22
2 ej

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

INTERFAZ GRAFICA-TEXTUAL PARA EL MECANISMO DE INFERENCIA MICROEXPERT

T E S I S

QUE PARA OBTENER EL TITULO DE
INGENIERO EN COMPUTACION
P R E S E N T A N :

CORTES BECERRIL JOSE HECTOR
ROSALES ONTIVEROS JUAN CARLOS
TRUJILLO ESTEVEZ ABRAHAM

ASESORES: ING. LUIS ADRIAN LETEPICHIA F. (†)
ING. MA. EUGENIA RAMIREZ N.

MEXICO, D. F. ABRIL 1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

INTRODUCCION

El desarrollo de una sociedad esta basado en el nivel del conocimiento aplicado. El conocimiento y aplicación de la computación en nuestra sociedad es básico; sin él no es posible un desarrollo equilibrado, tanto las personas como las empresas que no aplican la informática en sus respectivos ámbitos de trabajo están condenados a no ser competitivas.

Es por ello, que actualmente se requiere desarrollar programas de aplicación en computadora con diseños eficaces en cuanto a la interfaz hombre-máquina se refiere, para incrementar la eficiencia, funcionalidad y uso de la computadora.

Una interfaz hombre-máquina se debe diseñar utilizando diversas técnicas que existen actualmente para el manejo de menúes, ventanas, color, despliegue de texto, etc. Estas técnicas bien utilizadas, ofrecen al diseñador una base muy importante para crear interfases que cumplan con los requerimientos ergonómicos.

En este estudio se diseñará e implementará una interfaz que servirá como ayuda para la construcción de sistemas expertos, que son sistemas de software que tienen una alta demanda en el mercado actual.

Un sistema experto esencialmente tiene dos componentes principales, una máquina de inferencia y una base de conocimientos. La base de conocimientos es el almacén de los conocimientos de expertos humanos que se pueden representar formalmente de varias maneras; en general, se utilizan un conjunto de reglas que puede interpretar la computadora. La máquina de inferencia es el corazón del sistema experto y es la encargada de realizar las inferencias lógicas sobre la base de conocimientos para llegar a una o varias conclusiones.

Existe un tipo de herramientas denominadas esqueletos o cascarones (shells), dentro de los cuales un sistema experto puede ser construido. La diferencia primordial existente entre los sistemas expertos tradicionales y los cascarones, radica en la posibilidad que presentan estos últimos de modificar su base de conocimientos, permitiendo de esta forma utilizarlos en diversas aplicaciones.

Debido a lo anterior se desarrollará una interfaz basada en un shell de sistema experto. Dicho esqueleto será Microexpert, potente máquina de inferencia capaz de manipular cualquier base de conocimientos sin importar el campo de aplicación, siempre y cuando esta base cumpla con una sintaxis predefinida. Esta sintaxis no

es fácil de entender y es por ello necesario crear una interfaz transparente para el usuario y con la cual sea capaz de crear sus propios sistemas expertos sin que tenga que profundizar en conocimientos internos del mecanismo y de sistemas expertos. Además Microexpert carece de un ambiente accesible para el usuario, al no contar con herramientas tales como : administración y consulta de archivos, módulos de impresión de reportes y un módulo de representación gráfica de la base de conocimientos, características que la interfaz contemplará combinando un ambiente gráfico-textual.

El presente trabajo consta principalmente de cuatro partes :

Capítulo I .- Fundamentos de Sistemas Expertos
Capítulo II .- Mecanismo de Inferencia Microexpert
Capítulo III .- Análisis y Diseño de la Interfaz
Capítulo IV .- Ejemplos de Aplicación

En el primer capítulo se hace una reseña de los conceptos más importantes de los Sistemas Expertos incluyendo su entorno dentro de la Inteligencia Artificial, historia, características y aplicaciones en el mundo real.

En el capítulo dos se presenta una descripción detallada del manejo, la representación del conocimiento y funcionamiento de Microexpert, mencionando sus características principales.

El tercer capítulo presenta la metodología propuesta para la construcción de la interfaz, describiendo cada uno de los módulos de que constará, su función y relación con el resto de los módulos, además de describir su forma de implementación.

El último capítulo evalúa y analiza los resultados obtenidos con el sistema, mediante la presentación y el manejo de dos ejemplos prácticos y de ámbito general.

CAPITULO I

REVISION BIBLOGRAFICA DE SISTEMAS EXPERTOS

Los Sistemas Expertos -como nueva generación de software- reproducen el planteamiento de expertos en la solución de problemas; se entiende que es un tipo de programa que imita el comportamiento de un experto humano. Pueden almacenar conocimientos para un campo determinado -por cierto muy delimitado- y solucionar cuestiones mediante deducción lógica de conclusiones. Esto significa que en un programa de computador disponemos de estrategias de solución y conocimientos de expertos, con los que podemos solucionar problemas de una forma casi inteligente.

Los Sistemas Expertos son aplicables, ahí donde por la **complejidad del problema**, su **comportamiento dinámico** o la **explosión combinatoria** de múltiples soluciones no resulta posible o rentable una solución convencional mediante procesamiento de datos. Son en general problemas que implican un procedimiento de solución **basado en el conocimiento**. Este procedimiento comprende las siguientes capacidades:

- *Utilización de normas u otras estructuras que contengan conocimientos y experiencia de expertos especializados.*
- *Deducción lógica de conclusiones.*
- *Interpretación de datos inciertos.*
- *Manipulación de conocimientos vagos, es decir, conocimientos afectados por valores de probabilidad.*

Científicos e Ingenieros han demostrado cómo puede estructurarse, almacenarse y extraerse el conocimiento según el tipo de problema, es por ello que se habla más de procesamiento de conocimientos que de procesamiento de datos.

La programación ha experimentado nuevos impulsos al ocuparse del comportamiento humano ante la solución de problemas. Se han desarrollado nuevos métodos y procedimientos de programación, y el presente capítulo describe estos métodos y procedimientos.

I.1 .- Fundamentos de la Inteligencia Artificial.

Todo el mundo tiene una cierta idea de lo que es la inteligencia, pero aún en nuestros días es muy difícil llegar a definirla.

El ser humano como sistema inteligente, ha sido definido de diversas maneras entre ellas podemos mencionar: sujeto con capacidad de aprender, sujeto con capacidad de adaptarse al ambiente o sujeto con una tendencia a alcanzar la plenitud.

En general un sistema cualquiera que este sea, es inteligente cuando incorpora conocimientos de un tema en específico y es capaz de discernir acerca de él.

I.1.1 Definición y conceptos básicos.

La Inteligencia Artificial (I.A.) se define como la ciencia que trata de la comprensión de la inteligencia y del diseño de máquinas inteligentes, es decir, el estudio y la simulación de las actividades intelectuales del hombre (manipulación, razonamiento, percepción, aprendizaje, creación , etc.), aplicadas dentro del entorno de funcionamiento de una computadora.

El uso de la I. A. como herramienta de programación facilita la solución de problemas que involucran un alto grado de complejidad, ambigüedad e incertidumbre, ya que éste tipo de problemas no pueden ser resueltos por métodos tradicionales. Un sistema de I. A., es aquel que incorpora conocimientos sobre un tema, junto con los procedimientos para encontrar respuestas a los problemas sobre el mismo.

A lo largo de más de 30 años en investigación en I. A. se ha llegado a la conclusión de que esta última es mucho más difícil de lo que se esperaba, ya que existen muchos factores necesarios para llevar a cabo ésta tarea, los cuales son inherentes al ser humano; es por ello que aún cuando se ha logrado hoy en día un alto desarrollo de hardware los resultados son inferiores a lo esperado.

I.1.2 Reseña Histórica.

La I. A. como se concibe hasta nuestros días, aparece en la década de los 50's, cuando se empieza por escribir programas de computadora de tipo simbólico para la resolución automática de problemas.

El desarrollo de la I. A. ha estado unido desde su orígenes a los avances tecnológicos en el campo de la computación, y éstos están íntimamente ligados a los desarrollos de la microelectrónica.

Una visión de la historia de la I. A. es la dada por Forsyth

la cual divide a ésta en cuatro décadas :

-1950	*Redes Neuronales.*
-1960	*Búsqueda Heurística.*
-1970	*Sistemas Expertos.*
-1980	*Aprendizaje de las Máquinas.*

Ya en 1943, Warren, McCulloch y Pitts trabajaron con Redes Neuronales, que en pocas palabras consiste en el estudio de modelos que siguen la Arquitectura del cerebro humano con el fin de conseguir con ello, la realización de las tareas propias del cerebro de una forma artificial, incluyendo por supuesto, la inteligencia.

Es decir, se trataba de imitar la estructura interna o "hardware". La semejanza entre el cerebro humano y una computadora es extraordinariamente correcta, pues cada neurona es en realidad un procesador binario muy sencillo, que toma como salida el valor uno o cero, dependiendo de las señales que le llegan de otras neuronas; cada procesador binario puede estar activo o inactivo, dependiendo del proceso que se esté realizando. Hasta aquí el parecido, porque el acceso a la memoria no es por la dirección de la misma donde se encuentra físicamente el contenido, como ocurre en las computadoras actuales, sino que es directamente por contenido, y además una neurona se haya conectada con hasta 10,000 similares o más.

Todo lo cual hace pensar en estructuras y procesos nuevos como son : la búsqueda inteligente de la información, las organizaciones asociativas de la memoria, las redes jerárquicas, conexión de procesadores en paralelismo masivo , procesadores con microcódigo o firmware, etc.

Esta línea de investigación, perdió interés en un principio debido a la alta complejidad con que había que enfrentarse.

En 1950 Alan Turing ideó una prueba para reconocer comportamientos inteligentes. Esta prueba dice que la inteligencia de un sistema, viene dada por la sensación de inteligencia que recibimos por un terminal conectado a un interlocutor desconocido y el cual al darnos respuesta, a una pregunta específica, no sabemos diferenciar si nos ha respondido un humano o una máquina.

En 1955 Allan Newell, J.C. Chow, y Herbert Simons de la Universidad de Carnegie Mellon, desarrollan el lenguaje IPL2 que se considera el primer lenguaje de la I. A.

En 1957 se creó la primera máquina basada en la filosofía de Turing, que fue el PERCEPTRON, de Frank Roseblett (Universidad de Cornell), la cual tenía tres capas : Percepción, Asociación y Respuesta.

En 1960 John McCarthy crea el LISP en el M.I.T.[1], lenguaje basado en el procesamiento de listas y en el cual el flujo de control se conduce principalmente por recursividad.

En 1961 Marvin Minsky del M.I.T., es el primero en emplear el término de Inteligencia Artificial.

En 1964 Joseph Weizenbaum del M.I.T., escribe el primer programa de I. A., llamado ELIZA, que era un sistema informatizado que analizaba psicológicamente al estilo rogeriano.

En 1972 Alan Colmerauer crea PROLOG en Marsella Francia. Desde su concepción, PROLOG está orientado a apoyar a la Inteligencia Artificial y fué bien recibido. En 1981 los japoneses anuncian que utilizarían PROLOG como el lenguaje de sus máquinas de quinta generación.

Actualmente la tecnología de integración de dispositivos avanza rápidamente y se tienen capacidades de integración en memorias RAM próximas a las densidades del cerebro humano : un megabit/100mm^3.

I.1.3 Componentes de la Inteligencia Artificial.

La resolución de problemas es una constante que ha acompañado al hombre desde sus orígenes. Tanto los problemas que frenan las fronteras del saber (explicación de fenómenos y comportamientos, demostración de teoremas, ...), como aquellos otros que impiden las realizaciones prácticas (ingeniería, planificación, arquitectura, ...), o simplemente los creados por curiosidad o distracción (juegos, pasatiempos, ...), han llenado el tiempo y consumido esfuerzos intelectuales del hombre.

Es aquí donde la I. A. se ubica dentro del amplio campo del conocimiento humano.

Las áreas más importantes de aplicación de la Inteligencia Artificial, junto con sus componentes son mostradas en la figura 1; A continuación se da un a breve descripción de cada una de ellas:

Búsqueda de soluciones .- (Razonamiento) Estudia mejores métodos de búsqueda para encontrar, recuperar e interpretar información contenida en gigantescas bases de datos.

Sistemas Expertos .- (Razonamiento) estudian la simulación de los procesos intelectuales de los expertos humanos como pueden ser: la Interpretación de Datos, el Diagnóstico, la Corrección, el

---

[1] Massachusetts Institute Tech.

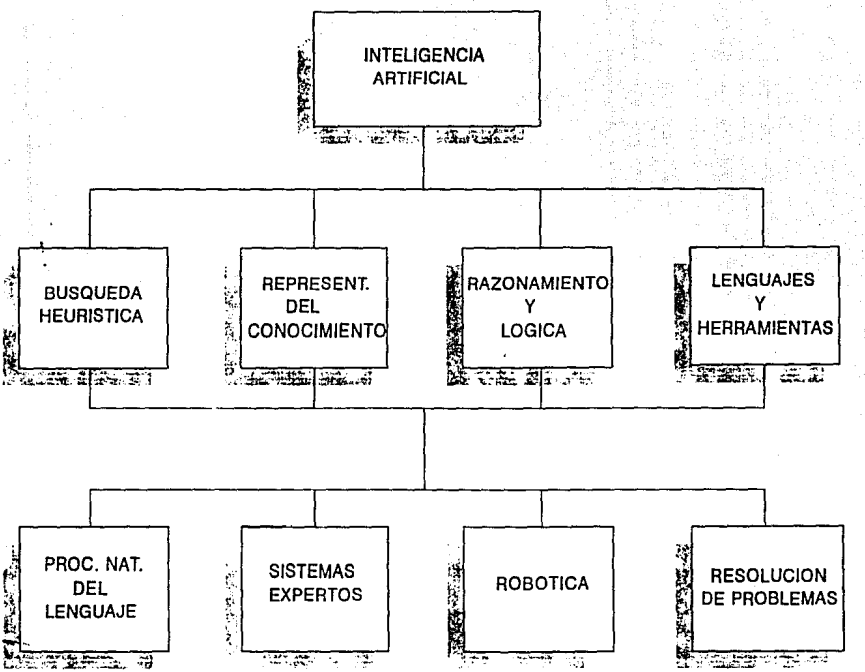


FIG. 1 ELEMENTOS DE LA INTELIGENCIA ARTIFICIAL

Control, Predicción, Planificación, Diseño y Enseñanza.

Procesamiento del Lenguaje Natural .- (Percepción) Es la habilidad que tiene una computadora para procesar el mismo lenguaje que los humanos usan en conversaciones cotidianas. Se divide en las siguientes áreas :

1) Interfaz en Lenguaje Natural con Bases de Datos : Es un sistema administrador de bases de datos que ejecuta acciones en respuesta a peticiones que el usuario realiza en lenguaje común.
2) Traducción de Idiomas por Computadora : Se refiere a programas que reciben como entrada un texto escrito en un idioma y lo traducen a otro idioma.
3) Sistemas de Habla : Permiten la interacción de la computadora a través de la voz.

Visión y Reconocimiento de Patrones : (Percepción) Estudia la identificación, inspección, localización y verificación de objetos por medio de computadora, analizando la cantidad de luz que es reflejada por el objeto en cuestión.

Robótica : (Manipulación) Estudia las máquinas capaces de realizar procesos mecánicos repetitivos y tareas manuales. Una definición de Robótica muy utilizada, es la de Allan Newell : " Es la ciencia que integra la inteligencia y la energía, esto es, el control inteligente de movimientos coordinados gracias a las percepciones que se tengan del medio ambiente."

I.1.4 Aplicaciones de la Inteligencia Artificial.

El rango potencial de la I. A. incluye una vasta cantidad de aplicaciones que virtualmente cubren el total de la actividad de la inteligencia humana. Aplicaciones generales son listadas a continuación:

Manejadores de Conocimiento :
- Bases de datos de acceso inteligente
- Adquisición de conocimiento
- Comprensión de texto
- Generación de texto
- Explicación y manejo de máquinas
- Operaciones lógicas avanzadas en bases de datos

Interacción Humana :
- Comunicaciones
- Diagnóstico y reparación de fallas en la industria
- Sistemas autónomos inteligentes
- Asistencia ejecutiva

Aprendizaje y Enseñanza :
- Aprendizaje basado en la experiencia

Generación de conceptos
Operación y mantenimiento
Diseño inteligente asistido por computadora

I.2 .- Generalidades Sobre los Sistemas Expertos.

En la década de los 50's apareció un interés especial por parte de maestros y psicólogos, por encontrar métodos generales de resolución de problemas. Se observó en aquel entonces, que la mayoría de las personas, aún conociendo toda la información necesaria para resolver correctamente un problema (definiciones, formulas, etc.) son muchas veces incapaces de conseguirlo, realizando con frecuencia razonamientos defectuosos.

Con la difusión de los primeros ordenadores, en la segunda mitad de la década de los 50's se trató de trasladar a estos las dificultades acerca de la resolución de problemas. Surgen así nuevos problemas como son la representación del conocimiento en la memoria del ordenador y representaciones de las relaciones entre los conocimientos; uno de los problemas que surgieron en aquel entonces, fué la aparición de la explosión combinatoria, los cálculos exhaustivos que limitan la profundidad en los mismos y el número de conocimientos que se podían procesar, es decir, se calculaban todas las posibles soluciones para luego elegir la óptima. Aparecen entonces los primeros algoritmos de poda.

En la década de los 70's, no conociendo los mecanismos generales de resolución de problemas de la mente humana, se pensó en simular la misma para campos muy concretos del conocimiento. El manejo eficaz de los conocimientos dio entonces sus primeros frutos: los Sistemas Expertos.

I.2.1 Definición.

Desde que se dieron los primeros pasos en el desarrollo de los Sistemas Expertos hasta la fecha, han surgido diversas definiciones acerca de éstos :

"En términos generales un Sistema Experto se puede definir como un programa de computadora cuyo objetivo es emular el comportamiento seguido por un experto humano en la solución de un problema." [Mejía]

"Los Sistemas Expertos son programa de computadora capaces de tomar decisiones o hacer recomendaciones a nivel de un experto humano, incorporando hechos, leyes y reglas prácticas, organizados de tal manera que dicho programa pueda hacer inferencias lógicas." [Ostrowsky]

"Los Sistemas Expertos son programas sofisticados de computa-

dora que manipulan conocimientos de expertos humanos para resolver eficiente y efectivamente problemas de un área específica, tal y como lo haría un experto humano; los sistemas expertos son creados para actuar como asistentes inteligentes en la toma de decisiones." [Vladimir Marik]

"Un Sistema Experto es un programa inteligente que utiliza el conocimiento y procesos de inferencia para resolver problemas lo suficientemente complicados como para no necesitar la guía de un experto humano." [Hunt]

"Si la ejecución de un conjunto de programas de computadora puede convencernos de que su comportamiento es el que tendría un experto humano, entonces este conjunto de programas es un verdadero sistema experto." [Forsyth]

Resumiendo, un Sistema Experto o Sistema Basado en el Conocimiento es un conjunto de programas de computadora, capaz de aplicar conocimientos y resolver programas de un área determinada del saber.

Como podemos observar el conocimiento es la base fundamental de un S. E. y este conocimiento lo podemos encontrar de dos formas: el primero son hechos comunes que son conocimientos compartidos por profesionales u otras fuentes, y el segundo es la heurística, que es en sí, sentido común.

I.2.2 Historia de los Sistemas Expertos.

El precursor de los S. E. actuales fué el sistema DENDRAL (Universidad de Standford 1967) que incorporaba una gran cantidad de conocimientos. Utilizaba para la representación del mismo, reglas de producción, si bien su modo de funcionamiento se acercaba más a la filosofía de la resolución automática, que a la de los S. E. Este sistema era capaz de determinar la estructura química de un compuesto orgánico a partir de los resultados obtenidos mediante un espectrógrafo de masas.

La historia de los S. E. puede dividirse en tres etapas :

- La primera época : que llega hasta el año 1964, y que podríamos denominar la prehistoria de los S. E.. Durante esta etapa se crearon las bases teóricas que van a posibilitar la concepción de los mismos. Se desarrollan los lenguajes de programación y las máquinas capaces de soportarlos.

- La segunda época : comprende desde 1974 a 1984 es llamada por algunos autores la década de los S. E.. En ella se construyen de una forma sistemática S. E. que han sido referencia obligada durante muchos años.

De los Sistemas Expertos construidos en esta época están: El PROSPECTOR (Standford Research Institute 1974), Sistema experto en prospecciones mineras, especialmente en las petroleras, que cuenta entre sus méritos el descubrimiento en 1980 de un yacimiento de molibdeno en Washington; y el MYCIN (Universidad de Standford 1987) S. E. en el diagnóstico y terapia de enfermedades infecciosas de origen bacteriano que contaba con unas 400 reglas.

En esta década se ponen en marcha los grandes proyectos de investigación y desarrollo que de una u otra forma incluyen a los S. E. (Proyecto de ordenadores de quinta generación).

- La tercera época : comienza en 1984 y todavía seguimos en ella. Se caracteriza por la gran difusión de lenguajes especializados, herramientas y sistemas vacíos (shells), gracias a su comercialización en computadoras personales.

I.2.3 Características de los Sistemas Expertos.

La tecnología de construcción de los S. E. está comprendida entre la Ingeniería de Programación, Ingeniería del Hardware y el Psicoanálisis.

Realmente un S. E., no se comporta como un experto humano pues no se conocen todavía los procesos mentales que se ponen en funcionamiento en el hombre cuando trata de resolver un problema y mucho menos cual es la fuente de inspiración. Es por esto que realmente un S. E. simula estos procesos, pero de una forma mejorada ya que de una u otra forma es capaz de justificar los resultados obtenidos.

Según Hayes Roth las características ideales que debe tener un S. E. son las siguientes :

- Resolver problemas muy difíciles tan bien o mejor que un experto humano.
- Razonar heurísticamente, utilizando reglas que los expertos humanos consideran eficaces.
- Interactuar activamente y en lenguaje natural, con los usuarios.
- Manipular descripciones simbólicas y razonar sobre ellas.
- Funcionar con datos erróneos y reglas imprecisas.
- Contemplar simultáneamente múltiples hipótesis alternativas.
- Justificar sus conclusiones.

Alcanzado este punto podemos ya diferenciar, en su forma externa, un S. E. de un programa tradicional.

Una de las diferencias básicas es que el conocimiento en un S. E. tiene que estar en forma de unidades elementales del mismo que

pueden relacionarse unas con otras y que le permiten conocer cual de ellas ha actuado, cómo y por qué.

Un experto humano va aumentando sus conocimientos con el transcurso del tiempo, y es por ello que un S. E. tiene que ser flexible, es decir, puede modificar el conocimiento que tiene almacenado de una forma sencilla y sin que ello afecte al resto del programa. Esto hace que las funciones de control (Mecanismo de Inferencia), los datos o hechos (Base de Hechos) y el conocimiento (Base de Conocimiento) sean completamente independientes.

Esta flexibilidad lo convierte en un programa muy rentable y productivo al no "envejecer" y poderse adaptar a las necesidades, criterios y políticas de cada situación.

Otra diferencia notable es que un S. E. tiene que ser capaz de resolver cualquier problema planteado dentro de su campo, y como no puede tener almacenada todas las soluciones, es por naturaleza declarativo, de ahí lo cual, utiliza preferentemente el razonamiento simbólico frente al algorítmico.

Ventajas y desventajas de los Sistemas Expertos.

Los S. E. además de las ventajas que se han señalado en los párrafos precedentes, presentan las siguientes ventajas respecto de los expertos humanos :

- Están siempre disponibles a cualquier hora del día y de la noche, y de forma ininterrumpida.
- Carecen de personalidad y por lo tanto,no son capaces de mostrarse indiferentes con el usuario.
- Pueden duplicarse, lo que permite tener tantos como se necesiten.
- Pueden situarse en el lugar donde sean necesarios; entornos hostiles, peligrosos, etc.
- Son fáciles de reprogramar.
- Pueden ser consultados por cualquier persona u otros sistemas informáticos.
- Almacenan y consolidan permanentemente el conocimiento.
- Auxilian a los humanos en el trabajo rutinario.

A pesar de las virtudes anteriores, no es posible utilizar un S. E. cuando suceden las siguientes situaciones :

- Cuando el problema incluye demasiado uso del sentido común humano.
- Cuando el problema es bien entendido y esta claramente estructurada su solución.
- Cuando se tienen conocimientos que varían con el tiempo, el S. E. no es capaz de actualizarse.

I.3 .- Arquitectura de los Sistemas Expertos.

I.3.1 Introducción.

Un programa tradicional está compuesto por entradas y salidas de datos (constantes o variables), algoritmos (lógicos o aritméticos) y unas sentencias de control; el cual podemos resumir como sigue :

DATOS+ALGORITMO+CONTROL+ENTRADAS/SALIDAS=PROGRAMA

y con ello tendremos un programa tradicional completo.

Ahora, para un S. E. tenemos más o menos los mismos elementos, solo que éstos son completamente independientes unos de otros; realizando una semejanza entre uno y otro un S. E. puede representarse como :

(BASE DE HECHOS, BASE DE CONOCIMIENTOS, MECANISMO DE INFERENCIA, ENTRADA/SALIDA)=SISTEMA EXPERTO

En la figura 2 aparece un esquema de bloques de los componentes de un S. E..

Centremos nuestra atención en la definición de dos términos importantes para la arquitectura de un S. E. que son : algoritmo y heurístico.

Un algoritmo es un procedimiento para ejecutar instrucciones de programa, según un orden determinado, un conjunto de operaciones elementales; son determinísticos, masivos y resolutivos; además se pretende que sean mínimos y únicos.

Un heurístico es un proceso creador que a partir de un conjunto de operaciones elementales suficientes, resuelve todos los problemas de una clase; son determinísticos, masivos y no resolutivos pues no garantizan la solución de un problema en específico; no son ni mínimos ni únicos.

Aclaradas las diferencias anteriores pasemos a definir cada una de las unidades funcionales del S. E..

I.3.2 Componentes de un Sistema Experto

I.3.2.1 El Mecanismo de Inferencia.

Es el software que realmente lleva el control del programa y con ayuda de los otros módulos e interactuando con ellos llega a la solución buscada; una definición mas formal es :

"El Mecanismo de Inferencia selecciona, decide, interpreta y aplica el conocimiento de la Base de Conocimiento sobre la Base de Hechos con el fin de obtener la solución buscada."

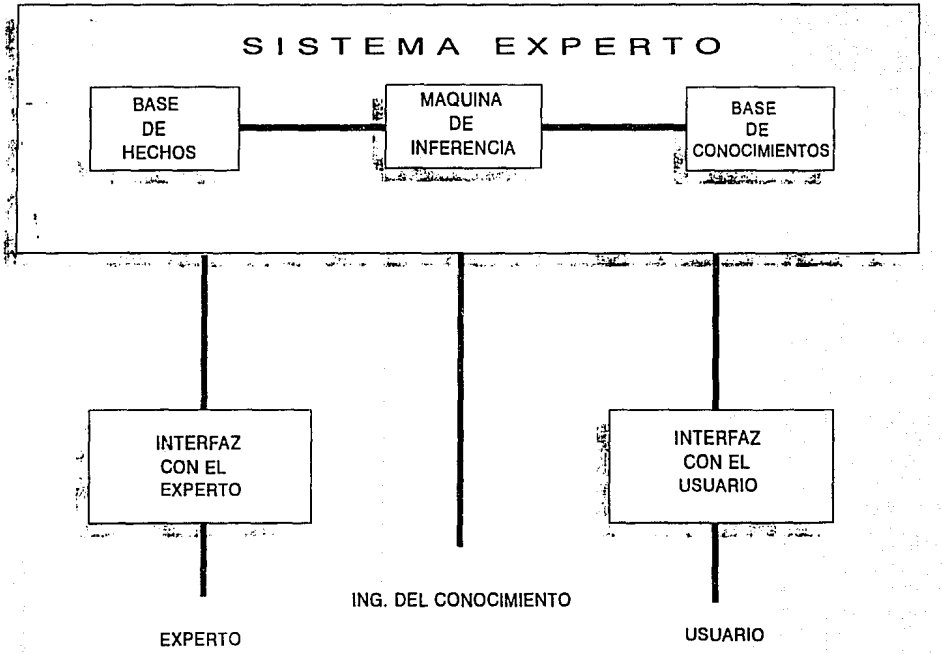


FIG. 2 COMPONENTES DE UN SISTEMA EXPERTO

Al operar sobre la Base de Conocimientos el Mecanismo de Inferencia realiza hilaciones, desarrolla nuevos hechos, actualiza información previa y crea nuevas reglas.

El funcionamiento del Mecanismo de Inferencia es el siguiente:
- evaluación, en el cual se selecciona el conocimiento a emplear.
- comprobación, de que el conocimiento sea aplicable.
- ejecución, en la cual se aplica el conocimiento.
- comprobar condición final.
- controlar reglas activas.

Las características del Mecanismo de Inferencia son :
- el lenguaje con el que ha sido escrito.
- la velocidad de trabajo (inferencias por segundo).
- la estrategia que utiliza.
- el sistema de elección del conocimiento.
- la posibilidad de utilizar Metaconocimientos.
- el orden de la lógica que emplea.
- el método de evaluación.

Mecanismos de búsqueda.

