.1, el desarrollo de un sistema experto es generalmente el resultado de la colaboración de uno o varios expertos humanos especialistas en el tema de estudio y los ingenieros del conocimiento. Los expertos humanos suministran el conocimiento básico en el tema de interés, y los ingenieros del conocimiento trasladan este conocimiento a un lenguaje, que el sistema experto pueda entender. La colaboración de los expertos humanos, los ingenieros del conocimiento y los usuarios es, quizás, el elemento más importante en el desarrollo de un sistema experto. Esta etapa requiere una enorme dedicación y un gran esfuerzo debido a los diferentes lenguajes que hablan las distintas partes y a las diferentes experiencias que tienen.

Los especialistas son responsables de suministrar a los ingenieros del conocimiento una base de conocimiento ordenada y estructurada, y un conjunto de relaciones bien definidas y explicadas. Esta forma estructurada de pensar requiere que los expertos humanos repiensen, reorganicen, y reestructuren la base de conocimiento y, como resultado, el especialista se convierte en un mejor conocedor de su propio campo de especialidad.

En el desarrollo de este trabajo, la fase de adquisición del conocimiento se basó, principalmente, en entrevistas con algunos usuarios del Sistema Eminus y con los desarrolladores del mismo, para conocer los principales problemas reportados por los diferentes tipos de usuarios y la explicación de las técnicas usadas para su solución.

En el proceso de adquisición del conocimiento se identificaron una serie de problemas, los cuales se agruparon de la siguiente forma:

- Logins: Para entrar al sistema, desde la página principal se debe insertar un nombre de usuario y una clave de acceso. Dichos datos se

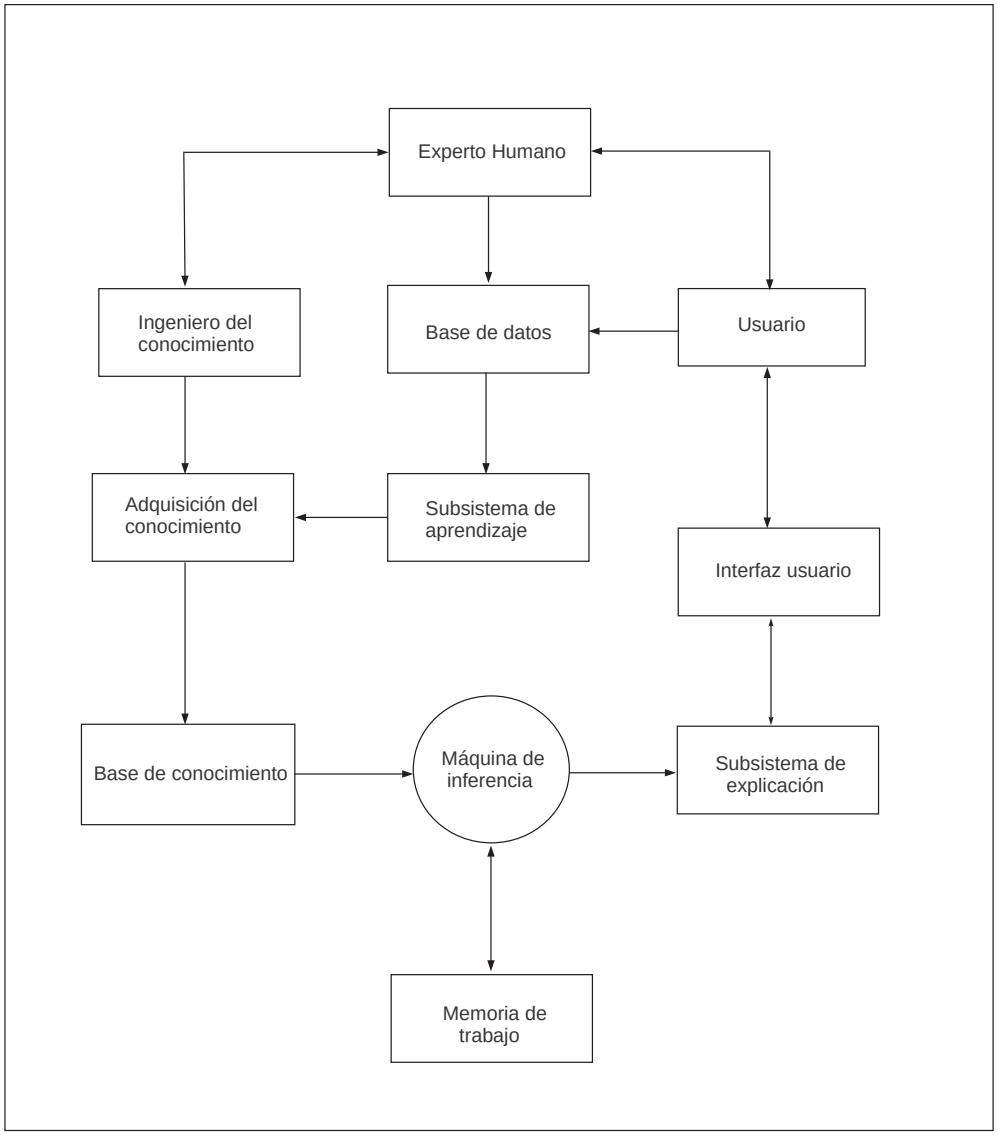

Figura 4.1: Componentes de un sistema experto. Las flechas representan el flujo de información.

almacenan en una cuenta distinta para cada usuario. Después de introducir los datos anteriores se deberá dar clic en el botón Entrar para tener acceso al sistema. Los usuarios pueden cambiar la contraseña en el momento que lo deseen una vez que entraron a Eminus. Los estudiantes y facilitadores presentan problemas cuando se ingresan su login y password al sistema, los cuales se registraron junto con su solución, en el sistema experto de ayuda.

- Cursos: En esta sección, se muestran los cursos a los que el usuario está asignado. Estos aparecen en una tabla con 3 columnas: el nombre del curso, la duración y el perfil. Cada uno de los cursos posee un programa, recursos del curso como son documentos, imágenes, videos, presentaciones (salón de clases) y varios, actividades del curso y sus fuentes de información (bibliografía). En el sistema experto se incluyeron los principales problemas junto con su solución de este rubro.

- Eventos y Anuncios: Aquí se presentan los Eventos y Anuncios de interés para los estudiantes. El sistema experto contiene soluciones para los principales problemas que hasta el momento se han reportado dentro de Eventos y Anuncios.

- Salón Virtual de Clase: Esta herramienta brinda los elementos necesarios para que el facilitador pueda impartir clases a personas ubicadas en diferentes lugares geográficos. Dicha herramienta, emula el ambiente de un salón tradicional de clases donde el facilitador cuenta con un pizarrón electrónico, chat, transmisión de presentaciones, etc. Dada la importancia de dicha herramienta, el sistema experto muestra las soluciones a los principales problemas reportados durante su uso.

- Herramientas de apoyo: Una herramienta es un programa que cumple una función específica y que funge como auxiliar en el aprendizaje a distancia. Dentro de las herramientas con que se cuenta Eminus están: el chat, la pizarra electrónica, la calculadora, los foros de discusión, el chat, el calendario, los comunicados y el salón de asesorías. En la implementación del sistema experto, se incluyeron las soluciones a los principales problemas reportados por los usuarios en el uso de las herramientas.

Se identificaron 3 perfiles de usuario: *Alumno, Facilitador y Administrador*, debido a que un mismo problema puede resolverse de distinta forma

de acuerdo al perfil dentro de la aplicación. El sistema experto, omite los problemas del administrador dado que dicho perfil, está reservado para un pequeño grupo de usuarios.

A continuación, se enumeran algunos de los problemas comunes de los usuarios de Eminus junto con su solución:

1. *Los módulos no aparecen ordenados de acuerdo al orden de captura*
   Solución: No es posible visualizar los módulos en el orden en que fueron capturados debido a que en Eminus se ordenan alfabéticamente por nombre, por ello, en algunas ocasiones no siguen el mismo orden con el que fueron dados de alta.

2. *En una sesión del Salón virtual de clases se desea mandar una página web pero aparece: "La página no ha sido encontrada"*
   Solución: Cuando se escribe la dirección de una página web en el Salón virtual de clases, se le debe anteponer http:// para que se pueda desplegar en Eminus.

3. *Existe una tarea en Eminus pero no la pueden ver todos los estudiantes del curso.*

   Solución: Al momento de enviar una tarea a Eminus se debe indicar el tipo al que pertenece:

   (a) Pública: La pueden ver todos los usuarios miembros de un curso independientemente del nivel de acceso que tengan.

   (b) Privada: Únicamente la van a poder ver el facilitador y el propietario de la tarea.

4. *Al entrar herramienta de salón de asesorías, después de elegir el curso al que pertenece el alumno, desaparece la pantalla principal y no se puede entrar.*

   Solución: Se debe verificar que en la computadora del usuario no se encuentre habilitado un bloqueador de "Popups" o en su defecto si existe, se debe configurar para que permita el uso de pantallas emergentes de Eminus.

5. *En la lista de usuarios de un curso no aparece un alumno.*

Solución: Se debe a que el usuario no fue asignado al curso, el único que puede asignar usuarios a un curso es el Administrador. En ese caso, el facilitador debe ponerse en contacto con el administrador para que se haga la corrección pertinente.

4.3.1 Uso de Protégé

El entorno de desarrollo Protégé (Gennari, 2002) se usó como editor de ontologías y para modelar la base de conocimiento (Devedzic, 2002). Protégé 2000 es una herramienta de desarrollo creada por un grupo de trabajo de la Universidad de Stanford. Está basado en clases y sus atributos, lo cual permite una sencilla organización y modificación del conocimiento mediante un lenguaje visual apropiado para todos los usuarios, adicionalmente, permite interactuar con Java.

