DISEÑO Y PROTOTIPADO DE UNA HERRAMIENTA INTELIGENTE PARA PROPONER LA SOLUCIÓN DE EJERCICIOS EN LA WEB

ANTONIO JOSE PINEDO ARRIETA

ALVARO NICOLAS QUINTANA BURGOS

Trabajo final de investigación para el Minor en Ingeniería de Software

Universidad Tecnológica de Bolívar

Programa de Ingeniería de Sistemas

Facultad de Ingenierías

Cartagena

2006

RESUMEN

El desarrollo de este proyecto está basado en los Sistemas Expertos como la base para la construcción de un tutor virtual que responda a las solicitudes de distintas fuentes que puedan requerir el razonamiento lógico de un experto humano para la solución de problemas en un ambiente educativo.

Se define a los sistemas expertos como un sistema de cómputo capaz de emular la habilidad de tomar decisiones de un especialista humano de manera que reaccione como tal ante una situación que se encuentra dentro de un dominio particular.

Estos se conforman por dos componentes básicos: la base de conocimientos y el mecanismo de inferencia, los cuales interactúan para emular el proceso de solución de problemas de un experto.

Un aspecto muy importante al considerar el desarrollo o utilización de un Sistema Experto es la necesidad real de este, ya que si un problema puede ser resuelto eficazmente por medio de programación convencional, entonces un sistema experto no es la mejor opción sino que podría generar respuestas erróneas.

Los componentes principales de este tipo de sistemas son: interfaz de usuario, medio de explicación, memoria activa, mecanismo de inferencia, la agenda y el medio para la adquisición del conocimiento.

Para diseñar un sistema experto es necesario conocer los esquemas de representación del conocimiento y su posible interacción. El sistema experto procesa símbolos y el mecanismo de inferencia los manipula por medio de la selección de reglas, agrupando los símbolos de hechos y luego disparando las reglas para establecer nuevos hechos.

El proceso de inferencia se puede realizar de muchas maneras, pero los métodos más populares son el Encadenamiento hacia delante (proceso basado en datos) y Encadenamiento hacia atrás (proceso basado en metas).

Para el desarrollo y puesta en marcha de este tipo de proyectos es necesario desarrollar algún tipo de metodología que defina el ciclo de vida de la

aplicación; el modelo Lineal ha sido aplicado con éxito en varios proyectos de desarrollo de sistemas expertos, el cual está conformado por varias etapas que van desde la planeación a la evaluación del sistema, realizándose varias repeticiones del proceso hasta que esté disponible para su uso.

Este modelo puede considerarse también como un circuito en espiral donde cada etapa esta formada por tareas, las cuales dependerán del tipo de aplicación que se este construyendo.

El prototipo de la aplicación realizada está diseñado en JESS el cual es un motor de las reglas que se desarrollo en el lenguaje de programación Java, y su propósito es mostrar la relación entre el código fuente desarrollado en JAVA y el sistema experto desarrollado en el lenguaje CLIPS. Como resultado de esta implementación obtenemos un sistema totalmente funcional capaz de realizar razonamiento basado en la lógica de reglas y hechos.

El programa desarrollado emula el conocimiento de un experto en la solución de ecuaciones de una variable utilizando distintos métodos numéricos definidos en la Base de Conocimiento.

La elaboración de este prototipo fue propuesta como una demostración de la capacidad de análisis de los sistemas inteligentes y las grandes ventajas que trae su integración con lenguajes de programación orientado a objetos, por lo

cual se pretende que en el futuro se continúe con su desarrollo llevando a cabo las siguientes tareas: desarrollar un analizador que permite evaluar cualquier tipo de función matemática ingresada, un mecanismo de explicación capaz de justificar las decisiones tomadas y una interfaz para incrementar el conocimiento del Experto el cual le da flexibilidad al sistema.

TABLA DE CONTENIDOS

LISTA DE TABLAS Y FIGURAS

INTRODUCCIÓN

Desde los inicios del estudio en el campo de la Inteligencia Artificial se ha buscado solucionar problemas empleando las computadoras y se ha buscado en estas un comportamiento similar al de un experto humano frente a dichas situaciones.

En un principio se pensó que esto no era posible o que era parte de la ciencia ficción que se veía en las cintas de cine, pero a medida que fue pasando el tiempo y con los grandes aportes realizados por investigadores en la materia se fueron ideando esquemas para llegar a este objetivo, y de allí surgieron los primeros prototipos de sistemas expertos los cuales tuvieron su dominio enfocado a la computación y la medicina. Luego el dominio se fue expandiendo hasta tal punto que ha abarcado un gran número de áreas del conocimiento y se han diseñado sistemas expertos que son usados tanto en ambientes investigativos como de producción.

Uno de los principales obstáculos que se ha encontrado en el desarrollo de sistemas expertos es la aplicabilidad de estos y la falta de conocimiento de sus usuarios potenciales, ya que muchas veces un problema puede ser resuelto por medio de paradigmas como la programación estructurada o la P.O.O. y en estos casos no sería la mejor alternativa seleccionar un Sistema Experto para dar solución a tales problemas; otros necesitan de un Sistema Experto para la solución de un problema pero por el temor o desconocimiento del mismo deciden no utilizarlo.

Son muchas las aplicaciones que se pueden dar hoy en día a los sistemas expertos, y se ha incursionado en campos como la medicina, química, ingeniería y la educación entre otras, siendo este último el objeto principal de nuestro estudio.

El objetivo de este escrito es dar al lector las bases para la construcción de un sistema experto que pueda ser utilizado en el ambiente educativo, por lo cual se describen en el los conceptos básicos de cualquier sistema experto haciendo énfasis en aquellas partes que son realmente importantes en el desarrollo del prototipo de la aplicación planteada.

También deseamos que este documento sirva como apoyo para futuras investigaciones en esta rama del conocimiento y que pueda dar soporte a los

grupos de investigación que basen su estudio en el desarrollo de sistemas expertos o redes neuronales.

El contenido consta de cinco capítulos donde se explican desde los conceptos básicos, hasta el diseño de un sistema experto y la implementación del sistema con herramientas y lenguajes modernas, pasando de esta manera a la práctica por medio de ejemplos y casos concretos que son producto de nuestra experiencia en el desarrollo del sistema experto, todo esto apoyado en gráficas y tablas que ayudan a una mejor comprensión de los temas tratados a lo largo del documento.

Esperamos que este documento sea de gran ayuda para todos aquellos que estén de alguna manera relacionados con el tema y que deseen continuar su estudio en esta línea de investigación.

CAPÍTULO 1:

Conceptos sobre Sistemas Expertos

Los sistemas expertos son unos de los primeros avances obtenidos en el campo de la inteligencia artificial y fueron diseñados para poner el conocimiento de los expertos a disposición de todos emulando sus patrones de pensamiento.

En la actualidad existen distintas aplicaciones para los sistemas expertos y se han implementado sistemas para medicina, electrónica, química y otras ramas de la ciencia.

Para comprender el alcance y la utilidad de los sistemas expertos es necesario conocer algunos conceptos que se presentarán a continuación.

¿Qué es un Sistema Experto?

Un sistema experto es un sistema de cómputo que emula la habilidad de tomar decisiones de un especialista humano[1], es decir, actúa en todos los sentidos como tal ante una situación que se encuentra dentro de un dominio particular del saber.

Está conformado por dos componentes básicos: la base de conocimientos y el mecanismo de inferencia, los cuales interactúan para simular (o emular) el proceso de solución de problemas de un experto.

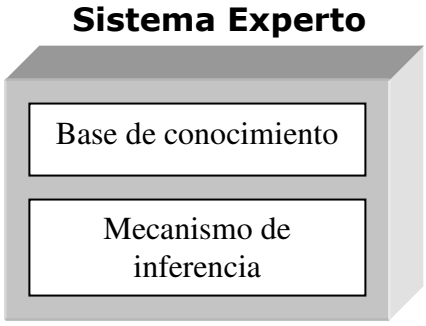


Ilustración 1: Componentes de un Sistema Experto

[1] Giarratano, pág. 2. Capítulo 1, Introducción a los sistemas expertos.