Dentro del Mecanismo de Inferencia del S. E. se pueden seguir 2 estrategias de búsqueda posibles, la primera es :

- La búsqueda no ordenada de soluciones la cual se divide en:
  - **Aleatoria** : con esta forma se recorre toda la Base de Conocimientos con lo cual es posible encontrar todas las posibles soluciones y con ello seleccionar la óptima.
  - **Heurística** : con esta forma sólo se busca en ciertos módulos, los cuales son indicados por el Metaconocimiento; con este método se ahorra muchisimo tiempo y las soluciones obtenidas son casi óptimas.

- La búsqueda ordenada de soluciones :
  Este método es utilizado cuando el conocimiento está expresado en forma de reglas y consiste en ir encadenando dicho conocimiento hasta encontrar la solución o grupo de soluciones; esto se logra haciendo que el consecuente de una regla se convierta en el antecedente de la siguiente.

  Dicho encadenamiento genera árboles, los cuales pueden ser podados por algoritmos o por métodos heurísticos y en ambos métodos deben ser considerados los conocimientos y no únicamente los coeficientes o estructuras incluidos en el árbol.

  En este tipo de búsqueda se identifican tres métodos :

* El encadenamiento hacia adelante o data-driven.
(Guiado por los datos, trata de encontrar un objeto)
En este método se comienza a encadenar el conocimiento a partir de los datos dados, con el fin de obtener una solución. Este método se basa en el "Modus Ponens" de la lógica formal.

Dado que este proceso genera nuevos hechos existen dos modos de tratarlos :

- En profundidad : cuando un hecho que se genere se introduce a la Base de Conocimiento.
- En anchura : No se incorporan los nuevos hechos a la base de hechos hasta que se termina de aplicar la base de conocimientos.

* El encadenamiento hacia atrás u object-driven.
(Guiado por los objetos, trata de validar un objeto)
Este método trata de probar que un objeto es cierto en base a los hechos que existen y a la Base de Conocimiento; está fundamentado en el "Modus Tollens" de la lógica formal.

En este método las soluciones deben ser conocidas de antemano y los datos pueden ser suministrados a lo largo de toda la ejecución, lo cual no ocurre con el método anterior, en el que los datos solo eran suministrados al inicio.

* Encadenamiento mixto .
Como su nombre lo indica es una combinación de los 2 métodos descritos anteriormente; aplicándose primero un encadenamiento hacia adelante con el cual se guía la búsqueda y se obtiene un conjunto de soluciones, después se aplica un encadenamiento hacia atrás con el objeto de verificar las soluciones encontradas.

Metaconocimiento.

El metaconocimiento es un parámetro que puede variar en determinado momento la estrategia de solución.

El metaconocimiento, es conocimiento que indica al Mecanismo de Inferencia que datos deben ser elegidos y aplicados durante la búsqueda de soluciones.

Algunos puntos importantes a considerarse cuando se utiliza metaconocimiento son :

- Se requiere una Mecanismo de Inferencia de orden superior, es decir una máquina que pueda ejercer cierta intuición acerca de los conocimientos que se están manejando.
- La Base de Conocimientos debe de dividir dicho conocimien-

to en grupos o clases.

Otro aspecto importante del Mecanismo de Inferencia es cómo se evalúa el conocimiento, de donde podemos encontrar tres métodos :

Determinístico.- Las condiciones de entrada afectan de forma total el resultado.

Probabilístico.- Se les da un cierto peso de probabilidad a los hechos y la probabilidad final es calculada en base a la fórmula general de probabilidad condicional de bayes.

Aproximado .- La certeza del resultado está en función de la certeza o falsedad de los hechos y conocimientos utilizados.

I.3.2.2 La Base de Conocimientos.

Es la parte del S. E. en el cual está codificado el conocimiento dentro del cual es experto nuestro sistema.

Dicho conocimiento debe de estar expresado en tal forma de que pueda ser manipulado con sencillez, ocupe poco espacio en memoria, y sea totalmente independiente del resto de los módulos del S. E. con la finalidad de que cualquier modificación que se le realice no cause ningún problema con los demás módulos; además dichas modificaciones deben ser fáciles de realizar y las debe poder realizar el experto o el usuario final, sin requerir la ayuda del programador. Por último debe ser transparente, es decir, debe de ser capaz de soportar cualquier Mecanismo de Inferencia que siga el modelo con el cual fue creado.

Ahora bien, esta Base de Conocimientos, difiere mucho de las base de datos ampliamente conocidas, ya que ésta incluye hechos, intuición, heurística; así como de métodos especiales para consultar dicha base.

Un punto muy importante a considerar, es que la calidad del sistema experto es una función de la cantidad y calidad de ésta base.

I.3.2.2.1 Representación del conocimiento.

Introducción

El poder representar la realidad, es uno de los mayores problemas a los que se ha enfrentado la humanidad, y es por ello que muchos problemas no han sido solucionados hasta que se han logrado desarrollar sus modelos de representación.

El proceso de aprender se basa en la adquisición de imágenes o representaciones de la realidad, ya sea de un proceso o de un

objeto (esto último se conoce como representación simbólica).

Es en ésta última, en la que el humano basa su aprendizaje describiendo su realidad en forma de símbolos.

Esta necesidad de aprender ha llevado al hombre a crear un lenguaje matemático para poder representar, de forma artificial su realidad, lo cual ha provocado por ende, que su estudio sea complicado. Otra herramienta de la cual se ha auxiliado es la Lógica, con la cual puede comprobar la veracidad o falsedad de un proceso u objeto en base a los hechos que se le presentan.

Existen varias formas de representar la realidad, entre las cuales podemos citar :

- *Proceso de observación objetiva de la realidad*, el cual permite describir de forma numérica su comportamiento, con lo cual se logra atraer la atención a aquella información que realmente describe al proceso.

- *Procesos estadísticos*, por medio del cual se pueden estudiar fenómenos regulares o de comportamiento complejo, es decir, analizando la totalidad de las observaciones para después tratar de asociar su comportamiento a algún modelo probabilístico, para con ello poder prever su comportamiento.

- *Proceso simbólico*, con el cual se estudian procesos irregulares ó que muy raramente llegan a repetirse, por ejemplo: que en dos jugadas de una partida de ajedrez se repita la misma posición de las piezas. Este proceso es muy potente pero poco preciso.

En todos los problemas, el encontrar una solución depende en gran medida de la representación que se haga del mismo y por ello consideremos que existen dos características básicas con las cuales debe contar la representación del conocimiento, para llegar a una solución y que son:

- *Poder expresivo*, facilidad de descripción. Cuanto más simbólico es un sistema, más potente es.

- *Eficiencia de cálculo*, cuanto más simbólico es un sistema menor es su eficacia de cálculo.

Durante mucho tiempo la Psicología Cognoscitiva ha investigado las formas de representar el conocimiento que más se aproximan al del cerebro humano. Basándose en éstos estudios, la Ingeniería del Conocimiento, ha creado diversos sistemas de representación que implantados sobre una computadora actúan según los modelos teóricos elaborados por la Psicología Cognoscitiva.

Existen múltiples formas de representar el conocimiento en una

computadora, y cada una de ellas presenta alguna ventaja sobre el resto. El deseo de alcanzar soluciones de compromiso ha motivado, la aparición de las llamadas representaciones híbridas, que intentan incorporar las ventajas de varios sistemas; si bien esto se consigue, se emplea una mayor complejidad en la realización del Mecanismo de Inferencia al no ser uniforme la representación del conocimiento.

No hay que confundir, la representación del conocimiento con los datos. Los datos no describen la realidad sino que únicamente informan del valor que toma un determinado parámetro.

Esquemas de Representación Lógica

Entre los trabajos más importantes sobre la Lógica Clásica, destacan los de George Boole sobre Lógica de Proposición y Fregge sobre Lógica de Predicados, que fueron desarrollados en la segunda mitad del siglo pasado y la primera de éste. La representación lógica del conocimiento generalmente utiliza los dos esquemas antes mencionados, aunque en los años 60's se desarrollaron procedimientos para la deducción automática, como el llamado "Método de Resolución" de Robinson.

La Lógica de Proposiciones o enunciados maneja variables proposicionales y conectivos. Un conjunto de reglas formadas por enunciados válidos definen rigurosamente el lenguaje, y se puede construir un sistema axiomático mediante la definición de unos axiomas y de unas reglas de transformación. Con ella se pueden formalizar muchos de los procesos racionales mediante la utilización de un sistema de reglas de inferencia.

La Lógica de Predicados (conocida como cuantificacional), introduce elementos para tratar con razonamientos lógicos, en los que intervienen propiedades de individuos y relaciones entre los individuos. Los elementos básicos de ésta representación son:

- *Universo del Discurso*.- es el conjunto de todos los valores posibles que puede tomar una variable.
- *Constantes y Variables*.- representan a los individuos o entidades.
- *Predicados*.- expresan una propiedad de algúna variable entre una o más variables.
- *Funciones*.- permiten representar transformaciones o relaciones entre dos ó más variables.

Las representaciones lógicas son fácilmente entendibles y tienen disponibles conjuntos de reglas de inferencia necesarias para operar por sí mismas. Un inconveniente de la representación lógica, es su tendencia a consumir grandes cantidades de memoria.

Las formas más importantes de representación del conocimiento

basados en ésta lógica son : las reglas de producción (rules), los marcos o tramas (frames), las redes semánticas o árboles (nets) y los objetos.

Reglas de Producción

La representación del conocimiento en forma de reglas de producción fue propuesta por Post en 1943.

La regla es la forma más común de representar el conocimiento, debido a su gran sencillez y a ser la formulación más inmediata del principio de causalidad (causa-efecto).

Una regla consta de un conjunto de acciones o efectos (uno o más) que son ciertos cuando se cumplen un conjunto de condiciones o causas (una o más). La potencia de la regla es función de la lógica que admita en las expresiones las condiciones y las conclusiones.

El "mundo" o Sistema de Producción es una representación del conocimiento mediante reglas de producción, y se hace mediante un conjunto de hechos que son ciertos y un conjunto de reglas de producción que muestran la forma de evolución de éstos hechos. Un Sistema de Producción consiste de los siguientes elementos:

- Un conjunto de reglas de producción, cada una de ellas consiste de un lado izquierdo llamado condición, que determina la aplicabilidad de la regla, y un lado derecho que describe la acción que será ejecutada si la regla es aceptada.

  CONDICION ---------> ACCION

  SI la condición se presenta, ENTONCES la acción es apropiada

- Una o más estructuras de datos, que contienen información necesaria para una tarea particular. Algunas partes de las estructuras de datos pueden ser permanentes o estáticas, mientras que otras pueden ser temporales o dinámicas ya que permanecen sólo en la solución del problema. Estas estructuras también se conocen por el nombre de contexto.

- Una estrategia de control que especifica el orden en el cual las reglas serán comparadas con las estructuras de datos y una forma de resolver los conflictos que se crean cuando diversas reglas pueden ser ejecutadas simultáneamente. Esto es, un interprete que controla la actividad del sistema.

Por lo anteriormente expuesto, la forma general de representación de una regla es la siguiente:

SI < CONDICION> ENTONCES <CONCLUSION O ACCIONES>

La conclusión se suele referir a la creación de un nuevo hecho válido, o la incorporación de una nueva característica a un hecho, mientras que la acción suele referirse a la transformación de un hecho.

Existe una restricción de la sintaxis de las reglas de producción que se denomina *'Cláusulas de Horn'*, utilizada por el lenguaje PROLOG y que consiste en:

1.- Sólo existe una conclusión por regla.
2.- La conclusión no puede aparecer negada.

El conocimiento acerca de las reglas de producción que siguen esta misma estructura, se denomina Metarregla. Las metarreglas facilitan la resolución de los problemas, pues si la Base de Conocimientos está construida de forma modular, las metarreglas pueden inhibir parte de los módulos, lo que hace más rápida la búsqueda de soluciones.

Existen dos clases de metarreglas que son :

- las metarreglas ciegas, son aquellas que contienen conocimiento sobre la estructura de las reglas, es decir sobre su sintaxis.
- las metarreglas no ciegas o inteligentes, son aquellas que contienen conocimiento sobre el contenido de las reglas, es decir sobre su semántica.

Existen un tipo de reglas que normalmente se les denomina estratégicas, que consisten en la ejecución de algoritmos o procedimientos para el control del proceso. De esta forma se puede permitir el metaconocimiento. Sin embargo, no parece excesivamente lógico o "correcto" construir un S. E. empleando este tipo de reglas. Un sistema así no estaría en forma declarativa, sino procedimental, esto quiere decir que en la base de conocimientos se incluirían algoritmos y procedimientos de control, lo cual implicaría de forma implícita o explícita la no independencia de la base de conocimientos y el control, ... lo que nos llevaría a una filosofía de programación tradicional en la que el mejor de los casos el control sería de forma funcional como es en el LISP, pero que nos alejaría de la de los S. E.

Las reglas de producción, pueden estar afectadas por factores de certidumbre, los cuales son una medida de que tan verdadero es cierto hecho. Así, se han encontrado diversas maneras del tratamiento de la incertidumbre, entre las que se encuentran la aritmética y la lógica difusa [Zadeh].

Las ventajas que presentan las reglas de producción son su carácter declarativo, su sencillez, su uniformidad que permite la representación del conocimiento como metaconocimiento, su independencia que permite la supresión o inclusión sin que se vea afectado el resto de la base de conocimientos y su modularidad al ser fácilmente agrupables. .

Las desventajas que presentan, es la dificultad de establecer relaciones, para lo cual hay que recurrir al uso de metarreglas, al crecimiento muy rápido del número de reglas, lo que disminuye la velocidad del proceso de inferencia y a la gran facilidad con la que se pueden introducir repeticiones y, lo que es peor, contradicciones.

Redes Semánticas

Otra de las técnicas de Representación de Conocimiento, es la de objetos estructurados, entre los cuales se tienen a las redes semánticas. Estas fueron propuestas inicialmente como modelo de la memoria humana por Quillian y Collins en 1968.

Las redes semánticas (Semantics Nets) o redes asociativas representan el conocimiento mediante nudos o nodos (elementos del conocimiento o conceptos) y ramas o arcos (relaciones entre los nodos, estas relaciones pueden ser de herencia o de descripción). La estructura de las redes semánticas es muy parecida a la de árboles (figura 7).

Las redes al igual que las reglas de producción pueden transformarse en una lista con facilidad, por este motivo, es por el que el LISP (List Inference Process) es un lenguaje básico en la construcción de S. E..

Una red semántica, es un enfoque para describir las propiedades y relaciones de objetos, eventos, conceptos, situaciones ó acciones por una gráfica dirigida (grafo), en el que los nodos corresponden a las constantes, variables y conjuntos y los arcos a relaciones funcionales y de pertenencia.

Las ventajas de las redes semánticas son su potencia a la hora de definir relaciones y su especial adaptación a sistemas interactivos.

Las desventajas de las redes semánticas son su poca flexibilidad, lo que dificulta las modificaciones y la complejidad que encierra su lectura cuando la Base de Conocimientos es grande, por lo que necesita de potentes sistemas gráficos.

La representación de conocimiento de un S. E., únicamente mediante redes semánticas no suele ser habitual, utilizándose conjuntamente con las reglas de producción, lo cual será tratado con profundidad en el Capítulo II del presente trabajo.

Marcos o Frames

Los objetos estructurados se consideran una extensión de las redes semánticas, que incluyen conocimientos procedimentales pero

no inferenciales.

Se han propuesto diversos tipos de representaciones en forma de objetos estructurados, todas ellas muy parecidas, de entre los que destacan : los marcos y los objetos.

Los marcos fueron propuestos por Marvin L. Minski del M.I.T. en 1974, que da la siguiente definición formal: "Un Frame es una estructura de datos que sirve para representar una situación estereotipada, como estar en algún tipo especial de salón o ir a la fiesta de cumpleaños de un niño. Aunado a cada Frame hay varios tipos de información. Parte de esta información hace referencia a cómo utilizar el frame; otra se refiere a lo que uno puede esperar que suceda en segundo lugar, y otra a su vez indica qué hacer si tales esperanzas no son confirmadas".

La mejor traducción del término original "frames" en cuanto a su sentido sería lo que normalmente conocemos como ficha ó registro de un archivo, si bien la traducción literal más próxima es la de marco o cuadro.

En primera instancia, un frame se basa en el postulado de evidencia psicológico que propone que la gente usa una gran cantidad de conocimiento, sobre experiencias anteriores, para interpretar nuevas situaciones en sus actividades cotidianas.

Los marcos combinan las características de las reglas y de las redes semánticas, son modulares por naturaleza y admiten la Representación del Conocimiento en forma declarativa o procedimental. Están formados por un nombre y por una serie de ranuras (slots), que también se denominan : casilleros, alojamientos, campos de información o variables. (ver figura 3)

Las ranuras pueden guardar ciertas relaciones de herencia entre ellos y contener valores, procedimientos para calcular estos valores o reglas para inferirlos. La información que debe de contener un frame es:

- cómo usar el frame
- que hacer si pasa algo imprevisto
- valores default para las ranuras

Un marco permite describir un concepto según un modelo, que visto desde esta perspectiva es como una plantilla.

Un marco puede presentar uno de los estados siguientes:

- *Modelo, prototipo, marco general o vacío*, que contiene la estructura del marco y los valores por defecto.

- *Particularizado, instanciado o lleno*, en el cual se han sustituido algunos valores por defecto, por valores particulares. Los marcos particularizados deben de incluir una

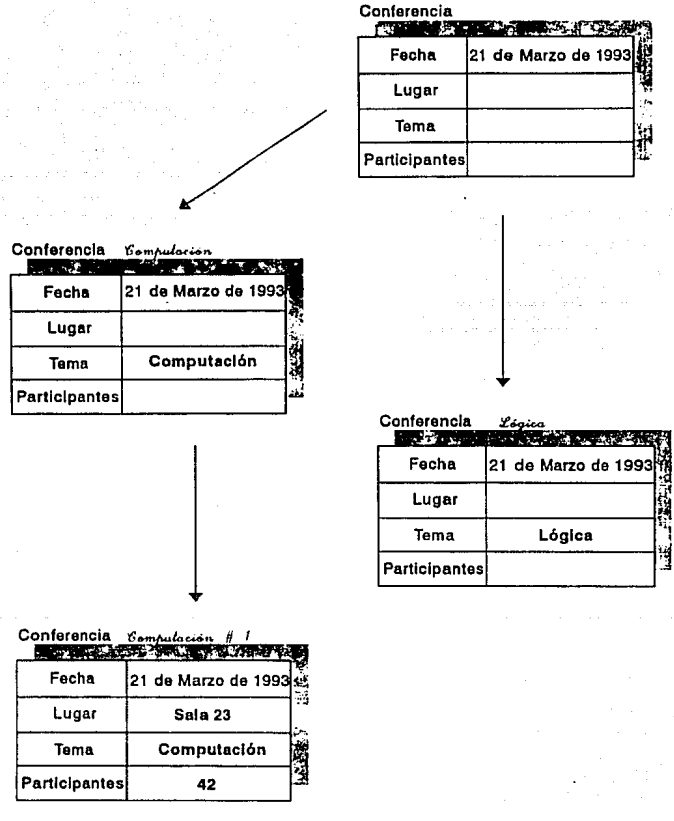


FIG. 3 EJEMPLO DE UN FRAME

referencia al modelo del cual proceden.

Un marco puede llenarse por defecto, es decir que para aquellos valores que no se definen de forma explícita se toma el valor del casillero del modelo. También se pueden definir los valores de los casilleros con relación a otros modelos, de reglas o de procedimientos. Existen dos formas de aplicar las reglas, bien cuando el valor de una variable no está definido de forma explícita o cuando lo está. Los marcos también pueden escribirse con facilidad en forma de listas.

El uso de los marcos está especialmente indicado para la representación de estereotipos, en los cuales se van rellenando con información que distingue a cada caso. Los marcos pueden emplearse fácilmente para bases de datos relacionales.

Objetos

Los objetos, son similares a los marcos, con la diferencia de que la relación entre los objetos viene definida por los mensajes.

Al igual que en los marcos se mantienen los conceptos de casilleros y relaciones de herencias. Aquí las propiedades describen a los objetos. Los objetos se agrupan en clases y subclases, y estas pueden relacionarse mediante mensajes y herencias.

Existen un gran número de lenguajes concebidos con esta filosofía, como es el caso de SMALLTALK.

I.3.2.3 La Base de Hechos.

Es la parte del S. E., en la cual se almacena el conjunto de la información que forma el universo del mismo.

Dichos hechos son la información que permanece invariable de un problema a otro, siempre y cuando se encuentren dentro del dominio del S. E.

Con estos hechos y aplicando las reglas de la base de conocimiento sobre de ellos, se infiere el conjunto de posibles soluciones.

I.3.2.4 Los Módulos de Comunicación.

Al referirse a los Módulos de Comunicación se habla de las interfases, estas deben de cumplir con los requisitos de ser rápidas, potentes, y sencillas en su uso.

Dentro de estas interfases las más importantes son : *El Módulo*

del Experto y El Módulo de usuario; el módulo del experto debe poder realizar depuraciones tales como la adquisición del conocimiento, mantenimiento de dicho conocimiento y debe de ser capaz de realizar operaciones tales como detección de inconsistencias y duplicidad. En lo referente al módulo del usuario debe de poder tener una comunicación de lo más sencilla aproximándose en la medida de lo posible al Lenguaje Natural; este módulo debe ser capaz de aceptar datos, seleccionar diferentes opciones, justificar y explicar procesos.

I.3.2.5 El Diseño de la Interfaz

Definición

El término Interfaz Hombre-Máquina es el nombre colectivo que se aplica a todos los componentes perceptibles de una máquina y que el ser humano utiliza para comunicarse con ella. [Charwat]

La interacción entre un humano y una máquina (computadora) se denomina "diálogo Hombre-Máquina" y resulta de un compromiso entre el lenguaje usado normalmente por la computadora y el usado por el hombre, por lo que se hace necesario un diálogo intermedio donde se conjuguen las necesidades psicológicas del usuario (factores humanos) y la eficiencia de la máquina.

Factores Humanos en el Diseño de Interfases

El desarrollo de una interfaz está determinado por los requerimientos y las características de los usuarios del sistema, ya que en la medida en que éstas sean satisfechas, se determinará la aceptación o rechazo del sistema por parte de dichos usuarios.

Uno de los principales problemas que presenta el diseño de interfases, es que en ello se refleja la posición predominante del Ingeniero, quien es el que determina el grado y tipo de automatización, especificando el hardware y software, sin tomar en cuenta los factores humanos, los cuales son revisados en etapas en las cuales las modificaciones son difíciles ó costosas de realizar [Rijnsdorp].

Para evitar estos problemas es necesario que el diseñador tome en cuenta diversos factores psicológicos y fisiológicos a nivel humano. Es aquí donde la ergonomía entra en juego, es decir la ciencia definida como la encargada del estudio de las necesidades del ser humano y las condiciones bajo las cuales trabaja.

Factores Psicológicos y Fisiológicos

La interacción del hombre con una computadora involucra

básicamente tres tipos de procesos: percepción, proceso cognoscitivo y actividad motora. El trabajo del diseñador de la interfaz consiste en desarrollar técnicas de interacción que minimicen el trabajo requerido por estos procesos. La identificación de los procesos es el primer paso importante al analizar y diseñar una técnica de interacción.

El Proceso de Percepción

Es un proceso mediante el cual los estímulos físicos "no inteligentes", son captados por los órganos receptores, transmitidos al cerebro y ahí reconocidos por un proceso determinado. Los estímulos dominantes provenientes de una computadora son los visuales, aunque ahora se intenta también llamar la atención del usuario por medio de tonos o "beeps" de la computadora. Cabe mencionar que aproximadamente el 80% de toda la información que acepta el cerebro la reciben los ojos.

Es por ello que las técnicas de interacción entre el hombre y la computadora comienzan con percepción visual: el usuario localiza el menú de selección, un elemento a ser borrado, el cursor, o reconoce una forma o un contorno. Así el factor más importante a considerar es la forma en que se va a desplegar la información, de tal manera que el usuario pueda reconocer rápidamente los elementos que necesita.

El Proceso Cognoscitivo

La psicología cognoscitiva trata la forma como los humanos adquirimos, organizamos y solicitamos información.

Desde el punto de vista de diseño de interfases, la Psicología Cognoscitiva proporciona un marco de trabajo para estudiar y simplificar las estructuras de información.

El estudio del proceso cognoscitivo nos adentra en diversas formas para estructurar jerárquicamente los menúes y manejar adecuadamente el número de opciones a presentar; para elegir los tipos de palabras a usar y las formas de llamar o abreviar los comandos.

Si la información se clasifica y presenta en categorías o conceptos que la mayoría de los usuarios manejan, el aprendizaje se llevará a cabo de una forma rápida, de lo contrario éste será lento.

El Proceso Motriz

La actividad motriz comienza cuando el usuario que ha

recibido, reconocido y decidido cómo reaccionar al estímulo, responde con una acción física. Esta proceso depende de la acción continua de los procesos perceptivo y cognoscitivo, creando un ciclo de retroalimentación.

Presentación de la Información

Como se mencionó anteriormente, este es el punto más importante a considerar en el desarrollo de la interfaz, y por lo tanto se deben de tomar consideraciones especiales.

Densidad de Información

Se define como la cantidad de información por unidad de superficie en la pantalla. La densidad máxima va a depender de la cantidad de información requerida por las tareas y de las condiciones de observación que tendrá el usuario al realizarlas. [Villavicencio]

Se recomienda utilizar de un 10% a un 20% de la superficie de la pantalla, de modo que ésta no aparezca sobrecargada.

Despliegue de Texto

Se cree que la mejor forma de dar énfasis a los textos es escribiéndolos en mayúsculas totalmente, aunque éstos sean largos, sin embargo es conveniente recordar que según investigaciones, [Thinker], se incrementa la velocidad y comprensión de la lectura si se escribe con letras minúsculas.

Es por ello que las palabras en mayúsculas sólo se deben usar cuando sea realmente necesario. Las cabezas de columnas y renglones pueden escribirse en mayúsculas para separarlas visualmente del cuerpo de una tabla. El resto del texto debe estar en minúsculas, con opción a usar mayúsculas en la primera letra.

Color

El uso del color en los despliegues de información es uno de los medios más atractivos para incrementar la efectividad de una interfaz. Se debe de cuidar el uso excesivo de éstos, ya que puede ocasionar confusiones en el diálogo. Cuando se decide agregar color a las pantallas, no sólo se afecta la percepción de la información sobre la que se usó color, también se afecta la demás información de la pantalla, considerando además que el cambio muy brusco de colores afecta al ojo humano, llegando a cansar los músculos oculares debido a la contracción continua.

Por otro lado los principales beneficios que presenta el uso del color son: el aumento de la velocidad de identificación de

datos y la acentuación del estado de cierto tipo de información.

Se recomienda utilizar de 4 a 8 colores, incluyendo el negro; además es conveniente considerar lo siguiente :

- no usar el azul en despliegues de texto. Es bueno utilizarlo como color de fondo.
- cuidar la combinación de colores con diferentes profundidades perceptuales, ya que grandes diferencias en la profundidad visual ocasiona fatiga y distracción.
- usar colores opuestos, como rojo y verde o amarillo y azul.

Ventanas

Una ventana es un área rectangular específica dentro de la pantalla de video, considerada como una entidad independiente en la cual se puede desplegar información.

El valor de un diseño utilizando ventanas en una interfaz hombre-máquina, radica en el tipo y extensión de la tarea que se esté llevando a cabo. Se pueden enumerar seis tareas en las cuales la técnica de ventanas puede ser aplicada aceptablemente [Maguire]:

1) Despliegue de grandes cantidades de información en la pantalla.
2) Acceso y combinación de múltiples fuentes de información.
3) Control independiente de múltiples programas.
4) Mantener congelada una tarea que será reanudada más tarde.
5) Cambiar el significado de comandos o teclas.
6) Presentación de diferentes representaciones de una misma tarea.

Control del Diálogo Hombre-Máquina

Los métodos para el control del diálogo hombre-máquina, proporcionan un medio a través del cual el usuario controla el orden y los tipos de desplegados en la pantalla de una aplicación dada, así como las operaciones que la misma provee.

Existen cuatro métodos para control de diálogo:

- *Menúes*
- *Comandos*
- *Diálogo pregunta-respuesta*
- *Documentación en Línea*

Menúes

Son el método de control de diálogo más frecuentemente usado. Presentan dos ventajas: ofrecen una estructura obvia del software en el que están siendo aplicados y toman ventaja de la mayor habilidad de las personas para reconocer información, en vez de

recordarla.

Los menúes también ofrecen ventajas para los diseñadores, ya que existen muchas herramientas para desarrollar software que facilitan la implementación de menúes.

Los menúes se presentan en cuatro diferentes tipos:
a) Menú de pantalla completa, que consiste de una lista vertical con opciones, desplegada en una pantalla completa o en una ventana.
b) Menú de lista con opciones de una sola palabra, desplegada horizontalmente abajo o sobre un área de aplicación de la pantalla.
c) Menú de íconos (dibujos), los cuales se presentan en una lista horizontal, que también es desplegada abajo.
d) Menú pop-up, que es desplegado junto a la aplicación o en algunos casos sobre ella y desaparece una vez seleccionada la tarea.

El diseño de un menú debe de seguir las siguientes reglas:
- poner un título significativo al principio de cada menú.
- poner una línea en blanco para separar grupos lógicos de opciones y después de aproximadamente 5 opciones en los menúes muy largos.
- limitar el número de opciones a una sola pantalla.
- proporcionar un medio para que el usuario pueda salir del menú sin tener que seleccionar una opción.
- en las opciones, utilizar palabras que describen clara y específicamente lo que el usuario está seleccionando.
- organizar la jerarquía de los menúes de acuerdo con las tareas que el usuario ejecutará, en lugar de hacerlo de acuerdo a la estructura de los módulos del software.