Para el desarrollo del presente trabajo, se hizo uso de Protége para facilitar el proceso de modelado de la base del conocimiento. La estructura del conocimiento modelada en Protége, facilita el mantenimiento de la base del conocimiento. El ingeniero del conocimiento puede agregar, eliminar y modificar el conocimiento de una manera muy sencilla.

La fase de adquisición del conocimiento estuvo apoyada por Protége. En la Figura 4.2, se muestra la estructura del nodo principal que se utilizó para el almacenamiento del conocimiento.

La clase *root* se definió para que fuera compatible mediante JessTab, con la regla base que fue desarrollada en JESS. Dicha clase está compuesta por los siguientes *slots*:

1. *Answer*. Se usa para almacenar la respuesta del usuario del sistema experto.

2. *Question*. Almacena las preguntas que se le hacen al usuario del sistema experto.

3. *n1, n2, n3, n4, n5 y n6*. Contienen las referencias hacia otras clases.

En la Figura 4.3, se muestra una instancia de la clase *root*, clase inicial del sistema, donde se determina el perfil del usuario del sistema Eminus (administrador, alumno y maestro o facilitador). Cada uno de los usuarios está representado por una clase que a su vez, está formado por otro conjunto

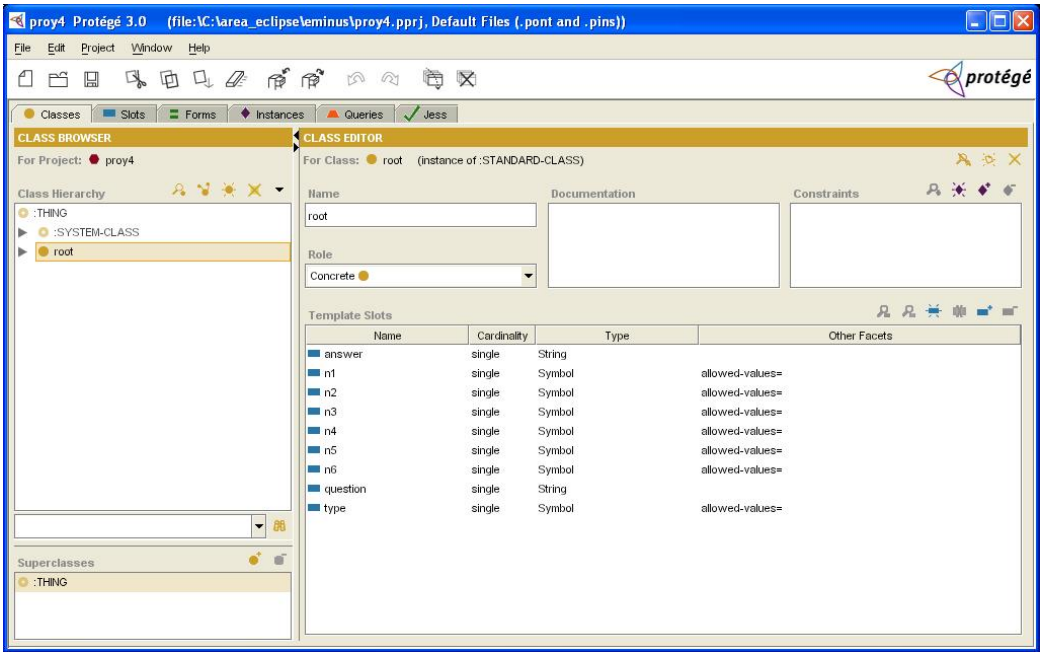

Figura 4.2: Protége

de subclases que almacenan el conocimiento necesario para ayudar al usuario a encontrar una solución a un problema determinado.

La Figura 4.4 muestra la estructura de la base del conocimiento del sistema experto en Protége. Se crearon clases para los principales problemas detectados en el uso de Eminus: *logins*, cursos, materiales, eventos y herramientas.

Es importante destacar que la base de conocimiento del sistema se irá incrementando en la medida en que se detecten más problemas. La arquitectura del sistema experto, permitirá capturar nuevos problemas de una manera sencilla y rápida mediante el uso de Protége.

4.4 Desarrollo del sistema de ayuda

En el desarrollo del Sistema Experto basado en Web se hizo uso de la metodología de los prototipos sucesivos y de los sistemas *open source*. Es importante destacar que el sistema experto es independiente del dominio de aplicación, es decir que se puede usar para implementar otros sistemas

Figura 4.3: Ejemplo de instancia de una clase en Prótegé.

expertos.

La arquitectura del sistema experto para el Apoyo en la Solución de Problemas de Eminus, se presenta en la Figura 4.5.

Como se ilustra en dicha figura, el sistema experto fue implementado en una arquitectura cliente-servidor haciendo uso del lenguaje de programación java [20].

El componente inteligente del centro de ayuda está basado en JESS (Java Expert System Shell). Jess es una variante de CLIPS -C Language Inte-

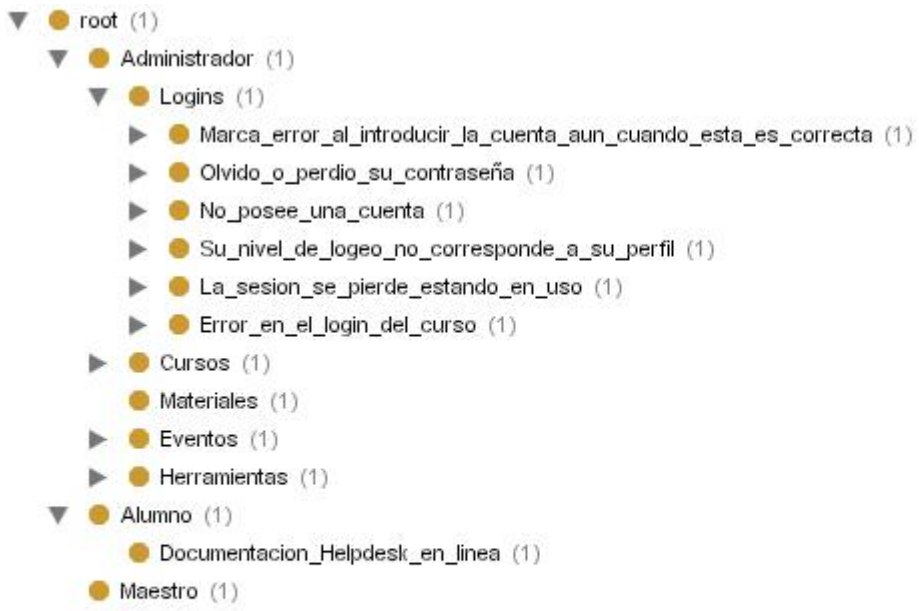

Figura 4.4: Estructura de la base de conocimiento en Prótegé.

grated Production System [29] desarrollado en la NASA y se utiliza como herramienta para el desarrollo de sistemas inteligentes en Internet [64].

Jess, brindó los elementos básicos para la implementación del Sistema Experto para el Apoyo en la Solución de Problemas en un Sistema de Administración de Ambientes Flexibles de Aprendizaje: la base de conocimiento, para almacenar el conocimiento con reglas, la memoria de trabajo, que contiene los hechos representando conclusiones, hipótesis y metas, y la máquina de inferencia (razonamiento hacia atrás).

Para la integración de Protégé y JESS se usó JessTab [19], [18]. Mediante este "plug-in", es posible desarrollar gráficamente las bases de conocimiento en Protégé [48] y Jess puede efectuar razonamientos sobre dichas bases de conocimiento.

En las siguientes secciones se describirá a detalle el uso de JessTab y Jess

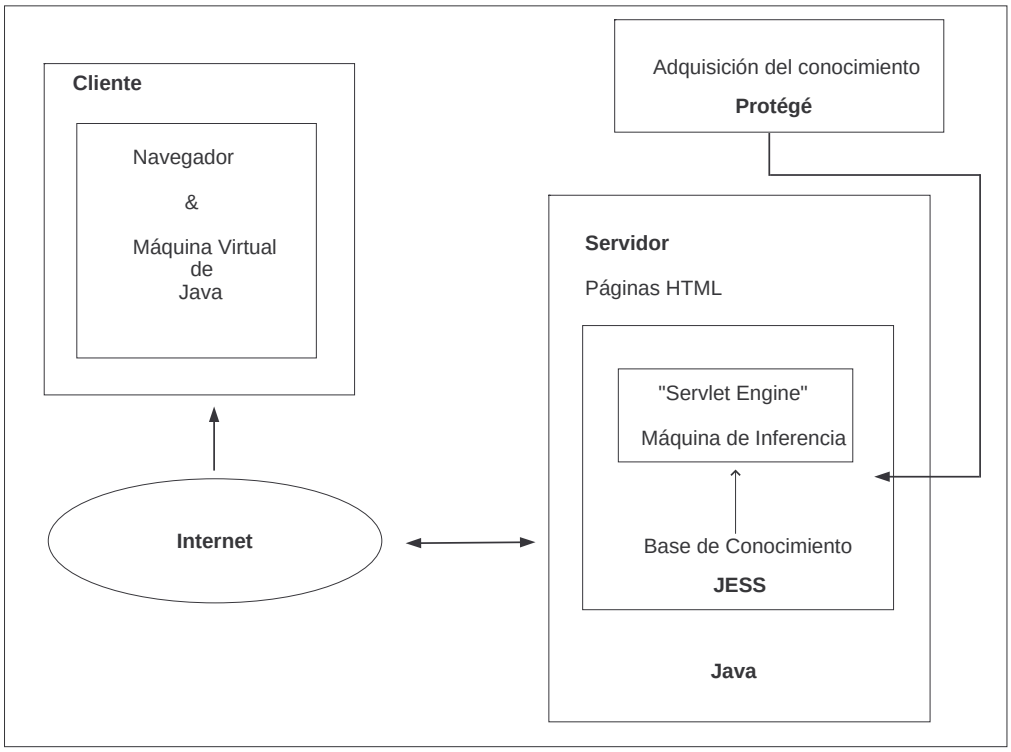

Figura 4.5: Arquitectura del Sistema Experto

para el desarollo del sistema experto.