Funcionamiento

La siguiente tabla define los componentes de un sistema experto[2]:

Componente	Funcionalidad
Base de conocimientos	Colección de conocimiento requerido para la solución del problema
Mecanismo de inferencia	Valida los hechos disponibles, selecciona la fuente de conocimiento la base de conocimiento, compara los hechos con el conocimiento y genera hechos adicionales.

Tabla 1: Componentes de un Sistema Experto

El mecanismo de inferencia saca conclusiones basado en el conocimiento contenido en la base de conocimientos, las cuales da como respuesta a las consultas del usuario. El razonamiento que este implementa, está basado en reglas heurísticas que los expertos consideran eficaces.

Puede funcionar con datos inciertos y reglas imprecisas, usando algún método para manipular y determinar grados de credibilidad de datos y conclusiones.

[2] Krishnamoorthy, cap. 3 sección 3.3.

Dominio

Es importante tener en cuenta que un sistema experto es un "especialista" en un área específica del conocimiento, tal como medicina, finanzas, ingeniería, etc. Incluso puede ser especialista en una rama del área de conocimiento; por ejemplo, no se espera que un sistema experto en pediatría tenga conocimientos sobre cirugías o ginecología, y todas estas son ramas de la medicina.

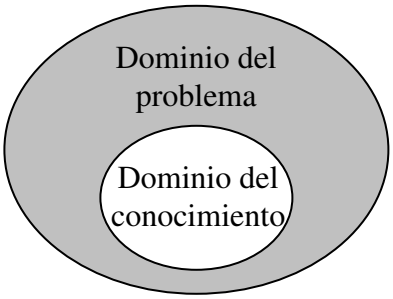


Ilustración 2: Dominio del problema

Un sistema experto resuelve problemas de su dominio restringido tan bien o mejor que un experto, pero si nos apartamos de su campo de especialidad puede fallar catastróficamente.

Características[3]

Entre las principales características de un sistema experto se encuentran:

- Eficiencia: la capacidad de respuesta debe ser igual o mayor a la de un experto humano.
- Tiempo de respuesta adecuado: Debe tomar una decisión en un tiempo comparable o menor que el del especialista.
- Confiabilidad: debe ser confiable cuando se esté trabajando dentro de su dominio del conocimiento.
- Comprensible: el sistema debe ser capaz de explicar los pasos de su razonamiento mientas se ejecutan, lo que hará que el sistema sea más comprensible y a su vez permita ver los errores en los mecanismos de inferencia o en la base de conocimiento.
- Flexibilidad: debe ser fácil agregar, editar o eliminar conocimiento.

Las ventajas y desventajas del uso de sistemas expertos se muestran en la tabla 2.

[3] Giarratano, págs. 4,5,8,9.

Ventajas	Desventajas
• Mayor accesibilidad de la experiencia gracias a su facilidad de duplicación y transferencia • Mayor disponibilidad • Costo reducido • El conocimiento puede ser usado de manera inmediata, descentralizada y duradera. • Experiencia múltiple (conocimiento de varios especialistas) • Pueden explicar claramente el razonamiento que conduce a una conclusión. • Da respuestas sólidas y sin emociones. • **Puede funcionar como un tutor inteligente, permitiendo que el**	• Está ligado a un dominio del conocimiento, y si el problema se sale de este, puede fallar catastróficamente. • La justificación de sus conclusiones generalmente se reduce a la enumeración de las reglas usadas. • No tiene conocimiento de sus alcances o limitaciones

estudiante ejecute programas de ejemplo y explicando el razonamiento del sistema.	

Tabla 2: Ventajas y desventajas de los S.E.

Diferencias entre un sistema convencional y un sistema experto

Un aspecto muy importante al considerar el desarrollo o utilización de un Sistema Experto, es la necesidad real de este. Si un problema puede ser resuelto eficazmente por medio de programación convencional, entonces un sistema experto no es la mejor opción.

Cuando intentamos dar solución a este tipo de problemas es posible que nos encontremos con una solución idéntica o menos eficaz que la solución algorítmica; es por esto que se hace necesario identificar los problemas que requieren de una estructura rígida, y en tal caso es preferible utilizar la programación estructurada.

Lenguajes, Shells, Herramientas

Con el paso de los años se ha venido utilizando los sistemas expertos como ayuda para asistir al ser humano en la toma de decisiones. Para esto se han desarrollado distintas maneras de intercomunicación hombre-máquina, los cuales se dividen en tres clases y son:

- **Shells:** es un entorno de programación que contiene utilerías necesarias para desarrollar y ejecutar un sistema experto[4]. Está compuesto por la base de conocimiento, el motor de inferencia, una memoria o área de trabajo, el mecanismo de explicación y las interfaces de usuario, desarrollo y sistema (APIs). Un ejemplo de Shell es EMYCIN (MYCIN –sistema médico– vacío).

- **Lenguajes:** son traductores de comandos escritos con una sintaxis específica. Estos proporcionan un mecanismo de inferencia que ejecuta las instrucciones del lenguaje. El uso de un lenguaje brinda flexibilidad en el desarrollo del S.E. ya que se puede desarrollar a la medida de lo necesario, pero al mismo tiempo requiere la programación del entorno completo de trabajo, incluido el motor de

[4] The handbook of applied expert systems. Capítulo 4, sección 1.

inferencia y el mecanismo de explicación. Como ejemplo de lenguajes tenemos PROLOG o C.

- **Herramientas:** es un lenguaje asociado con un programa de utilería para facilitar el desarrollo, depuración y uso de los programas de aplicación [5]. Puede contener editores de texto, depuradores y administradores de archivos, entre otros.

[5] Giarratano, Capítulo 1. Pág. 23.

CAPÍTULO 2:

Sistemas Expertos Basados en Conocimiento

En el estudio de los sistemas expertos han surgido distintas apreciaciones sobre el concepto de Base de conocimientos y algunos lo limitan a *solo un conjunto de reglas*, pero William Siler lo describe como más que eso, "la suma de conocimientos y habilidades" [6] ; el conocimiento relevante al problema se determina por medio de los datos, y las habilidades necesarias para usar el conocimiento disponible y dar solución a un problema se determina como reglas.

Arquitectura de un sistema experto basado en reglas

Un sistema experto basado en reglas está compuesto principalmente por:

[6] Siler, William. Capítulo 2, pág. 15.

- Interfaz de usuario: permite la comunicación entre el usuario y el sistema.
- Medio de explicación: explica el razonamiento del sistema.
- Memoria activa: registra los hechos usados por las reglas.
- Mecanismo de inferencia: decide que reglas satisfacen los hechos, les asigna prioridades y ejecuta las de prioridad más alta.
- Agenda: lista con prioridades asignadas a las reglas, creada por el mecanismo de inferencia.
- Medio para la adquisición del conocimiento: recibe información directamente del usuario sin necesidad de ser codificada por el ingeniero de conocimiento.