Comandos

El lenguaje de comandos es un conjunto de palabras y abreviaciones, que antes de los menúes fueron el método más usado.

El diseño de los nombres de los comandos es complicado, ya que idealmente, estos nombres deben estar formados por verbos cortos, claros y carentes de ambigüedad en su significado para el usuario.

El tiempo tomado en desplegar un menú de pantalla completa, puede ser mayor al tiempo tomado para desplegar un "prompt" de un comando. Esto es una ventaja compensada, ya que es posible que el usuario cometa un error al teclear y tenga que reescribirlo. Cuando el usuario comete un error, se debe desplegar un mensaje claro y específico que indique cual es el error y que se debe hacer. Además una lista de prompts en forma de ayuda es esencial.

Diálogo Pregunta - Respuesta

Este tipo de control, es también llamado "prompts", y son pequeños mensajes que permiten al usuario realizar una simple elección o proporcionar alguna información.

Al diseñar éste tipo de diálogo, es importante ser claro y específico, ya que de esta forma ayudaremos al usuario a entender exactamente lo que se requiere, reduciendo así la posibilidad de posibles errores.

Se recomienda dirigirse al usuario, hablándole de usted; usar oraciones imperativas o interrogativas y usar texto que incluya minúsculas y mayúsculas.

Documentación en Línea

El término documentación en línea, se utiliza para referirse al texto que es presentado en una pantalla de video y consiste en un conjunto de técnicas que mejoran la comunicación entre una aplicación y sus usuarios. Los tipos de documentación en línea son los siguientes:

a) Mensajes de estado
b) Mensajes de error
c) Ayuda en Línea

Cada tipo de documentación en línea debe seguir determinadas directrices, que son:

- usar verbos sencillos para describir acciones. No usar sustantivos para reemplazar pronombres, verbos o adjetivos.
- describir las acciones siguiendo un orden lógico.
- usar palabras comunes, sin utilizar términos computacionales o anglicismos, a menos que sea muy necesario.
- evitar utilizar el sentido del humor.

Una documentación en línea bien diseñada pude marcar la diferencia entre una interfaz amigable y fácil de usar, y una que sea confusa, frustrante y que propicie errores.

a) *Mensajes de Estado*

Se utilizan para informar al usuario sobre que proceso está ejecutando el software, en que parte de una secuencia se encuentra, y que opciones ha seleccionado o cuales están disponibles.

Este tipo de mensajes se clasifican en:
1.- Mensajes de estado de procesamiento.- son mensajes que indican que se está llevando a cabo una determinada tarea; por ejemplo el ícono del reloj de arena usado con mouse, o el mensaje que nos dice: Archivando documento...

2.- Mensajes de Estado de Opción.- son mensajes que indican en que modo se está interactuando con la computadora; como por ejemplo: CAPS ON, Ins, etc. Los cuales son desplegados usualmente en la periferia de la pantalla

3.- Mensajes de Localización del Usuario.- Estos mensajes indican al usuario en que posición de la pantalla se encuentra el cursor, usados normalmente en paquetes gráficos o procesadores de texto; por ejemplo: Ren 4, Col 7.

b) *Mensajes de Error*

Los mensajes de error sirven para indicar al usuario que algo ha fallado durante su interacción con la aplicación. Un mensaje de error efectivo, permite al usuario recuperarse después de ocurrido un error, indicándole además, la causa del mismo, reduciendo de esta forma la probabilidad de que vuelva a ocurrir.

La deficiencia más común de éste tipo de mensajes es que no indican la causa del problema que los provocó, por lo cual al diseñarlos se debe intentar ser lo más específico posible en la descripción, minimizando el uso de códigos de error. Además siempre que sea posible, el mensaje debe explicar la acción correctiva a tomar.

Por último, los mensajes de error deben ser consistentes en su formato, en las palabras usadas en su redacción y en el lugar de la pantalla donde son desplegables.

c) *Ayuda en Línea*

Es una característica relativamente nueva, incluída en las aplicaciones de software. Su objetivo es auxiliar al usuario en la comunicación e interacción con una aplicación y se usa principalmente para explicar conceptos, procedimientos, mensajes, opciones de menúes, comandos y teclas de función.

Las habilidades requeridas para diseñar un sistema de ayuda en línea, son diferentes de las que se necesitan para diseñar otros componentes de una interfaz, ya que la explicación de procesos y conceptos mediante palabras no es una tarea fácil.

Existe tres fases para crear un sistema de ayuda eficiente:

a) Planeación.- El diseñador deberá contestarse las siguientes 5 preguntas:
   1 ¿Cual es el alcance del texto de ayuda?
   2 ¿Cuál es su propósito?
   3 ¿Hacia quién está dirigido?
   4 ¿Cuáles son las tareas que se están ejecutando?

5 ¿Cuáles son las limitantes que se presentan?

b) Redacción.- Una vez que la planeación está completa y se está listo para escribir, se deben seguir los siguientes pasos:
   1.-Identificar el contenido del texto de ayuda. Dadas las respuestas de las 5 preguntas anteriores, ¿que es lo que se desea decir?.
   2.-Organizar el texto.
   3.-Escribir el texto evitando utilizar términos computacionales y expresar las ideas en oraciones cortas.
   4.-Cuando sea posible, usar gráficas o dibujos.

c) Evaluación.- Después de la redacción, se deben revisar y realizar las modificaciones pertinentes. Para hacerlo con mayor eficiencia, se debe pedir a otras personas que lean el texto, principalmente a usuarios potenciales.

Sistema de Ayuda Sensitivo al Contexto

Un sistema de éste tipo es aquel que es capaz de responder, con la información adecuada, a las necesidades de ayuda del usuario, sin requerir que éste sea muy específico al hacer el llamado. El usuario simplemente invoca al sistema de ayuda dentro del contexto donde se encuentre ejecutando una tarea, y el software "sabrá" que tipo de ayuda presentar.

Cabe aclarar que la realización de éste tipo de sistema de ayuda no es sencillo de hacer, y en algunos casos se hace susceptible de presentar información inadecuada y/o innecesaria.

I.4.- Lenguajes de Programación de Sistemas Expertos.

En la actualidad existen infinidad de lenguajes y herramientas para programar computadoras. En muchos de ellos se ha trabajado para la realización de S. E., y de entre todos ellos sólo veremos los más utilizados.

Para construir un S. E. se pude partir de diferentes niveles,que por orden de menor a mayor especialización son:

- de un lenguaje ensamblador.
- de un lenguaje de programación imperativo y orientado a los procedimientos.
- de un lenguaje de programación funcional y orientado a las estructuras de datos.
- de un lenguaje imperativo y orientado a los objetos.
- de un lenguaje de programación declarativo y orientado a las reglas.
- de una herramienta o un sistema o entorno de desarrollo.
- de un sistema vacío o concha (shell).

La elección de una u otra herramienta estará condicionada por el objetivo que se desee alcanzar.

I.4.1.- Lenguajes Imperativos.

Los lenguajes imperativos, son aquellos en los que el control del programa pasa siempre a la siguiente línea del mismo, salvo que se le ordene lo contrario.

El programador será el encargado de marcar en el programa, ya sea mediante el número de la línea, o una etiqueta, el flujo que deberán seguir las instrucciones para la ejecución del programa. El modo de funcionamiento del computador se dará en forma clara para el programador.

En la figura 4, se muestra, el diagrama de funcionamiento de un programa escrito en este tipo de lenguaje.

Si se parte de un lenguaje de este tipo o incluso todavía de uno de nivel más bajo como sería el lenguaje de máquina o algún Lenguaje Ensamblador, para la construcción de un S. E., hay que realizar todas las tareas que el lenguaje no sea capaz de hacer.

Algunas de las ventajas que representan los lenguajes imperativos son las siguientes :

- flexibilidad total.
- conocimiento muy generalizado de los lenguajes tanto a un nivel teórico como a uno práctico, por lo cual se ahorra tiempo de aprendizaje.
- máxima eficacia.

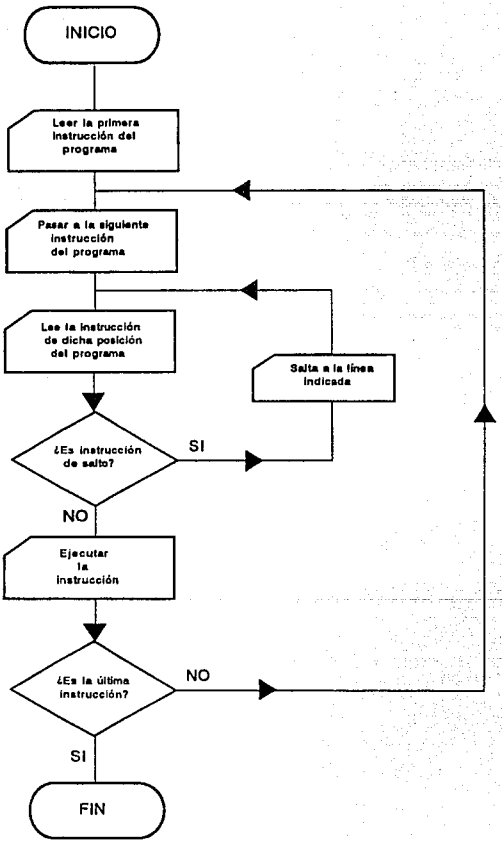


FIG. 4 DIAGRAMA DE FUNCIONAMIENTO DE UN LENGUAJE IMPERATIVO

Dentro de los inconvenientes nos podemos encontrar con los siguientes :

- desarrollos muy largos (costosos) y complejos.
- los sistemas expertos finales son poco reutilizables en otras aplicaciones.
- los programas presentan dificultad al leerlos y modificarlos.

Debido a todas estas razones raramente se parte desde este nivel, solo que las especificaciones de diseño del sistema así lo requieran, se desee una gran optimización en el sistema final, se traten de alcanzar fines didácticos o por satisfactores personales.

Los lenguajes imperativos que se utilizan con mayor frecuencia son : *el Basic, el Pascal, el C y el Fortran.*

I.4.2.- Lenguajes Funcionales.

Estos lenguajes también llamados aplicativos, son aquellos en los que el flujo del programa viene marcado por las necesidades que aparecen al evaluar una función.

El programador solamente se dedicará a indicar el orden de evaluación de las funciones, no siendo necesario que se preocupen por hacer referencia a la posición física que ocupa dentro del programa.

En éste tipo de lenguajes, el control de tipo secuencial, se ha sustituido por cuatro pasos de evaluación, similares a los que se utilizan cuando se desea resolver una función matemática.

El proceso seguido por los lenguajes de tipo funcional lo podemos vislumbrar en la figura 5.

Haciendo un resumen de éste proceso obtendríamos la secuencia siguiente :

- análisis de la función a evaluar.
- búsqueda de la primera subfunción.
- evaluación de la subfunción.
- retorno a la función anterior o valor final de la función si se encuentra en la primera función.

Como podemos observar el proceso es por definición *recursivo*. Al programa se le puede aplicar una entrada de datos de modo que se obtengan una serie de resultados, para realizar una evaluación del mismo.

Si partimos del desarrollo de S. E., los lenguajes funcionales nos ahorran el tratamiento simbólico, gracias a la potencia en su estructura de datos.

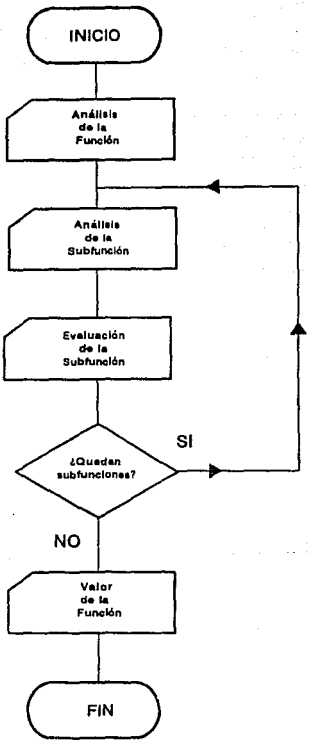


FIG. 5 DIAGRAMA DE FUNCIONAMIENTO DE UN LENGUAJE FUNCIONAL

Por otra parte la construcción de un Mecanismo de Inferencia resulta más sencillo a partir de un control con ésta característica, que con una del tipo imperativo.

Dentro de las ventajas que tiene el utilizar un lenguaje de este tipo para la realización de S. E. tenemos :

- poseen sistemas de tratamiento simbólico.
- sencillez en la construcción de Mecanismos de Inferencia.
- sencillo de aprender y utilizar.

Sus desventajas son las citadas a continuación :

- es caro por los requerimientos mínimos que necesita para funcionar correctamente (necesita gran capacidad de memoria, uso de la memoria virtual y velocidades de ejecución muy altas).
- poco eficaz sobre arquitecturas tradicionales.
- entornos cerrados.
- pueden crecer con facilidad, debido a lo cual no existen dos versiones de lenguaje iguales, ni están basadas en un mismo estándar.

Los lenguajes funcionales mas conocidos son : *Lisp y Logo.*

I.4.3.- Lenguajes Orientados a Objetos.

Los lenguajes orientados al objeto (LOOB's), se distinguen de los demás porque no existe una distinción entre los datos y los procedimientos.

Los programas se forman por los objetos que son a la vez los procedimientos (conocidos también como Procedimientos Locales o Comportamientos) y los datos (datos locales o facetas).

La estructura del programa en objetos es la base fundamental de su gran modularidad.

Las acciones que este realiza son realmente mensajes que son enviados por los objetos y cada uno interpreta el mensaje que le ha llegado.

Un objeto lo podemos ver como la particularización de una clase la cual hereda en este proceso las propiedades de la clase, molde o prototipo, que le dio origen. A causa de este proceso generativo de los objetos, están estructurados jerárquicamente en clases y subclases.

Los objetos tienen propiedades que les caracterizan y que reciben el nombre de atributos. Cada uno de los atributos posee una serie de facetas como pueden ser los valores permitidos, los valores por defecto, etc.

La posibilidad de que los valores sean tomados por defecto, es muy importante, por ser una de las principales características que posee el razonamiento humano.

Los objetos pueden aparecer en tres estados:

activos : esto significa que se encuentran en la lista de hipótesis.

semiactivo: si ha sido propuesta como hipótesis solamente que en forma un tanto confusa.

inactivo : cuando o no se considera o ya se ha rechazado como hipótesis.

Este tipo de programación se realiza siguiendo los tres pasos citados a continuación :

- identificar los objetos que aparecen a lo largo del problema y en la solución.
- clasificación de los objetos por sus semejanzas y diferencias.
- redactar los mensajes que interaccionan a los objetos.
- implantar los métodos o procedimientos en los correspondientes objetos.

El control en un lenguaje orientado a objetos es el siguiente: "entran los datos, se sugieren hipótesis, se ordenan estas hipótesis, se verifican, si se crearon nuevas hipótesis se vuelve de nuevo al punto dos y si no queda una más por verificar se propone una solución en base a las hipótesis ya mencionadas".

Este sistema de control es fácilmente convertible en un Mecanismo de Inferencia. Además de forma dinámica se pueden crear y modificar los objetos.

Algunos de los inconvenientes que se tienen al desarrollar S. E. con lenguajes de este tipo son :

- poco eficaz.
- filosofía de programación completamente nueva.

Dentro de los lenguajes orientados a objetos solo citaremos unos ejemplos : *Duck, Keops, Krl, Omega, Krypton, Qsl, C++,* etc.

I.4.4.- Lenguajes Declarativos.

Los lenguajes declarativos son aquellos en los que solamente hay que indicarle al programa el objetivo que se pretende alcanzar, especificando en el programa el universo en el que debe demostrarlo y las reglas que puede utilizar, dicho en otras palabras especificaremos el "que" pero no el "cómo".

Para lograr éste objetivo es imprescindible que se incorpore un Mecanismo de Inferencia que controle la demostración.

Los lenguajes declarativos más conocidos en la actualidad son: *Prolog y Ops5*.

Haciendo uso de un lenguaje declarativo en la construcción de un S. E., nos ahorramos algunas tareas ya que este tipo de lenguajes cuentan con :

- sistema de Representación de Conocimientos en forma de reglas de producción.

- un Mecanismo de Inferencia de orden 1 que es el propio interprete o el compilador, el cual se basa en la unificación, la búsqueda en profundidad y la marcha atrás en algunos casos, además de hacer uso también de la comparación de patrones, la estrategia de resolución de conflictos. Permite los encadenamientos hacia adelante o guiado por datos y hacia atrás o guiado por objetivos de las reglas.

La realización de un S. E. basado en este tipo de lenguajes se reduce a la presentación y a los módulos de justificación de los resultados y explicación del proceso.

Otra de las ventajas de los lenguajes declarativos para la construcción de S. E. tenemos que son lenguajes muy compactos al no necesitar ninguna estructura de control.

Dentro de los inconvenientes nos encontramos con los que a continuación se mencionan :

- las operaciones de entrada/salida en estos lenguajes son algo complejas de programar.
- el Mecanismo de Inferencia es fijo o no parametrizable.
- sistema de Representación del Conocimiento único.
- carencia de entornos, o sea, que son lenguajes cerrados.
- poco eficaz sobre arquitecturas tradicionales.
- poco fiable al no aparecer explícito el comportamiento del programa, ya que el motor de inferencia es implícito.

Recientemente están apareciendo más lenguajes declarativos con representaciones del conocimiento en formas de reglas de producción como podríamos citar a *Small X*.

I.4.5.- Herramientas, Entornos de Desarrollo y Sistemas Vacíos.

Las herramientas, entornos o sistemas de desarrollo, se encuentran formados por un conjunto de modos de representación del conocimiento (esquemas, reglas de producción, etc.), Lenguajes, Mecanismos de Inferencia Programables o Parametrizables, etc., que dan mayor facilidad para la construcción de S. E..

La mayoría de estos entornos de desarrollo se hacen pensando en que resulten de fácil utilización para cualquier técnico en el

desarrollo de sus prototipos de S. E. sin que necesariamente requiera de grandes conocimientos en computación.

Algunas de las ventajas que representan son :
- gran rapidez en el desarrollo.
- gran comodidad y sencillez de manejo.
- dispone de un abanico muy amplio de utilidades entre las que se puede ir eligiendo las más convenientes en cada caso, y estas suelen ser :
  - *un editor o motor de desarrollo de la base de conocimientos* el cual generalmente se basa en un sistema multitareas y uso del ratón.
  - *subsistema de detección de contradicciones*, repeticiones y errores de tipo sintáctico de las bases de hechos y de conocimientos.
  - *subsistema de aplicación* y justificación del proceso seguido por el Mecanismo de Inferencia.

La desventajas que estos entornos representan son una gran rigidez y una portabilidad muy pequeña con respecto al S. E. desarrollado.

Los sistemas vacíos ("vide"), concha ("shell"), esenciales ("essentiel") o armazón ("skeleton"), a los cuales se les denomina de todas estas formas, son programas en los que únicamente hay que introducirles el conocimiento para hacerlos operativos, pudiendo así implantarse S. E. sin grandes conocimientos en Informática.

La desventaja principal es que son productos prácticamente terminados, con lo que no se tiene ninguna flexibilidad. Esto nos obliga a comprar el producto que cumpla estrictamente nuestros requerimientos.

La división entre entornos de desarrollo y sistemas vacíos es difícil, si bien el criterio más empleado habitualmente es el grado de complejidad que presentan para la construcción de un S. E.; si se requieren conocimientos importantes se denomina herramienta, si solo se necesitan unas ideas generales de programación es un sistema de desarrollo y si lo puede construir "cualquier" persona se dice que es un sistema vacío.

Se darán algunas de las características más destacadas de algunos de los productos existentes actualmente en el mercado:

APES (Augmented Prolog for Expert System / 1982)
Tipo : Sistema de desarrollo compuesto por una serie de módulos que son extensiones del lenguaje Prolog.
Representación del conocimiento : Reglas de producción, restringidas a las *Cláusulas de Horn*.
Mecanismo de Inferencia
  Tipo : determinístico.

Búsqueda : Exhaustiva en profundidad.
Encadenamiento : Hacia atrás.
Utilerías : Editor, lenguaje pseudonatural (inglés), detector de contradicciones en la base de conocimientos, programación directa en Prolog, interfase con el lenguaje C.
Escrito en : Prolog
Disponible para : Minicomputadores y estaciones de trabajo.
Desarrollado por : Hammond y Sergon del departamento de Computer Science del Imperial College de Londres.
Distribuido por : Logic Basic Systems Ltd.

ARITY EXPERT SYSTEM
Tipo : Sistema de desarrollo de sistemas expertos, basado en una extensión del lenguaje Prolog.
Representación del conocimiento : Mediante reglas de producción y marcos.
Factores de certeza : Si.
Mecanismo de Inferencia:
Búsqueda : Exhaustiva en profundidad.
Modos de encadenamiento : Hacia atrás.
Utilerías : Compilador e interprete.
Escrito en : Prolog.
Disponible para : Minicomputadoras.
Desarrollado por : Arity Corp.
Distribuido por : Arity Corp.

CIA (Centrale Intelligence Artificialle / 1985)
Tipo : Sistema de desarrollo de sistemas expertos.
Representación del conocimiento : Mediante reglas de producción.
Relaciones : Grafos.
Mecanismo de Inferencia:
Parametrizable : Si.
Escrito en : Basic (Quick Basic)
Disponible para : Microcomputadoras.
Desarrollado por : AAF.
Distribuido por : AAF.

ESE (Expert System Environment)
Tipo : Sistema de desarrollo de S. E.
Representación del conocimiento : mediante reglas de producción.
Representación del metaconocimiento : si, FCB (Focus Control Block).
Motor de Inferencia
Modos de encadenamiento : hacia adelante y hacia atrás.
Utilerías : editor de reglas, representación en árbol, acceso a rutinas escritas en COBOL, FORTRAN, PL/1, PASCAL, etc, gráficos y ayuda en línea.
Escrito en : Pascal
Disponible para : computadoras IBM bajo S.O. VMS.
Desarrollado por : IBM

MICRO EXPERT Y MACRO EXPERT
Tipo : Sistema vacío.
Disponible para : Microcomputadoras.
Desarrollado por : ISIS system Ltd.
Distribuido por : ISIS system Ltd.
NOTA: Descrito con mayor detalle en el Capítulo II
Tema de desarrollo de la Interfaz.

TURBO EXPERT
Tipo : Sistema de desarrollo de S. E.
Representación del Conocimiento : mediante reglas de producción. Representa tambien metarreglas.
Motor de Inferencia
Modos de encadenamiento : hacia adelante y hacia atrás.
Utilerías : verificación de sintaxis y apertura de ciclos, procedimientos en C, sistema de multiventanas.
Disponible : para microcomputadoras (IBM y compatibles).

I.5.- Construcción de Sistemas Expertos

La adquisición del conocimiento de un experto humano, es un proceso gradual que puede llevarse semanas, meses ó años.

El prototipo de modelos de creciente complejidad son construidos y refinados hasta que el sistema rinde en un nivel aceptable. El diseño de un S. E. es un arte y el diseñador del modelo debe estar dispuesto para adaptarlo a un medio ambiente que cambie dinámicamente. Es de fundamental importancia la selección de una aplicación real. En la figura 6, se puede observar un diagrama a bloques de manera general, que facilita la comprensión del proceso de construcción de un S. E.

Se debe también considerar muy cuidadosamente si el planteamiento del sistema de aplicación sería de valor si se implementase. Mientras esto puede parecer obvio, muchos sistemas han progresado a períodos relativamente avanzados, pero se ha visto que después no hay mercado para dichos sistemas.

La parte más difícil de desarrollo de un sistema, es el diseño de las primeras etapas. Una de las características distinguibles del diseño de un S. E., es que uno está constantemente dirigiéndose hacia una aplicación en constante evolución. Mientras muchos sistemas de software pueden ser diseñados con especificaciones firmes, los S. E. usualmente no pueden ser fácilmente especificados antes de que el sistema sea construido.

El prototipo inicial del modelo puede ser poco claro, estar incompleto y contener equivocaciones, pero al menos esto provee un punto de enfoque para que el experto pueda hacer sus sugerencias.

Uno de los sacrificios que se tienen que hacer cuando se construyen S. E., es estar constantemente identificando las debilidades y defectos del sistema que está siendo diseñado. Se tiene que estar expectante de revisiones mayores que se le hagan al sistema, particularmente en las primeras etapas. Esto es algunas veces provechoso para ver el ritmo con el cual se está adquiriendo el conocimiento una vez que el prototipo del modelo ha sido construido.

La carga de trabajo en el diseño del modelo recaerá en el Ingeniero del Conocimiento, quien extraerá el conocimiento de la manera más adecuada para representarla en la computadora. Existen algunas guías preliminares que serán valiosas para que el diseñador del modelo y el experto vayan en la dirección adecuada para la construcción inicial del prototipo, particularmente si el modelo puede ser planteado como una clasificación de modelos.

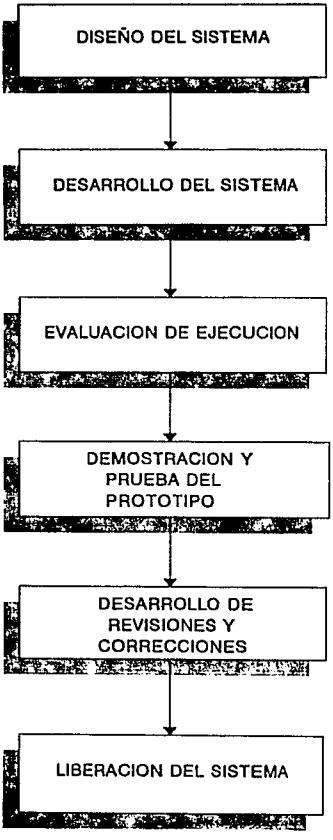


FIG. 6 PASOS PARA LA CONSTRUCCION DE UN SISTEMA EXPERTO

I.5.1 Metodología de Construcción

1.- Diseñar un modelo concentrándose primero, en un pequeño conjunto de hipótesis que incluya en un primer prototipo, sólo aquellas que sean más productivas.

2.- Identificar grupos de conclusiones que sean más discriminantes.

3.- En las reglas de decisión, combinar el menor número de conclusiones necesarias para aceptar o rechazar entre hipótesis. Incrementar el número de reglas conjuntivas cuando la regla resultante, incremente significativamente el poder del Sistema, para aceptar o rechazar la conclusión. Existen potencialmente, un número grande de combinaciones de conclusiones. Un modelo deberá contener el menor número de éstas que sean suficientemente específicas para confirmar, rechazar y discriminar.

4.- Incluir conclusiones que no puedan ser poderosamente predictivas o discriminatorias entre ellas mismas, pero las cuales puedan significativamente mejorar la calidad de decisión, poniendo un contexto o punto de atención para el proceso de toma de decisiones. El prototipo del modelo llega a ser más realista en ésta etapa.

5.- Determinar si las abstracciones pueden ser hechas; por ejemplo, algunas reglas de producción pueden ser declaradas como satisfechas si *n* o más que no estén en la lista de conclusiones son satisfechas. Tales reglas pueden ser útiles para expresar los efectos combinados para la relativa independencia del conjunto de conclusiones que, propiamente, contribuyen marginalmente en una inferencia.

6.- Ver si las hipótesis adicionales intermedias pueden ser introducidas para simplificar el razonamiento. Puede haber muchas diferentes combinaciones de conclusiones que introducen reglas para cada conjunto de hipótesis.

7.- Probar el modelo con una base de datos de casos complejos. Considerar aquellos casos que producen conclusiones equivocadas, y determinar cómo modificar el modelo para corregir los errores. Revisar el modelo y examinar las conclusiones para los casos y el efecto de éstas revisiones en otros casos no vistos.

I.5.2 "Rapid Prototyping". Construcción rápida de prototipos

El desarrollo de S. E. impone los siguientes riesgos:
- no existen implementaciones similares que puedan servir de orientación al encargado del desarrollo
- en muchos puntos, los requisitos necesarios están esbozados con muy poca precisión.

El método ideal para desarrollar un S. E. es el llamado "Rapid Prototyping". Es un método iterativo que consta de los siguientes apartados:

- **Identificación del problema**
- **Conceptualización del mismo**
- **Formalización e implementación**
- **Verificación y prueba**

estos pasos son repetidos hasta el cumplimiento de un criterio de interrupción. De esta forma es posible crear un prototipo capaz de ejecutar una función y de mejorarlo cada vez más con un esfuerzo de desarrollo relativamente pequeño.

I.5.3 Ingeniería del Conocimiento

En el desarrollo de un S. E. existen tres tareas muy diferentes que en principio deben desarrollar tres personas o grupos dado lo diverso de las tareas a realizar.

Las personas que intervienen son :

El experto.- es la persona o grupo de personas que proporcionan los conocimientos sobre el campo de actuación del S. E. y validan el funcionamiento del mismo desde el punto de vista de su aptitud para desarrollar las tareas para las que se ha creado.

El Ingeniero del Conocimiento.- que es la persona que aplica el conocimiento y métodos que permiten adquirir y representar el conocimiento del experto en un sistema informático.

Es sin duda alguna la persona clave en el desarrollo, y existe ya un perfil que debe reunir para que su labor sea útil:

- Espíritu abierto: que le permita la comprensión de las exposiciones de los expertos, el ser conciente de su ignorancia sobre el campo del experto y el saber tratar a personas de muy diversos campos y niveles de formación.
- Conocimientos multidisciplinarios: con el fin de poder abordar problemas de dominios muy diversos y que lo mantengan al día tanto a nivel de Software como de Hardware.
- Curiosidad intelectual; que le lleve a profundizar en los problemas y en las soluciones de los mismos.
- Organización: con el fin de poder manejar grandes cantidades de información muy diversa.