4.4.1 JessTab

En el desarrollo de este proyecto, fue posible hacer uso de las bases de conocimiento creadas en Prótegé desde JESS mediante el uso de JessTab. A través de una consola en la interfaz de Protégé, se efectuó el mapeo de la base de conocimiento a hechos de Jess (consulte la Figura 4.6).

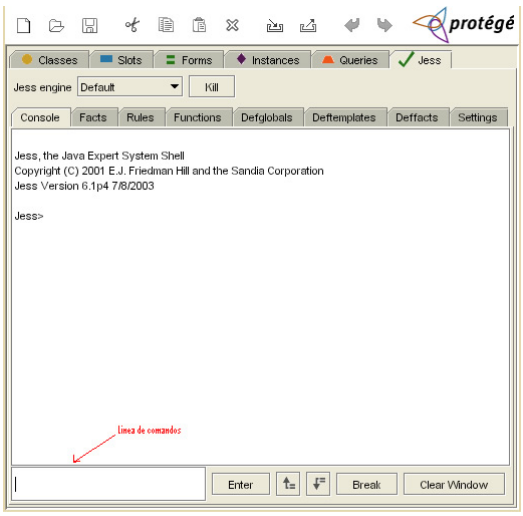

Figura 4.6: JessTab

Figura 4.7: Pestaña de Jess en Protégé

Después de la creación de todas las clases de la base de conocimiento con sus respectivas instancias, en la interfaz de Protégé se selecciona la pestaña denominada *Jess* (véase la Figura 4.7), en caso de no estar activa, se elige el menú *Project* y en la opción *Configure*, aparece una tabla con dos pestañas. *TabWidgets* se coloca visible y se habilita *JessTab*. En la consola de JessTab

se pasan las instancias de Protégé a hechos de Jess mediante el comando *(mapclass nodo)*. La Figura 4.8 ilustra la lista de hechos con todas las instancias que se introdujeron desde la interfaz de Protégé.

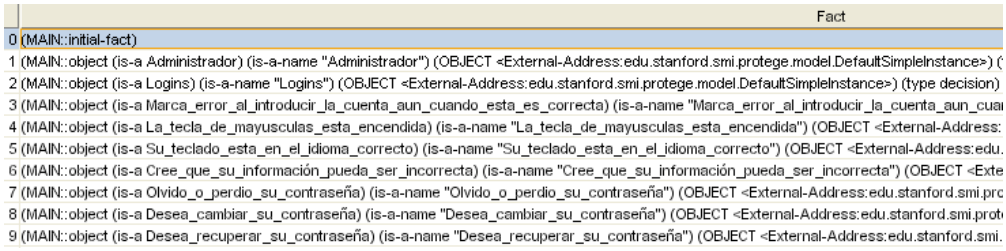

Figura 4.8: Base de conocimiento en JessTab

Para la compatibilidad de la base de conocimiento con la regla base que se definió en JESS, fue necesario definir un *template* desde la consola de Jess en Protégé con la misma estructura de la clase *root* que se usó en la fase de adquisición del conocimiento. A continuación, se muestra la definición del *template*:

```
deftemplate node (slot name) (slot type) (slot question) (slot n1) (slot n2)
                 (slot n3) (slot n4) (slot n5) (slot n6) (slot answer))
```

El siguiente paso, consistió en la definición de una regla para la conversión de las instancias que fueron generadas por el *mapclass*, al formato del *template* usado en la regla base definida en Jess, como se muestra en la Figura 4.9.

```
(defrule MAIN::replace
  (MAIN::object (is-a ?isa) (is-a-name ?isan) (OBJECT ?obj) (type ?type)
   (question ?question) (n1 ?n1) (n2 ?n2) (n3 ?n3) (n4 ?n4) (n5 ?n5) (n6 ?n6) (answer ?ans))
   =>
   (assert (MAIN::node (name ?isa) (type ?type) (question ?question)
   (n1 ?n1) (n2 ?n2) (n3 ?n3) (n4 ?n4) (n5 ?n5) (n6 ?n6) (answer ?ans))))
```

Figura 4.9: Definición del *template* de la regla base

Después, se obtuvieron los hechos en el formato deseado, como se muestra en la Figura 4.10.

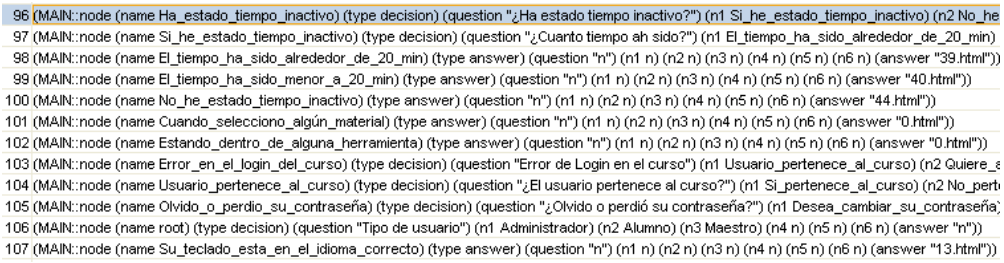

Figura 4.10: Base del conocimiento con la estructura final

Las soluciones de los principales problemas de Eminus, se almacenaron en un archivo con extensión *.html* dentro de la base de conocimiento. En la Figura 4.11, se muestra el almacenamiento de la solución a un problema de visualización de contenidos de un curso o experiencia educativa en el archivo *direccionweb.html*.

```
(MAIN::node (name Pagina_web_no_encontrada) (type answer) (question "n") (n1 n) (n2 n)
(n3 n) (n4 n) (n5 n) (n6 n) (answer "direccionweb.html"))
```

Figura 4.11: Integración del archivo de la solución del problema en la base del conocimiento

La base de conocimiento se almacenó en un archivo con extensión *.dat*, mediante la siguiente instrucción:

$$(save - facts \quad c : /areaeclipse/eminus.dat \quad MAIN :: node)$$

Finalmente, se hizo uso del comando *batch* desde la consola de Jess dentro de Protégé para ejecutar y almacenar los datos en un archivo:

$$(batch \quad c : /ruta/reglas - eminus.txt)$$

4.4.2 JESS

El sistema experto fue desarrollado en Java como una aplicación *web* mediante el uso de *Servlets* y *Java Server Pages (JSP)* para el desarrollo de la interfaz. En este caso, Jess se usa como una librería incorporada en el contenedor de *servlets* Apache Tomcat.

Para el desarrollo del proyecto, se inició con la creación de un archivo index.jsp que presenta la pantalla inicial del sistema experto mostrada en la Figura 4.12. La pantalla inicial muestra una descripción del sistema y el botón de entrada al sistema experto.

Figura 4.12: Interfaz del sistema experto

Al inicio de la consulta del sistema experto se limpia la memoria de trabajo a través de la función *clear*.

El *servlet* se encarga de enviar la opción elegida (iniciar el sistema experto, continuar con la interacción con el usuario o retroceder en la consulta del sistema experto) al algoritmo RETE de JESS y acceder a la base del conocimiento.

La Figura 4.13 muestra el código de la implementación del servlet en el sistema experto (el código completo se puede consultar en el Anexo A). La clase *HttpServlet* incluye métodos para procesar las solicitudes del cliente en el Web. El *servlet* es *hosteado* en un servidor web y accedido a través del web usando el protocolo HTTP e incorpora a JESS mediante las instrucciones *import jess.** e *import jess.JessException*. También se crean instancias de la clase RETE mediante la instrucción:

$$Rete\ r = new\ Rete();$$

```
package test;
import javax.servlet.http.*;
import java.util.*;
import java.io.IOException;
import javax.servlet.http.HttpSession;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletResponse;
import jess.*;
import jess.JessException;

public class Servlet extends HttpServlet{
   HttpSession session;
   ReteControl reteCon;
   String template = "(deftemplate node (slot name) (slot type) (slot question)"+
  "(slot n1) (slot n2) (slot n3) (slot n4) (slot n5) (slot n6) (slot answer))";
   String cargaReglas = "(defrule inicializa => (load-facts C:/area_eclipse/eminus.dat))";

   public void doPost(HttpServletRequest req, HttpServletResponse res)
   throws IOException, ServletException {
    String boton   = req.getParameter("res1");
    String action2 = req.getParameter("accion2");
    String retrac  = req.getParameter("retrac");

      if (action2.equals("1") )
        session(req,res,"root",1);
      if (action2.equals("Continuar") )
        session(req,res,boton,2);
      if (action2.equals("Regresar") )
        {boton = req.getParameter("res2");
        session(req,res,boton,2);}
      if (action2.equals("Retractar") )
        session(req,res,retrac,3);
      if (action2.equals("Ver"))
        {session(req,res,boton,5);}
      if (action2.equals("Ayuda"))
        {session(req,res,boton,6);}
  }

public void session( HttpServletRequest req, HttpServletResponse res, String accion,int opcion)
   throws IOException, ServletException {
   session = req.getSession(true);
   List lista  = new ArrayList();
   List lnodos = new ArrayList();
   try {
   Rete r = new Rete();
        if (session.isNew())  {
            r.executeCommand(template);
            r.executeCommand(cargaReglas);}
        else {
        List oldlista  = (List)session.getAttribute("lista");
        List oldlnodos = (List)session.getAttribute("lnodos");
        Rete oldr = (Rete)session.getAttribute("r");
        if (oldlnodos != null) {
        lista  = new ArrayList();
            lista.addAll(oldlista);
        lnodos = new ArrayList();
            lnodos.addAll(oldlnodos);
        r = new Rete(); r = (Rete)session.getAttribute("r");}
    }
        session.setAttribute("lista",  lista);
   session.setAttribute("lnodos", lnodos);
   session.setAttribute("r",r);

   if (opcion==1){
        lista.clear();
            lnodos.clear();
       r.executeCommand(template);
            r.executeCommand(cargaReglas);
       reteCon = new ReteControl(res,accion,lista,lnodos,r);}
  if (opcion==2)
       reteCon = new ReteControl(res,accion,lista,lnodos,r);
  if (opcion==3)
       reteCon = new ReteControl(res,accion,lista,lnodos,r);
  if (opcion==5)
            reteCon = new ReteControl(res,accion,r);
  if (opcion==6)
     reteCon = new ReteControl(res,lista,lnodos);

       }catch (JessException JE){ System.out.println("Error Rete Servlet"); }

       try { session.setAttribute("ReteCon", reteCon);
   } catch (IllegalStateException e) {System.out.println("Problemas en session");}
}  }
```

Figura 4.13: Servlet.java

Una vez que se crea un objeto *Rete*, se hace uso del método *executeCommand* para manipular Jess desde Java, el cual, acepta un argumento de tipo *String* y regresa un *jess.Value*. El *String* se interpreta como una expresión en el lenguaje Jess y regresa un valor como resultado de la evaluación de la expresión.