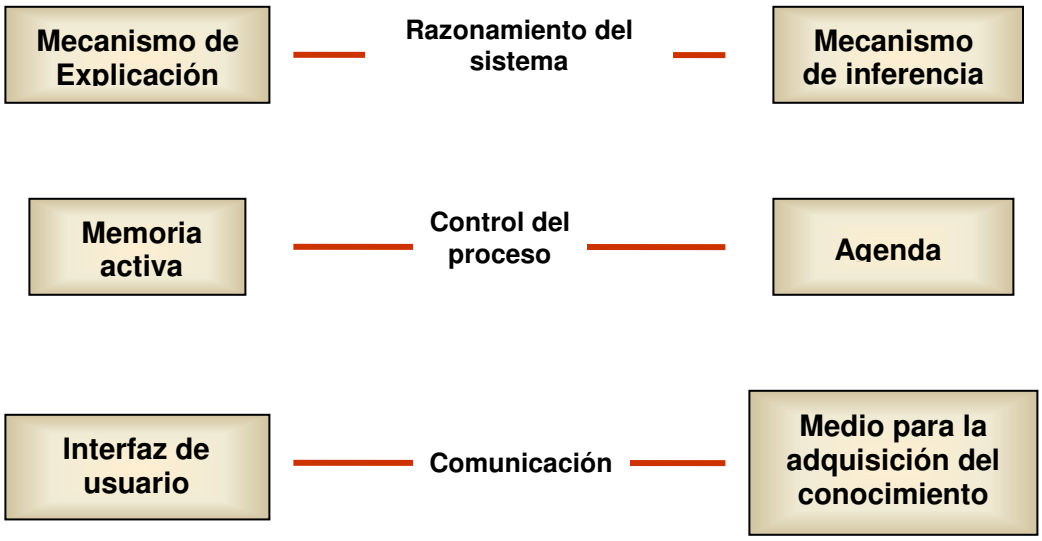


Ilustración 3: Arquitectura de un sistema experto

Representación del conocimiento

La base de conocimiento posee el conocimiento de un dominio específico para resolver problemas. Esta es creada por el ingeniero de conocimiento quien entrevista a los expertos y organiza el conocimiento de tal forma que pueda ser utilizado directamente por el sistema.

Podemos clasificar este conocimiento en tres categorías, como lo muestra la siguiente tabla.

Conocimiento	Descripción
Compilado	Resulta de la experiencia de los expertos en un dominio, libros, registros y especificaciones.
Cualitativo	Teorías aproximadas, modelos causales y el sentido común.
Cuantitativo	Teorías matemáticas, métodos numéricos

Tabla 3: Clasificación del conocimiento

Tanto el conocimiento compilado como el cualitativo se pueden subclasificar como declarativo (identificación de las características físicas del problema) y procedimental (técnicas de solución de problemas).

Para diseñar un sistema experto es necesario conocer los esquemas de representación del conocimiento y su posible interacción. La dificultad de esto no se encuentra en la representación del conocimiento sino en el uso.

La selección del esquema adecuado depende del tipo de aplicación que se esté desarrollando.

Los métodos de representación del conocimiento más comunes son:

- Lógica de predicados: provee mecanismos para la representación de hechos y razonamiento basado en la manipulación sintáctica de fórmulas lógicas. Usa reglas predefinidas para la deducción de los hechos.
- Reglas de producción: combinan la representación procedimental y declarativa dando así una mayor flexibilidad. Están formadas por un conjunto de *antecedentes* y un conjunto de *consecuencias*. Los antecedentes especifican un conjunto de condiciones y las consecuencias un conjunto de acciones.

  Un ejemplo sencillo es el siguiente:

```
Regla: Luz_roja
SI
      La luz es roja
ENTONCES
      Alto

Regla: Luz_verde
SI
      La luz es verde
ENTONCES
      adelante
```

Ilustración 4: Regla de producción

- Marcos (objetos) y redes semánticas: son ideales para la representación del conocimiento declarativo, describiendo entidades físicas y relaciones semánticas entre ellos. Es apropiado usar objetos con atributos encapsulados para tener una representación más estructurada de los hechos en el contexto de ejecución de un sistema experto, y las reglas deben interactuar con estos.

  Los marcos son estructuras que simulan el conocimiento común, representando conocimiento relacionado con un tema concreto que cuenta con mucho conocimiento predeterminado. Están compuestos por Ranuras (los registros) y sus Rellenos.

Ranuras	**Rellenos**
nombre	automóvil
tipos	(sedan, deportivo, convertible)
ruedas	4
motor	(gasolina, diesel)

Ilustración 5: Ejemplo de un marco

- Programas procedimentales: en algunas de las etapas del proceso de solución de problemas, es posible que el sistema realice cálculos

numéricos los cuales pueden ser representados por medio de funciones o programas en lenguajes de alto nivel tal como C. Estos programas forman parte de la base de conocimiento del sistema experto.

Mecanismos de Inferencia

Un mecanismo de inferencia es un conjunto de estrategias de control o técnicas de búsqueda que recorren toda la base de conocimientos para producir decisiones[7].

Un sistema experto procesa símbolos y el mecanismo de inferencia los manipula por medio de la selección de reglas, agrupando los símbolos de hechos y luego disparando las reglas para establecer nuevos hechos.

Este proceso se realiza continuamente como una cadena hasta alcanzar una meta. Una cadena es entonces "un conjunto de inferencias que conectan un problema con su solución"[8].

[7] Krishnamoorthy, Capítulo 3 Sección 3.2.
[8] Giarratano, Capítulo 3 – Métodos de inferencia, pág. 143.

En un sistema experto la inferencia se puede realizar de muchas maneras, pero los métodos más populares son el Encadenamiento hacia adelante (proceso basado en datos) y Encadenamiento hacia atrás (proceso basado en metas).

Encadenamiento hacia adelante

Es un grupo de inferencias que conectan un problema con su solución. En este proceso el usuario debe dar todos los datos disponibles antes de que comience la inferencia. Este mecanismo intenta establecer los hechos tal como aparecen en la base de conocimientos hasta alcanzar el objetivo.

En bases de conocimiento muy grandes es necesario hacer varias iteraciones antes de poder establecer un objetivo.

El orden en que aparecen las reglas juega un papel muy importante ya que de este depende el orden en que se formulen las preguntas al usuario y se dificulta la explicación debido a que no se conocen claramente los subobjetivos, sino a media que se van presentando.

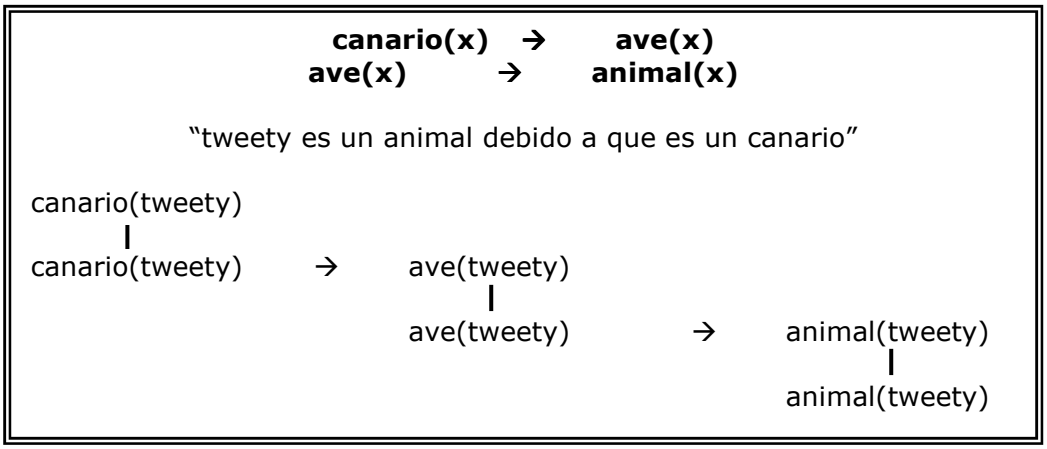


Ilustración 6: Cadena causal hacia adelante

Encadenamiento hacia atrás

Este proceso intenta establecer las metas en el orden en que estas aparecen en la base de datos. En este se define una variable objetivo y cuando esta obtiene un valor el proceso se detiene.

Las tres estructuras de datos dinámicas usadas durante el proceso de inferencia son: la memoria de trabajo (contexto), una pila de reglas y una pila de metas.

Cuando se realiza alguna de las acciones del proceso de inferencia tales como selección, agrupación o ejecución, una o más de estas estructuras se modifica.