El Programador.- es el encargado de transferir el conocimiento obtenido a un lenguaje o entorno de desarrollo para un computador en concreto. Indudablemente debe conocer los lenguajes y entornos de programación de S. E. más usuales.

Esta estructura es completamente teórica, ya que a la hora de emprender el desarrollo de un S. E. se empieza por descubrir la carencia de técnicos que reúnan estos conocimientos, por lo que hay

que proceder a su formación lo que vuelve lento el desarrollo e induce a que se unan las tareas de Ingeniero del Conocimiento, Experto y Programador.

I.5.4 Prueba y evaluación de Sistemas Expertos.

Durante el diseño de S. E. se pueden realizar muchos cambios probables sobre el diseño original.

Para realizar estos cambios es necesario analizarlo mediante algún método específico, ya que pueden surgir dificultades o problemas.

Existen dos aproximaciones que ilustran los extremos en la evaluación de S. E.. Uno es el de la aproximación *"anecdotal"*; con esta aproximación el modelo que se ha diseñado describe todas las experiencias que han sido exitosas, u otras situaciones donde los programas se ejecuten en forma correcta.

Los diseños típicos sólo mantienen una cantidad pequeña de términos cortos para checar la memoria sobre posibles cambios debidos a opciones o casos de problemas anteriores para los cambios comunes en el modelo.

Al igual que la inteligencia humana, se espera que la ejecución de un S. E. deba mejorar con la experiencia de acuerdo al tiempo. Existen S. E. que carecen de la poderosa capacidad de aprender y son sometidos a pruebas para cada caso y cada área específica.

La otra aproximación hace énfasis en la evaluación empírica de la ejecución sobre los casos del problema, almacenados en una base de datos. Se someten a pruebas rigurosas bien específicas para comparar el procedimiento del modelo y la independencia para la interpretación obtenida. La prueba tradicional se realiza desde fuera del sistema; ésta es relativamente bien desarrollada y con eficiencia.

Para este tipo de evaluación se toma un ejemplo de las opciones para probar la base de datos, y se puede examinar bien el cambio en la ejecución para el caso presentado en la base; existen tipos diversos de herramientas para poder desarrollar el uso de la información en la parte del diseño de un modelo experto.

En estas dos aproximaciones observamos que la aproximación empírica es superior a la anecdotal, ya que en la empírica los métodos implementados se realizan sobre la base de datos que por lo regular presenta severos obstáculos en la práctica del programa.

Existen varias herramientas relativamente fáciles para auxiliar la evaluación de modelos o diseños, y su ejecución.

Observemos las siguientes directrices :

1.- Checando la consistencia de resultados desde algunos cambios del modelo.
Durante el transcurso de una consulta se pueden almacenar las opciones a lo largo de una base de datos para asignarle un peso a las instrucciones del programa, indicando cómo se cubrieron las hipótesis.

2.- Buscando en la base de datos por paréntesis.
La técnica para búsqueda de paréntesis sobre una base de datos especifica, puede provenir de la evaluación de herramientas, ambos en formulación de reglas de decisión y checando la consistencia subjetiva de la construcción de reglas de decisión.

3.- Partiendo de las computadoras y las conclusiones expertas
Si no es una evaluación muy simple de las herramientas, puede ser aplicada cuando se tienen las conclusiones de expertos almacenadas por cada caso en la Base de Conocimientos. Una información semejante no es siempre evaluada pero cuando esto sucede, se puede obtener una evaluación definitiva para lo cual se partirá de las conclusiones del experto humano.

El diseño de un modelo y el análisis de su ejecución debe hacerse dentro de los pasos siguientes:

- iniciar el proceso del modelo de aplicaciones
- entrada de datos: casos y conclusiones expertas
- ejecución del análisis del modelo
- generación de un modelo de experimentos refinados
- revisión de los efectos en el cambio de modelo sobre los casos de conclusiones.

Para facilidad nuevamente se identifica la conclusión final, el diseño del modelo puede listar mayores o menores observaciones y componentes específicos de las reglas, los cuales suelen llegar a la conclusión.

El proceso de improvisación de la ejecución de una aplicación típica del modelo suele constar de las iteraciones de los pasos siguientes:

- ejecución de las reglas sobre los casos almacenados
- analizar las reglas
- revisar las reglas

I.5.5 Consulta de un Sistema Experto

Los S. E. desempeñan el papel de asistentes inteligentes y competentes del experto. El conocimiento de los expertos es necesario para muchas personas que, por regla general, no poseen una formación especializada en dominios muy concretos.

Si en un sector determinado hay pocos expertos, un S. E. puede entonces poner a disposición, de forma constante, los conocimientos de dichos expertos. Pero no se trata de sustituir al experto o de mantener usuarios y expertos alejados entre sí, sino más bien se trata de poner en las manos de los usuarios una herramienta efectiva que libere a los expertos de trabajos rutinarios.

El usuario, no obstante, debe disponer de un conocimiento especializado lo suficientemente amplio sobre el entorno en el cual trabaja para poder manejar el S. E. Como regla general puede decirse que la diferencia entre usuario y experto no debe ser demasiado grande . Ambos deben poder "conversar" en un lenguaje común sobre el problema mediante el S. E. Esta es una consecuencia sencilla del lenguaje especializado que describe el problema.

Esta situación se agudiza aún más ante el hecho de que no está básicamente asegurado que el S. E. pueda ofrecer una respuesta utilizable. No en todos los casos pueden captarse o preeverse del todo las condiciones marginales bajo las cuales se emite la respuesta del S. E. El usuario estará obligado a realizar una mínima valoración de la relevancia que tiene la respuesta ofrecida por el sistema. La responsabilidad de las acciones que se desencadenan basadas en una respuesta del sistema recae en el usuario.

La consulta misma transcurre en general según el esquema siguiente:

Primero se plantean al usuario algunas preguntas generales para alcanzar una determinación aproximada del contexto, Una primera valoración, que se produce dependiendo del método prefijado de procesamiento de los conocimientos del S. E. en uso, desemboca entonces en un verdadero diálogo con el usuario, orientado al objetivo. El diálogo, por parte del sistema, está a menudo dimensionado para ir confirmando o rechazando hipótesis (por ejemplo, comprobación de una causa o fallo), o para realizar una aproximación sucesiva hacia un objetivo introducido de antemano (por ejemplo, una configuración perfecta de una red de ordenadores).
En el marco de este diálogo, el sistema realiza constantemente las actividades necesarias para alcanzar la solución del problema y para explicar el estado del sistema.

El sistema se comporta como un experto y:

- plantea preguntas precisas
- informa (según el caso sólo a petición del usuario) sobre los resultados intermedios y las hipótesis modificadas
- determina el resultado, por ejemplo, un diagnóstico
- justifica el resultado y explica, y si es necesario permite consultar el historial completo de la consulta.

Es posible:

- visualizar todas las entradas del usuario
- confrontar el resultado obtenido con todos los demás resultados posibles
- visualizar las reglas activas y no utilizadas; incluso las reglas activadas, que sin embargo fueron rechazadas en una misma hipótesis, pueden elegirse y visualizarse.

El resultado alcanzado por el S. E. dependerá de la calidad de las respuestas del usuario. En general, el S.`E. no puede comprobar la consistencia de las diferentes respuestas del usuario. Especialmente cuando el usuario se ve obligado a realizar correcciones respecto a algún dato introducido con anterioridad, el sistema no puede determinar qué otras respuestas dadas y procesadas pueden estar afectadas. La responsabilidad de la consistencia de este "mundo meta" -desde el punto de vista del sistema- recae en el usuario y no en el sistema, como ya se había mencionado con anterioridad.

I.6.- Aplicación de los Sistemas Expertos

En esta parte, se definirán los campos de aplicación de los Sistemas Expertos en la actualidad.

La utilidad que se espera obtener del presente inciso, no es dar a conocer las múltiples aplicaciones de los S. E., sino más bien tratar de inducir en que problemas puede ser adecuado aplicar los mismos.

I.6.1 Tareas que realizan los Sistemas Expertos

Interpretación

Consiste en encontrar el significado de los datos de entrada obtenidos por sensores o introducidos por el usuario. Con frecuencia aparecen datos contradictorios por lo que hay que dotar al S. E. de conocimiento para resolver este tipo de problema y que en general puede tratarse mediante: la valoración en forma conjunta de los datos, la eliminación de algunos de ellos o la imposibilidad de realizar la interpretación debida a errores.

Existen dos tipos de interpretación:
- *Análisis.*- la interpretación de los datos se obtiene mediante la separación o distinción de las partes que forman a los mismos.
- *Síntesis.*- la interpretación de los datos se obtiene mediante la combinación de los mismos.

Diagnóstico

El diagnóstico consiste en identificar las causas internas que provocan un problema, avería o disfunción a partir de una serie de datos o síntomas que son consecuencia de la misma y que son observables.

Para un diagnóstico es necesario la interpretación de los datos por parte de un experto, de otro S. E. independiente o incluido en el mismo de diagnóstico, y el conocimiento de la estructura y funcionamiento del sistema.

Existen dos formas de búsqueda de un diagnóstico correcto mediante un S. E. La primera de ellas consiste en seleccionar el mejor diagnóstico de entre un conjunto de diagnósticos predefinidos, y la segunda consiste en construir el diagnóstico.

Los S. E. en diagnóstico encuentran múltiples dificultades a la hora de realizar su tarea como son:
- manifestaciones nuevas, es decir síntomas que no se habían observado con anterioridad.

- manifestaciones debidas a varias causas.
- relaciones no biyectivas entre los datos y las causas.
- fallos o averías de aparición intermitente.
- existencia de varios fallos en cadena.

Reparación o corrección

Consiste en la proposición de las acciones correctoras necesarias para la resolución de un problema. Para realizar una reparación es necesario realizar previamente un diagnóstico, que puede ser hecho por un experto, un sistema independiente o una parte del mismo S. E. de prueba.

Este tipo de sistemas tienen que cumplir los siguientes objetivos:
- reparación lo más rápida y económica posible
- orden de las reparaciones cuando hay que realizar varias
- evitar los efectos secundarios de la reparación, es decir, la aparición de nuevas averías por la reparación.

Control

Un sistema de control consiste en la realización de las tareas de interpretación, diagnóstico y reparación de forma secuencial. Con ello se consigue conducir o guiar un proceso o sistema.

Los sistemas de control pueden ser en **lazo abierto** si en el mismo, la realimentación o el paso de un proceso a otro lo realiza el operador o, en **lazo cerrado**, si no tiene que intervenir el operador en ninguna parte del mismo.

Simulación, pronóstico o predicción

La simulación es una técnica consistente en crear modelos basados en hechos, observaciones e interpretaciones, sobre el ordenador con el fin de estudiar el comportamiento de los mismos mediante la observación de las salidas para un conjunto de entradas (sensibilidad del sistema frente a los datos).

Las técnicas tradicionales de simulación requieren modelos matemáticos y lógicos que describen el comportamiento del sistema bajo estudio.

En la aplicación de S. E. para simulación, hay que diferenciar cinco configuraciones posibles que son :

1.- Un S. E. puede disponer de un simulador con el fin de comprobar las soluciones y en su caso rectificar el proceso que sigue.

2.- Un sistema de simulación puede contener como parte del mismo a un S. E. y por lo tanto el S. E. no tiene que ser necesariamente de simulación.

3.- Un S. E. puede controlar un proceso de simulación es decir que el modelo está en la Base de Conocimiento y su evolución es función de la Base de Hechos y el Mecanismo de Inferencia, y no de un conjunto de ecuaciones aritmético-lógicas.

4.- Un S. E. puede utilizarse como consejero del usuario y del sistema de simulación.

5.- Un S. E. puede utilizarse como máscara o sistema frontal de un simulador con el fin de que el usuario reciba explicación y justificación de los procesos.

Los problemas de predicción o pronóstico no son más que aplicaciones de la simulación.

Otras Tareas

De estas 5 aplicaciones derivan casos especiales que son :

- *Monitorización*.- Es parte de la interpretación, y consiste en la comparación continua de los valores de las señales o datos de entrada y unos valores que actúan como criterios de normalidad.

- *Planificación*.- Es la realización de planes o secuencias de acciones y es un caso particular de la simulación. Está compuesto por un simulador y un sistema de control. El efecto final es la ordenación de un conjunto de acciones con el fin de conseguir un objetivo global.

- *Diseño*.- El diseño es un proceso cíclico, en el que se va modificando una solución, con el fin de que satisfaga los objetivos propuestos, tanto los referentes a recursos y medios empleados, como a los resultados.

I.6.2 Areas de aplicación de los Sistemas Expertos

Los S. E. tienen una infinidad de áreas de aplicación, pero consideramos necesario solo mencionar aquellas que han tenido mayor éxito debido a su alta efectividad.

Medicina

Los S. E. en medicina podemos dividirlos para su estudio en dos grandes grupos :

- los S. E. desarrollados, y que son aquellos en los que se ha partido desde el nivel más bajo, por lo que los mismos son realizados por tareas de Ingeniería del Software y del Conocimiento.

- los sistemas que denominaremos implantados, que son aquellos en los que se ha partido de un sistema vacío o shell ya comercializado.

Fué en los años 60's que comienza a existir la inquietud sobre lo que se denominó diagnóstico médico automático. Entre 1964 y 1965 se desarrollaron los primeros sistemas expertos en el campo de la Medicina, que fueron: MYCIN, PIP, CASNET e INTERNIST.

Los S. E. en Medicina han tratado diversos temas como son : Diagnóstico y tratamiento de Glaucoma, Neurología, enfermedades infecciosas bacterianas, Medicina interna, Insuficiencia renal, Radiología, Control de diabetes, etc.

Finanzas y Gestión

Es el segundo campo en importancia en cuanto a aplicación de los S. E., debido principalmente a las fuertes inversiones que se han realizado.

Algunas de las aplicaciones son las siguientes :
- Análisis de mercado.
- Análisis de riesgos y tasación de seguros.
- Asesoría jurídica y fiscal.
- Aplicación de impuestos y tasas.
- Concesión de créditos y prestamos.
- Gestión de personal.
- Planes de inversión de capitales.
- Supervisión de estados financieros.

Algunas de las características comunes a todos estos desarrollos son : trabajo en tiempo real, flexibilidad, vida operativa corta y alta portabilidad.

Educación

Todos los S. E. son por sí mismos herramientas utilizables para la enseñanza, ya que explican los procesos que realizan y justifican su comportamiento. De esta forma los S. E. pueden :
- diagnosticar los problemas de aprendizaje en los alumnos.
- recomendar una metodología sobre impartición de clases.

Un S. E. completo en enseñanza debe de tener un módulo en diagnóstico, uno de conexión y un tercer que contenga la Base de Conocimiento del tema que debe de enseñar.

Industria

De los distintos tipos de S. E. los más empleados en la

industria son los de diagnóstico y reparación de averías tanto de producto como de maquinaria, debido a lo siguiente :

- los equipos y productos tiene una complejidad creciente.
- los sistemas industriales se diseñan a prueba de fallas.
- las reparaciones deben de realizarse sin interrumpir el servicio.
- presencia cada vez mayor de equipos específicos y sofisticados.

Algunas de las aplicaciones industriales son : Diagnósticos de control de calidad, Detección y actuación en caso de alarmas y emergencias, Configuración de equipos y sistemas bajo demanda, Generación de especificaciones y manuales de utilización, Control de procesos y Gestión óptima de recursos.

Electrónica, Informática y Telecomunicaciones

En la industria de la electrónica el sistema más utilizado es el de diseño, diagnóstico y reparación.

Veamos una serie de aplicaciones :

- diseño de circuitos de alto grado de integración.
- sistemas inteligentes de autodiagnóstico.
- configuración de equipos y sistemas.
- control de redes de comunicación.
- programación automática.
- ajuste de equipos y sistemas.

Militar

Desde su origen, la informática esta presente en el campo militar pasando a ser uno de los factores claves en los desarrollos actuales y más cuando el poder de los mismos es incluso excesivo para los fines que fueron desarrollados.

Dentro de la milicia existen múltiples aplicaciones dentro de las cuales podemos citar :

- elección inteligente de contramedidas electrónicas.
- guía de vehículos y proyectiles de forma semiautomática.
- reconocimiento automático de patrones.
- interpretación de señales provenientes de sensores.
- tiempos mínimos de comunicación.
- optimización de carga.

I.6.3 Responsabilidad de los Sistemas Expertos

El grado de confianza en el S. E. no solamente depende de la eficacia demostrada, sino también del nivel de conocimientos del usuario; cuanto mayor sea este nivel, menor será la confianza que

se deposite en el sistema.

Los S. E. según la responsabilidad que se deposita en ellos pueden clasificarse en :

- *Sistema de Comprobación :* Es el menor grado de confianza y de responsabilidad, el experto solo lo consulta después de haber decidido.

- *Sistemas de ayuda o consulta :* Es un grado medio de confianza. Se considera al S. E. una herramienta válida que facilita las tareas y la resolución de los problemas.

- *Sistemas de decisión :* La confianza depositada en él es la máxima y el S. E. es autónomo.

Por el momento, el papel de los S. E. es secundario y se limita a tareas de ayuda, consejo y comprobación.

I.7.- Alcances y Futuro de los Sistemas Expertos.

Después de haber realizado una revisión histórica de los S. E. se puede concluir que las expectativas de los años 60's fueron hasta cierto punto un poco exageradas ya que a la fecha no ha sido posible alcanzar los objetivos esperados, y esto mismo ocasionó que el estudio de la I. A. y de los S. E. se redujera significativamente a sólo Universidades y algunos Institutos.

Mismos que, al pasar de los juegos y problemas sencillos a la realización de S. E. para problemas reales se dieron cuenta de que esta era una tarea realmente difícil; por lo cual haciendo una revisión real se podrá observar que sólo algunos cuantos S. E. han dado los resultados esperados. Ya que esto no es una tarea fácil, se requiere de un grupo de trabajo que involucra al experto o los expertos humanos y a los Ingenieros de Conocimiento.

Así pues, vemos también el hecho de que el desarrollo de S. E. se ha enfocado a sistemas de tiempo compartido debido a que la necesidad de recursos es muy grande y aún cuando el desarrollo de microcomputadores ha ido hacia arriba, estos no cuentan con los recursos necesarios.

Otro problema al cual se ha enfrentado el desarrollo de S. E. es el impacto social ya que no todo mundo le ha tendido los brazos, esto en parte se debe a que todavía se tiene la idea de que la computadora no sirve para nada, o el temor a usarla; pero afortunadamente esta idea a ido cambiando poco a poco y algunos centros de investigación y profesionales ya las observan como una herramienta muy poderosa que puede incrementar la productividad.

Por otro lado el uso de S. E. podría crear la necesidad de tener una cierta habilidad o conocimientos especiales, por lo cual en muchas industrias la gente no ve con buenas intenciones la llegada de dichos sistemas sin realmente comprender que este tipo de sistemas nunca llegarán (por lo menos en esta década) a reemplazar a los expertos humanos.

Hacia donde se dirige la Investigación.

Hasta ahora se ha probado que el método de reglas es una buena manera de representar conocimientos, y aún cuando éste modelo tiene sus defectos, nosotros pensamos que partiendo del mismo (como se esta haciendo en laboratorios e institutos) se podrán obtener modelos mucho más poderosos para representar conocimiento.

Avances en la Representación del Conocimiento

Dada la importancia de éste rubro, se ha motivado el estudio de carácter teórico sobre el mismo, con los siguientes avances para el futuro próximo :

- Representación del sentido común mediante estructuras jerárquicas y lógicas borrosas (fuzzy logic).
- Eliminación de las incoherencias de las bases de conocimientos, presencia de un objeto y su contrario, mediante la utilización de redes semánticas.
- Adquisición automática de conocimientos.
- Depuración automática de bases de conocimientos.

Adicionando nuevos tipos de conocimiento al sistema.

Aún cuando se ha observado que el modelo de reglas de producción es útil para representar el conocimiento, es cierto que este último es incapaz de representar cierta información que es usada por el experto humano como es la información casual, la estructura o función del conocimiento y las relaciones matemáticas; para poder representar dicho tipo de conocimiento es necesario un modelo más profundo, pero desgraciadamente el construir dicho modelo aparte de ser muy complicado, muchas veces en el análisis final muestra que el esfuerzo realizado no valió la pena; así como también se tiene muy poca información disponible de este tipo de modelos.

Facilitando la tarea de adquisición del conocimiento.

Uno de los principales problemas en el desarrollo de S. E. es el tiempo necesario para codificar el conocimiento del experto, es por ello que se ha tratado de que el propio experto sea capaz de codificar su conocimiento; y una forma relativamente sencilla sería el crear un marco general de captura aún cuando no todos los problemas son iguales y presentan diferentes características; bastaría con unas pequeñas modificaciones a dichos formatos generales, con lo cual el tiempo necesario para adquirir el conocimiento se reduciría de forma muy significativa.

Otro punto realmente importante es el hecho de tratar de realizar S. E. que aprendan de la experiencia; por ejemplo el que un sistema después de realizar varias inferencias visualice una nueva regla y la almacene en su base. Desgraciadamente esto todavía no se logrà, por lo cual no es posible crear S. E. que por sí solos creen S. E. y por ende es todavía necesario el Ingeniero de Conocimientos y el Experto.

CAPITULO II

MECANISMO DE INFERENCIA MICROEXPERT

La eficiencia en la creación de Sistemas Expertos puede aumentarse en gran medida con la aplicación de *Shells*

Un Shell -expresado en forma sencilla- es un Sistema Experto que contiene una base de conocimientos vacía. Existen el Mecanismo de Inferencia y a veces también la interfase del usuario.

Ya que el Mecanismo de Inferencia depende del problema o grupos de problemas, no existe ningún shell para todas las aplicaciones, sino que hay que buscar un shell apropiado para cada aplicación.

También es posible que haya que desarrollar adicionalmente partes del Mecanismo de Inferencia. Según el tamaño que tenga esta parte se tendrá que pensar si la aplicación de un shell determinado sigue siendo apropiada.

Si el Ingeniero de Conocimiento conoce bien este shell, es decir, si por ejemplo, conoce exactamente cómo son procesadas las reglas, entonces sólo tendrá que concentrarse en la creación de la Base de Conocimientos.

A menudo el Shell contiene Frames. Estos son marcos previamente preparados, en los que, por ejemplo, sólo se introduce el nombre del objeto, sus cualidades y los correspondientes valores. Las relaciones entre los objetos se indican mediante señalización de los dos objetos y del tipo de relación que exista entre ellos. El trabajo de implementación debe reducirse al mínimo.

Microexpert es un shell de un Sistema Experto, con esto queremos decir que es un mecanismo de inferencia que puede trabajar sobre cualquier Base de Conocimientos que haya sido construida siguiendo la sintaxis de Microexpert. Microexpert permite llegar a una conclusión a partir de datos dados por el usuario, es decir, es un sistema de consulta. A continuación describiremos detalladamente la sintaxis de dicho shell.

II.1.- Sintaxis

Regla

Una regla es la codificación correspondiente a un enunciado de tipo condicional en español. Un enunciado condicional en nuestro idioma se compone de una conclusión y varias condiciones. Si todas las condiciones se cumplen, la conclusión es verdadera, de otra manera la conclusión es falsa. Veamos un ejemplo:

Si es vivíparo
y come carne
y ladra
es un perro.

Si es ovíparo
y no es un ave
y no se arrastra
y vive en los pantanos
es un cocodrilo.

Sintaxis de una regla en Microexpert

La sintaxis de una regla en Microexpert es la siguiente:

```
numero
IF atributo1 IS valor1
AND atributo2 IS valor2
AND atributo3 IS valor3
      .
      .
      .
THEN objeto1 IS valori
```

donde número es el número consecutivo que se le debe dar a cada regla escrita en Microexpert, atributo es cada una de las cualidades o propiedades de un objeto, valori son los diferentes valores que pueden llegar a tomar los atributos y objetos, son las acciones o conclusiones a que se llega después de haber hecho una consulta y validado las condiciones.

Para construir una regla en Microexpert se debe de:

1.- Asignar un número entero consecutivo a cada regla, empezando con el número 1.

2.- Sobre un renglón escribir exclusivamente el número de la regla.

3.- En el siguiente renglón comenzar con la palabra reservada IF, seguir con el nombre de un atributo, enseguida la palabra reservada IS y finalmente el valor que toma ese

atributo.

4.- Opcionalmente: la palabra reservada AND, seguir con el nombre de un atributo, enseguida la palabra reservada IS y finalmente el valor que toma ese atributo.

5.- Repetir el paso 4 tantas veces como sea necesario.

6.- Colocar al inicio de un renglón la palabra reservada THEN seguir con el nombre de un objeto, enseguida la palabra reservada IS y finalmente el valor que toma ese objeto.

7.- Terminar la regla con un punto en un renglón aparte.

Como se puede observar de las condiciones anteriores para la construcción general de una regla en Microexpert, las palabras reservadas para este shell son: *IF, IS, AND, THEN*.

Adicionalmente se cuenta con una función de comparación, llamada *function_compare*, que se utiliza para realizar comparaciones entre atributos y números. La sintaxis para la función de comparación es:

```
function_compare(atributo,'operador','número')
```

lo cual significa que:

1.- Se debe de colocar el nombre de la función de comparación.

2.- Abrir un paréntesis.

3.- Colocar el nombre de un atributo seguido de una coma.

4.- Escribir el operador relacional deseado encerrado entre apóstrofes.

5.- Colocar una coma.

6.- Escribir entre apóstrofes el número con el cual se desea comparar.

7.- Cerrar el paréntesis.

Los operadores relacionales permitidos son:

>	mayor que
<	menor que
=	igual
>=	mayor o igual
<=	menor o igual
<>	diferente

Por lo anteriormente expuesto, notamos que se tiene otra palabra reservada para Microexpert, aparte de los símbolos de

comparación, que es: *function_compare*.

Hasta el momento se ha visto que se utilizan la coma, el punto y el paréntesis, que no son palabras reservadas, pero que para Microexpert forman su conjunto de delimitadores.

Por cada atributo diferente que se utilice en la definición de las reglas, debe hacerse una declaración de "Prompt". Un prompt es una estructura adicional de Microexpert que nos sirve para que el sistema obtenga datos; de manera general es la pregunta que el sistema debe hacernos una vez que se ha empezado la consulta, es la forma como el sistema obtiene datos que le permiten llegar a una conclusión. La sintaxis para la declaración del prompt es la siguiente:

```
PROMPT atributo
     .
     .
     .
texto
     .
     .
```

Lo cual significa que se debe:

1.- Colocar la palabra reservada PROMPT seguida de un nombre de atributo que vayamos a usar posteriormente en la definición de las reglas.

2.- En un renglón aparte, escribir el texto que se vaya a desplegar en pantalla cuando el usuario esté consultando la base de conocimientos.

3.- Terminar la definición de este prompt con un punto en un renglón aparte.

Los atributos que se vayan a utilizar con la función de comparación function_compare, deben ser declarados un poco diferentes. La sintaxis para los prompts numéricos es:

```
NUMERIC PROMPT atributo
     .
     .
texto
     .
     .
```

cuya única diferencia con el texto anterior es que la palabra PROMPT le antecede la palabra NUMERIC.

Comandos de Microexpert

Una vez que se ha diseñado la base de conocimientos (por medio

de la construcción de reglas), lo que sigue es capturarla con la ayuda de algún editor de textos. Microexpert asume que la extensión de sus archivos es .KB (Knowledge Base, Base de Conocimientos). Ya que tenemos la base de conocimientos en un archivo, podemos proceder a su consulta.

Cuando Microexpert comienza su ejecución, nos pide el nombre del archivo que contiene la Base de Conocimientos. A continuación, la base será analizada con el fin de determinar si está correctamente escrita. En caso de que haya algún error (de sintaxis) será necesario hacer la corrección de la base. Si no hay error, podemos entonces utilizar los comandos de Microexpert, que son:

WHY	Explica porqué se hace una pregunta en especial
WHAT	Despliega los hechos que se han determinado hasta el momento durante la consulta
RULE n	Despliega el contenido de la regla número 'n'
HOW n	Explica cómo puede probarse la regla n
WHATIF valor	Indica lo que sucede si se da un 'valor'
QUIT	Termina la acción que está desarrollando en ese momento

A una pregunta del Mecanismo de Inferencia, podemos dar <ENTER> y nos mostrará en pantalla cuáles son las opciones válidas. Nuestra respuesta debe ser una opción válida o bien un comando. En caso de que las respuestas del usuario no lleven a ninguna conclusión, el Sistema Experto nos lo indicará.

II.2.- Representación del conocimiento

Como ya se mencionó anteriormente, el conocimiento en Microexpert se representa por medio de Reglas.

Para construir las reglas y determinar adecuadamente los atributos que definen un objeto en particular, frecuentemente se utiliza el árbol de conocimientos. Un árbol de conocimientos es en forma estricta, una red semántica, vista en el Capítulo I. Un árbol es una gráfica acíclica, es decir que no debe de contener ciclos, compuesta por nodos y ramas, como se ve en la figura 7.

Los círculos se llaman nodos y las líneas se denominan ramas. El nodo que se encuentra en el extremo izquierdo se llama "raíz", el primero de izquierda a derecha (nodo inicial); a partir de él se desprenden todos los demás nodos. Los nodos que se encuentran en el extremo derecho (nodos finales), sin importar su distancia a la raíz, se denominan "hojas".