En el *servlet* se define el template que se usó desde JessTab para hacer compatibles los formatos de las reglas y se carga la base de conocimiento generada en Protégé, facilitando el manejo de las reglas y de la base del conocimiento, como se ilustra en la Figura 4.14.

```
String template = "(deftemplate node (slot name) (slot type) (slot question)"+
"(slot n1) (slot n2) (slot n3) (slot n4) (slot n5) (slot n6) (slot answer))";
  String cargaReglas = "(defrule inicializa => (load-facts C:/area_eclipse/eminus.dat))";
```

Figura 4.14: Definición del *template* y carga de la base del conocimiento

En el archivo que maneja los *servlets* y en el archivo que contiene al algoritmo *Rete*, se agregan hechos a la memoria de trabajo mediante la instrucción:

r.executeCommand(template);

Para el intercambio de objetos entre Jess y Java, se usaron los siguientes métodos de la clase *Rete* de Jess:

- *Rete.store(String, Object).* Almacena cualquier objeto en una tabla *hash* de Java.
- *Rete.store(String, Value).* Almacena un *jess.Value* en la tabla de Java.
- *Rete.fetch(String, Value).* Recupera un *jess.Value* de Java.
- *(store symbol value).* Guarda un valor en una tabla *hash* de Jess.
- *(fetch symbol).* Recupera un valor de la tabla de Jess.

En el *servlet* también se manejan listas para emular el retroceso en la consulta del sistema experto y en todo momento se muestra la ruta del problema que se está resolviendo.

```
String reglabase = "(defrule ask-decision-node " +
  "(declare (auto-focus TRUE))" +
  "(node (name ?*n*) (type ?type) (question ?text) " +
  "(n1 ?n1) (n2 ?n2) (n3 ?n3) (n4 ?n4) (n5 ?n5) (n6 ?n6) (answer ?ans))" +
  " => " +
  "(if (eq ?type pregunta) then" +
  "(store TEXT ?text) (store TIPO ?type) " +
  "(store R1 ?n1) (store R2 ?n2) (store R3 ?n3) (store R4 ?n4) (store R5 ?n5) (store R6 ?n6)" +
  "else " +
  "(store TEXT ?ans) (store TIPO h)" +
  "(store R1 ?n1) (store R2 ?n2) (store R3 ?n3) (store R4 ?n4) (store R5 ?n5) (store R6 ?n6)" +
  "  ) )";
```

Figura 4.15: Definición de la regla base del sistema experto

En el archivo *ReteControl.java* se implementó una regla base para facilitar la actualización de la base del conocimiento -consulte el Anexo A-. La estructura de la regla definida en JESS se muestra en la Figura 4.15. El conocimiento del dominio del sistema experto para el Sistema de Administración de Ambientes Flexibles de Aprendizaje está separado del algoritmo RETE que se usa en JESS y está almacenado en un arhivo por separado. Se trató que el código del funcionamiento del sistema experto fuera independiente de la base del conocimiento, de esta manera resulta muy fácil integrar el conocimiento de otro dominio de aplicación.

En la Figura 4.16, se puede observar un segmento del código implementado en el archivo *ReteControl.java* que hace uso de los principales métodos de la clase *Rete* de Jess.

```
r.store("PNODO",new Value(pnodo,RU.ATOM));
r.executeCommand("(bind ?*n* (fetch PNODO) )");
r.executeCommand(reglabase);
r.executeCommand(" (run)");
```

Figura 4.16: Segmento de código desarrollado en JESS

La instrucción *r.store("PNODO",new Value(pnodo,RU.ATOM));*, se utiliza para almacenar un símbolo en *"PNODO"*, mediante el tipo *RU.ATOM* de Jess.

Con *r.executeCommand("(bind ?*n* (fetch PNODO))");*, se recupera el valor del átomo que se almacenó en *PNODO* y a través de la instrucción *r.executeCommand(reglabase);*, se define la regla base (consulte la Figura 4.15) y con *r.executeCommand("(run)");* se dispara la regla y se almacena el resultado.

4.5 Pruebas

La interfaz del sistema experto es sencilla, agradable y fácil de usar. En todo momento el usuario puede visualizar la ruta del problema que se está resolviendo.

Para estas pruebas, se inició con una base de conocimiento, donde se recopilaron los principales problemas con los que se enfrentan los usuarios de Eminus. Posteriormente se irá incrementando el número de reglas con la liberación de la versión 2.0 de Eminus.

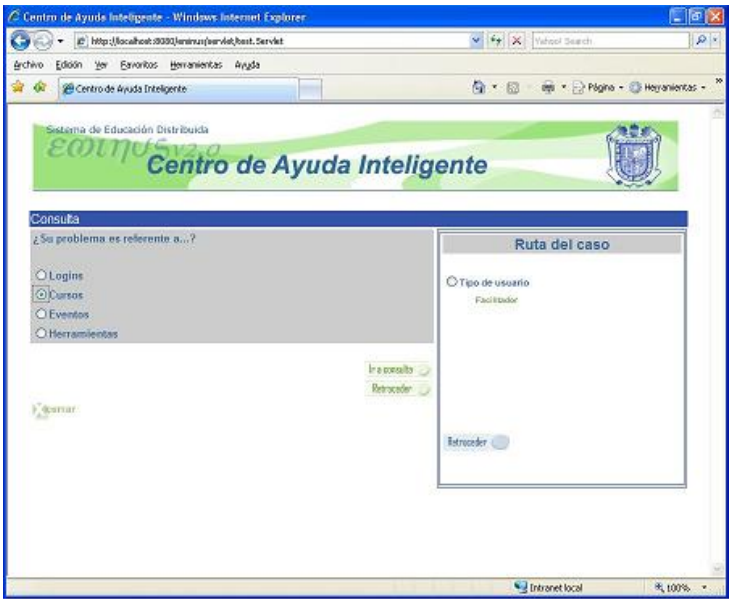

Figura 4.17: El facilitador presenta un problema con un curso

Se tomó como prueba el caso en que un facilitador o maestro no puede visualizar correctamente los módulos de su curso o experiencia educativa que está impartiendo, tal y como lo ilustran las Figuras 4.17, 4.18 y 4.19.

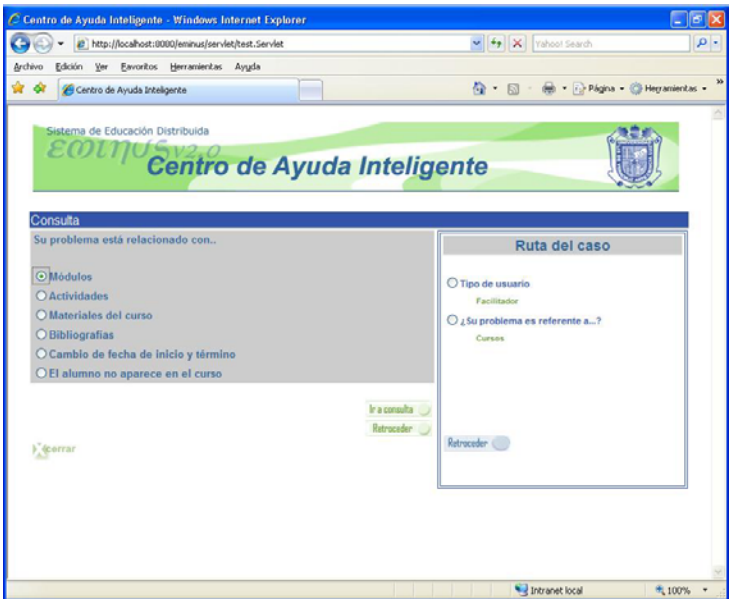

Figura 4.18: El facilitador tiene un problema con los módulos

La interfaz del sistema experto muestr el *trace*, es decir el camino o ruta del caso que se ha seguido para llegar a una solución, la cual está visible en todo momento para aumentar la confianza del usuario en el uso del sistema. Véase la Figura 4.17.

Se hicieron evaluaciones con los desarrolladores y algunos usuarios del sistema Eminus y las soluciones presentadas por el sistema fueron satisfactorias.

El reto consiste en el desarrollo de una robusta base de conocimiento del sistema con el retroalimentación de los usuarios, mediante un formulario donde se le de seguimiento a la solución de los problemas reportados. Y se alimente la base de conocimiento fácilmente mediante el uso de Protégé que brinda una interfaz fácil de usar, sin necesidad de actualizar las reglas de producción directamente en Jess.