El sistema de explicación provee un mecanismo para consultar la memoria de trabajo, y esto se realiza con facilidad debido a que el sistema puede decir con exactitud qué objetivo trata de cumplir.

De esta manera se puede obtener contenido del contexto (responde a preguntas del tipo *¿que?*), y saber como se estableció un hecho (responde a preguntas del tipo *¿como?*). Para responder a la primera consulta simplemente se devuelve el valor del contexto, mientras que para responder la segunda consulta es necesario hacer referencia a la base de conocimiento.

Encadenamiento hacia adelante	Encadenamiento hacia atrás
Planeación, supervisión, control	Diagnóstico
Presente a futuro	Presente a pasado
Antecedente a consecuencia	Consecuencia a antecedente
Controlado por datos	Controlado por objetivos
Razonamiento de abajo hacia arriba	Razonamiento de arriba hacia abajo

Intenta encontrar cuales soluciones se desprenden de los hechos	Intenta encontrar los hechos que sustentan las hipótesis
Los antecedentes facilitan la búsqueda (a lo ancho)	Las consecuencias facilitan la búsqueda (a fondo)
Explicación no facilitada	Explicación facilitada

Tabla 4: Características del encadenamiento hacia delante y hacia atrás

Razonamiento inexacto

Una de las principales capacidades de los expertos humanos y la más difícil de replicar en un sistema experto es la habilidad de manejar datos imprecisos, incompletos y algunas veces información incierta. En la actualidad se han hecho distintos esfuerzos para incorporar el razonamiento inexacto basado en información incierta.

La incertidumbre está presente en la mayoría de los algoritmos para sistemas expertos debido a que raramente se puede estar totalmente seguro de las afirmaciones que estos realicen. Esta proviene de distintas fuentes en una variedad de formas.

Las técnicas que han sido implementadas en los sistemas existentes son en su mayoría extensiones de unas cuantas técnicas principales, tales como la teoría de probabilidades, aproximación heurística, niveles de certeza asociados a hechos, conjuntos difusos (diseñados para manejar imprecisiones lingüísticas), entre otros. Algunos de ellos son de tipo intuitivo y de difícil representación matemática.

Las mayores fuentes de incertidumbre en una base de conocimientos pueden ser las siguientes: información no confiable, lenguaje descriptivo impreciso, inferencia con información incompleta y una mala combinación del conocimiento de distintos expertos.

El proceso de razonamiento requerido para inferir en situaciones prácticas que están generalmente sin estructurar, debe tener algunas de estas características:

- Al menos en alguna instancia del proceso de toma de decisiones, alguna parte de la información está incompleta.
- Las condiciones pueden cambiar en el tiempo
- Es necesario hacer deducciones eficientes, pero posiblemente incorrectas cuando el razonamiento termine sin resultados.

CAPÍTULO 3:

Diseño de un Sistema Experto

Conociendo los aspectos fundamentales sobre un sistema experto podemos proceder al diseño y desarrollo de estos, utilizando en este caso particular el lenguaje CLIPS debido a que es uno de los más importantes en la actualidad y provee herramientas de gran ayuda.

Metodologías para el desarrollo de Sistemas Expertos

Uno de los métodos más importantes en la ingeniería de software es el ciclo de vida, que es la descripción de todo el software desde su inicio hasta el retiro de su uso.

El concepto de ciclo de vida proporciona una continuidad que conecta todas las etapas desde la planeación hasta el desarrollo y mantenimiento.

El ciclo de vida se considera una metametodología debido a que determina el orden en el que se aplican los métodos para el software común y su duración. También define etapas comunes a todos como son: planeación, requisitos, adquisición de conocimiento, pruebas, representación de productos de etapa, documentación, código y diagramas.

Algunos de los modelos más utilizados en el desarrollo de sistemas expertos son:

Modelo de cascada

Este es el modelo clásico para desarrollo de software convencional donde las etapas se desarrollan en forma secuencial y al final de cada una se realizan actividades de validación y verificación (V & V) para minimizar problemas en estas.

Las etapas de este modelo con sus respectivas (V & V) son:

- Factibilidad del sistema / validación
- Planeación y requisitos del software / validación
- Diseño del producto / verificación
- Diseño detallado / verificación
- Código / prueba de unidad

- Integración / verificación del producto
- Implementación / prueba del sistema
- Operaciones de mantenimiento / revalidación

Modelo de código y reparación

Es uno de los modelos más primitivos y consiste en escribir código y cuando este deje de funcionar correctamente, realizar las correcciones necesarias.

Esto también llevó al concepto de hacer las cosas dos veces, ya que es posible realizar un prototipo sin tener aun los requisitos del sistema, y luego se rediseña en base a estos.
Este modelo es usado aun por principiantes en el desarrollo de programas convencionales y sistemas expertos.

Modelo progresivo

Es una extensión del modelo en cascada donde cada etapa se desarrolla por medio de incrementos progresivos haciendo más fácil la

corrección de problemas, la verificación y validación de cada producto individual por etapas.

Es similar a un prototipo rápido continuo que se extiende sobre todo el desarrollo y que finalmente evoluciona en lo que es el sistema.

Modelo en espiral

Esta es una adaptación del modelo progresivo donde cada circuito de la espiral agrega alguna capacidad funcional al sistema. El punto final del espiral indica el inicio del mantenimiento y la evolución del sistema.

Modelo lineal

Este es un modelo utilizado con éxito en varios proyectos de desarrollo de sistemas expertos. Está formado por varias etapas que van desde la planeación a la evaluación del sistema, realizándose varias repeticiones del proceso hasta que esté disponible para su uso.

Puede considerarse también como un circuito en espiral donde cada etapa está formada por tareas, las cuales dependerán del tipo de aplicación que se esté construyendo.

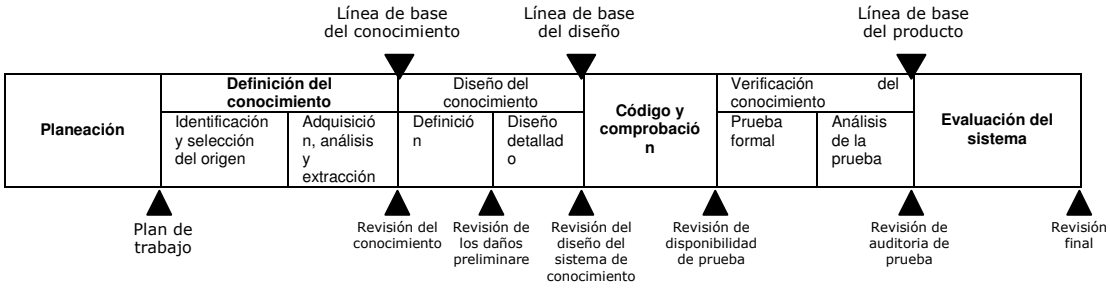


Ilustración 7: Modelo lineal del ciclo de vida

Descripción de las etapas:

- **Planeación**

  Consiste en la elaboración de un plan de trabajo formal para desarrollar el sistema experto, el cual está formado por un grupo de documentos que guiarán y evaluarán el desarrollo.

  Aquí se realizan distintas tareas tales como: valoración de factibilidad, administración de recursos, determinación de las

fases de las tareas, calendarización, disposición funcional preliminar y requisitos de alto nivel.

- **Definición del conocimiento**

  Define el conocimiento requerido por el sistema experto. Está compuesta por a) identificación y selección del origen del conocimiento, y b) adquisición, análisis y extracción del conocimiento.

  El objetivo principal de estas tareas es producir y verificar el conocimiento que necesita el sistema. Además del método convencional de entrevista con el experto para la adquisición del conocimiento, es posible utilizar técnicas automáticas tales como las rejillas de repertorio o la teoría de construcción personal.