Asimismo, se denomina Nivel 1 del árbol, al nivel que contiene al nodo inicial; a partir de ahí se empieza por numerar sucesivamente los niveles (desde el 2 hasta el 'n') para nombrar los niveles subsiguientes en cuanto a nodos intermedios y finales se refiere.

La teoría de los Sistemas Expertos utiliza árboles para representar el conocimiento, a estos árboles se les denomina árboles de conocimientos. En éstos, cada nodo representa un atributo, cada rama representa el valor que toma el atributo o nodo anterior y cada hoja representa el objeto que se está definiendo. Los nodos deben agruparse dentro de cada nivel, dependiendo de las características inherentes que tengan con respecto al atributo que fué definido para cada nivel.

De esta manera, cuando ya contamos con un pequeño árbol de conocimientos, un objeto es definido por la lista de atributos y valores de atributos por los que se pasa cuando se recorre el árbol desde la raíz hasta la hoja que contiene el objeto que está siendo definido.

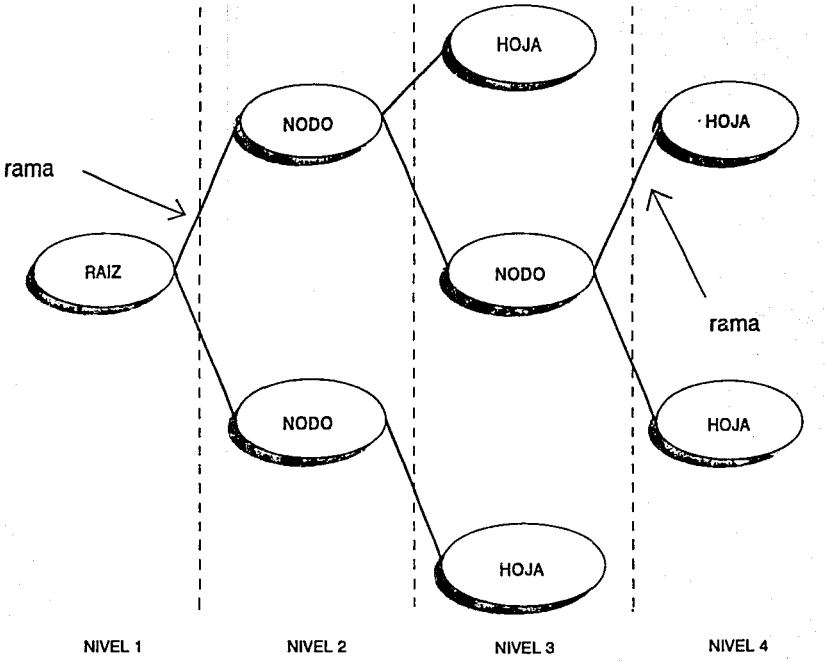


FIG. 7 ARBOL DE CONOCIMIENTOS

II.3 .- Operación del Mecanismo de Inferencia Microexpert

Inicialmente Microexpert realiza un análisis de la base de conocimientos para verificar que esta no presente inconsistencias en cuanto a la sintaxis que se definió, en caso de detectar algún error se detiene el análisis y se emite un mensaje de error, de no existir errores se inicia con el proceso de inferencia; así mismo durante esta primera etapa se analizan los valores válidos para cada atributo. Cabe aclarar que aun cuándo existan errores de sintaxis en la base, Microexpert tratará de llegar a una conclusión.

El siguiente paso es un barrido de la base de hechos (atributos) de arriba hacia abajo (Top-Down), toma el primer atributo y despliega la pregunta correspondiente, dependiendo de la respuesta del usuario, la cual solo puede ser uno de los valores válidos, se realiza una poda de la base de conocimientos, es decir sólo se seguirán infiriendo aquellas reglas en las cuales el valor del atributo leído sea igual al consecuente del mismo atributo definido en la regla.

A continuación se toma el siguiente atributo y se pregunta nuevamente al usuario por el valor de dicho atributo, con lo cual se realiza otra poda de la base de conocimientos, este proceso continúa hasta que solo una regla es capaz de igualar todos los valores de atributos con lo cual el mecanismo de inferencia es capaz de emitir una conclusión.

Al emitir la conclusión, ésta es acompañada de todos los hechos que fueron necesarios para validar la misma.

Debido a lo anterior podemos concluir que Microexpert es capaz de aplicar *metaconocimiento* ya que utiliza éste para saber cual es el conocimiento que se puede aplicar y es *determinístico* ya que sólo es capaz de emitir una sola conclusión a la vez.

II.4 .- Ejemplo de aplicación y codificación

Para ejemplificar el diseño de una Base de Conocimientos, siguiendo la sintaxis de Microexpert, vamos a suponer que deseamos un Sistema Experto que por la descripción de sus características físicas identifique el nombre de un animal (objetivo).

Los objetos que vamos a definir son: perro, león, leopardo, puma, gato, hombre, ballena, mantarraya, mono, víbora, cocodrilo, águila, halcón, gallina, pavo, pato, canario y perico.

Los atributos a través de los cuales vamos a describir estos animales son: vivíparo, carnívoro, ladra, marino, es_grande, tiene_garras, bípedo, tiene_melena, tiene_manchas, es_ave, se_arrastra, vuela, vive_pantano, tiene_cresta, se_come, come_alpiste.

Los valores que puede tomar cada atributo son: si o no exclusivamente.

El árbol de conocimientos que describe este Sistema es el de la figura 8.

Para obtener el conjunto completo de reglas, a partir del árbol de conocimientos deben de seguirse los siguientes pasos:

1.- Se debe tomar como base la regla en Microexpert ya dada anteriormente.

2.- Cada uno de los atributos está representado por un nodo en el árbol, por lo cual es muy fácil reconocerlos.

3.- Los valores de los atributos se toman de las etiquetas de cada rama del árbol. Cabe aclarar que a cada nodo le corresponden dos ramas, por lo cual son dos los posibles valores que puede tomar.

4.- Por último los objetos están representados en el árbol por medio de las hojas ó nodos finales. Como se sabe éstos nodos finales son las conclusiones.

Por cada animal hay una regla que lo describe, y son las siguientes:

```
IF vivíparo IS si
AND carnívoro IS si
AND ladra IS si
THEN animal IS perro

IF vivíparo IS si
AND carnívoro IS si
AND ladra IS no
```

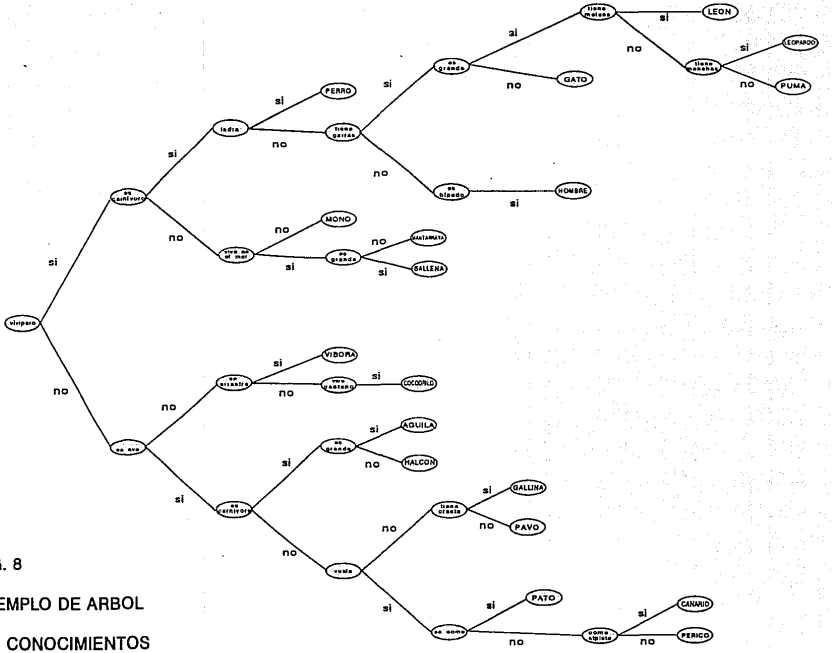


FIG. 8

EJEMPLO DE ARBOL

DE CONOCIMIENTOS

```
AND tiene_garras IS si
AND es_grande IS si
AND tiene_melena IS si
THEN animal IS león

IF vivíparo IS si
AND carnívoro IS si
AND ladra IS no
AND tiene_garras IS si
AND es_grande IS si
AND tiene_melena IS no
AND tiene_manchas IS si
THEN animal IS leopardo

IF vivíparo IS si
AND carnívoro IS si
AND ladra IS no
AND tiene_garras IS si
AND es_grande IS si
AND tiene_melena IS no
AND tiene_manchas IS no
THEN animal IS puma

IF vivíparo IS si
AND carnívoro IS si
AND ladra IS no
AND tiene_garras IS si
AND es_grande IS no
THEN animal IS gato

IF vivíparo IS si
AND carnívoro IS si
AND ladra IS no
AND tiene_garras IS no
AND bípedo IS si
THEN animal IS hombre

IF vivíparo IS si
AND carnívoro IS no
AND marino IS si
AND es_grande IS si
THEN animal IS ballena

IF vivíparo IS si
AND carnívoro IS no
AND marino IS si
AND es_grande IS no
THEN animal IS mantarraya

IF vivíparo IS si
AND carnívoro IS no
AND marino IS no
```

```
THEN animal IS mono

IF vivíparo IS no
AND es_ave IS no
AND se_arrastra IS si
THEN animal IS víbora

IF vivíparo IS no
AND es_ave IS no
AND se_arrastra IS no
AND vive_pantano IS si
THEN animal IS cocodrilo

IF vivíparo IS no
AND es_ave IS si
AND carnívoro IS si
AND es_grande IS si
THEN animal IS águila

IF vivíparo IS no
AND es_ave IS si
AND carnívoro IS si
AND es_grande IS no
THEN animal IS halcón

IF vivíparo IS no
AND es_ave IS si
AND carnívoro IS no
AND vuela IS no
AND tiene_cresta IS si
THEN animal IS gallina

IF vivíparo IS no
AND es_ave IS si
AND carnívoro IS no
AND vuela IS no
AND tiene_cresta IS no
THEN animal IS pavo

IF vivíparo IS no
AND es_ave IS si
AND carnívoro IS no
AND vuela IS si
AND se_come IS si
THEN animal IS pato

IF vivíparo IS no
AND es_ave IS si
AND carnívoro IS no
AND vuela IS si
AND se_come IS no
AND come_alpiste IS si
```

```
THEN animal IS canario

IF vivíparo IS no
AND es_ave IS si
AND carnívoro IS no
AND vuela IS si
AND se_come IS no
AND come_alpiste IS no
THEN animal IS perico
```

Hagamos ahora las preguntas que deseamos que el Sistema Experto nos haga al momento de hacer la consulta; cabe aclarar que para cada posible atributo es necesario hacer una pregunta:

```
1.- el animal es vivíparo ?
2.- el animal es carnívoro ?
3.- el animal ladra ?
4.- el animal vive en el mar ?
5.- el animal tiene garras ?
6.- el animal es grande de tamaño ?
7.- el animal tiene dos patas (bípedo) ?
8.- el animal tiene melena ?
9.- el animal tiene manchado el cuerpo ?
10.- el animal es una ave ?
11.- el animal se arrastra ?
12.- el animal gusta de vivir en los pantanos ?
13.- el ave vuela ?
14.- el ave tiene cresta ?
15.- es comestible el ave ?
16.- el ave se alimenta de alpiste ?
```

Una vez hecho todo lo anterior, es necesario codificar la base de conocimientos de acuerdo a la sintaxis establecida por Microexpert. A continuación se muestra el programa completo sobre animales codificado para el shell:

```
PROMPT vivíparo
el animal es vivíparo ?
.
PROMPT carnívoro
el animal es carnívoro ?
.
PROMPT ladra
el animal ladra ?
.
PROMPT marino
el animal vive en el mar ?
.
PROMPT tiene_garras
el animal tiene garras ?
.
PROMPT es_grande
```

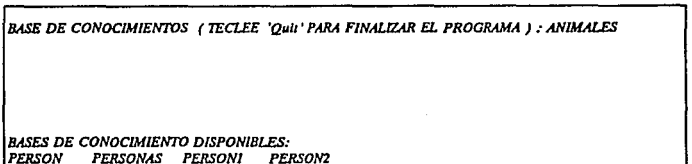
```
el animal es grande de tamaño ?
.
PROMPT bípedo
el animal tiene dos patas (bípedo) ?
.
PROMPT tiene_melena
el animal tiene melena ?
.
PROMPT tiene_manchas
el animal tiene manchado el cuerpo ?
.
PROMPT es_ave
el animal es una ave ?
.
PROMPT se_arrastra
el animal se arrastra ?
.
PROMPT vive_pantano
el animal gusta de vivir en los pantanos ?
.
PROMPT vuela
el ave vuela ?
.
PROMPT tiene_cresta
el ave tiene cresta ?
.
PROMPT se_come
es comestible el ave ?
.
PROMPT come_alpiste
el ave se alimenta de alpiste ?
.
1
IF vivíparo IS si
AND carnívoro IS si
AND ladra IS si
THEN animal IS perro
.
2
IF vivíparo IS si
AND carnívoro IS si
AND ladra IS no
AND tiene_garras IS si
AND es_grande IS si
AND tiene_melena IS si
THEN animal IS león
.
3
IF vivíparo IS si
AND carnívoro IS si
AND ladra IS no
AND tiene_garras IS si
```

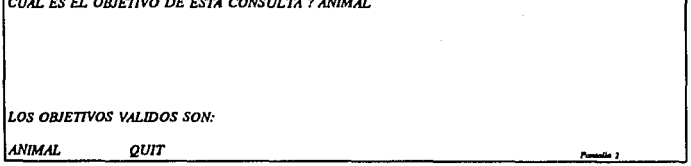
```
AND es_grande IS si
AND tiene_melena IS no
AND tiene_manchas IS si
THEN animal IS leopardo
.
4
IF vivíparo IS si
AND carnívoro IS si
AND ladra IS no
AND tiene_garras IS si
AND es_grande IS si
AND tiene_melena IS no
AND tiene_manchas IS no
THEN animal IS puma
.
5
IF vivíparo IS si
AND carnívoro IS si
AND ladra IS no
AND tiene_garras IS si
AND es_grande IS no
THEN animal IS gato
.
6
IF vivíparo IS si
AND carnívoro IS si
AND ladra IS no
AND tiene_garras IS no
AND bípedo IS si
THEN animal IS hombre
.
7
IF vivíparo IS si
AND carnívoro IS no
AND marino IS si
AND es_grande IS si
THEN animal IS ballena
.
8
IF vivíparo IS si
AND carnívoro IS no
AND marino IS si
AND es_grande IS no
THEN animal IS mantarraya
.
9
IF vivíparo IS si
AND carnívoro IS no
AND marino IS no
THEN animal IS mono
.
```

```
10
IF vivíparo IS no
AND es_ave IS no
AND se_arrastra IS si
THEN animal IS víbora
.
11
IF vivíparo IS no
AND es_ave IS no
AND se_arrastra IS no
AND vive_pantano IS si
THEN animal IS cocodrilo
.
12
IF vivíparo IS no
AND es_ave IS si
AND carnívoro IS si
AND es_grande IS si
THEN animal IS águila
.
13
IF vivíparo IS no
AND es_ave IS si
AND carnívoro IS si
AND es_grande IS no
THEN animal IS halcón
.
14
IF vivíparo IS no
AND es_ave IS si
AND carnívoro IS no
AND vuela IS no
AND tiene_cresta IS si
THEN animal IS gallina
.
15
IF vivíparo IS no
AND es_ave IS si
AND carnívoro IS no
AND vuela IS no
AND tiene_cresta IS no
THEN animal IS pavo
.
16
IF vivíparo IS no
AND es_ave IS si
AND carnívoro IS no
AND vuela IS si
AND se_come IS si
THEN animal IS pato
.
```

```
17
IF vivíparo IS no
AND es_ave IS si
AND carnívoro IS no
AND vuela IS si
AND se_come IS no
AND come_alpiste IS si
THEN animal IS canario
.
18
IF vivíparo IS no
AND es_ave IS si
AND carnívoro IS no
AND vuela IS si
AND se_come IS no
AND come_alpiste IS no
THEN animal IS perico
```

Se muestra una corrida del ejemplo desarrollado usando el paquete Microexpert haciendo inferencias sobre la Base de Conocimiento anteriormente descrita.





el animal es vivíparo ?

SI

RESPUESTAS VALIDAS PUEDEN SER:

NO	*SI*
WHY	*WHAT*
RULE	*HOW*
WHATIF	*QUIT*

Pantalla 3

el animal es carnívoro ?

NO

RESPUESTAS VALIDAS PUEDEN SER:

NO	*SI*
WHY	*WHAT*
RULE	*HOW*
WHATIF	*QUIT*

Pantalla 4

el animal vive en el mar ?

SI

RESPUESTAS VALIDAS PUEDEN SER:

NO	*SI*
WHY	*WHAT*
RULE	*HOW*
WHATIF	*QUIT*

Pantalla 5

el animal es grande de tamaño ?

NO

RESPUESTAS VALIDAS PUEDEN SER:

NO	*SI*
WHY	*WHAT*
RULE	*HOW*
WHATIF	*QUIT*

Pantalla 6

Conclusion :
ANIMAL *ES MANTARRAYA*

PRESIONE <BARRA ESPACIADORA> PARA CONTINUAR.

Pantalla 7

LOS SIGUIENTES HECHOS SE HAN DETERMINADO :

1) ANIMAL	*ES MANTARRAYA*
2) ES_GRANDE	*ES NO*
3) MARINO	*ES SI*
4) CARNIVORO	*ES NO*
5) VIVIPARO	*ES SI*

PRESIONE <ENTER> PARA CONTINUAR

Pantalla 8

CAPITULO III

DISEÑO DE LA INTERFAZ

El desarrollo de aplicaciones de software demanda tener diversas habilidades y conocer diferentes metodologías para entender inicialmente el problema y encontrar una posible solución.

La implementación para la solución de un problema específico, puede ser realizada aplicando un conjunto de técnicas que son en esencia independientes. Estas técnicas forman la base de la metodología de la Ingeniería de Software.

Los objetivos de la Ingeniería de Software son :

1) Una metodología bien definida que permita identificar la planeación, el desarrollo y el mantenimiento del sistema.

2) Un conjunto establecido de componentes del software que documentan cada etapa en el desarrollo del sistema y muestran la ruta a seguir entre cada una de ellas.

3) Un conjunto de segmentos predecibles que pueden ser revisados a intervalos regulares a lo largo de las etapas de desarrollo del sistema.

El analista de sistemas es el principal encargado de llevar a cabo estas tareas y para lo cual se auxilia de modelos. Un modelo es una forma de abstracción de la realidad y ayudan al analista en el diseño y documentación de sistemas de software. Algunos de los modelos más ampliamente usados se describen a continuación.

Los modelos lógicos matemáticos son representaciones cuantitativas de la realidad, se aplican en sistemas para apoyar las funciones de procesamiento de transacciones. Una tabla de decisiones es la representación tabular del proceso lógico de "si esto ocurre, entonces haz aquello". Un árbol de decisiones es un modelo de una secuencia de condiciones en donde cada decisión que se toma depende del valor actual de la variable que se está probando y de todas las decisiones tomadas anteriormente. Otro modelo para describir procedimientos es el demoninado pseudocódigo que se emplea con frecuencia para especificar modelos de procedimientos en una forma cercana a un lenguaje de programación real. Los diagramas de flujo de datos son modelos que documentan el flujo de información entre procesos del sistema, estos han sido la forma tradicional de ilustrar las entradas y salidas de un programa. La técnica de diagramas de cajas se emplea principalmente para describir la lógica detallada de los programas y es una alternativa de los diagramas de flujo tradicionales. Una técnica de diagramación que prepara la entrada, la salida y las funciones de un sistema es la técnica HIPO (jerarquía mas entrada-proceso-salida), es en realidad

la integración de un diagrama jerárquico y un conjunto de diagramas funcionales que muestran la entrada, los procesos y la salida para cada uno de ellos. Un diagrama de estructura es una alternativa viable para HIPO, esta se desarrolla en una forma descendente (top-down), presentando primeramente una vista general del sistema y exponiendo después una cantidad de detalle sucesivamente mayor a medida que se abren porciones del modelo.

El diseño del sistema, motivo del presente trabajo, utiliza tres de las herramientas mencionadas anteriormente para una mejor aproximación de objetivos y compresión del mismo. Como primera parte se desarrolla un diagrama de flujo de datos para analizar las transformaciones de la información. Después se procede a realizar una carta de estructura donde se modulariza el proceso y se muestra la jerarquía y comunicación entre dichos módulos. Por último se desarrolla el seudocódigo general del sistema, que tiene como característica principal mostrar las microespecificaciones de cada módulo.

III.1 .- ANALISIS ESTRUCTURADO

El análisis estructurado tiene como propósito fundamental especificar, en la forma más precisa posible, los requerimientos del usuario para un programa o sistema.

El diagrama de flujo de datos, es la principal herramienta gráfica del análisis y tiene como objetivo mostrar las transformaciones de los datos a medida que éstos fluyen a través de los procesos del programa; es decir, ayuda a analizar los cambios que ocurren a los datos de entrada a fin de lograr los datos de salida.

El Diagrama de Flujo de Datos

Es un instrumento de modelación que permite mostrar a un sistema como una red de subsistemas conectados unos a otros mediante flujos de datos que muestran las relaciones entre ellos.

Un diagrama de flujo de datos consta de los siguientes elementos:

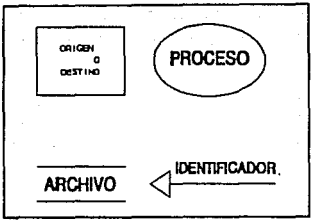
1.- Un óvalo, con un nombre inscrito, para indicar un proceso, el cual indica la función del mismo y es el que actúa sobre los datos para transformarlos o generar nueva información.

2.- Una flecha, con un nombre asociado, para indicar la entrada y salida de datos a un proceso. La dirección de la flecha indica la dirección del flujo de datos.

3.- Dos líneas paralelas, con un nombre entre ellas para indicar un contenedor de datos, es decir, un archivo en disco o cinta. Un archivo con una flecha saliente indica que se extrae información de él; con una flecha entrante se indica escritura de información.

4.- Un rectángulo que indica dónde se origina o a donde se destina la información (sentido de la flecha). Un mismo rectángulo puede ser fuente o destino. El rectángulo tiene un nombre que lo identifica como fuente y/o destino.

Todos los elementos anteriormente descritos se muestran a continuación :



Obsérvese que del diagrama de flujo de datos se obtiene una estructura del tipo entrada-proceso-salida; es decir, datos de entrada a un proceso que actúa sobre los mismos y datos de salida transformados.

Las características del diagrama de flujo de datos son las siguientes:

a) *Es gráfico,* es decir, de un vistazo permite percibir rápidamente las funciones del sistema.

b) *Es modular,* es decir, muestra la partición de un sistema en funciones tan independientes entre sí como sea posible.

c) *Enfatiza el flujo de datos,* muestra solamente el flujo de datos que se transforman a medida que pasan a los procesos (funciones), desde la entrada a la salida.

d) *Desenfatiza el flujo de control,* es decir, no muestra información de control, ni secuencia de acciones en el tiempo.

e) *Es modificable,* es decir, se pueden reconsiderar algunas partes del DFD con las cuales no se haya quedado satisfecho y volver a trabajarlas.

f) *No es redundante,* una función se registra sólo una vez para que el sistema, al cual dará origen el DFD, sea consistente y de fácil actualización.

Cabe señalar que un ejemplo completo de Diagrama de Flujo de Datos se desarrollará en un apartado posterior, que será el correspondiente al utilizado para la implementación de la interfaz, motivo del presente trabajo.

III.2 .- DISEÑO ESTRUCTURADO

La herramienta principal del Diseño Estructurado es la **Carta de Estructura**, la cual muestra la partición del sistema en módulos y la relación jerárquica entre éstos. Además muestra los flujos de datos y control entre los módulos.

Elementos de una Carta de Estructura.

Una carta de estructura cuenta con los siguientes elementos:

1.- Un rectángulo con un nombre inscrito para indicar un módulo. El nombre indica la función del mismo.

2.- Líneas que indican la liga entre módulos (llamadas a módulos).

3.- Flechas que indican el flujo de datos y de control (comunicación entre módulos).

4.- El nombre del módulo debe resumir los nombres de sus subordinados inmediatos o resumir su función y las funciones de sus subordinados inmediatos.

Atributos básicos de los Módulos

Son cuatro atributos básicos:

1.- Entrada: Los datos que le pasa quien lo invoca.
Salida: Los datos que regresa a quien lo invoca.

2.- Función: Lo que hace a sus datos de entrada para producir sus datos de salida.

3.- Mecánica: Cómo realiza su función, es decir su lógica.

4.- Datos: Su propio espacio de trabajo, es decir, las variables locales.

Características de la Carta de Estructura

Una carta de estructura muestra:

A. La partición del programa, es decir, los módulos de que consta.

B. La estructura jerárquica, es decir, la relación entre módulos.

C. Los nombres de módulos y por consiguiente su función.

D. El grado de acoplamiento entre módulos.

F. Flujo de datos entre módulos.

G. Las decisiones e iteraciones que involucran la llamada a un módulo.

Una carta de estructura no muestra:

A. El número de veces que se llama a un módulo.

B. La secuencia en que se llaman los módulos.

C. Cómo realiza su función un determinado módulo.

D. Datos internos del módulo.

Acoplamiento y Cohesión

Acoplamiento es la medida del grado de la interdependencia entre dos módulos. Existen tres formas de acoplamiento, a continuación se describen cada una de ellas, partiendo de la mejor a la menos recomendada :

Acoplamiento por Datos.- dos módulos están acoplados por datos si se comunican por datos que no sean banderas, ni arreglos, ni registros.

Acoplamiento por Estampilla.- dos módulos están acoplados por estampilla si se comunican mediante registros o arreglos.

Acoplamiento por Control.- dos módulos están acoplados por control si se comunican al menos por una bandera.

Cohesión es la medida del grado de asociación de los elementos dentro de un módulo. Un elemento puede ser: a) una instrucción; b) un grupo de instrucciones; c) una llamada a otro módulo. Se describen tres tipos de cohesión partiendo del mejor al más pobre:

Cohesión Funcional.- un módulo con cohesión funcional es aquel en que todos los elementos contribuyen a una y sólo una tarea.

Cohesión Secuencial.- un módulo con cohesión secuencial es aquél en que los datos de salida de un elemento sirven como datos de entrada a otro elemento.

Cohesión Comunicacional.- un módulo con este tipo de cohesión es aquél cuyos elementos contribuyen a tareas diferentes, pero cada tarea tiene los mismos parámetros de entrada y salida.

III.3 .- CARACTERISTICAS DE LA INTERFAZ

Definición del problema

La necesidad del usuario es la de contar con una herramienta de software que sea capaz de proporcionarle un ambiente de trabajo sencillo de usar para la creación de Sistemas Expertos.

Se parte del hecho de contar con un Mecanismo de Inferencia, que es capaz de realizar inferencias lógicas sobre una base de conocimientos que es alimentada por el propio usuario en forma de reglas de conocimiento.

El problema inicial estriba en crear dicha base, sin tener conocimientos profundos de como codificar reglas de inferencia de acuerdo a la sintaxis predefinida de un determinado shell.

En base a los requerimientos iniciales descritos anteriormente, seleccionamos el mecanismo de inferencia Microexpert, a partir del cual se desarrollará una interfaz gráfica textual que cumpla con los mismos, mediante la interacción continua con el usuario por medio de una serie de preguntas sencillas de contestar; cabe mencionar que aún cuando la interfaz será capaz de crear la base de conocimientos completamente con la información dada, es necesario que el usuario conozca la forma de poder representar el conocimiento en forma de árbol (Ver Capitulo II).

Ambiente de la Interfaz

Como se ve, la idea principal de la tesis presente, es la implementación de un software para la construcción de Sistemas Expertos, y es por ello que consideramos hacer un programa que cuente con las siguientes características:

- **Presentación accesible de utilerías.-** Esto se logra con la creación de menúes pop-up, es decir menúes que presentan otro submenú al ser seleccionados.

- **Facilidad de selección de utilerías.-** Esta característica se deriva de la anterior, por lo cual consideramos que la forma más sencilla de manejo de opciones es por medio de la utilización de un mouse, y todos los menúes principales serán accesibles de esta forma, y sólo será necesario el teclado a la hora de capturar información a la base de conocimientos. Cabe mencionar que si el equipo del usuario no cuenta con un mouse, todo el programa debe ser accesible por medio del teclado.

- **Facilidad de comprensión.-** Con esto queremos decir que el sistema está pensado para gente con poco ó ningún

conocimiento en equipos PC, y por ello todas las utilerías presentadas tienen nombres concretos, es decir el nombre indica que tarea realiza.

- **Facilidad de cancelación de opciones.-** Esto es necesario en todo momento ya que el usuario puede equivocarse y cometer errores, y es por ello que el sistema es cancelable en todas sus opciones.

- **Ayuda.-** Parte esencial del programa, ya que con esta característica se le dota de independencia, al hacerlo completamente entendible y manejable sin la necesidad de herramientas auxiliares de software o hardware.