El sistema experto para el Apoyo en la Solución de Problemas presenta

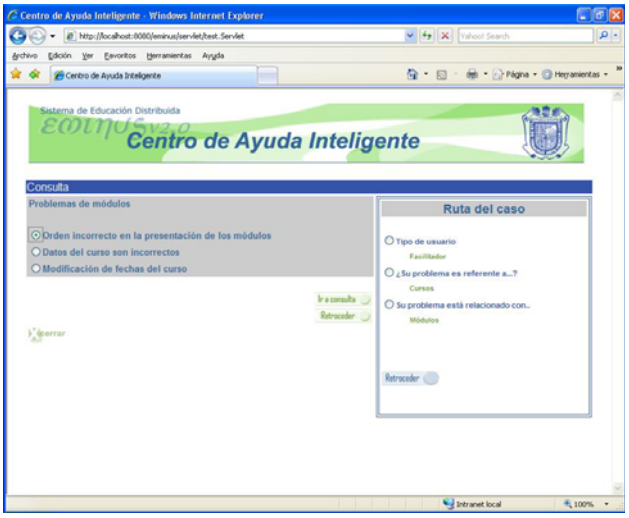

Figura 4.19: Los módulos no se presentan en el orden correcto

la solución al problema presentado por el facilitador en la Figura 4.20.

Figura 4.20: Solución al problema planteado

Capítulo 5

Conclusiones y Trabajo Futuro

5.1 Conclusiones

La metodología de los Sistemas Expertos basados en Web, constituye una viable solución para el desarrollo de centros de ayuda inteligentes que brinden soluciones a los problemas oportunamente durante las 24 horas del día, con la calidad técnica de expertos de alto nivel.

Sin duda, el Sistema Experto para el Apoyo en la Solución de Problemas en un Sistema de Administración de Ambientes Flexibles de Aprendizaje será de gran ayuda para el estudiante en línea evitando que abandone los cursos, lo cual, es de vital importancia para garantizar el éxito de Eminus.

Un punto relevante del presente trabajo, consiste en el desarrollo de una arquitectura genérica de un sistema experto para el apoyo en la solución de problemas basados en tecnología web. Dada la arquitectura propuesta, se puede integrar fácilmente, el conocimiento de otro dominio en el ámbito de la solución de problemas, debido a que la base de conocimiento es independiente de la arquitectura propuesta.

El resultado del presente trabajo, consiste en la propuesta de una metodología que permite integrar un componente inteligente mediante el uso de técnicas de inteligencia artificial, las cuales son transparentes para el usuario, en una aplicación con tecnologías web.

El entorno de desarrollo *Jess -Java Expert System Shell-* brindó todas las ventajas de desarrollo de *Java Enterprise Edition (J2EE)* para la creación de este proyecto.

Se tiene considerado agregar el manejo de incertidumbre en el forma-

lismo de representación del conocimiento y el desarrollo de un módulo de aprendizaje para actualizar la base de conocimiento de manera automática.

La función del módulo de aprendizaje contribuye en la construcción y el refinamiento de la base de conocimiento, mediante un mecanismo de aprendizaje que permita al sistema adquirir nuevos conocimientos y refinar el conocimiento existente en la base de conocimiento de manera automática.

Es importante destacar que se tiene que incrementar el número de reglas de la base de conocimiento, en la medida en que se vayan registrando más problemas reportados por los usuarios e incluir un mecanismo para la formalización del control de la coherencia de las reglas de la base del conocimiento, para evitar inconsistencias en la base de conocimiento en las reglas y/o hechos.

Una vertiente de desarrollo que ofrece Jess es la representación de las reglas de producción en formato *XML -Extensible Markup Language-*. *XML* es un estándar ampliamente utilizado para el intercambio de información estructurada entre diferentes plataformas y constituye una excelente vía para el almacenamiento de reglas debido a que puede usarse como editor de reglas, permitiendo directamente a los expertos capturar el conocimiento del dominio. Como extensión del proyecto, se sugiere elaborar la definición de los principales conceptos del dominio en el formato de *XML* para aprovechar las ventajas de interoperabilidad, facilidad de edición y búsqueda que posee *XML* y hacer uso de un *script de XSLT -Extensible Stylesheet Language Transformations-* para traducir fácilmente, el formato de *XML* al formato de reglas de Jess.

5.2 Trabajo Futuro

Se tiene contemplado incorporar el uso de *Agentes* para el manejo de adaptatividad, mediante un sistema interno que modele al usuario a través de diversas técnicas, muchas de ellas a través de agentes, que soporten una interacción asistida y por lo tanto, una ayuda personalizada al usuario.

La arquitectura del sistema experto propuesto es extendible a una arquitectura de sistemas multiagentes análoga al modelo usado para la asistencia personalizada en el proceso de minería de datos mediante agentes inteligentes [32]. A través de Sistemas Mutiagentes basados en el modelo *BDI*, se puede implementar un agente que coordine las solicitudes de los problemas a resolver, otro para agente que funja como un especialista con el conocimiento

del dominio "preprocesado" para la solución de problemas y algunos agentes que implementen algoritmos de minería de datos (por ej. ID3, C4.5, redes bayesianas y algoritmos TAN) que compitan para producir la "mejor hipótesis" para resolver un problema dado y cooperar con el agente que contenga el conocimiento "preprocesado" cuando sea necesario. El *MAS* puede implementarse en *Jason* que es un intérprete del lenguaje de programación orientado en agentes *BDI* conocido *AgentSpeak(L)*. *Jason* permite la implementación de sistemas de planeación reactivos basados en la arquitectura *BDI -Beliefs-Desires-Intention-*. Los agentes *BDI*, son sistemas basados en sus actitudes mentales: creencias, deseos e intenciones que continuamente perciben su ambiente y ejecutan acciones para modificarlo. Las creencias representan el estado informativo del agente. Los deseos o metas representan el estado final buscado. Las intenciones representan el estado deliberativo del agente. La idea es que estos agentes sean capaces de ejecutar acciones en *Waikato Enviroment for Knowledge Analysis (WEKA)* como parte de sus planes. Weka es una herramienta implementada en Java para el desarrollo de aplicaciones basadas en minería de datos, desarrollado por la Universidad de Waikato (Nueva Zelanda). Weka contiene las herramientas necesarias para realizar transformaciones sobre los datos, tareas de clasificación, regresión, clustering, asociación y visualización [65].

Debido a que Jason y Weka fueron implementados en Java y la arquitectura del sistema experto propuesto fue desarrollada en Protégé, JessTab y Jess, también basados en Java, es ampliamente factible la implementación de los agentes que se centren en la ejecución de las tareas que el sistema experto no puede hacer.

El uso de las funcionalidades de *ARTIMIS* constituye una viable solución, para robustecer las interacciones del sistema multiagente con los usuarios mediante el uso de lenguaje natural. *Artimis* es un *framework* genérico para establecer ricas interacciones entre agentes y usuarios humanos [34]. En Artimis, la comunicación se considera un caso especial del comportamiento inteligente que permite el diálogo con usuarios humanos y con medios de comunicación (teléfono, mensajería instantánea, interacción multimodal y dispositivos móviles). Los agentes presentan avanzadas funcionalidades tales como lenguaje natural, negociación y cooperación y pueden interactuar con agentes de software, *web services*, bases de datos, etc. Ofrece funcionalidades tales como: identificación de las mejores propuestas de las solicitudes de los usuarios y también brinda soluciones alternativas.

Por las características mencionadas anteriormente, Artimis se ha usado en

el ámbito del *e-learning*. En la solución de problemas del sistema Eminus, se requiere un diálogo natural entre el usuario y el sistema y que la interfaz sea capaz de adaptarse a los objetivos, expectativas e intereses de las diferentes clases de usuarios para brindar una ayuda personalizada, sin duda Artimis ofrece elementos suficientes para satisfacer dichos requerimientos.

Apéndice A

Código del Sistema Experto

Servlet.java

```
package test;
import javax.servlet.http.*;
import java.util.*;
import java.io.IOException;
import javax.servlet.http.HttpSession;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletResponse;
import jess.*; //Uso de Jess
import jess.JessException;

public class Servlet extends HttpServlet{
   HttpSession session;
   ReteControl reteCon;
   \\ Definicion del template con la misma estructura de los slots de la base del conocimiento
   generado en Protege.
   String template = "(deftemplate node (slot name) (slot type) (slot question)"+
   "(slot n1) (slot n2) (slot n3) (slot n4) (slot n5) (slot n6) (slot answer))";
   \\Carga del archivo con la base del conocimiento generada por JessTab.
   String cargaReglas = "(defrule inicializa => (load-facts C:/area_eclipse/eminus.dat))";

   public void doPost(HttpServletRequest req, HttpServletResponse res)
   throws IOException, ServletException {
    String boton   = req.getParameter("res1");
    String action2 = req.getParameter("accion2");
    String retrac  = req.getParameter("retrac");

      if (action2.equals("1") )
       session(req,res,"root",1);
      if (action2.equals("Continuar") )
       session(req,res,boton,2);
      if (action2.equals("Regresar") )
        {boton = req.getParameter("res2");
       session(req,res,boton,2);}
      if (action2.equals("Retractar") )
       session(req,res,retrac,3);
```

```
      if (action2.equals("Ver"))
        {session(req,res,boton,5);}
      if (action2.equals("Ayuda"))
        {session(req,res,boton,6);}
  }

public void session( HttpServletRequest req, HttpServletResponse res, String accion,int opcion)
   throws IOException, ServletException {
session = req.getSession(true);
List lista  = new ArrayList();
List lnodos = new ArrayList();
try {
Rete r = new Rete(); \\Instancia de la clase RETE.
        if (session.isNew())  {
          r.executeCommand(template);
             r.executeCommand(cargaReglas); }
        else {
     List oldlista  = (List)session.getAttribute("lista");
     List oldlnodos = (List)session.getAttribute("lnodos");
     Rete oldr = (Rete)session.getAttribute("r");
        if (oldlnodos != null) {
     lista  = new ArrayList();
         lista.addAll(oldlista);
     lnodos = new ArrayList();
         lnodos.addAll(oldlnodos);
     r = new Rete();
         r = (Rete)session.getAttribute("r");}
    }
    session.setAttribute("lista",  lista);
session.setAttribute("lnodos", lnodos);
session.setAttribute("r",r);
if (opcion==1){
     lista.clear();
         lnodos.clear();
      r.executeCommand(template);
         r.executeCommand(cargaReglas);
    reteCon = new ReteControl(res,accion,lista,lnodos,r);}
if (opcion==2)
     reteCon = new ReteControl(res,accion,lista,lnodos,r);
if (opcion==3)
     reteCon = new ReteControl(res,accion,lista,lnodos,r);
if (opcion==5)
         reteCon = new ReteControl(res,accion,r);
if (opcion==6)
     reteCon = new ReteControl(res,lista,lnodos);
        }catch (JessException JE){ System.out.println("Error Rete Servlet"); }
       try { session.setAttribute("ReteCon", reteCon);
} catch (IllegalStateException e) {System.out.println("Problemas en session"); }
}   }
```