- **Diseño del conocimiento**

  Consiste en la producción del diseño detallado para el sistema experto. Está compuesto por la definición del conocimiento (representación del conocimiento, estructura de control detallada, estructura interna de hechos, interfaz preliminar del usuario y plan de pruebas inicial) y el diseño detallado

(estructura del diseño, estrategias de implantación, interfaz detallada del usuario, especificaciones e informe de diseño, plan detallado de prueba).

El producto de esta etapa es el documento de diseño que se toma como referencia para proceder a la codificación. Este documento debe ser revisado y verificado antes de codificar.

- **Código y verificación**

  Esta etapa consiste en un grupo de tareas que abarcan la codificación, las pruebas del mismo utilizando datos de prueba, producción de la documentación, elaboración de manuales de usuario para que los especialistas y usuarios puedan retroalimentar el sistema, guía de instalación y operación, y el documento de descripción del sistema donde se describe la funcionalidad, limitantes y problemas globales.

  Esta etapa finaliza con la revisión de disponibilidad de prueba que determina si el sistema está listo para pasar a la siguiente etapa.

- **Verificación del conocimiento**

  El objetivo de esta etapa consiste en determinar que el sistema sea correcto, esté completo y sea congruente. Está compuesta por las pruebas formales y el análisis de pruebas.

  El análisis de prueba busca los siguientes problemas importantes: respuestas incorrectas, respuestas incompletas y respuestas incongruentes.

- **Evaluación del sistema**

  Esta es la etapa final del proceso y consiste en resumir todo lo aprendido con las mejoras y correcciones recomendadas.

  Al final de cada iteración es necesario producir informes provisionales que describan la funcionalidad incrementada del sistema a medida que se añade nuevo conocimiento.

CAPÍTULO 4:

JESS Desarrollo de Sistemas Expertos en JAVA

Este capitulo tiene como objetivo dar una introducción a JESS y familiarizarlo con las funcionalidades básicas de este, de tal forma que pueda Interpretar el código elaborado en el Capitulo 5.

JESS (Java Experts Systems Shell) es un motor de reglas que se desarrollo utilizando 100% el lenguaje de programación Java. En un principio JESS estuvo bajo la influencia de otro poderoso motor de reglas llamado CLIPS (C Language Integrated Production System), pero en los recientemente JESS ha evolucionado de tal forma que ya es totalmente independiente de CLIPS.

Como se ha venido comentando en el transcurso de este documento, los Sistemas Expertos tratan de representar el conocimiento para su posterior utilización, JESS ofrece al programador 3 formas de representar el conocimiento:

- Reglas: Que se basan fundamentalmente en el conocimiento Heurístico basado en la experiencia.
- Funciones: Enfocadas al Conocimiento Procedimental.
- Programación Orientada a Objetos: Enfocado al Conocimiento Procedimental pero representado mediante un paradigma diferente a las funciones.

Todo shell para el desarrollo de sistemas expertos debe proveer al programador los elementos básicos para el desarrollo de un Sistema Experto:

- Hechos: Fragmentos de información que le permiten al Sistema Experto razonar.
- Base de Conocimiento: Contiene la lista de todas las reglas que tiene el Sistema Experto.
- Motor de Inferencia: Controla la ejecución de las reglas.

Notación

Para mayor entendimiento de los textos que se presentaran en este capitulo la notación a utilizar para explicar la sintaxis de JESS es la siguiente:

- Todos los caracteres que no se encuentren encerrados en los caracteres [], <> deberán introducirse exactamente como se muestra.

- Los caracteres que se encuentran encerrados entre [] indican que son opcionales.
- Los caracteres encerrados en <> indican que deberán ser reemplazados por un tipo de dato especifico que representan.

Por ejemplo:

- (ejemplo): deberá ser digitado igual.
- (ejemplo [1]): Podrá ser digitado de 2 formas, (ejemplo) y (ejemplo 1).
- <entero>: Indica que se debe reemplazar con un valor entero.

Hechos

Como ya se ha mencionado anteriormente, los hechos son fundamentales en los Sistemas basados en reglas, ya que, estas operan sobre los hechos existentes generando nuevos hechos que dispararan nuevas reglas y así hasta llegar a una solución optima para un problema determinado.

Plantillas de Definición

En JESS como en CLIPS, deben definirse plantillas para los grupos de hechos que comparten un mismo nombre de relación y que tienen información en común. Una plantilla se define de la siguiente manera:

(deftemplate <nombre-relacion> [<comentario-opcional>]

<definición-ranura>*)

Ilustración 8: Definición de una plantilla

y <definición-ranura> se define de la siguiente forma:

(slot <nombre-ranura>) | (multislot <nombre-ranura>)

Ilustración 9: Definición de un hecho

Hechos Ordenados

Se les llama hechos ordenados a aquellos que no están relacionados por ninguna plantilla de definición, aunque el uso de las plantillas es muy recomendado pues facilita la legibilidad de los hechos.

Visualización de la Lista de Hechos

Para visualizar todos los hechos que se encuentran creados en nuestra Base de Conocimiento se utiliza el comando **(facts)**, el cual mostrara una lista de los hechos, cada uno con un numero que lo identifica.

Adición y Eliminación de Hechos

Para la adicion y eliminación de los hechos se utilizan 2 sentencias propias del lenguaje:

- assert: Agrega un hecho a la lista de hechos. La sintaxis para su utilización es la siguiente: **(assert <hecho>+)**
- retract: Remueve un hecho determinado. La sintaxis para su utilización es la siguiente: **(retract <hecho-indice>)**.

Modificación y Duplicación de Hechos

La sintaxis para la modificación de los hechos es la siguiente:

(modify <indice-hecho> <modificador>)

, donde <modificador> se define de la siguiente forma

(<nombre-slot> <nuevo-valor>).

La sintaxis para la duplicación de los hechos es la siguiente:

(duplicate <indice-hecho> <modificador>)

, donde <modificador> se define de la siguiente forma

(<nombre-slot> <nuevo-valor>).

La diferencia de utilizar modify y duplicate radica en el hecho que modify remueve el hecho modificado y crea uno nuevo, mientras que duplicate, crea un nuevo hecho pero no borra el anterior.

Creación de Hechos Iniciales

La mayoría de los Sistemas expertos, tienen una base de conocimiento inicial que les permite definir nuevos hechos y tomar un determinado comportamiento al cual nosotros le denominamos "Conocimiento Inicial". Para crear hechos iniciales JESS tiene el comando **(deffacts)** el cual tiene la siguiente sintaxis:

(deffacts <nombre> <hechos>*)

Reglas

Las reglas como los Hechos, son la base fundamental de un Sistema Experto, JESS como CLIPS maneja las reglas del tipo:

SI <condiciones> ENTONCES <acciones>

La sintaxis para la definición de una regla es la siguiente:

(defrule <nombre> <condicion>* => <accion>*)

Hay que tener ciertas consideraciones presentes a la hora de trabajar con las reglas:

- Las reglas se ejecutan cuando se cumplen las condiciones, es decir, las acciones solo se ejecutaran si las condiciones se satisfacen.
- El orden en que son creadas no tiene nada que ver con el orden en que se ejecutan**.**
- Las reglas no se disparan mas de una vez para un mismo hecho.

Variables

Como otros lenguajes de programación JESS maneja variables para almacenar datos y a diferencia de los hechos que son estáticos y no puede ser cambiados (como se vio anteriormente, cuando se modifica un hecho, en realidad se crea uno nuevo y se borra el anterior), el contenido de una variable es dinámico.

La sintaxis para la definición de una variable es la siguiente:

?<identificador-variable>

Hay que tener en cuenta que el alcance de una variable, esta definido para la regla en que esta es declarada, exceptuando las variables globales, las cuales están disponibles para cualquier regla que la quiera utilizar, si se

intenta utilizar una variable que no esta definida en la regla que la esta invocando, JESS arrojara un error indicándolo.