III.4 DIAGRAMA DE FLUJO DE DATOS DE LA INTERFAZ

Para cumplir con la meta definida en la introducción partimos de las siguientes condiciones iniciales :

A partir de lo anterior se considera que son necesarias las siguientes herramientas básicas :

Creación de la base de conocimientos .- Esta parte es la encargada de crear la base de conocimientos. Entrega como salida el archivo de conocimientos *.Kb

Consulta del programa de la base de conocimientos .- Mediante este proceso el usuario va ser capaz de consultar el programa creado por el sistema y que sirve como alimentación al Mecanismo de Inferencia.

Presentación gráfica del árbol de conocimientos .- En esta parte el usuario tendrá la facilidad de consultar el árbol de conocimientos creado por el sistema contra el árbol de conocimientos creado por él mismo y de esta forma compararlos y verificar que los datos hayan sido interpretados correctamente.

Llamada al mecanismo de Inferencia .- Ejecutar a Microexpert, que es el encargado de realizar las inferencias lógicas.

Creación de reportes .- Facilidad para crear impresiones en papel tanto del programa, como del gráfico. Además será posible mandar a imprimir las tablas de conexiones generadas en base al interrogatorio inicial y que contienen los nombres y los atributos de los nodos del árbol.

Además consideramos necesaria la creación de herramientas adicionales como son :

Ayuda general .- Texto presentado en pantalla que sirve como una guía general para el manejo de la interfaz y que lleva paso a paso al usuario a través de la misma.

Visualización de directorio de trabajo .- Herramienta necesaria para la consulta de los archivos presentes en la ruta y unidad de trabajo actual.

Cambio de unidad o subdirectorio .- Utilería necesaria para la correcta administración de archivos creados por el sistema y que es capaz de cambiar la unidad de trabajo a cualquiera accesible por el sistema y dentro de esta unidad cambiar la ruta de trabajo a subdirectorios existentes.

Eliminación de archivos .- Facilidad para depurar directorios de trabajo al permitir borrar archivos ya no utilizables que fueron creados por el sistema.

Todas las herramientas anteriormente mencionadas deben de quedar englobadas dentro de un mismo ambiente de trabajo para hacer estas más accesibles al usuario. Es por ello que se crea una herramienta adicional que será la encargada de enlazar cada una de las utilerías anteriormente propuestas. Esta parte es la encargada de interactuar directamonte con el usuario al permitirle seleccionar la opción deseada.

La interacción de todos los módulos descritos y el flujo de datos que va a existir entre ellos pueden verse con mayor claridad en la figuras 9, 10 y 11.

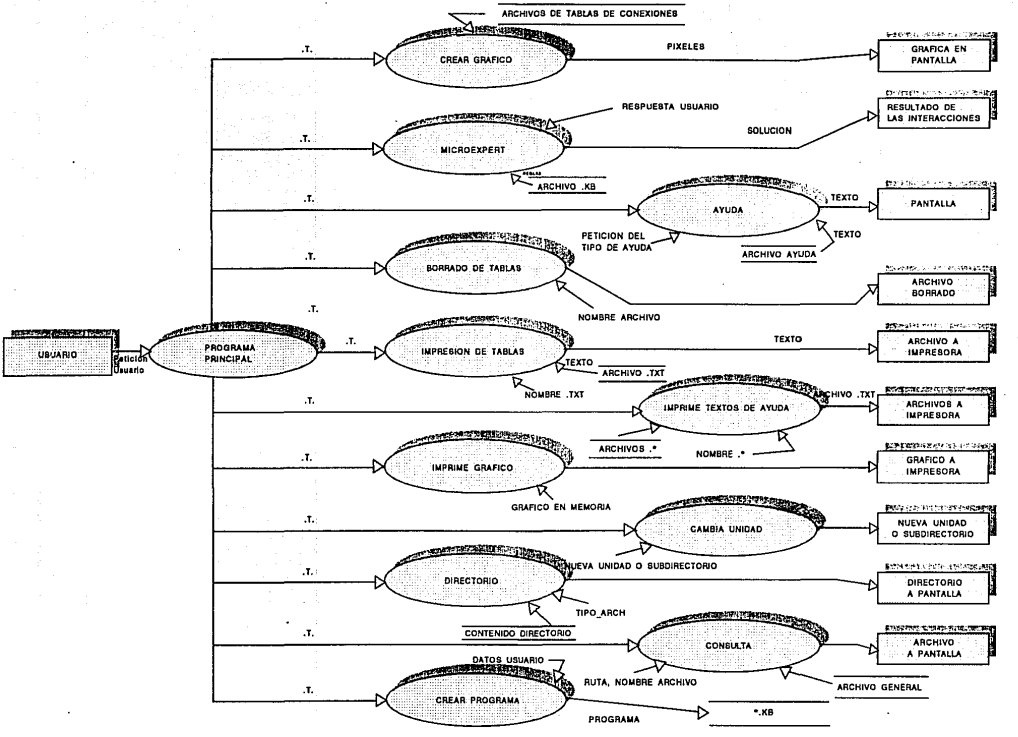


FIGURA 9. DIAGRMA DE FLUJO DE LA INTERFAZ

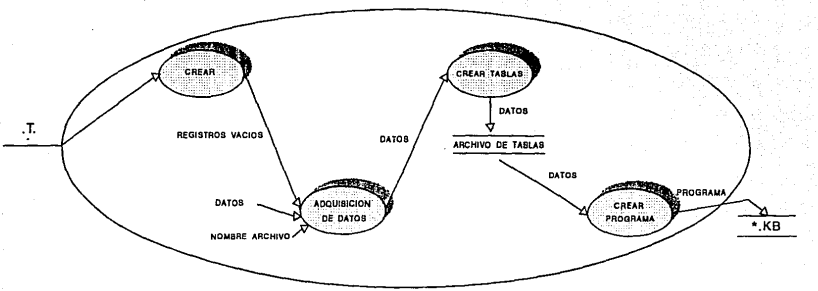


FIGURA 10. DIAGRAMA DE FLUJO DEL MODULO CREAR TABLAS

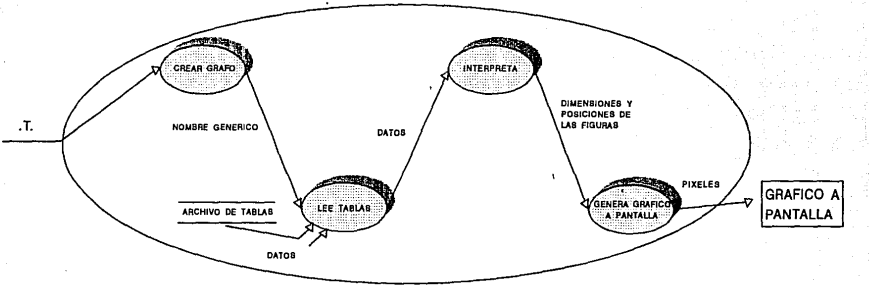


FIGURA 11. DIAGRAMA DE FLUJO DEL MODULO CREAR GRAFICO

III.5 CARTA DE ESTRUCTURA DE LA INTERFAZ

Partiendo del flujo de datos anteriormente descrito se procede a la creación de la Carta de Estructura de la Interfaz, misma que contendrá la partición del sistema en módulos y la jerarquía que existe entre ellos.

Para una mejor estructuración de los módulos consideramos necesario agruparlos por características inherentes que puedan tener, es por ello que los clasificamos como sigue :

◊ Crear programa , Consulta, Directorio y Cambia unidad corresponden a una misma tarea que la denominaremos *Introduce árbol*.

◊ Crear gráfico e Imprime gráfico corresponden a la tarea *Grafica árbol*.

◊ *Ejecuta Árbol* es un módulo individual.

◊ *Borrado de archivos* engloba el procedimiento de borrado de archivo individual.

◊ *Imprimir* módulo que engloba los procedimientos correspondientes a la impresión de un archivo en específico o una impresión de los textos de ayuda.

◊ *Ayuda*, módulo encargado de guiar al usuario, presentándole auxilio sensible al contexto.

En base a esta clasificación se procede a la creación del módulo encargado de presentar las herramientas al usuario. Esta primer carta puede consultarse en la figura 12.

Como puede observarse aquí, únicamente existe flujo de control entre el programa principal y los submódulos de las tareas arriba descritas.

El programa principal es el encargado de realizar la presentación del sistema, así como del menú principal y recibir la selección del usuario.

Siguiendo el orden propuesto en la carta anterior, el primer módulo a desglosar es **Introduce árbol**, este módulo como se dijo anteriormente tiene por objeto crear el programa, consultar archivos, ver directorio y cambiar unidad, tareas que son separadas en módulos individuales tal y como puede verse en la carta correspondiente (figura 13).

El submódulo Crear se encarga de inicializar y enviar las variables *nom_arch* (nombre de archivo) y *var_reg* (variables de

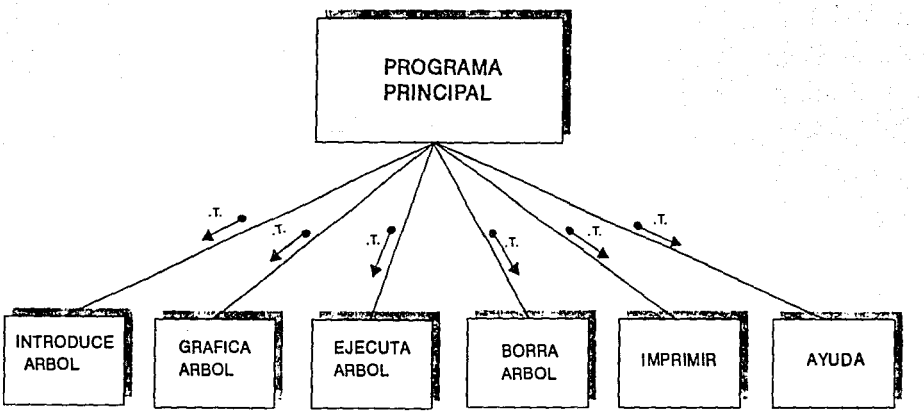


FIG. 12 CARTA DE ESTRUCTURA
INTERFAZ

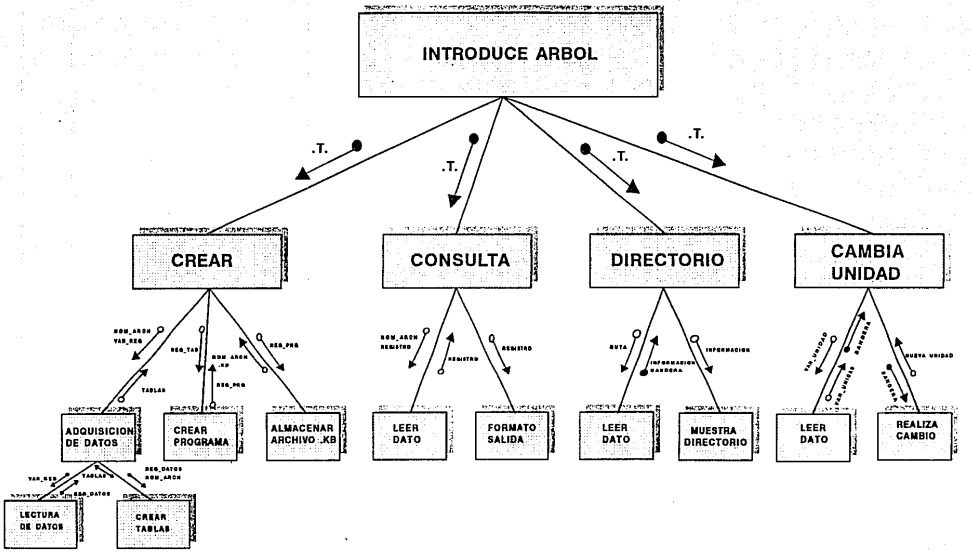


FIG. 13 CARTA DE ESTRUCTURA DEL MODULO INTRODUCE ARBOL

datos) al módulo Adquisición de Datos, este se comunica con el bloque de Lectura de Datos encargado de preguntar y capturar los datos proporcionados por el usuario en las variables mencionadas; al terminar su tarea regresa estas variables a Adquisición de Datos, que se encarga de turnarlas al bloque de Crear Tablas que como su nombre lo dice, crea los archivos de tablas cuyo contenido son las conexiones entre nodos; estas tablas se regresan a Adquisición de Datos, que a su vez las regresa al módulo Crear.

El módulo Crear envía estas variables recibidas al bloque auxiliar Crear Programa que crea la base de conocimientos basada en reglas de inferencia de acuerdo a la sintaxis de Microexpert, misma que es regresada en *reg_prg* (programa) y *nom_arch* a Crear.

Como último paso Crear envía esta información al módulo auxiliar Almacenar Archivo, que es el encargado de guardar físicamente en disco el programa .Kb.

El submódulo Consulta reinicializa las variables *nom_arch* (Nombre de archivo) y *registro* (Variable de datos), verificando la existencia física del archivo y en caso de existir, carga los mismos en la variable *nom_arch* y la envía al submódulo Leer Dato quien lee la información y la almacena en *registro* que es regresada a Consulta.

Paso seguido Consulta redirecciona la variable *registro* al bloque Formato Salida el cual se utiliza para presentar el archivo a pantalla con un formato entendible para el usuario.

El siguiente módulo ha diseñar es el de Directorio; utiliza una sola variable: *ruta*, la cual captura del sistema el directorio actual de trabajo, mismo que ha de ser presentado por medio del submódulo Leer Dato, éste regresa una bandera de ruta correcta, junto con la información en un archivo tipo texto a Directorio, el cual redirecciona el flujo de datos a Muestra Directorio, submódulo encargado de direccionar el mencionado archivo texto a pantalla y de esta forma poder ser consultado por el usuario.

El último módulo de esta subcarta de estructura es el módulo Cambia Unidad, que al igual que el anterior sólo requiere el identificador lógico de la nueva unidad o la ruta completa de un nuevo subdirectorio, dicha información se toma del usuario y se almacena en la variable *var_unidad* enviada al submódulo Leer Dato, que regresa ésta y una bandera que indica que la ruta fue correcta a Cambia Unidad, este redirecciona dicha bandera a Realiza Cambio, submódulo encargado de realizar el cambio, el cual retorna a Cambia Unidad la nueva unidad solicitada.

Continuando con el orden propuesto en la Carta de Estructura principal (figura 12), el módulo ha diseñar es el de **Grafica Arbol**, cuya carta se muestra en la figura 16.

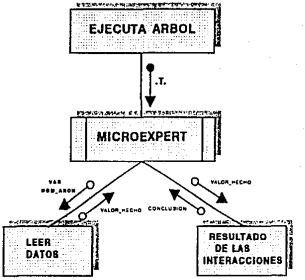


FIG 14. CARTA DE ESTRUCTURA
MODULO EJECUTA ARBOL

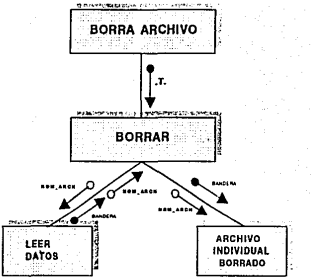


FIG 15. CARTA DE ESTRUCTURA
MODULO BORRA ARBOL

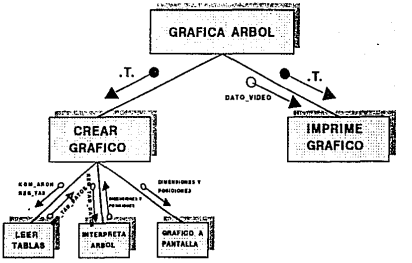


FIG 16. CARTA DE ESTRUCTURA
MODULO GRAFICA ARBOL

Este módulo tiene dos tareas encomendadas, Crear el gráfico y/o mandarlo a impresión. Es por ello que tenemos dos submódulos inherentes a él, descritos a continuación.

El módulo encargado de la creación del gráfico es Crea Gráfico, el cual lee la información de los archivos generados en el modulo Crear Tablas; para leer dicha información, utiliza la variables *nom_arch* y *reg_tab*, las cuales son enviadas al módulo Leer Tablas, el cual regresa la variable *reg_tab_datos* a Crear Gráfico, este la redirecciona al módulo Interpreta árbol, de acuerdo a esta información, se calculan las dimensiones y posiciones de las figuras a utilizar para la creación del gráfico y se regresan a Crear Gráfico el cual las redirecciona a Gráfico a Pantalla que genera finalmente éste mostrándolo en pantalla.

El siguiente bloque ha diseñar es **Ejecuta Arbol**, módulo medular de la interfaz; este es un módulo predefinido ya ejecutable, por lo cual se ejemplifica su funcionamiento en la figura 14.

El funcionamiento es el siguiente: el módulo principal envía *nom_arch* (nombre de archivo de base de conocimientos) al bloque Leer Datos que interpreta el archivo y verifica que esté de acuerdo a la sintaxis; si es correcta, cada regla es enviada al módulo Resultado de las Interacciones el cual infiere lógicamente sobre cada regla en base a preguntas hechas y que son respondidas por el usuario hasta emitir una conclusión que se envía al módulo principal para que sea presentada en pantalla.

Continuando con el orden establecido el siguiente bloque es **Borra Archivo,** módulo encargado de una sola tarea (figura 15); éste módulo utiliza un bloque funcional subdividido en dos partes Leer Datos y Archivo Individual Borrado; el primero se encarga de leer el identificador del archivo dado por el usuario (var *nom_arch*) y verificar su existencia, en caso de existir envía una bandera al módulo principal junto con la variable *nom_arch* para que cargue el módulo Archivo Individual Borrado, proceso que se encarga de realizar la tarea.

Volviendo a la Carta Principal, el siguiente bloque funcional es **Imprimir**, ver figura 17; se encuentra dividido en dos partes: Impresión de Tablas e Impresión de archivos de Ayuda. El primero se comunica con Imprimir por medio de dos variables: *nom_arch* (nombre de archivo) y *registro* (datos del archivo); la función de éste módulo es la de verificar la existencia del archivo cuyo nombre fue dado en el bloque antecesor; en caso de ser cierta la existencia, el bloque almacena todos los datos del archivo en la variable *registro* y la regresa a Impresión de Tablas. Este la envía ha Imprimir Archivo .txt, cuya única función es la de verificar que la impresora esté encendida, en caso de no estarlo envía la bandera *(bandera lpt1)* de rechazo del dispositivo, si la impresora está en línea sólo envía la información a la misma. El segundo bloque tiene exactamente el mismo modo de operación que el anterior pero sólo

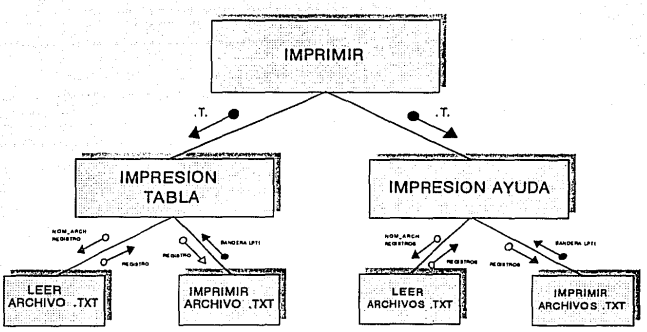


FIGURA 17. CARTA DE ESTRUCTURA DEL MODULO IMPRIMIR

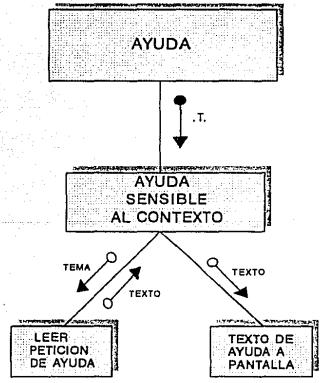


FIGURA 18. CARTA DE ESTRUCTURA DEL MODULO AYUDA

que ahora se envían a impresora todos los archivos involucrados en la ayuda del sistema y se da la posibilidad al usuario de imprimir sólo un archivo específico; esto se logra siguiendo el proceso descrito para el bloque anterior por medio de un ciclo repetitivo.

El último módulo dependiente del módulo principal a describir es el módulo de **Ayuda** (ver figura 18), el cual como ya se dijo vuelve al programa independiente de cualquier otra herramienta de software, ya que al utilizarlo el usuario no necesita de nada más.

El funcionamiento de este módulo es el siguiente: se crea primero un archivo tipo texto, el cual contiene toda la información concerniente a la interfaz. Después el módulo de Ayuda llama a un submódulo que se encarga de presentar al Usuario Ayuda Sensible al Contexto, esto quiere decir que el usuario será capaz de pedir ayuda de acuerdo al proceso que esté utilizando; este módulo tendrá dos partes funcionales: una, requiere de un bloque encargado de obtener la pregunta del usuario (Leer Petición de Ayuda) y segunda, la presentación del texto referente a la misma.

De forma general, esta es la descripción propuesta inicial para la creación de la Interfaz Gráfica-Textual para el Mecanismo de Inferencia de Microexpert, en su parte operativa.

III.6 SEUDOCODIGO DE PROGRAMACION

PROGRAMA PRINCIPAL

```
/* Declaración de Bibliotecas */
/* Declaración de Constantes */
/* Declaración de Variables Globales*/
/* Prototipo de Funciones */

INICIO
     Presenta pantalla de inicio
     Activa menú de acceso a la interfaz
     REPITE
          SI selección = Introduce Arbol
               Llama menú correspondiente
               SI selección = Crear
                    Llama módulo Crear Arbol o Tablas
               O SI selección = Consulta
                    Llama módulo Consulta de Archivos
               O SI selección = Cambiar
                    Llama módulo Cambiar Unidad
               O SI selección = Ver
                    Llama módulo Ver Directorio
               O SI selección = Ayuda
                    Llama módulo de Ayuda
               O SI selección = Cancelar
                    Cierra menú
               FIN
          O SI selección = Grafica Arbol
               Llama menú correspondiente
               SI selección = Grafica
                    Llama módulo Graficar Arbol de Conocimientos
               O SI selección = Ayuda
                    Llama módulo de Ayuda
               O SI selección = Cancelar
                    Cierra menú
               FIN
          O SI selección = Ejecuta
               Llama menú correspondiente
               SI selección = Ejecutar Arbol
                    Llama módulo Microexpert ( Inferencia )
               O SI selección = Ayuda
                    Llama módulo de Ayuda
               O SI selección = Cancelar
                    Cierra menú
               FIN
          O SI selección = Borra
               Llama menú correspondiente
               SI selección = Borra archivo
                    Llama módulo Borrar archivos
               O SI selección = Ayuda
```

```
                Llama módulo de Ayuda
            O SI selección = Cancelar
                Cierra menú
            FIN
        O SI selección = Imprime
            Llama menú correspondiente
            SI selección = Imprime archivo
                Llama módulo Imprime Archivos
            O SI selección = Imprime Ayuda
                Llama módulo Imprime Archivos de Ayuda
            O SI selección = Ayuda
                Llama módulo de Ayuda
            O SI selección = Cancelar
                Cierra menú
            FIN
        FIN
    HASTA selección = Salir
```

CREAR ARBOL O TABLAS

```
/* Declaración de Variables Locales */
/* Declaración de Palabras Reservadas */
/* Define máximo # de niveles = 20 */
/* Define máximo # de nodos = 100 */

INICIO
     Pantalla de presentación
     Pide el nombre donde se almacenarán los datos del árbol
     Pregunta el No. de Niveles que tendrá el árbol
     Pregunta el No. de Nodos que tendrá el árbol
     i <-- Contador
     DESDE ( i HASTA i <= No. de niveles ) REPETIR
          SI ( i < No. de Niveles ) LUEGO
               Pregunta nombre del atributo del nivel, su tipo y
                    prompt
          O BIEN
               Pregunta nombre del atributo del nivel
          FIN
     FIN
     Crea archivo de Niveles extensión .NIV
          DESDE ( i < No. de Niveles ) REPETIR
               Pregunta No. de nodos que pertenecen al nivel i
          FIN
     Crea archivo de nodos finales extensión .NSF
          DESDE ( i < No. de nodos finales ) REPETIR
               Pregunta las conclusiones del último nivel
          FIN
     Crea archivo de nodos no finales
     Crea archivo de conexiones
          EN TANTO ( No se conecten todos los nodos ) REPETIR
```

```
                Pregunta conexiones entre nodos ( origen, destino y
                    conexión )
                Checa que la conexión sea válida entre el nodo
                    origen y el nodo destino
                Checa que la conexión sea válida de acuerdo al tipo
                    del nivel al que pertenece el nodo origen
          FIN
     Crea archivo de rutas extensión .RUT
     Crea archivo de Base de Conocimientos extensión .KB
```

CONSULTA DE ARCHIVOS

```
/* Declaración de Variables Locales */

/* Programa Principal */
INICIO
     Limpia pantalla
     Llamada a función Pantalla
     Llamada a función Menú
FIN

/* Función Pantalla */
     /* mediante esta función se crea una pantalla de presentación
     para la lectura de datos */

/* Función Menú */
INICIO
     EN TANTO ( Verdadero ) REPETIR
          ch = caracter en buffer
          SI ( ch = 0 ) LUEGO
               cargo = 0
               EN TANTO ( no cargo ) REPETIR
                    Cierra todos los archivos
                    Llamada a función Limpia
                    Llamada a función Detecta
                    Llamada a función Tamaño
               FIN
          O SI ( ch = arriba ) LUEGO
               Pantalla Abajo
          O SI ( ch = abajo ) LUEGO
               Pantalla Arriba
          O SI ( ch = ESC ) LUEGO
               Termina Ejecución
          O BIEN
               ch = 0
          FIN
     FIN
FIN
```

```
/* Función Limpia */
     /* con esta función se limpia el arreglo que contendrá el
archivo */

/* Función Detecta */
INICIO
     Llamada a función Lee
     EN TANTO ( archivo no encontrado ) REPETIR
          Emitir mensaje de Error
          Llamada a función Lee
     FIN
FIN

/* Función Lee */
INICIO
     Se limpia el arreglo en el que se guardará el nombre
     Se solicita el nombre genérico de los archivos
     REPETIR
          ch2 = caracter en buffer
          SI ( ch2 = ESC ) LUEGO
               Termina Ejecución
          O SI ( ch2 = 0 ) LUEGO
               Lee caracter de buffer
          O SI ( ch2 = BACKSPACE ) LUEGO
               Decrementa contador de arreglo
               Borra caracter del arreglo
          O SI ( ch2 = ENTER ) LUEGO
               Pon caracter de fin de línea en arreglo
          O BIEN ( ch2 = otro caracter ) LUEGO
               Incrementa contador del arreglo
               Escribe caracter en el arreglo
          FIN
     HASTA ( ch2 = enter o contador > 49 )
FIN

/* Función Tamaño */
INICIO
     EN TANTO ( No sea fin de archivo ) REPETIR
          Lee línea de archivo y guarda en arreglo
          SI ( Lineas leídas > 599 ) LUEGO
               Cierra archivo
               Rompe ciclo en tanto
          FIN
     FIN
     SI ( Lineas leídas > 599 ) LUEGO
          Emite mensaje de error
     O BIEN
          Cierra archivo
          Muestra las primeras lineas que quepan en pantalla
     FIN
FIN
```

CAMBIAR DIRECTORIO

```
/* Declaración de Variables Locales */
INICIO
     Haz Menú de Presentación y de captura de Selección
     Mensaje ("Que deseas cambiar")
               Mensaje ("Unidad Lógica", valor)
               Mensaje ("Subdirectorio", valor)
     EN TANTO bandera = 1
          SI valor = unidad LUEGO
               Mensaje ("A que unidad deseas cambiarte",respuesta)
               Lee respuesta
               Checa ( respuesta <> NULO )
                    SI error = 1 LUEGO
                         Mensaje ("Error Unidad no Existente")
                         Rompe
                    O BIEN
                         Haz Cambio de unidad
                         Rompe
                    FIN
          O SI valor = subdirectorio LUEGO
               Mensaje ("A que subdirectorio deseas cambiarte",res-
               puesta)
               Lee respuesta
               Checa ( respuesta <> NULO )
                    SI error = 1 LUEGO
                         Mensaje ("Error subdirectorio no Existen-
                         te")
                         Rompe
                    O BIEN
                         Haz Cambio de subdirectorio
                         Rompe
                    FIN
          FIN
     FIN
FIN
```