ReteControl.java

```
package test;
import jess.*;
import jess.JessException;
import java.io.IOException;
import java.io.*;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
import java.util.*;

  public class ReteControl {
   String x, xtipo, aux1,aux2, anodos[]= new String[7];
Value atn[] = new Value[7], an[] = new Value[7];
   //Definicin de la regla base
   String reglabase = "(defrule ask-decision-node " +
"(declare (auto-focus TRUE))" +
"(node (name ?*n*) (type ?type) (question ?text) " +
"(n1 ?n1) (n2 ?n2) (n3 ?n3) (n4 ?n4) (n5 ?n5) (n6 ?n6) (answer ?ans))" +
" => " +
"(if (eq ?type pregunta) then" +
"(store TEXT ?text) (store TIPO ?type) " +
"(store R1 ?n1) (store R2 ?n2) (store R3 ?n3) (store R4 ?n4) (store R5 ?n5) (store R6 ?n6)" +
"else " +
"(store TEXT ?ans) (store TIPO h)" +
"(store R1 ?n1) (store R2 ?n2) (store R3 ?n3) (store R4 ?n4) (store R5 ?n5) (store R6 ?n6)" +
"  ) )";
// Variables globales
 String globales = "(set-reset-globals FALSE) (defglobal ?*n* = \"\")";

 public ReteControl(HttpServletResponse res,String pnodo,List lista,List lnodos,Rete r)
 throws IOException,ServletException  {
 //Manejo del "back" del navegador o de un "retract" del sistema experto.
  for (int i=0; i<lnodos.size()-1; i++)
   if (lnodos.get(i).equals(pnodo)) {
   lista.subList(i ,lista.size() ).clear();
   lnodos.subList(i,lnodos.size()).clear();
   }
   System.out.println(pnodo); //Imprime el nodo actual en la salida del sistema
   res.setContentType("text/html");
   PrintWriter out = res.getWriter();
   out.println("<html> <head> <title>Centro de Ayuda Inteligente </title>" +
   "<link href=\"../Estilo.css\" rel=\"stylesheet\" type=\"text/css\"> " +
   "<SCRIPT LANGUAJE=\"JavaScript\"> var val1=\"root\"; val2=0; " +
   "window.onload=maximixar;"+ "function maximixar(){"+
   "window.resizeTo(900,screen.availHeight); }" + "function newWindow()" +
   "{window.open('', 'newWin','width=600,height=400,scrollbars=yes,menubar=0,
    toolbar=0','location=0','status=0'); }" + "function cerrarw()" +
   "{window.opener = \"madre\"; window.close(); } </SCRIPT>" +
   " </head> <body vlink=\"#FFFFFF\"> <center>");
   out.println("<table width =\"800\"  border=0 >" +
   "<tr> <td colspan=2> <table width=100%> <tr> <td width=800>
   <img src=\"../imagenes/signo.gif\" heigth=60>" +
   "</td> </tr> </table>  " + " <br> </td> </tr> ");
   try{
       r.executeCommand(globales);
```

```
r.store("PNODO",new Value(pnodo,RU.ATOM));
r.executeCommand("(bind ?*n* (fetch PNODO) )");
r.executeCommand(reglabase);
r.executeCommand("(reset) (run)");
for(int ires=1; ires<=6; ires++){ //Se obtiene el nombre de los nodos a insertar
an[ires] = r.fetch("R"+ires);
anodos[ires] = an[ires].stringValue(r.getGlobalContext());
}
//Se obtiene el tipo de pregunta
Value  tipo = r.fetch("TIPO");
xtipo = tipo.stringValue(r.getGlobalContext());
//Se obtiene la pregunta o el resultado
Value sT = r.fetch("TEXT");
x = sT.stringValue(r.getGlobalContext());
if (!xtipo.equals("pregunta")){
   out.println("<SCRIPT LANGUAJE=\"JavaScript\"> " +
          "window.open('../resultadoshtml/"+x+"','Resultado','width=840,height=600,
          scrollbars=yes'); </SCRIPT>");
}
if (xtipo.equals("pregunta"))
   out.println("<tr bgcolor=3354A9><td colspan=2> <font face=\"Arial\" color=FFFFFF >
                        Consulta  </font> </td> </tr> " + "<tr><td valign=top>");
   else
   out.println("<tr><td colspan=2> " + "<table bgcolor=3354A9 width=800> <tr><td>
            <font face=\"Arial\" color=CCCC99 > <strong> Solucion </strong> </font>
            </td> " + "<td> <img src=\"../imagenes/icoRutadelcaso-R.gif\" border=0>
            </td> </tr> </table> </td> </tr>" + "<tr><td valign=top>");
//Se agrega la pregunta en la lista de preguntas para mostrar el "trace"
lista.add(new String(x));
lnodos.add(new String(pnodo));
}catch (JessException JE){System.out.println("ReteControl: error in executeCommand /
        run: " + JE);}
if (!pnodo.equals("root"))
aux2=lnodos.get(lnodos.size()-2).toString();
else
aux2="root";
out.println(" <FORM name=fp1 method=POST >");
if(xtipo.equals("pregunta") ){ //Si es una pregunta, de despliegan sus opciones
out.println("<table bgcolor=CCCCCC border=0 ><tr><td class=\"preg_resp\">"+x+ "
                    </td></tr> <tr><td>   </td></tr>");
String auxnodos;
for (int ires=1; ires<=6; ires++)
if(!anodos[ires].equals("n")){
auxnodos=anodos[ires];
auxnodos = auxnodos.toLowerCase();
if (auxnodos.charAt(0)=='s' && auxnodos.charAt(1)=='i' && auxnodos.charAt(2)=='1')
                    {auxnodos="Si";}
else
if (auxnodos.charAt(0)=='n' && auxnodos.charAt(1)=='o' && auxnodos.charAt(2)=='1')
                    {auxnodos="No";}
else
auxnodos=anodos[ires];
out.println("<tr> <td width=500> <font class=\"preg_resp\"> " +
         "<INPUT TYPE=\"radio\" NAME=\"res1\" VALUE="+anodos[ires]+" CHECKED >"+
auxnodos.replaceAll("_"," ")+" </td> </tr>");
}
out.println("</table> <br> ");
```

```
   }

   if(xtipo.equals("pregunta") ){
   out.println("<table width=500><tr><td align=\"right\"><input TYPE=hidden name=accion2 >");
   out.println("<input name=bi2 TYPE=image src=\"../imagenes/aceptar-R.gif\"
                       onClick=document.fp1.accion2.value=\"Continuar\"; >" + "</td></tr>");
   }
   else{
   out.println("  <table width=500 ><tr bgcolor=CCCCCC><td> " +
            "<font face=\"Arial\" color=444444> <input TYPE=hidden name=accion2 >" +
            "<INPUT TYPE=\"radio\" NAME=\"res1\" VALUE=\"ayuda\" CHECKED >
            La respuesta no resuelve mi caso </font> </td> </tr>" +
            "<tr> <td align=right> <input name=bi2 TYPE=image src=\"../imagenes/aceptar-R.gif\"
            onClick=document.fp1.accion2.value=\"Ayuda\"; >" + "</td> </tr>");
            out.println("<tr><td bgcolor=FFFFFF align=right><a href=\"../index.jsp\">
            <img src=\"../imagenes/nuevaconsulta-A.gif\"> </a> </td></tr> ");
   }
   out.println("</FORM> <FORM name=fpx method=POST>");
   out.println(" <tr> <td align=right>");
   out.println("<INPUT NAME=\"retrac\" VALUE=\""+aux2+"\" type=\"hidden\">");
   if(!pnodo.equals("root")){
   out.println("<input TYPE=hidden  name=accion2 value=Retractar>");
   out.println("<input name=bi1 TYPE=image src=\"../imagenes/retroceder-E.gif\" >");
   }
   else {
   out.println("<a href=\"../index.jsp\"> <image src=\"../imagenes/retroceder-E.gif\"> </a>");}
   out.println("</td></tr> </FORM> <tr> <td>");
   out.println("<a href=\"javascript:cerrarw()\"> <img src=\"../imagenes/salir-A.gif\" border=0>
                </a> </td>  </tr> </table>");
   out.println("</td> <td valign=top >");
   out.println("<table border bordercolorlight=#3354A9 bordercolordark=#3354A9 HEIGHT=320 > <tr>
                <td valign=top>");
   if (!pnodo.equals("root"))
     muestralista(res,lista,lnodos,out);
   else
   out.println("<table> <tr> <td align=right bgcolor=CCCCCC width=300> <center> " +
   "<font face=\"Arial\" color=336699 size=4> <b>Ruta del caso </b> </font> </center> </td>" +
   "</tr> <tr> <td height=300> <br> </td></tr> </table>");
   out.println("</td></tr> </table>");
   out.println("</td> </tr> </table> </body> </html>");
   }
//Muestra la pregunta
 public ReteControl(HttpServletResponse res,String accion,Rete r)
throws IOException,ServletException{
res.setContentType("text/html");
PrintWriter out = res.getWriter();
out.println("<html> <head> <title> Opciones que tenia la pregunta </title>" +
"<link href=\"../Estilo.css\" rel=\"stylesheet\" type=\"text/css\"> </head>" +
"<body vlink=\"#FFFFFF\" link=\"#FFFFFF\"> " +
"<table width=\"550\" border=0> <tr> <td>"+
"<div class=\"signoprincipal\"> <img src=\"../imagenes/signo.gif\" heigth=60> </div>" +
"<br> <br> <br>" +
"<br> <br> <br> </td> </tr> </table>");
try{
    r.executeCommand(globales);
r.store("PNODO",new Value(accion,RU.ATOM));
r.executeCommand("(bind ?*n* (fetch PNODO) )");
```

```
r.executeCommand(reglabase);
r.executeCommand("(reset) (run)");
for(int ires=1; ires<=6; ires++){
atn[ires] = r.fetch("R"+ires);
anodos[ires] = atn[ires].stringValue(r.getGlobalContext());
}
//Obtiene la pregunta
Value sT = r.fetch("TEXT");
x = sT.stringValue(r.getGlobalContext());
out.println("<table border=0> <tr class=\"celda_trace_opcion\"> <td>
         <font face=\"Arial\" size=4>"+x+"</font> </td></tr>");
for (int ires=1; ires<=6; ires++)
   if(!anodos[ires].equals("n"))
   out.println("<tr class=\"celda_trace_titulo\"> <td > "+
   anodos[ires].replaceAll("_"," ")+" </td> </tr>");
   out.println("</table>");
}catch (JessException JE){System.out.println("ReteControl: error in executeCommand /
               run: " + JE);}
out.println("<a href=\"javascript:window.close();\">" +
         "<br><img src=\"../imagenes/salir-A.gif\"></a> </body> </html>");
}//Fin de muestra pregunta

 //Pantalla de ayuda
public ReteControl(HttpServletResponse res,List lista,List lnodos) throws IOException {
res.setContentType("text/html");
PrintWriter out = res.getWriter();
out.println("<html> <head> <title> Centro de Ayuda Inteligente </title>" +
        "<link href=\"../Estilo.css\" rel=\"stylesheet\" type=\"text/css\"> " +
         "<SCRIPT language=\"JavaScript\" type=\"text/javascript\">" +
        "window.onload=maximixar;"+ "function maximixar(){"+
        "window.resizeTo(900,screen.availHeight); }" + "</SCRIPT>" +
        "<SCRIPT LANGUAJE=\"JavaScript\"> var val1=\"root\"; " +
        "function newWindow()" + "function cerrarw()
        {window.opener = \"madre\"; window.close(); } " +
        " </SCRIPT>" + " </head> <body vlink=\"#FFFFFF\"> <center>");
out.println("<center> <img src=\"../imagenes/signo.gif\" heigth=60> </center>");
out.println("<table width =\"800\"  border = 0 >" + "<tr> <td colspan=2 > <br>  <br>
           <br> " + "<br> <br> </td> </tr>  " + "</table><br>");
out.println("<table width=600><tr><td bgcolor=CCCCCC> <font face=\"Arial\" size=4 color=3377AA>
           <center> Solicitud de Atenci&oacute;n Personalizada </center> </font> </td></tr>" +
          "<tr><td> <font color=3377AA face=\"Arial\" size=2> <p align=justify>
              Una vez recibida su solicitud, &eacute;sta ser&aacute; atendida
              por el personal competente " + "en la problematica descrita,
              el cual se pondra en contacto con usted para ayudarlo en el procedimiento"
              + "de salida del problema o bien informarle sobre la disponibilidad de
              su soluci&oacute;n en el sistema. </p> </font> </td> </tr>" + "<tr>
              <td bgcolor=337799> <font face=\"Arial\" color=FFFFFF size=4>
              Datos del usuario </font> </td></tr> ");
out.println("<tr><td><FIELDSET width=800> " +
            "<form name=fayu method=\"post\" action=\"mailto:administrador@uv.mx\"
            ENCTYPE=\"TEXT/PLAIN\">" + "<font color=3377AA face=\"Arial\" size=3>
            Nombre: <input type=\"text\" name=\"nombre\" size=50><br>" +
            "Dependencia: <input type=\"text\" name=\"dependencia\" size=50><br>" +
            "<font face=\"Arial\" size=1 >Ejemplo: Direccin de Tecnologas y Apoyo Acadmico:
             Multimedia </font> </font>" + "<table ><tr><td valign=top> <font
             face=\"Arial\" color=3377AA> Descripcin del <br> problema: </font> </td>" +
            "<td><textarea name=\"descripcion\" cols=50 rows=10> </textarea> </td></tr>" +
```

```
          "</table> "+"</form> <table width=500> <tr> <td align=right> " +
          "<input type=image name=\"imagen\" SRC=\"../imagenes/enviar-A.gif\"
           onClick=document.fayu.submit(); > <br>" + "<input name=bi1
           TYPE=image src=\"../imagenes/limpiar-A.gif\" onClick=document.fayu.reset();
           >" + "</td> </tr> </table> </FIELDSET> </td></tr> </table>");
out.println("<a href=\"../index.jsp\"> <img src=\"../imagenes/nuevaconsulta-A.gif\"> </a> ");
}

public void muestralista(HttpServletResponse res,List lista, List lnodos,PrintWriter out)
throws IOException,ServletException{
   String muestra="",auxnodos;
   out.println("<table border=0 width=300> <tr> <td align=right bgcolor=CCCCCC
               HEIGHT=20> <center> " + "<font face=\"Arial\" color=336699 size=4>
               <b> Ruta del caso </b> " + " </font> </center> </td> </tr> </table>");
    out.println("<FORM name=uno method=POST action=/eminus/servlet/test.Servlet >");
    out.println("<div id=\"cContenido\" style=\"position:relative;width:300px;height:200px;
            scrollbar-face-color:white; scrollbar-3dlight-color:green; left:0px; top:0px;
            z-index:0; overflow: auto;\">");
    out.println("<table>");
    for(int i=0; i<lista.size()-1; i++){
    out.println("<tr> <td class=\"ruta_preg\" HEIGHT=20> <INPUT TYPE=radio NAME=res1 VALUE="+
        lnodos.get(i)+ " onClick=val1=this.value;>"+ lista.get(i).toString() + "</td> </tr>");
    muestra = lnodos.get(i+1).toString();
    auxnodos=muestra;
        auxnodos = auxnodos.toLowerCase();
    if (auxnodos.charAt(0)=='s' && auxnodos.charAt(1)=='i' && auxnodos.charAt(2)=='1')
            {auxnodos="Si";}
    else
     if (auxnodos.charAt(0)=='n' && auxnodos.charAt(1)=='o' && auxnodos.charAt(2)=='1')
            {auxnodos="No";}
    else
        auxnodos=muestra;
    out.println("<tr> <td class=\"ruta_resp\" HEIGHT=20> <dd>"+
           auxnodos.replaceAll("_"," ")+ " </td> </tr> ");
    }
    out.println("<tr> <td>   </td> </tr>");
    out.println("</table> ");
    out.println("</div>");
    out.println("<table border=0 ><tr><td width=200 valign=top> ");
    out.println("<INPUT NAME=res2 type=hidden>");
    out.println("<Input TYPE=hidden name=accion2 Value=Regresar > ");
    out.println("<input TYPE=image src=\"../imagenes/retroceder-A.gif\"
                onClick=document.uno.res2.value=val1; >");
    out.println("</form> </td>");
    out.println("<td> <FORM name=fo2 method=POST action=/eminus/servlet/test.Servlet
                target=newWin onSubmit=\"newWindow()\">");
    out.println("<INPUT NAME=res1 type=hidden>");
    out.println("<INPUT NAME=accion2 TYPE=hidden value=Ver>");
    out.println("</FORM> </td> </tr> </table>");
 }
}
```