CAPÍTULO 5:

Prototipo de una herramienta de evaluación

Este capitulo esta enfocado a explicar como fue el proceso de desarrollo del prototipo para la evaluación de Ejercicios. Este prototipo contiene información de gran utilidad para el programador principiante de Sistemas Expertos, ya que muestra la integración entre JESS con el lenguaje de programación JAVA y como una aplicación puede reaccionar en Base a las instrucciones del experto.

El propósito de este prototipo es mostrar la relación entre el código fuente escrito en JAVA y el Sistema experto desarrollado en el lenguaje CLIPS enlazados por medio de la herramienta JESS para obtener como resultado un Sistema Experto totalmente funcional capaz de realizar operaciones, funciones avanzadas (cálculos matemáticos, etc..). y conservando la capacidad de realizar razonamiento basado en la lógica de las reglas establecidas en su Base de Conocimiento.

El desarrollo del prototipo se dividió en las siguientes etapas:

- Selección e Investigación del Tema
- Desarrollo del Prototipo
- Funcionamiento del Prototipo

Selección e Investigación del Tema

Todo Sistema experto que se desee desarrollar debe tener un dominio de conocimiento específico en el cual el sistema cuenta con un conjunto de reglas y hechos que le permiten resolver apropiadamente situaciones que requieran una solución lógica.

El dominio que se trabajó para el desarrollo del prototipo es **"Solución de ecuaciones de una variable"** utilizando métodos numéricos definidos para este propósito:

- Método de la Bisección
- Método de Newton-Raphson

Definido el tema, se realizó un estudio de cada uno de los métodos, para obtener un conjunto de hechos y reglas que permitan al sistema evaluar un problema dado y determinar su solución.

Es importante destacar que el dominio de aplicación es amplio y su complejidad puede variar, por lo que el prototipo desarrollado se limita a la evaluación de funciones de una variable.

Desarrollo del Prototipo

El prototipo desarrollado en este trabajo es una demostración de las ventajas que ofrece la elaboración de un sistema experto, por lo que los temas que no fueron desarrollados se dejan como una iniciativa para su posterior implementación.

El esquema que se utilizo para el desarrollo del prototipo es el siguiente:

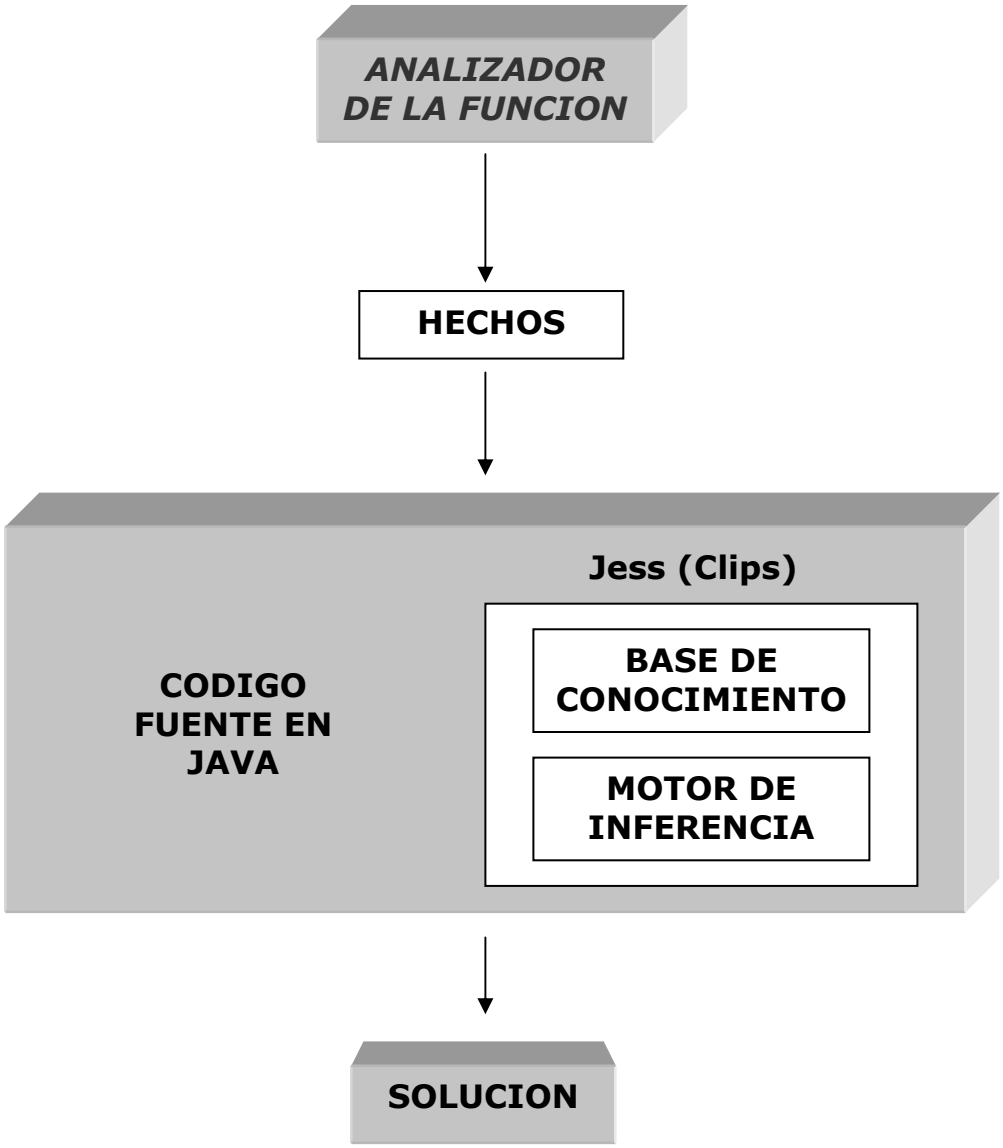


Ilustración 10: Esquema del prototipo

- **Analizador:** No fue desarrollado debido a que no hace parte del desarrollo de este trabajo. Su función es generar los hechos que serán

utilizados por la Base de Conocimiento a partir de un problema planteado por un usuario, para efectos de demostración, las funciones planteadas están predefinidas y se utiliza una pequeña clase para resolver la función dependiendo del método seleccionado por el experto.

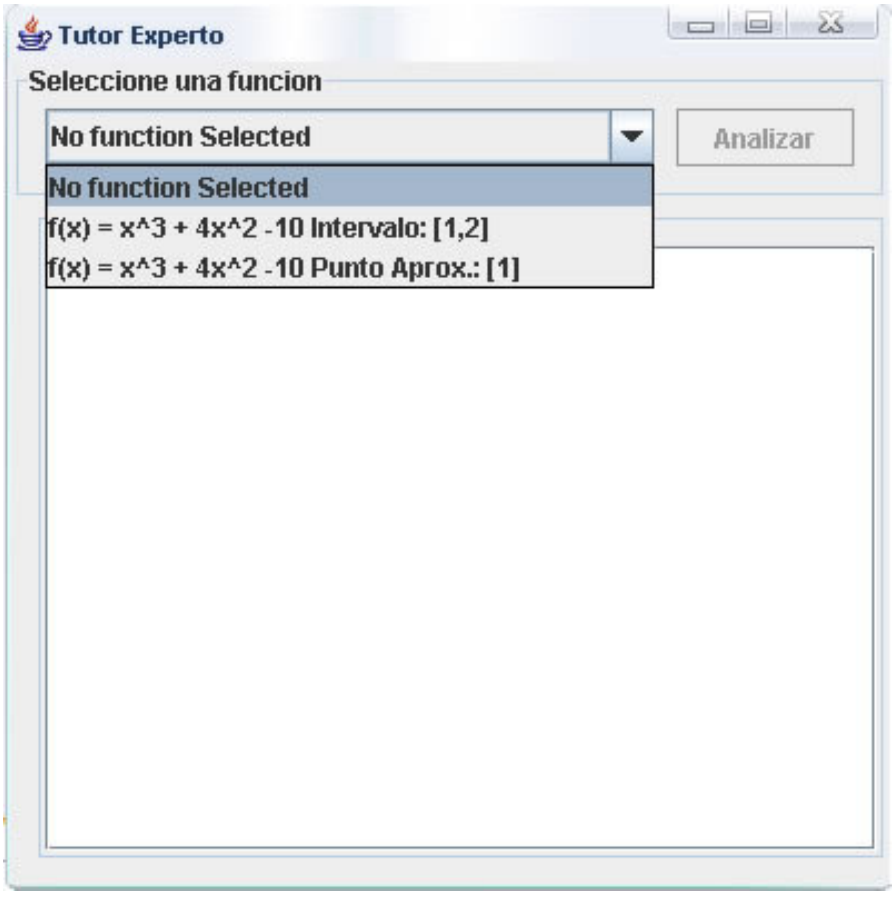
El analizador deberá tomar la función ingresada por el usuario y realizar un análisis profundo de esta para poder generar hechos que el experto entienda y pueda tomar como referencia para darle solución a cualquier problema propuesto, por ejemplo una de las principales características del Método de Newton Raphson sobre los demás métodos de resolución como el Método de la Bisección es que el método de Newton resuelve el problema en menor número de Iteraciones que el método de la Bisección, por lo que por eficiencia el experto deberá elegir resolver un método por Newton en lugar de Bisección, pero este es más útil utilizarla en lugar de Newton cuando la primera derivada de la función tiene una solución difícil de hallar, por eso el analizador debe de alguna generar hechos de este tipo para poder definir reglas en la Base de Conocimiento que permitan una análisis mas profundo y detallado de las características de cada uno de los métodos de solución de ecuaciones de una sola variable.