VER DIRECTORIO

```
/* Declaración de Variables Locales */

INICIO
     Haz menú de Presentación y de captura de Selección
     Mensaje ("Nodos no Finales", selección)
     Mensaje ("Nodos Finales", selección)
     Mensaje ("Conexiones", selección)
     Mensaje ("Niveles", selección)
     Mensaje ("Rutas", selección)
     Mensaje ("Todos los archivos", selección)
     EN TANTO bandera = 1
```

```
          Lee valor de selección
          SI selección = nodos_no_finales LUEGO
               Haz filtro de archivos con extensión .NNF
               Haz sorteo de archivos
               Lista los archivos
               Rompe
          O SI selección = nodos_finales
               Haz filtro de archivos con extensión .NSF
               Haz sorteo de archivos
               Lista los archivos
               Rompe
          O SI selección = conexiones
               Haz filtro de archivos con extensión .CNX
               Haz sorteo de archivos
               Lista los archivos
               Rompe
          O SI selección = niveles
               Haz filtro de archivos con extensión .NIV
               Haz sorteo de archivos
               Lista los archivos
               Rompe
          O SI selección = rutas
               Haz filtro de archivos con extensión .RUT
               Haz sorteo de archivos
               Lista los archivos
               Rompe
          O SI selección = todos
               Haz filtro de archivos con extensión .*
               Haz sorteo de archivos
               Lista los archivos
               Rompe
          FIN
     FIN
FIN


GRAFICAR ARBOL DE CONOCIMIENTOS

/* Declaración de Variables Locales */

/* Programa Principal */
INICIO
     Limpiar Pantalla
     Llamada a función Pantalla
     Llamada a función Lee
     Llamada a función Sigue
FIN

/* Función Pantalla */
     /* mediante esta función se crea una pantalla de presentación
     en modo texto */
```

```
/* Función Lee */
INICIO
     Se limpia el arreglo en el que se guardara el nombre
     Se solicita el nombre genérico de los archivos
     REPETIR
          ch2 = caracter en buffer
          SI ( ch2 = ESC ) LUEGO
               Termina ejecución
          O SI ( ch2 = 0 ) LUEGO
               Lee caracter de buffer
          O SI ( ch2 = BACKSPACE ) LUEGO
               Decrementa contador de arreglo
               Borra caracter del arreglo
          O SI ( ch2 = ENTER ) LUEGO
               Pon caracter de fin de linea en arreglo
          O BIEN ( ch2 = otro_caracter ) LUEGO
               Incrementa contador del arreglo
               Escribe caracter en el arreglo
          FIN
     HASTA ( ch2 = ENTER o contador > 49 )
FIN

/* Función Detecta */
INICIO
     REPETIR
          Copia nombre genérico de archivo y concatena extensión
          SI ( Archivo no existe ) LUEGO
               Error
          FIN
     HASTA ( Verificar todos los archivos )
     SI ( Error ) LUEGO
          Emite mensaje de Error
          Llamada función Lee
     FIN
FIN

/* Función Sigue */
INICIO
     Llamada a función Limpia
     Llamada a función Lee_archivo
     Llamada a función Ini_val_list
     Llamada a función Ini_graficos
     Se escriben las lineas de la primer tabla mostrada
     Llamada a función árbol
     SI ( Llamada a función existe_raton ) LUEGO
          /* mediante esta llamada se verifica la existencia de un
          ratón, en caso de no existir, se emulará éste con el
          teclado */
          Esconde ratón
          Muestra ratón
     FIN
     Esconde ratón
```

```
     Muestra ratón
     REPETIR
          c = Llamada a función espera_entrada ( boton_izq )
          Llamada a función se_procesa ( c )
     HASTA ( Verdadero )
FIN

/* Función Limpia */
     /* Con esta función se limpia el arreglo que contendrá el
     archivo mostrado junto con el gráfico */

/* Función Lee_archivo */
     /* Esta función lee el archivo correspondiente a la informa-
     ción activa que se despliega junto con el gráfico, y la aloja
     en el arreglo previamente limpiado */

/* Función ini_val_list */
     /* Aquí se guardan los valores de las teclas activas, las
     coordenadas de los íconos y la función que se debe de ejecutar
     para cada caso */
     Llama función Cambia Tabla
     Llama función Imprime
     Llama función Salir

/* Función ini_graficos */
INICIO
     Se ajustan parámetros para trabajar sólo en modo gráfico
          VGA de 640x480
     SI ( Error en modo gráfico) LUEGO
          Emite mensaje de error
          Termina ejecución
     FIN
     Se crea pantalla de presentación en modo gráfico
     Se dibujan los íconos en su posición correspondiente
FIN

/* Función existe_raton */
INICIO
     Ejecuta interrupción 33h
     SI ( Existe ) LUEGO
          Mueve a posición (0,0)
          Muestra ratón
     O BIEN
          Inicializa parámetros para emulación de ratón
          Muestra ratón
     FIN
FIN

/* Función espera_entrada ( boton_izq ) */
     /* con esta función se analiza la entrada, de no existir ratón
     se toma el valor de la tecla presionada del buffer, si existe
     el mouse se procesa la petición */
```

```
/* Función se_procesa */
INICIO
     SI ( No_existe_raton ) LUEGO
          DESDE ( i <- 0 HASTA 5 ) REPETIR
               SI ( llave = c  ||  llave1 = c ) LUEGO
                    Ejecuta proceso correspondiente
               FIN
          FIN
     O BIEN
          Obtén coordenadas del ratón
          DESDE ( i <-  HASTA 5 ) REPETIR
               SI ( Coordenadas se hallan en área de ícono ) LUEGO
                    Ejecuta proceso correspondiente
               FIN
          FIN
     FIN
FIN

/* Función Arbol */
/* Esta función construye la gráfica del árbol de conocimientos */
/* Declaración de Variables Locales */
INICIO
     Abre archivo de Nodos Finales ( Conclusiones )
     i <-- Contador
     EN TANTO ( Archivo <> NULO ) REPETIR
          Copia ( Línea_archivo, C1 )
          Conclusión [i] = C1
          Incrementa i
     FIN
     Abre archivo de Conexiones
     i <-- Contador
     EN TANTO ( Archivo <> NULO ) REPETIR
          Copia ( Línea_archivo, C1, C2, C3, C4 )
          Conclusión [i].origen = C1
          Conclusión [i].destino = C2
          Conclusión [i].nivel = C3
          Conclusión [i].conexión = C4
          Incrementa i
     FIN
     Abre archivo de Niveles
     i <-- Contador
     EN TANTO ( Archivo <> NULO ) REPETIR
          Copia ( Línea_archivo, C1 )
          Conclusión [i] = C1
          Incrementa i
     FIN
     Calcula altura de elipses dependiendo No. de conclusiones
     Calcula ancho de elipses dependiendo No. de niveles
     Calcula separación entre niveles
     Dibuja nodos finales
     EN TANTO ( No se dibujen todos los nodos del árbol ) REPETIR
          DESDE ( i = 1 HASTA i <= # nodos finales ) REPETIR
```

```
                SI ( Se dibujaron nodos hijos de nodo padre ) LUEGO
                     Dibuja nodo padre
                     Dibuja líneas de conexión entre nodo padre e
                     hijos
                FIN
           FIN
     FIN
FIN

/* Función Cambia Tablas */
INICIO
     Analizar valor de variable indicador_tabla
     SI ( indicador_tabla = 0 ) LUEGO
          Muestra tabla conexiones
     O SI ( indicador_tabla = 1 ) LUEGO
          Muestra tabla nodos finales
     O SI ( indicador_tabla = 2 ) LUEGO
          Muestra tabla niveles
     FIN
FIN

/* Función Imprime */
/* Función que imprime Gráfico */

INICIO
     REPITE
          Checa estado de Impresora
          SI ( Impresora deshabilitada ) LUEGO
               Error y Mensaje
          FIN
     HASTA ( Impresora Lista)
     SI ( Impresora habilitada ) LUEGO
          Imprime Imagen
     FIN
FIN

/* Función Salir */
INICIO
     Cerrar modo gráfico
     Limpiar Pantalla
     Llamada a función Pantalla
     Llamada a función Lee
     Llamada a función Sigue
FIN


EJECUTAR ARBOL

/* Proceso predefinido y que es el correspondiente a la máquina de
Inferencia Microexpert */
```

BORRAR ARCHIVOS

```
/* Declaración de Variables Locales */

/* Programa Principal */
INICIO
     Limpia Pantalla
     Llamada a la Función Pantalla
FIN

/* Función Pantalla */
INICIO
     Hacer Pantalla de Presentación
     Llamada a la Función Ventana Izquierda
FIN

/* Función Ventana Izquierda*/
INICIO
     Construir ventana de petición de nombre de archivo genérico
     Llamada a la Función Presenta
FIN

/* Función Ventana Derecha */
INICIO
     Construir ventana de manejo de nombres de archivos
     Llamada a la Función Presental
FIN

/* Función Borra */
INICIO
     i <-- Contador
     Copiar ( archivo2 <-- nombre_genérico )
     Concatenar ( archivo2 con lista_extensiones [i-1] )
     Borrar archivo ( archivo2 )
     SI ( Error = 0 ) LUEGO
          Mensaje ( "Archivo ",archivo2,"borrado" )
          Mensaje ( "Oprima ENTER para continuar" )
          Acciona Sonido
          Dar Tiempo
          Desactiva Sonido
          Leer Tecla
     O BIEN
          Mensaje ( "Error: NO existe el archivo ",archivo2)
          Mensaje ( "Oprima ENTER para continuar" )
          Acciona Sonido
          Dar Tiempo
          Desactiva Sonido
          Leer Tecla
     FIN
          Error = 0
FIN
```

```
/* Función de Movimiento Hacia Arriba */
INICIO
     i <-- Contador
     SI ( i <= 1 ) LUEGO
          Limpiar Línea
          Hacer ( i = 6 )
          Colocar Cursor en (1,i)
          Imprime Caracter ASCII 175 como Selector
     O BIEN
          Limpiar Línea
          Colocar Cursor en (1,i)
          Hacer ( i = i-1 )
          Colocar Cursor en (1,i)
          Imprime Caracter ASCII 175 como Selector
     FIN
FIN

/* Función de Movimiento Hacia Abajo */
INICIO
     SI ( i <= 6 ) LUEGO
          Colocar Cursor en (1,i)
          Limpiar Línea
          Hacer ( i = 1 )
          Colocar Cursor en (1,i)
          Imprime Caracter ASCII 175 como Selector
     O BIEN
          Colocar Cursor en (1,i)
          Limpiar Línea
          Hacer ( i = i+1 )
          Colocar Cursor en (1,i)
          Imprime Caracter ASCII 175 como Selector
     FIN
FIN

/* Función Monitoreo de Teclas */
INICIO
     EN TANTO valor = 1
          Hacer ( ch <-- tecla ) presionada
          SI ( ch = 0 ) LUEGO
               Hacer ( ch <-- tecla presionada )
          FIN
          SI ( tecla presionada <-- ↑ ) LUEGO
               Llama a la Función Arriba
               Rompe
          O SI ( tecla presionada <-- ↓ ) LUEGO
               Llama a la Función Abajo
               Rompe
          O SI ( tecla presionada <-- SUPR ) LUEGO
               Llama a la Función Borra
               Llama a la Función Monitoreo
               Rompe
          O SI ( tecla presionada <-- ESC ) LUEGO
```

```
                Llamada a la Función Presenta
                Rompe
          O SI ( tecla presionada <-- default ) LUEGO
                Hacer ( Valor <-- tecla ) presionada
     FIN
FIN

/* Función Presenta */
INICIO
     Mensaje ("Petición de Archivo")
     Llamada Función Lee
FIN

/* Función Presenta1 */
INICIO
     i <-- Contador
     Mensaje ("Posibles archivos a ser Borrados")
     Mensaje ( Posibles archivos a ser Borrados, Nombres )
     DESDE ( i = 1 HASTA i = 6 ) REPETIR
          Copiar ( archivo1 <-- nombre_archivo )
          Concatenar ( archivo1 con lista_extensiones [i-1] )
          Imprime ( archivo1 )
     FIN
     Imprime ( Caracter ASCII 175 como selector )
     Valor <-- Tecla Presionada
     Llamada a Función monitoreo pasando valor
FIN

/* Función Lee */
INICIO
     Lectura del nombre del archivo y teclas relacionadas con la
          misma
     Llamada a la función ventana derecha
FIN


IMPRIMIR ARCHIVOS

/* Declaración de Variables Locales */

/* Programa Principal */
INICIO
     Limpia Pantalla
     Llamada a la Función Pantalla
FIN

/* Función Pantalla */
INICIO
     Hacer Pantalla de Presentación
     Llamada a la Función Ventana
FIN
```

```
/* Función Ventana */
INICIO
     Construir ventana de petición de nombre de archivo genérico
     Llamada a la Función Presenta
FIN

/* Función Ventana_Der */
INICIO
     Construir ventana de presentación de nombres de archivos
     Llamada a la Función Presenta1
FIN

/* Función Imprime */
INICIO
     i <-- Contador
     Hacer ( Cuenta = 4 )
     Copiar ( archivo2 <-- nombre_archivo )
     Concatenar ( archivo2 con lista_extensiones [i-1] )
     SI ( salida = Abertura de archivo ( archivo2 ) <> nulo ) LUEGO
          REPETIR
               Hacer ( Valor = 0 )
               Hacer ( Valor = señal puerto de la impresora )
               SI ( Valor <> 144 ) LUEGO
                    Acciona Sonido
                    Dar Tiempo
                    Desactiva Sonido
                    Mensaje ( "ERROR: Detección de Impresora
                         deshabilitada ")
                    Mensaje ( "Oprima tecla para Continuar" )
                    Leer Tecla
               FIN
          HASTA ( Valor <> 144 ) o ( Tecla Presionada <> Escape )
          Imprime ( Encabezado de Archivo )
          MIENTRAS ( No sea fin de Archivo )
               SI ( Cuenta <= 64 ) LUEGO
                    Haz Cuenta = Cuenta + 1
                    Imprime ( Cadena [cuenta-1] de texto, archivo2)
               O BIEN
                    Imprime ( Encabezado de Archivo )
                    Imprime ( Cadena [cuenta-1] de texto, archivo2)
                    Hacer ( Cuenta = 4 )
               FIN
          Imprime
          Enviar Caracter de Salto de Página
          Imprime ( Archivo [archivo2] Impreso )
          Mensaje ( "Oprima Tecla para Continuar" )
          Acciona Sonido
          Dar Tiempo
          Desactiva Sonido
          Leer Tecla
          Limpiar Ventana
     O BIEN
```

```
          Mensaje ( "ERROR: Inexistencia de archivo", archivo2 )
          Mensaje ( "Oprima tecla para Continuar" )
          Acciona Sonido
          Dar Tiempo
          Desactiva Sonido
          Leer Tecla
     FIN
FIN

/* Función de Movimiento Hacia Arriba */
INICIO
     i <-- Contador
     SI ( i <= 1 ) LUEGO
          Colocar Cursor en (1,i)
          Limpiar Línea
          Hacer ( i = 6 )
          Colocar Cursor en (1,i)
          Imprime Caracter ASCII 175 como Selector
     O BIEN
          Colocar Cursor en (1,i)
          Limpiar Línea
          Hacer ( i = i-1 )
          Colocar Cursor en (1,i)
          Imprime Caracter ASCII 175 como Selector
     FIN
FIN

/* Función de Movimiento Hacia Abajo */
INICIO
     SI ( i <= 6 ) LUEGO
          Colocar Cursor en (1,i)
          Limpiar Línea
          Hacer ( i = 1 )
          Colocar Cursor en (1,i)
          Imprime Caracter ASCII 175 como Selector
     O BIEN
          Colocar Cursor en (1,i)
          Limpiar Línea
          Hacer ( i = i+1 )
          Colocar Cursor en (1,i)
          Imprime Caracter ASCII 175 como Selector
     FIN
FIN

/* Función Monitoreo de Teclas */
INICIO
     EN TANTO valor = 1
          Hacer ( ch <-- tecla ) presionada
          SI ( ch = 0 ) LUEGO
               Hacer ( ch <-- tecla presionada )
          FIN
          SI ( tecla presionada <-- ↑ ) LUEGO
```

```
                Llama a la Función Arriba
                Rompe
          O SI ( tecla presionada <-- ↓ ) LUEGO
                Llama a la Función Abajo
                Rompe
          O SI ( tecla presionada <-- SUPR ) LUEGO
                Llama a la Función Borra
                Llama a la Función Monitoreo
                Rompe
          O SI ( tecla presionada <-- ESC ) LUEGO
                Llamada a la Función Presenta
                Rompe
          O SI ( tecla presionada <-- default ) LUEGO
                Hacer ( Valor <-- tecla ) presionada
     FIN
FIN

/* Función Presenta */
INICIO
     Mensaje ( Petición de Archivo )
     Llamada Función Lee
FIN

/* Función Presenta1 */
INICIO
     i <-- Contador
     Mensaje ("Posibles archivos a ser Impresos")
     Mensaje ( Posibles Archivos a ser impresos, Nombres )
     DESDE (i = 1 HASTA i = 6) REPETIR
          Copiar ( archivo1 <-- nombre_archivo )
          Concatenar ( archivo1 con lista_extensiones [i-1] )
          Colocar cursor en (5,i+5)
          Imprime ( archivo1 )
     FIN
     Imprime (Caracter ASCII 175 como selector)
     Valor <-- Tecla Presionada
     Llamada a Función monitoreo pasando valor
FIN

/* Función Lee */
INICIO
     Lectura del nombre del archivo y teclas relacionadas con la
          misma
     Llamada a la función ventana derecha
FIN
```

```
AYUDA

/* Declaración de Variables Locales */

INICIO
     Creación de archivos texto
     Crear liga de referencia entre archivos texto
     Presentación de ícono de ayuda en Programa Principal
     Activar ícono de ayuda sensible al contexto
     Crear ventana de presentación de ayuda
     EN TANTO bandera = 1 REPETIR
          Presentar archivo específico con opción de manejo
          interactivo entre todos los textos de ayuda
          Cerrar ventana de ayuda
     FIN
FIN
```

III.7 ANALISIS DE REQUERIMIENTOS

Después de definir que se va a hacer y como se va a hacer se debe definir con que se desarrollará, es decir se define el software que se utiliza en el desarrollo del sistema.

Antes de poder elegir la herramienta en que vamos a trabajar, es necesario definir las diferentes posibilidades que contemplamos, así como sus ventajas y desventajas.

Es importante en este punto tomar en cuenta el tiempo de programación, la disponibilidad del compilador, la versatilidad y documentación del mismo.

Para elegir las herramientas de acuerdo a los objetivos del sistema y la forma en que se desea que interactúe con el usuario, buscamos que dichas herramientas tengan características tales como entrada/salida de datos, funciones de manejo de gráficos, manejo de potentes estructuras de datos y de dispositivos periféricos, y que sea capaz de operar a bajo nivel.

Entre las herramientas posibles están:

- manejadores de bases de datos (Fox Base, Dbase, etc.).
- lenguajes de alto nivel (Pascal, Quick Basic, etc.).
- lenguajes de nivel medio (C).
- sistemas de programación de multimedios (Linkway).

Los manejadores de base de datos fueron desechados, ya que aunque facilitan el proceso de adquisición de datos, presentaban una gran limitante en cuanto al manejo de bases de datos. Los lenguajes de alto nivel se consideran no aptos para el desarrollo del sistema, debido a su difícil manejo en cuanto a rutinas de bajo nivel se refiere (Ensamblador).

En base a lo anterior se desarrollará el sistema en lenguaje de programación "C", debido a la amplia popularidad con que cuenta, además permite un fácil manejo de archivos, cuenta con una amplia gama de funciones gráficas, facilita el uso de programación a bajo nivel lo que permite explotar al máximo las capacidades del equipo de computo y además porque es fuertemente estructurado. Por otro lado, se cuenta con una gran portabilidad de los códigos entregados por dicho compilador.

Además, debido a que se desea interactuar activamente con el usuario, es necesario contar con un sistema que permita crear un entorno gráfico accesible fácilmente, característica que está disponible con el uso de un Sistema Multimedios, que en nuestro caso será Linkway, ya que es fácil de programar.

CAPITULO IV

PRUEBAS DE LA INTERFAZ

Este capítulo tiene por objetivo, mostrar las capacidades de que consta la interfaz, poniéndola a prueba con la construcción de dos ejemplos reales, partiendo de uno muy simple y avanzando en complejidad en el segundo.

Sistema Experto en Caracterización de Animales

Este Sistema Experto consta de la clasificación de 42 animales diferentes, tomando en cuenta sus características como lo son: el tamaño, el color, su clase, el medio en el que se desarrollan, etc.

El árbol de conocimientos consta de 18 niveles, 83 nodos y como ya se dijo, 42 conclusiones.

A continuación se presentan algunas de las pantallas que se obtienen al capturar el árbol de conocimiento por parte del usuario, usando el sistema.

CREACION DE TABLAS Y ARCHIVO DE CONOCIMIENTOS

```
Editor
Indique la ruta y el archivo donde se almacenaran los datos del árbol
  --> b:animal

Indique el numero de niveles del árbol (2-20) : 18

Cuantos nodos tiene el árbol de conocimientos (al menos 18) ?83

Mensajes

Esc -> Salir
```

CREACION DE TABLAS Y ARCHIVO DE CONOCIMIENTOS

Editor

NIVEL : 1

Introduzca el nombre del atributo para el nivel
--> vertebrado

Indique el tipo del nivel

1) Numérico
2) No numérico OPCION : 2

Introduzca el texto del 'prompt' para el nivel
--> es vertebrado?

Mensajes

ARCHIVO : b:animal NO. DE NIVELES : 18 NO. DE NODOS : 83

CREACION DE TABLAS Y ARCHIVO DE CONOCIMIENTOS

Editor

CONCLUSIONES

Cual es la conclusión para el nodo 1 del ultimo nivel (18) ?

--> ALCE

Mensajes

ARCHIVO : b:animal NO. DE NIVELES : 18 NO. DE NODOS : 83

CREACION DE TABLAS Y ARCHIVO DE CONOCIMIENTOS

```
                    Editor
                CONEXIÓN ENTRE NODOS

Con cuantos nodos se conecta el nodo 1    ? 2


  Nodo Origen : 1              Nodo Destino :  2
       Nivel : 1                     Nivel :  2


Cual es el valor de la conexión entre el nodo origen y el nodo destino ?
  --> SI



                    Mensajes

ARCHIVO : b:animal          NO. DE NIVELES : 18          NO. DE NODOS : 83
```

CREACION DE TABLAS Y ARCHIVO DE CONOCIMIENTOS

```
                    Editor
                CONEXION ENTRE NODOS

Con cuantos nodos se conecta el nodo 41   ? 2


  Nodo Origen : 41             Nodo Destino :  59
       Nivel : 17                    Nivel :  18


Cual es el valor de la conexión entre el nodo origen y el nodo destino ?
  --> SI



                    Mensajes
Aviso : ya se creo el archivo de conocimientos

ARCHIVO : b:animal          NO. DE NIVELES : 18          NO. DE NODOS : 83
```

El orden en el que se presentan las pantallas de captura es el mismo en que se le irán presentando al usuario, obviamente las pantallas se presentan el número de veces que sean necesarias, excepto la primera que sólo se presenta una vez.

Ahora presentamos las tablas que se obtienen después de

terminar la captura del árbol. En cada tabla se explica su significado.

Tabla de niveles:

Esta tabla consta de todos los niveles que forman el árbol el tipo de cada uno (1 numérico, 2 no numérico), junto con sus atributos y el prompt que tendrá el archivo .KB

NIVEL	TIPO	ATRIBUTO	TEXTO
1	2	vertebrado	es vertebrado?
2	2	clase	que clase?
3	2	medio	en que medio se desarrolla?
4	2	tamaño	que tamaño tiene?
5	2	cuernos	tiene cuernos?
6	2	pelo	tamaño del pelaje?
7	2	carnívoro	es carnívoro?
8	2	agua dulce	vive en agua dulce?
9	2	forma	se deforma con el tiempo?
10	2	concha	tiene concha?
11	2	cervido	es cervido?
12	2	clima	que tipo de clima habita?
13	2	beneficio	beneficia al hombre?
14	2	subreino	a que subreino pertenece?
15	2	movimiento	se mueve por sí mismo?
16	2	tentáculos	tiene tentáculos?
17	2	color	es de colores?
18	0	animal	

Tabla de nodos no finales:

Esta tabla consta del total de nodos que son intermedios o no finales. En ella se indica el número del nodo y el nivel al que pertenece.

NODO	NIVEL
1	1
2	2
3	2
4	3
5	3
6	3
7	3
8	4
9	4
10	4
11	5
12	6
13	6
14	7
15	7
16	7
17	7
18	7
19	8
20	9
21	10
22	11
23	12
24	12
25	12
26	12
27	12
28	13
29	13
30	13
31	13
32	13
33	13
34	13
35	14
36	15
37	15
38	15
39	16
40	17
41	17

Tabla de nodos finales:

Esta tabla contiene el número de nodo (final) y su conclusión correspondiente.

NODO	CONCLUSION

42	alce
43	cebú
44	toro almisclero
45	león
46	oso hormiguero
47	perro
48	leopardo
49	conejo
50	ratón
51	delfín
52	manati
53	águila
54	zopilote
55	avestruz
56	flamenco
57	ave del paraíso
58	cardenal
59	perico australiano
60	tiburón
61	barracuda
62	pez luna
63	esturión
64	piraña
65	carpa
66	hipocampo
67	salmón
68	pez espino
69	caracol
70	flagelado
71	sanguijuela
72	lombriz
73	caracol marino
74	anguila
75	espora
76	camarón
77	almeja
78	ostión
79	medusa
80	erizo
81	estrella de mar
82	pulpo
83	calamar

Tabla de Conexiones:

Esta tabla contiene todas las conexiones que existen entre los nodos del árbol, junto con su valor. Además se presenta el nivel al cual pertenece el nodo origen.

ORIGEN	DESTINO	CONEXION	NIVEL_ORIGEN
1	2	si	1
1	3	no	1

2	4	viviparo	2
2	5	oviparo	2
3	6	viviparo	2
3	7	oviparo	2
4	11	terrestre	3
4	35	marino	3
5	8	terrestre	3
5	9	marino	3
6	69	terrestre	3
6	70	marino	3
7	32	terrestre	3
7	10	marino	3
8	16	grande	4
8	41	pequeño	4
9	17	grande	4
9	18	pequeño	4
10	19	pequeño	4
10	39	grande	4
11	12	si	5
11	13	no	5
12	22	corto	6
12	44	largo	6
13	14	largo	6
13	15	corto	6
14	45	si	7
14	46	no	7
15	28	si	7
15	29	no	7
16	23	si	7
16	24	no	7
17	25	si	7
17	26	no	7
18	27	no	7
18	64	si	7
19	33	si	8
19	20	no	8
20	21	si	9
20	38	no	9
21	37	si	10
21	34	no	10
22	42	si	11
22	43	no	11
23	53	regular	12
23	54	cálido	12
24	40	regular	12
24	57	cálido	12
25	60	regular	12
25	61	cálido	12
26	62	regular	12
26	63	cálido	12
27	30	regular	12
27	31	cálido	12

28	47	si	13
28	48	no	13
29	49	si	13
29	50	no	13
30	65	si	13
30	66	no	13
31	67	si	13
31	68	no	13
32	71	si	13
32	72	no	13
33	73	si	13
33	36	no	13
34	78	si	13
34	79	no	13
35	51	cetáceos	14
35	52	sirenios	14
36	74	si	15
36	75	no	15
37	76	si	15
37	77	no	15
38	80	si	15
38	81	no	15
39	82	si	16
39	83	no	16
40	55	no	17
40	56	si	17
41	58	no	17
41	59	si	17

Tabla de Rutas:

Esta tabla contiene el total de las rutas que forman al árbol de conocimientos. De manera fácil es la tabla en la cual se puede hallar un regla completa (en nodos), ya que muestra la conexión entre nodos que cumplirán la regla específica.