Bibliografía

[1] *eXpertise2Go*, <http://www.expertise2go.com/>.

[2] *ExSys*, <http://www.exsys.com/>.

[3] *Blaze Advisor*, <http://www.fairisaac.com/ficx>.

[4] *Expert Systems and the Helpdesk*, <http://www.gslis.utexas.edu/ palmquis/ courses/project98/help/helpdsk1.htm>.

[5] *JessTab Manual. Integration of Protégé and Jess*, <http://www.ida.liu.se/her/JessTab/JessTab.pdf>.

[6] *ILOG's Business Rule Management System*, <http://www.ilog.es/products/businessrules/products.cfm>.

[7] *OPSJ*, <http://www.pst.com/opsj.htm>.

[8] *XPertRule KBS Shell*, <http://www.xpertrule.com/>.

[9] T.L. Acorn. Smart: Support management automated reasoning technology for compaq customer service. *AAI*, pages 2–18, 1992.

[10] Julie E. Adams. The feasibility of distributed web based expert systems. *Proceedings of the 2001 IEEE International Conference on Systems, Man, and Cybernetics, Conference*, 2001.

[11] Barr Avron and Tessler Shirley. Expert systems: A technology before its time. *AI Expert*, 1995.

[12] R. Barletta. Building a case-based help desk application. *IEEE Expert*, pages 18–26, December 1993.

[13] R. Barletta. Case-based reasoning and information retrieval: Opportunities for technology sharing. *IEEE Expert*, pages 2–8, December 1993.