- **Base de Conocimiento:** Define el conjunto de reglas y acciones que permiten al experto tomar la mejor decisión en un momento determinado. La base de Conocimiento desarrollada contiene solo las reglas fundamentales que caracterizan a cada uno de los métodos de solución de ecuaciones de una variable; estas se definieron para el Método de la Bisección y para Newton-Raphson ya que eran suficientes para mostrar las capacidades del experto. El Shell utilizado para el desarrollo de la base de conocimiento fue JESS, por la facilidad que este presenta en la manipulación de objetos elaborados en el lenguaje JAVA, de esta forma pueden definirse Objetos en JAVA que interactúen con el experto para obtener un mayor rendimiento en la aplicación y dándole una especie de inteligencia a la misma.

- **Interfase de Usuario:** Desarrollada en JAVA, brinda una interfaz amigable donde el usuario puede interactuar con la Base de Conocimiento. Aunque la interacción ofrecida por el programa es poca debido a que la Base de Conocimiento esta en sus inicios, se pretende que mas adelante el usuario pueda formular preguntas y obtener explicaciones para un problema determinado, dentro de los limites del dominio del experto.

Funcionamiento del Prototipo

Antes de explicar el funcionamiento del prototipo hay que dejar claro que su funcionalidad es limitada debido a la gran cantidad de restricciones en el dominio de nuestra investigación, pero su aporte a aquellas personas que están iniciando sus conocimientos en el desarrollo de Sistemas Expertos se basa en la integración de JESS con el lenguaje de programación JAVA lo cual permite mayor flexibilidad al momento de programar este tipo de sistemas.



El programa desarrollado emula el conocimiento de un experto en la solución de ecuaciones de una variable utilizando uno de los métodos numérico definidos en la Base de Conocimiento.

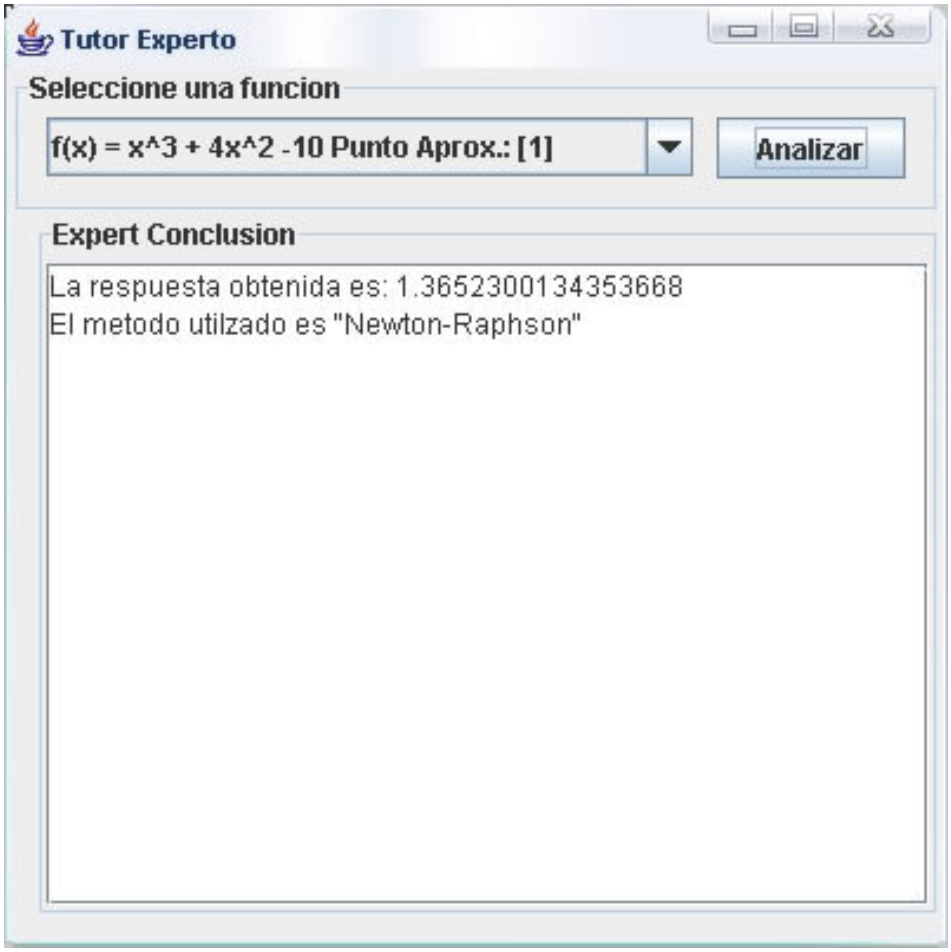
Esto se logro mediante la definición de reglas que permiten al sistema identificar las características fundamentales de cada uno de los métodos numéricos y tomar una decisión acertada al momento de dar solución a un problema planteado.

Para llegar a una solución acertada el experto debe partir de unos hechos que le permitirán decidir cual de los métodos definidos es el que mejor resuelve el problema planteado, estos hechos son generados por la interfase y enviados al experto, el cual los analiza y le indica a la aplicación cual es la forma en que debe solucionarlo.
El sistema propone un conjunto de funciones para probar el conocimiento del experto y mostrar que sus respuestas están influenciadas por la Base de Conocimiento. Los hechos exigidos por la Base de Conocimiento para llegar a una conclusión también están predefinidos para cada una de las funciones propuestas por el sistema.

Una vez el usuario elige una función se deben tomar los hechos relacionados con ella y enviárselos al experto para que este le de una solución una optima

basándose en su conocimiento, como los hechos están predefinidos, esta operación resulta sencilla y una vez los hechos han sido enviados al experto este ejecuta la función adecuada para resolver el problema planteado.

La función ejecutada por el experto varia dependiendo de la solución que halla determinado, esta función esta definida en una clase desarrollada en JAVA, la cual es una solución algorítmica al problema planteado.



Se realizo de esta forma siguiendo las pautas descritas en capítulos anteriores para el desarrollo de un Sistema Experto, en la cual se trata de omitir toda solución procedimental de la Base de Conocimiento.