[14] Bench-Capon. *Knowledge Representation an Approach to Artificial Intelligence*. Manning Publications Co., United States of America, 2003.

[15] Ho Kang Byeong, Yoshida Kenichi, Motoda Hiroshi, and Compton Paul. *Help Desk System with Intelligent Interface*. <http://www.ar.sanken.osaka-u.ac.jp/papers/motoda/helpdesk-kang.pdf>.

[16] J.P. Callan and W. B. Croft. As approach to incorporating cbr concepts in information retrieving systems. In AAAI Press, editor, *In AAAI Spring Symposium*, pages 28–34, 1993.

[17] Brewster Christopher and O'Hara Kieron. Knowledge representation with ontologies: The present and future. *IEEE Intelligent Systems*, pages 72–81, January–February 2004.

[18] Sleeman D. Corsar D. Reusing jesstab rules in protégé. *Knowledge Based Systems Journal*, 19, 2006.

[19] H. Eriksson. Using jesstab to integrate protégé and jess. *IEEE Intelligent Systems*, 18(2):43–50, 2003.

[20] Friedman-Hill Ernest. *JESS in Action: Rule Based Systems in Java*. The A.P.I.C. Series 32, Academic Press, San Diego, United States of America, 2003.

[21] Friedman-Hill Ernest. *JESS Java Expert System Shell*, <http://herzberg.ca.sandia.gov/jess>.

[22] Friedman Ernest-Hill. Jess and the javax.rules api. 2003. <http://www.theserverside.com/tt/articles/article.tss?l=Jess>.

[23] Borges F. La frustración del estudiante en línea, causas y acciones preventivas. *Digithum UOC*, (7), 2005.

[24] Grove Ralph F. Design and development of knowledge-based systems on the web. buscar la URL. Indiana University of Pennsylvania, Computer Science Department, Indiana.

[25] Lara Felipe. Expert systems applications in mexico. *4 World Congress on Expert Systems. Application of Advanced Information Technologies*, 1:10–11, March 1998.

[26] Stanford Center for Biomedical Informatics Research. *Protégé*, <http://protege.stanford.edu/>.

[27] Charles L. Forgy. Rete: a fast algorithm for the many pattern/many object pattern match problem. pages 324–341, 1990.

[28] Fergerson R. Gennari J., Musen M. The evolution of protege: An environment for knowledge-based systems development. Technical report, Stanford University CA, 2002.

[29] J.C. Giarratono. *CLIPS*, <http://www.ghg.net/clips/CLIPS.html>.

[30] Simic Goram and Devedzic. Building an intelligent system using modern internet technologies. *Expert Systems with Applications*, 25:201–210, 2003.

[31] Hulse Arthur C. Grove, Ralph F. An internet-based expert system for reptile identification. pages 165–173, 1999. The First International Conference on the Practical Application of Java, London, UK.

[32] Mondragón R. Guerra A. and Cruz N. Explorations of the bdi multiagent support for the knowledge discovery in databases process. *Advances in Computer Science and Artificial Intelligence. Research in Computing Science*, 39:221–238, 2008.

[33] Bordini Rafael H., Hubner Jomi Fred, and Wooldridge Michael. *Programming Multi-agent Systems in AgentSpeak using Jason*. Wiley Series in Agent Technology, 2007.

[34] Bordini Rafael H., Dastani Mehdi, Dix Jurgen, and Seghrouchni El Fallah Amal. *Multi-agent Programming*. Springer Science Business Media, 2005.

[35] Eriksson H. *JessTab: Using together with Protégé*, <http://protege.stanford.edu/conference/2003/HenrikErikssonJessTab ProtegeWS.pdf>.

[36] Kling R. Hara N. Students' frustration with a web-based distance education course. *First Monday*, 4(12), 1999.

[37] Owen James. Budget-friendly brms. 2004. <http://www.infoworld.com/article/04/03/12/11TCopsj1.html>.

[38] Negrete José, González Pedro Pablo, and Guerra Alejandro. *Pericia Artificial: Un Aprendizaje Constructivista de Sistemas Expertos*. Universidad Veracruzana, 1996.

[39] P. Kang B., Compton. A maintenance approach to case-based reasoning. pages 226–239, 1994.

[40] R. Kriesgman M., Barletta. Building a case-based help desk application. *IEEE Expert*, pages 18–26, December 1993.

[41] Forgy Charles L. A fast algorithm for the many pattern / many object pattern match problem. *Artificial Intelligence*, 19:17–37, 1982.

[42] Shu-Hsein Liao. Expert systems methodologies and applications - a decade review from 1995 to 2004. *Expert Systems with Applications*, 28(1):93–103, january 2005.

[43] Mestizo Gutiérrez Sonia Lilia. *Estudio Comparativo de Dos Entornos de Desarrollo de Sistemas Expertos*. Facultad de Estadística e Informática. Universidad Veracruzana, junio 1996.

[44] Linkoping University. *JessTabManual. Integration of Protg and Jess*, 2004.

[45] Vazey Megan and Richards Debbie. Achieving rapid knowledge acquisition in a high-volume call centre. http://www.freyatech.com.au/freyatech/publications/2004PKAW.pdf.

[46] Guerra A. Negrete J., González P. *Pericia Artificial: un Aprendizaje Constructivista de Sistemas Expertos*. Universidad Veracruzana, Xalapa, México, 1995.

[47] A. Newell. The knowledge level. *Artificial Intelligence*, 18:87–127, 1982.

[48] Mcguiness D. Noy N.F. Ontology development 101: A guide to creating your first ontology. Technical report, Stanford University, 2005.

[49] Cairó O. Arquitecturas de sistemas expertos. *Soluciones Avanzadas*, pages 41–47, Septiembre 1994.

[50] Maes Patti. Agents that reduce work and information overload. *Communications of the ACM*, 37(7), July 1994.

[51] Maes Patti. Intelligent software: Programs that can act independently will ease the burdens that computers put on people. *Scientific American*, 273(3):84–86, 1995.

[52] Jackson Peter. *Introduction to Expert Systems*. Addison-Wesley, Harlow, England, 3rd edition, 1999.

[53] G. Salton, J.and Buckley C. Allan, and Singhal A. Automatic analysis, theme generation and summarization of machine-readable texts. *Science*, 264:1421–1426.

[54] G. Salton and M. J.McGill. *Introduction to modern information retrieval*. USA:McGraw-Hill, 1986.

[55] H. Shimazu, A. Shibata, and K. Nihei. Case-based retrieval interface adapted to customer-initiated dialogues in help desk operations. In J. Mylopuoulus and R. Reiter, editors, *The Twelfth National Conference on Artificial Intelligence*, pages 513–518. Seattle, USA:AAAI Press.

[56] Nihei K. Shimazu H., Shibata A. Case-based retrieval interface adapted to customer-initiated dialogues in help desk operations. pages 513–518, 1994.

[57] Devedzic V. Shimic G. Building an intelligent system using modern internet technologies. *Expert Systems with Applications*, 25(3):231–246, 2003.

[58] Liao Shu-Hsien. Expert system methodologies and applications – a decade review from 1995 to 2004. *Expert Systems with Applications*, 28:93–103, 2004.

[59] E. Simoudis. Using case-based retrieval for custumer technical support. *IEEE Expert*, pages 7–12, October 1992.

[60] Russell Stuart and Norving Peter. *Inteligencia Artificial. Un enfoque moderno*. Prentice Hall, 1996.

[61] Devedzić Tomić Bojan, Jovanović Jelena. Javadon: an open-source expert system shell. *Expert Systems with Applications*, 31:595–606, 2006.

[62] Levy Tuthill. *Knowledge based Systems. A Manager's Perspective*. TAB Books, USA, first edition edition, 1991.

[63] Devedzic V. Understanding ontological engineering. *Communications of the ACM*, 45(4):136–144, April 2002.

[64] Kristof Van Laerhoven. Comparison of the clips and jess expert system shells. *Project report for Industrial Applications of AI*, June 1999.

[65] Frank E. Witten I. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufman, San Francisco, 2nd edition, 2005.

[66] Kim Wooju, Song Yong U., and Hong June. Web enabled expert systems using hyperlink-based inference. *Expert Systems with Applications*, 28:79–91, january 2005.

Lista de Figuras

2.1 Arquitectura clásica de un Sistema Experto 19
2.2 Una red RETE no optimizada para las reglas ejemplo1 y ejemplo2. 33
2.3 Una red RETE que comparte los nodos de patrones. 35
2.4 Una red RETE que comparte nodos de patrones y nodos *join*. 36
2.5 Historia de JessTab 40

3.1 Interfaz de Eminus 44

4.1 Componentes de un sistema experto. Las flechas representan el flujo de información. 52
4.2 Protége 56
4.3 Ejemplo de instancia de una clase en Prótegé. 57
4.4 Estructura de la base de conocimiento en Prótegé. 58
4.5 Arquitectura del Sistema Experto 59
4.6 JessTab 60
4.7 Pestaña de Jess en Protégé 60
4.8 Base de conocimiento en JessTab 61
4.9 Definición del *template* de la regla base 61
4.10 Base del conocimiento con la estructura final 62
4.11 Integración del archivo de la solución del problema en la base del conocimiento 62
4.12 Interfaz del sistema experto 63
4.13 Servlet.java 64
4.14 Definición del *template* y carga de la base del conocimiento . . 65
4.15 Definición de la regla base del sistema experto 66
4.16 Segmento de código desarrollado en JESS 66
4.17 El facilitador presenta un problema con un curso 67
4.18 El facilitador tiene un problema con los módulos 68

4.19 Los módulos no se presentan en el orden correcto 69
4.20 Solución al problema planteado 69

Lista de Tablas

2.1 Categorías de Sistemas Expertos basados en Web. 26