Como resultado el sistema indica al usuario el método que selecciono para la solución del problema planteado, adicionalmente le muestra la solución del mismo, para que se pueda apreciar el razonamiento realizado por el experto, ya que, como es sabido, los resultados obtenidos por estos métodos manejan un margen de error dependiendo del número de iteraciones realizadas.

Estos sistemas han tenido un gran auge en diferentes campos de la ciencia, dada su capacidad para emplear el conocimiento, pudiendo ser aplicados en la Industria, el Comercio y la Educación, entre otros, siendo esta ultimo nuestro principal objetivo de investigación.

Las aplicaciones a nivel educativo que se pueden desarrollar con un Sistema Experto son muchas dada la capacidad de razonamiento que este ofrece en un dominio de conocimiento, entre estas aplicaciones se destacan:

- Tutores: Los cuales ofrecen una guía para los estudiantes en la solución de problemas.

- Evaluadores: Verifican la solución entregada por el estudiante y proveen un mecanismo de explicación para profundizar en los temas que casen mayor dificultad a los estudiantes.

La funcionalidad ofrecida por el prototipo desarrollado para el desarrollo de este trabajo, es solo una base para el desarrollo de un tutor más complejo y con mayor capacidad de razonamiento.

RECOMENDACIONES

Por medio de la experiencia obtenida al desarrollar este proyecto aprendimos que los Sistemas Expertos son un arma de doble filo ya que estos pueden resolver problemas de una forma fácil y eficiente mientras que su solución siguiendo la programación convencional resulta compleja, pero su desarrollo requiere de cierta experiencia en lo que se conoce como el paradigma de la programación declarativa, la cual puede representar un obstáculo para un programador inexperto.

En el desarrollo de los sistemas expertos uno de los errores en los que frecuentemente se incurre es en creer que todo lo conocemos; esto puede resultar en un mal diseño de la base de conocimiento, pues si no se posee la suficiente experiencia en el dominio del sistema, el modelado de la base de conocimiento podría ser inadecuado. Por esto sugerimos que para realizar un buen modelado de la base de conocimiento deben tenerse en cuenta los siguientes pasos:

- Integrar al grupo de desarrollo uno o más expertos quienes podrán colaborar en la elaboración de los modelos que servirán para diseñar estrategias que le permitan a la Base de Conocimiento aprender nuevas reglas, desarrollando de esta forma un Sistema Experto robusto y escalable.
- Limitarse al desarrollo de los temas del dominio de aplicación del sistema experto. Agregar temas fuera del dominio puede hacer el sistema más ineficiente e inexacto.

Se propone para continuar con el desarrollo de esta investigación, lo siguiente:

- Desarrollo de un Analizador: El cual permitirá evaluar cualquier tipo de función matemática ingresada para de esta forma generar más hechos que permitan al experto tomar una mejor decisión.
- Mecanismo de Explicación: Esta parte es fundamental si se quiere desarrollar un tutor, ya que, este debe ser capaz de explicar el porque de su respuesta.
- Interfase para incrementar el conocimiento del Experto: Fundamental si se quiere desarrollar un sistema flexible y que pueda obtener conocimiento de una manera fácil sin tener que reprogramar la Base de Conocimiento.

Los temas desarrollados en este documento realizaron una introducción al lector en el mundo de los Sistemas Expertos. Esperamos que la lectura de cada uno de los capítulos haya generado dudas e inquietudes, las cuales el lector podrá resolver investigando otras fuentes de información que abarquen con mayor profundidad los temas aquí tratados.

BIBLIOGRAFÍA

GIARRATANO, Joseph. SISTEMAS EXPERTOS, Principios y programación. Tercera Edición. International Thomson Editores. México, 2001.

LIEBOWITZ, Jay. The Handbook of APPLIED EXPERT SYSTEMS. CRC Press LLC, 1999.

SILER, William. FUZZY EXPERT SYSTEMS AND FUZZY REASONING. John Wiley & Sons, Inc. New Jersey (USA), 2005.

KRISHNAMOORTHY, C. Artificial Intelligence and Expert Systems for Engineers. CRC Press, CRC Press LLC. 1996.

Clips Reference Manual. Volume 1: Clips Basic Programming Guide. Versión 6.23, 2005.

Clips Reference Manual. Volume 2: Clips Advanced Programming Guide. Versión 6.23, 2005.

Clips Reference Manual. Volume 3: Clips Interfaces Guide. Versión 6.23, 2005.

Artículo:

EL SOFTWARE EDUCATIVO

http://www.filos.unam.mx/POSGRADO/seminarios/pag_robertp/paginas/soft_edu.html

Artículo:

LAS NUEVAS TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN EN LA EDUCACIÓN

http://tecnologiaedu.us.es/bibliovir/pdf/tema6.pdf

Web docente de Tecnología Educativa. Universidad de La Laguna. 2002

Manuel Area Moreira

Lectura:

"Simulación: la revolución educativa"

Escribe: **Adriana Jiménez Revorio**

http://contexto-educativo.com.ar/2001/3/nota-07.htm

Artículo:

Introducción a los sistemas expertos

http://www.redcientifica.com/doc/doc199908210001.html

Definición de sistema experto en WIKIPEDIA:

http://es.wikipedia.org/wiki/Sistema_experto

APÉNDICES Y ANEXOS

SINTÁXIS BÁSICA EN JESS - CLIPS

Ejecución de un programa:

- Inicialización

  (reset)

  Borra contenido de la lista de hechos y la agenda, carga en la lista de hechos los hechos iniciales, carga en la agenda las reglas activadas

- Evaluación

  (mientras la agenda no esté vacía)

  (run)

  Selecciona la primera regla de la agenda, dispara la regla seleccionada, activar nuevas reglas

- Órdenes básicas en momento de ejecución

(reset): inicialización

(run): evaluación del programa hasta que se vacíe la agenda o encuentre (halt)

(run x): x evaluaciones consecutivas

(clear): limpia el entorno

Hechos ordenados

- Sintaxis:

  (<nombre-de-relación> <valor>*)

  Ejemplos:

  (alarma-conectada)

  (temperatura 20)

  (es-un pajaro animal)

- Carga de los hechos iniciales en la base de conocimiento:

  (deffacts <nombre> [<comentario>] <hecho>*)

  Ejemplo:

  (deffacts hecho (animal pajaro))

  Los hechos iniciales se cargan después de ejecutar (reset)

- Carga de un hecho inicial por defecto

  (initial-fact) <nombre> <hecho>

  (facts) : lista los hechos en la memoria de trabajo

  (get-fact-list) : lista las direcciones de los hechos

  Ejemplos:

  (facts)

  (deffacts hecho0 (fase 2))

  (deffacts hecho1 (nivel a))

  (deffacts hecho2 (nivel c))

  (reset)

  (facts)

- Para borrar hechos de la base de conocimiento:

  Individualmente: **(undeffacts <nombre>)** o **Browse**/**Deffacts Manager**

  Todos los hechos: **(clear)** o **Execution**/**Clear**

  Ejemplos:

  (clear)

(deffacts hecho0 (fase 2))

(deffacts hecho1 (nivel a))

(reset)

(facts)

(undeffacts hecho2 (nivel c))

(reset)

(facts)

Reglas

- Reglas en la base de conocimiento

  (defrule <nom-regla> [<comentario>] <elemento-codicional>* => <acción>*)

- Acciones con las reglas
  - Activar:
    i. todos los elementos condicionales se satisfacen
    ii. La agenda recoge las reglas activas en cada momento
  - Disparar
    i. Se ejecutan secuencialmente las acciones de la parte derecha de la regla

Ej.:

(clear)

(deffacts h0 (fase 2))

(defrule r0 (fase 2) => (assert (x0)))

(defrule reglaB => (reset))

(reset)

- Elementos condicionales booleanos

  Conjunción lógica: (and <elemento-condicional>+)

  Disyunción lógica: (or <elemento-condicional>+)

  Negación lógica: (not <elemento-condicional>)