NUMERO	RUTA							
1	1	2	4	11	12	22	42	
2	1	2	4	11	12	22	43	
3	1	2	4	11	12	44		
4	1	2	4	11	13	14	45	
5	1	2	4	11	13	14	46	
6	1	2	4	11	13	15	28	47
7	1	2	4	11	13	15	28	48
8	1	2	4	11	13	15	29	49
9	1	2	4	11	13	15	29	50
10	1	2	4	35	51			
11	1	2	4	35	52			
12	1	2	5	8	16	23	53	
13	1	2	5	8	16	23	54	
14	1	2	5	8	16	24	40	55
15	1	2	5	8	16	24	40	56

```
16      1   2   5   8   16  24  57
17      1   2   5   8   41  58
18      1   2   5   8   41  59
19      1   2   5   9   17  25  60
20      1   2   5   9   17  25  61
21      1   2   5   9   17  26  62
22      1   2   5   9   17  26  63
23      1   2   5   9   18  64
24      1   2   5   9   18  27  30  65
25      1   2   5   9   18  27  30  66
26      1   2   5   9   18  27  31  67
27      1   2   5   9   18  27  31  68
28      1   3   6   69
29      1   3   6   70
30      1   3   7   32  71
31      1   3   7   32  72
32      1   3   7   10  19  33  73
33      1   3   7   10  19  33  36  74
34      1   3   7   10  19  33  36  75
35      1   3   7   10  19  20  21  37  76
36      1   3   7   10  19  20  21  37  77
37      1   3   7   10  19  20  21  34  78
38      1   3   7   10  19  20  21  34  79
39      1   3   7   10  19  20  38  80
40      1   3   7   10  19  20  38  81
41      1   3   7   10  39  82
42      1   3   7   10  39  83
```

Programa de Base de Conocimientos:

Este es el resultado más útil para el usuario ya que este es el archivo sobre el cual el Mecanismo de Inferencia Microexpert realizará las inferencias para obtener las conclusiones.

```
prompt vertebrado
es vertebrado?
.

prompt clase
que clase?
.

prompt medio
en que medio se desarrolla?
.

prompt tamaño
que tamaño tiene?
.

prompt cuernos
tiene cuernos?
```

```
prompt pelo
tamaño del pelaje?
.

prompt carnívoro
es carnívoro?
.

prompt agua dulce
vive en agua dulce?
.

prompt forma
se deforma con el tiempo?
.

prompt concha
tiene concha?
.

prompt cervido
es cervido?
.

prompt clima
que tipo de clima habita?
.

prompt beneficio
beneficia al hombre?
.

prompt subreino
a que subreino pertenece?
.

prompt movimiento
se mueve por si mismo?
.

prompt tentáculos
tiene tentáculos?
.

prompt color
es de colores?
.

1
if vertebrado is si
```

```
and clase is viviparo
and medio is terrestre
and cuernos is si
and pelo is corto
and cervido is si
then animal is alce
.

2
if vertebrado is si
and clase is viviparo
and medio is terrestre
and cuernos is si
and pelo is corto
and cervido is no
then animal is cebú
.

3
if vertebrado is si
and clase is viviparo
and medio is terrestre
and cuernos is si
and pelo is largo
then animal is toro almisclero
.

4
if vertebrado is si
and clase is viviparo
and medio is terrestre
and cuernos is no
and pelo is largo
and carnívoro is si
then animal is león
.

5
if vertebrado is si
and clase is viviparo
and medio is terrestre
and cuernos is no
and pelo is largo
and carnívoro is no
then animal is oso hormiguero
.

6
if vertebrado is si
and clase is viviparo
and medio is terrestre
and cuernos is no
```

```
and pelo is corto
and carnívoro is si
and beneficio is si
then animal is perro
.

7
if vertebrado is si
and clase is viviparo
and medio is terrestre
and cuernos is no
and pelo is corto
and carnívoro is si
and beneficio is no
then animal is leopardo
.

8
if vertebrado is si
and clase is viviparo
and medio is terrestre
and cuernos is no
and pelo is corto
and carnívoro is no
and beneficio is si
then animal is conejo
.

9
if vertebrado is si
and clase is viviparo
and medio is terrestre
and cuernos is no
and pelo is corto
and carnívoro is no
and beneficio is no
then animal is ratón
.

10
if vertebrado is si
and clase is viviparo
and medio is marino
and subreino is cetáceos
then animal is delfín
.

11
if vertebrado is si
and clase is viviparo
and medio is marino
and subreino is sirenios
```

```
then animal is manati
.

12
if vertebrado is si
and clase is oviparo
and medio is terrestre
and tamaño is grande
and carnívoro is si
and clima is regular
then animal is águila
.

13
if vertebrado is si
and clase is oviparo
and medio is terrestre
and tamaño is grande
and carnívoro is si
and clima is cálido
then animal is zopilote
.

14
if vertebrado is si
and clase is oviparo
and medio is terrestre
and tamaño is grande
and carnívoro is no
and clima is regular
and color is no
then animal is avestruz
.

15
if vertebrado is si
and clase is oviparo
and medio is terrestre
and tamaño is grande
and carnívoro is no
and clima is regular
and color is si
then animal is flamenco
.

16
if vertebrado is si
and clase is oviparo
and medio is terrestre
and tamaño is grande
and carnívoro is no
and clima is cálido
```

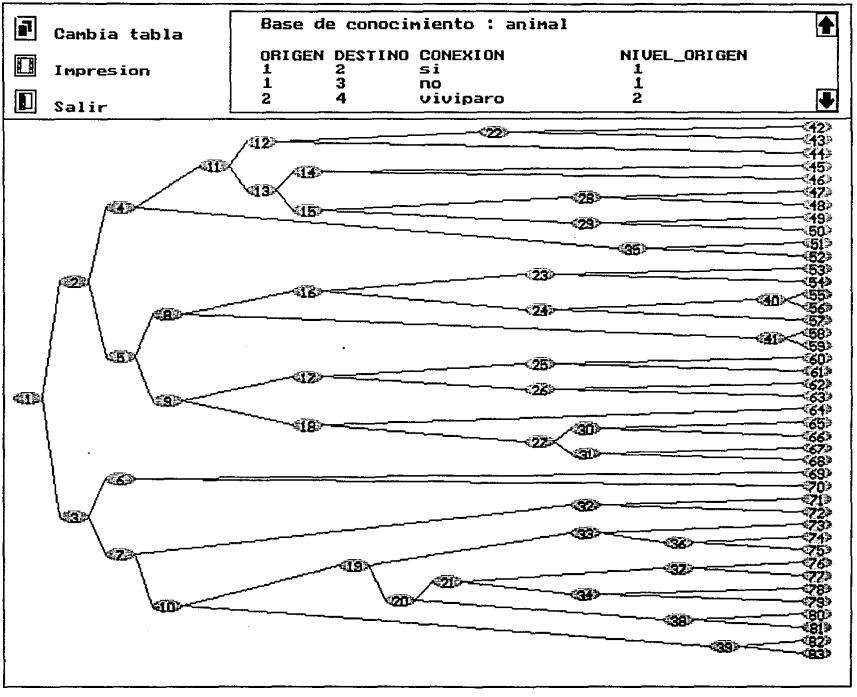
```
then animal is ave del paraíso
.

17
if vertebrado is si
and clase is oviparo
and medio is terrestre
and tamaño is pequeño
and color is no
then animal is cardenal
.

18
if vertebrado is si
and clase is oviparo
and medio is terrestre
and tamaño is pequeño
and color is si
then animal is perico australiano
.

19
if vertebrado is si
and clase is oviparo
and medio is marino
and tamaño is grande
and carnívoro is si
and clima is regular
then animal is tiburón
.

20
if vertebrado is si
and clase is oviparo
and medio is marino
and tamaño is grande
and carnívoro is si
and clima is cálido
then animal is barracuda
.

21
if vertebrado is si
and clase is oviparo
and medio is marino
and tamaño is grande
and carnívoro is no
and clima is regular
then animal is pez luna
.

22
```

```
if vertebrado is si
and clase is oviparo
and medio is marino
and tamaño is grande
and carnívoro is no
and clima is cálido
then animal is esturión
.

23
if vertebrado is si
and clase is oviparo
and medio is marino
and tamaño is pequeño
and carnívoro is si
then animal is piraña
.

24
if vertebrado is si
and clase is oviparo
and medio is marino
and tamaño is pequeño
and carnívoro is no
and clima is regular
and beneficio is si
then animal is carpa
.

25
if vertebrado is si
and clase is oviparo
and medio is marino
and tamaño is pequeño
and carnívoro is no
and clima is regular
and beneficio is no
then animal is hipocampo
.

26
if vertebrado is si
and clase is oviparo
and medio is marino
and tamaño is pequeño
and carnívoro is no
and clima is cálido
and beneficio is si
then animal is salmón
.

27
```

```
if vertebrado is si
and clase is oviparo
and medio is marino
and tamaño is pequeño
and carnívoro is no
and clima is cálido
and beneficio is no
then animal is pez espino
.

28
if vertebrado is no
and clase is viviparo
and medio is terrestre
then animal is caracol
.

29
if vertebrado is no
and clase is viviparo
and medio is marino
then animal is flagelado
.

30
if vertebrado is no
and clase is oviparo
and medio is terrestre
and beneficio is si
then animal is sanguijuela
.

31
if vertebrado is no
and clase  is oviparo
and medio is terrestre
and beneficio is no
then animal is lombriz
.

32
if vertebrado is no
and clase is oviparo
and medio is marino
and tamaño is pequeño
and agua dulce is si
and beneficio is si
then animal is caracol marino
.

33
if vertebrado is no
```

```
and clase is oviparo
and medio is marino
and tamaño is pequeño
and agua dulce is si
and beneficio is no
and movimiento is si
then animal is anguila
.

34
if vertebrado is no
and clase is oviparo
and medio is marino
and tamaño is pequeño
and agua dulce is si
and beneficio is no
and movimiento is no
then animal is espora
.

35
if vertebrado is no
and clase is oviparo
and medio is marino
and tamaño is pequeño
and agua dulce is no
and forma is si
and concha is si
and movimiento is si
then animal is camaron
.

36
if vertebrado is no
and clase is oviparo
and medio is marino
and tamaño is pequeño
and agua dulce is no
and forma is si
and concha is si
and movimiento is no
then animal is almeja
.

37
if vertebrado is no
and clase is oviparo
and medio is marino
and tamaño is pequeño
and agua dulce is no
and forma is si
and concha is no
```

```
and beneficio is si
then animal is ostion
.

38
if vertebrado is no
and clase is oviparo
and medio is marino
and tamaño is pequeño
and agua dulce is no
and forma is si
and concha is no
and beneficio is no
then animal is medusa
.

39
if vertebrado is no
and clase is oviparo
and medio is marino
and tamaño is pequeño
and agua dulce is no
and forma is no
and movimiento is si
then animal is erizo
.

40
if vertebrado is no
and clase is oviparo
and medio is marino
and tamaño is pequeño
and agua dulce is no
and forma is no
and movimiento is no
then animal is estrella de mar
.

41
if vertebrado is no
and clase is oviparo
and medio is marino
and tamaño is grande
and tentaculos is si
then animal is pulpo
.

42
if vertebrado is no
and clase is oviparo
and medio is marino
and tamaño is grande
```


Cambia tabla
Impresion
Salir
Base de conocimiento : animal
ORIGEN DESTINO CONEXION NIVEL_ORIGEN
1 2 si 1
1 3 no 1
2 4 viviparo 2

```
and tentaculos is no
then animal is calamar
.
```

Sistema Experto en Diagnóstico de Enfermedades

Este segundo sistema experto consiste en ser un auxiliar para el diagnóstico de enfermedades (muy básico).

Se distinguen enfermedades tales como lo son: la gripa, la viruela, el sarampión, el infarto, etc.

Consta de un total de 71 nodos, 19 niveles y 36 conclusiones. Al igual que el anterior se muestran las tablas generadas con la interfaz pero no se explican, ya que tienen el mismo significado que las anteriores.

Tabla de Niveles:

NIVEL	TIPO	ATRIBUTO	TEXTO
1	2	contagioso	es contagiosa?
2	2	fiebre	presenta fiebre?
3	2	afecta_piel	presenta lesiones en la piel?
4	2	dolor cabeza	presenta dolor de cabeza?
5	2	taquicardia	presenta taquicardia?
6	2	anemia	sufre de anemia?
7	2	convulsiones	dan convulsiones?
8	2	venas	afecta las venas?
9	2	musculos	duelen los musculos?
10	2	contracción	existe asincronia en los ventrículos?
11	2	sentidos	afecta los sentidos?
12	2	rehabilitación	la enfermedad es rehabilitable?
13	2	defecto	es un defecto fisiológico?
14	2	aparato	utilizas alguna aparato

			externo?
15	2	huesos	hay dolor en huesos?
16	2	mal agudo	se agudiza la enfermedad?
17	2	infección	se da por infección?
18	2	porción	afecta la porción externa del ojo?
19	0	enfermedad	

Tabla de Nodos no Finales:

NODO	NIVEL
1	1
2	2
3	3
4	3
5	4
6	4
7	4
8	5
9	5
10	5
11	6
12	7
13	7
14	7
15	8
16	9
17	9
18	9
19	10
20	11
21	11
22	12
23	12
24	13
25	14
26	15
27	16
28	16
29	16
30	16
31	16
32	17
33	18
34	18
35	18

Tabla de Nodos Finales:

NODO	CONCLUSION
36	tifus epidemico
37	escarlatina
38	viruela
39	herpes
40	rubéola
41	paludismo
42	tuberculosis
43	bronquitis
44	gripa
45	laringitis
46	hepatitis
47	peste bubónica
48	hidrofobia
49	tetanos
50	solitaria
51	arteriosclerosis
52	infarto
53	ventriculacion
54	soplo
55	arritmia
56	astigmatismo
57	sordera temporal
58	sordera
59	miopia
60	cataratas
61	leucoma
62	desprendimiento de retina
63	glaucoma
64	defecto neurológico
65	paralisis cerebral
66	síndrome de down
67	reumatismo
68	artritis
69	descalsificacion
70	pielonefritis
71	asma

Tabla de Conexiones:

ORIGEN	DESTINO	CONEXION	NIVEL_ORIGEN
1	2	si	1
1	10	no	1
2	3	si	2
2	4	no	2
3	5	si	3
3	6	no	3
4	46	si	3

4	7	no	3
5	8	si	4
5	40	no	4
6	9	si	4
6	11	no	4
7	47	si	4
7	14	no	4
8	12	si	5
8	13	no	5
9	27	si	5
9	45	no	5
10	15	si	5
10	18	no	5
11	43	si	6
11	44	no	6
12	36	si	7
12	37	no	7
13	38	si	7
13	39	si	7
14	48	si	7
14	17	no	7
15	51	si	8
15	16	no	8
16	19	si	9
16	55	no	9
17	49	si	9
17	50	no	9
18	20	si	9
18	21	no	9
19	28	si	10
19	54	no	10
20	22	si	11
20	24	no	11
21	23	si	11
21	26	no	11
22	56	no	12
22	25	si	12
23	65	si	12
23	66	no	12
24	32	si	13
24	64	no	13
25	57	no	14
25	29	si	14
26	30	si	15
26	31	no	15
27	41	si	16
27	42	no	16
28	52	si	16
28	53	no	16
29	58	si	16
29	59	no	16
30	35	si	16

30	69	no	16
31	70	si	16
31	71	no	16
32	33	si	17
32	34	no	17
33	60	si	18
33	61	no	18
34	62	si	18
34	63	no	18
35	67	no	18
35	68	si	18

Tabla de Rutas:

NUMERO	RUTA							
1	1	2	3	5	8	12	36	
2	1	2	3	5	8	12	37	
3	1	2	3	5	8	13	38	
4	1	2	3	5	8	13	39	
5	1	2	3	5	40			
6	1	2	3	6	9	27	41	
7	1	2	3	6	9	27	42	
8	1	2	3	6	11	43		
9	1	2	3	6	11	44		
10	1	2	3	6	9	45		
11	1	2	4	46				
12	1	2	4	7	47			
13	1	2	4	7	14	48		
14	1	2	4	7	14	17	49	
15	1	2	4	7	14	17	50	
16	1	10	15	51				
17	1	10	15	16	19	28	52	
18	1	10	15	16	19	28	53	
19	1	10	15	16	19	54		
20	1	10	15	16	55			
21	1	10	18	20	22	56		
22	1	10	18	20	22	25	57	
23	1	10	18	20	22	25	29	58
24	1	10	18	20	22	25	29	59
25	1	10	18	20	24	32	33	60
26	1	10	18	20	24	32	33	61
27	1	10	18	20	24	32	34	62
28	1	10	18	20	24	32	34	63
29	1	10	18	20	24	64		
30	1	10	18	21	23	65		
31	1	10	18	21	23	66		
32	1	10	18	21	26	30	35	67
33	1	10	18	21	26	30	35	68
34	1	10	18	21	26	30	69	
35	1	10	18	21	26	31	70	
36	1	10	18	21	26	31	71	

Programa de Base de Conocimientos:

```
prompt contagioso
es contagiosa?
.

prompt fiebre
presenta fiebre?
.

prompt afecta_piel
presenta lesiones en la piel?
.

prompt dolor cabeza
presenta dolor de cabeza?
.

prompt taquicardia
presenta taquicardia?
.

prompt anemia
sufre de anemia?
.

prompt convulsiones
dan convulsiones?
.

prompt venas
afecta las venas?
.

prompt músculos
duelen los músculos?
.

prompt contracción
existe asincronia en los ventrículos?
.

prompt sentidos
afecta los sentidos?
.

prompt rehabilitación
la enfermedad es rehabilitable?
.

prompt defecto
```

```
es un defecto fisiológico?
.

prompt aparato
utilizas algún aparato externo?
.

prompt huesos
hay dolor en huesos?
.

prompt mal agudo
se agudiza la enfermedad?
.

prompt infección
se da por infección?
.

prompt porción
afecta la porción externa del ojo?
.

1
if contagioso is si
and fiebre is si
and afecta_piel is si
and dolor cabeza is si
and taquicardia is si
and convulsiones is si
then enfermedad is tifus epidemico
.

2
if contagioso is si
and fiebre is si
and afecta_piel is si
and dolor cabeza is si
and taquicardia is si
and convulsiones is no
then enfermedad is escarlatina
.

3
if contagioso is si
and fiebre is si
and afecta_piel is si
and dolor cabeza is si
and taquicardia is no
and convulsiones is si
then enfermedad is viruela
.
```

```
4
if contagioso is si
and fiebre is si
and afecta_piel is si
and dolor cabeza is si
and taquicardia is no
and convulsiones is si
then enfermedad is herpes
.

5
if contagioso is si
and fiebre is si
and afecta_piel is si
and dolor cabeza is no
then enfermedad is rubeola
.

6
if contagioso is si
and fiebre is si
and afecta_piel is no
and dolor cabeza is si
and taquicardia is si
and mal agudo is si
then enfermedad is paludismo
.

7
if contagioso is si
and fiebre is si
and afecta_piel is no
and dolor cabeza is si
and taquicardia is si
and mal agudo is no
then enfermedad is tuberculosis
.

8
if contagioso is si
and fiebre is si
and afecta_piel is no
and dolor cabeza is no
and anemia is si
then enfermedad is bronquitis
.

9
if contagioso is si
and fiebre is si
and afecta_piel is no
and dolor cabeza is no.
```

```
and anemia is no
then enfermedad is gripa
.

10
if contagioso is si
and fiebre is si
and afecta_piel is no
and dolor cabeza is si
and taquicardia is no
then enfermedad is laringitis
.

11
if contagioso is si
and fiebre is no
and afecta_piel is si
then enfermedad is hepatitis
.

12
if contagioso is si
and fiebre is no
and afecta_piel is no
and dolor cabeza is si
then enfermedad is peste bubonica
.

13
if contagioso is si
and fiebre is no
and afecta_piel is no
and dolor cabeza is no
and convulsiones is si
then enfermedad is hidrofobia
.

14
if contagioso is si
and fiebre is no
and afecta_piel is no
and dolor cabeza is no
and convulsiones is no
and musculos is si
then enfermedad is tetanos
.

15
if contagioso is si
and fiebre is no
and afecta_piel is no
and dolor cabeza is no
```

```
and convulsiones is no
and musculos is no
then enfermedad is solitaria
.

16
if contagioso is no
and taquicardia is si
and venas is si
then enfermedad is arteroesclerosis
.

17
if contagioso is no
and taquicardia is si
and venas is no
and musculos is si
and contraccion is si
and mal agudo is si
then enfermedad is infarto
.

18
if contagioso is no
and taquicardia is si
and venas is no
and musculos is si
and contraccion is si
and mal agudo is no
then enfermedad is ventriculacion
.

19
if contagioso is no
and taquicardia is si
and venas is no
and musculos is si
and contraccion is no
then enfermedad is soplo
.

20
if contagioso is no
and taquicardia is si
and venas is no
and musculos is no
then enfermedad is arritmia
.

21
if contagioso is no
and taquicardia is no_ .
```

```
and musculos is si
and sentidos is si
and rehabilitacion is no
then enfermedad is astigmatismo
.

22
if contagioso is no
and taquicardia is no
and musculos is si
and sentidos is si
and rehabilitacion is si
and aparato is no
then enfermedad is sordera temporal
.

23
if contagioso is no
and taquicardia is no
and musculos is si
and sentidos is si
and rehabilitacion is si
and aparato is si
and mal agudo is si
then enfermedad is sordera
.

24
if contagioso is no
and taquicardia is no
and musculos is si
and sentidos is si
and rehabilitacion is si
and aparato is si
and mal agudo is no
then enfermedad is miopia
.

25
if contagioso is no
and taquicardia is no
and musculos is si
and sentidos is no
and defecto is si
and infeccion is si
and porcion is si
then enfermedad is cataratas
.

26
if contagioso is no
and taquicardia is no
```

```
and musculos is si
and sentidos is no
and defecto is si
and infeccion is si
and porcion is no
then enfermedad is leucoma
.

27
if contagioso is no
and taquicardia is no
and musculos is si
and sentidos is no
and defecto is si
and infeccion is no
and porcion is si
then enfermedad is desprendimiento de retina
.

28
if contagioso is no
and taquicardia is no
and musculos is si
and sentidos is no
and defecto is si
and infeccion is no
and porcion is no
then enfermedad is glaucoma
.

29
if contagioso is no
and taquicardia is no
and musculos is si
and sentidos is no
and defecto is no
then enfermedad is defecto neurologico
.

30
if contagioso is no
and taquicardia is no
and musculos is no
and sentidos is si
and rehabilitacion is si
then enfermedad is paralisis cerebral
.

31
if contagioso is no
and taquicardia is no
and musculos is no
```

```
and sentidos is si
and rehabilitacion is no
then enfermedad is sindrome de down
.

32
if contagioso is no
and taquicardia is no
and musculos is no
and sentidos is no
and huesos is si
and mal agudo is si
and porcion is no
then enfermedad is reumatismo
.

33
if contagioso is no
and taquicardia is no
and musculos is no
and sentidos is no
and huesos is si
and mal agudo is si
and porcion is si
then enfermedad is artritis
.

34
if contagioso is no
and taquicardia is no
and musculos is no
and sentidos is no
and huesos is si
and mal agudo is no
then enfermedad is descalsificacion
.

35
if contagioso is no
and taquicardia is no
and musculos is no
and sentidos is no
and huesos is no
and mal agudo is si
then enfermedad is pielonefritis
.

36
if contagioso is no
and taquicardia is no
and musculos is no
and sentidos is no
```

```
and huesos is no
and mal agudo is no
then enfermedad is asma
.
```

Ahora en éste ejemplo mostraremos las inferencias realizadas por Microexpert sobre la Base de Conocimientos, de forma que se pueda tener un complemento con el ejemplo anterior. Las pantallas mostradas son para la regla número 34.

```
BASE DE CONOCIMIENTOS  ( TECLEE 'Quit' PARA FINALIZAR EL PROGRAMA ) : MEDICO


BASES DE CONOCIMIENTO DISPONIBLES:
ANIMALES     MEDICO
```

```
CUAL ES EL OBJETIVO DE ESTA CONSULTA ? ENFERMEDAD


LOS OBJETIVOS VALIDOS SON:

ANIMAL     ENFERMEDAD      QUIT
```

```
es contagiosa ?

NO

RESPUESTAS VALIDAS PUEDEN SER:

NO                      SI
WHY                     WHAT
RULE                    HOW
QUIT
```

```
presenta taquicardia?

NO

RESPUESTAS VALIDAS PUEDEN SER:

NO                      SI
WHY                     WHAT
RULE                    HOW
WHATIF                  QUIT
```

```
duelen los músculos?

NO

RESPUESTAS VALIDAS PUEDEN SER:

NO                      SI
WHY                     WHAT
RULE                    HOW
WHATIF                  QUIT
```

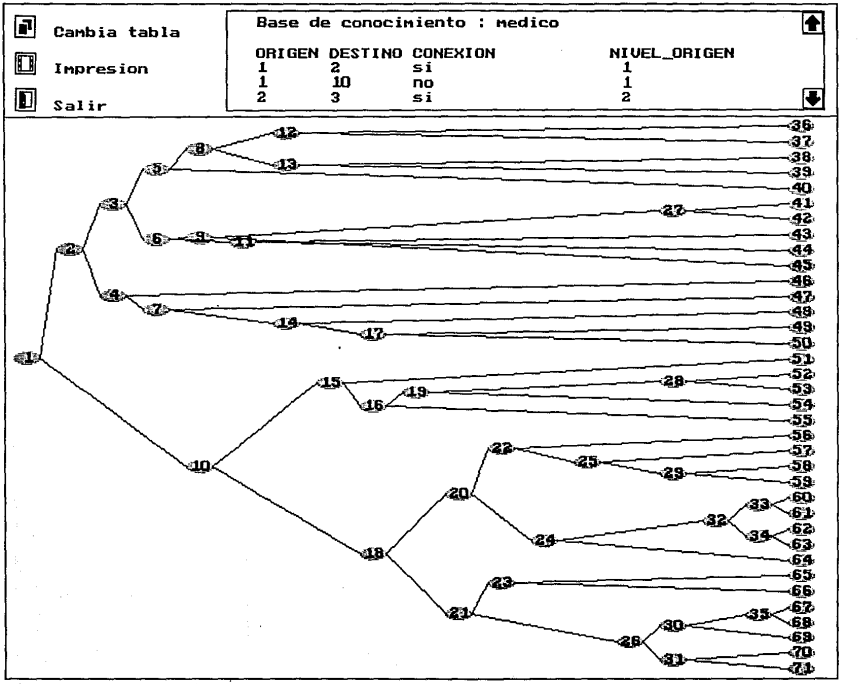
```
afecta los sentidos?

NO

RESPUESTAS VALIDAS PUEDEN SER:

NO                      SI
WHY                     WHAT
RULE                    HOW
WHATIF                  QUIT
```

```
hay dolor en huesos?

SI

RESPUESTAS VALIDAS PUEDEN SER:

NO                      SI
WHY                     WHAT
RULE                    HOW
WHATIF                  QUIT
```

```
se agudiza la enfermedad?

NO

RESPUESTAS VALIDAS PUEDEN SER:

NO                      SI
WHY                     WHAT
RULE                    HOW
WHATIF                  QUIT
```

```
Conclusion :
ENFERMEDAD          ES DESCALSIFICACION



PRESIONE   <BARRA ESPACIADORA> PARA CONTINUAR.
```

```
 LOS SIGUIENTES HECHOS SE HAN DETERMINADO :
  1) ENFERMEDAD          ES DESCALSIFICACION
  2) MAL AGUDO           ES NO
  3) HUESOS              ES SI
  4) SENTIDOS            ES NO
  5) MUSCULOS            ES NO
  6) TAQUICARDIA         ES NO
  7) CONTAGIOSA          ES NO

                    PRESIONE   <ENTER>   PARA CONTINUAR
```


Cambia tabla
Impresion
Salir
Base de conocimiento : medico
ORIGEN DESTINO CONEXION NIVEL_ORIGEN
1 2 si 1
1 10 no 1
2 3 si 2

CONCLUSIONES

Lo que podemos esperar de los S. E. es su diseminación en varios campos; una mayor aceptación de la gente que tendrá que cambiar su forma de pensar y observar que estos no vienen a reemplazarlos, sino sólo a ayudarlos en tareas repetitivas que a ellos les toman una gran cantidad de tiempo; así como aceptar el hecho de que dichos S. E. sólo lo serán en un campo con dominio muy reducido y no como se pensó en lo años 60's.

También, podemos esperar que con la propagación de dichos sistemas, más gente se interese en la investigación de la I. A. y de los S. E., entre ellos, expertos humanos que ayudarán en gran forma al desarrollo de dichos Sistemas.

Con la integración de la interfaz hombre-maquina al mecanismo de inferencia MICROEXPERT, se proporcionan al usuario utilerías que hacen mas fácil el manejo de dicho mecanismo de inferencia para el desarrollo de Sistemas Expertos.

Las ventajas que ofrece la interfase implementada son:

- el uso de menúes y de ventanas.

- la utilización de un intérprete para la creación del archivo de conocimientos que usa directamente MICROEXPERT.

- la facilidad para consultar las tablas que definen al sistema experto que se este desarrollando y que son creadas por el mismo intérprete.

- la creación del árbol de conocimientos que representa al sistema experto.

- la posibilidad de imprimir todos los archivos relacionados al sistema experto: archivo de conocimientos, tablas de niveles, de nodos, de hojas o conclusiones, de ramas o conexiones, de rutas y de la gráfica del árbol de conocimientos.

La interfaz se logro desarrollar siguiendo las normas propuestas para un buen diseño de interfases hombre-maquina, es decir, facilidad de uso, entorno amigable con el usuario, rapidez al accesar utilerías, claridad al incluir mensajes de estado y de error, proporcionar ayuda en el manejo del sistema, así como manejar elementos de despliegue de texto.

Es importante señalar que el alcance de la tesis no es incrementar la capacidad del mecanismo de inferencia, sino de hacer mas fácil y amigable el uso del mismo; por lo que se deja abierta

la posibilidad para que en un futuro se continúe el desarrollo de ésta interfase, mejorándola, al incorporarle un editor que permita la fácil modificación de las Bases de Conocimientos, sin que en ello tenga que intervenir el usuario a nivel programación, sino sólo a dialogo pregunta-respuesta.

MANUAL DE USUARIO

1.- REQUERIMIENTOS DE HARDWARE

- Computadora 286 o superior.
- Monitor VGA (Recomendable a color)
- Ratón (Recomendable)
- Disco duro
- Unidad de disco flexible de 3.5" o 5.25" de alta densidad

2.- MANEJO DE LA INTERFAZ

Al desarrollar la interfaz siempre se pensó en presentar al usuario menúes fáciles de accesar, cuestión que creemos se cumple cabalmente; así también en casi todas las pantallas de presentación se indica cuales son las teclas programadas para realizar alguna tarea en específico.

A continuación se listan aquellas teclas que auxiliarán al usuario a realizar las diferentes tareas :

- ESC.- En casi todas las herramientas de la interfaz con esta tecla se detiene la ejecución de la tarea en cuestión y se regresa al menú inmediato superior.

- SUPRIMIR.- Con esta tecla se realiza la selección del archivo en ciertos menúes que se encuentran en aplicaciones tales como impresión o eliminación de archivos.

- F10.- Esta tecla únicamente esta activa dentro del editor de archivos. Siempre que se desee visualizar un archivo será necesario presionar esta tecla par poder entrar el nombre del archivo deseado.

- FLECHAS DEL CURSOR.- Estas teclas son necesarias en ciertas aplicaciones que requieren hacer desplazamientos de pantalla.

El resto de las teclas funcionan de la misma manera en que fueron configuradas al iniciar la operación de la computadora.

El uso de la ayuda es interactivo y se tiene específica en cualquier tópico que se desee, siempre y cuando nos encontremos en el menú principal de la interfaz, con sólo presionar el botón de Ayuda, el cual se encuentra al calce de la pantalla de la computadora, se presenta ayuda dependiendo del menú que esté abierto o se presenta la ayuda general e índice. Si se presenta el

índice se puede elegir un tipo de ayuda más específico si se selecciona la el número resaltado con el cursor del mouse. El manejo de la ayuda es sencillo, se presenta una ventana con el texto solicitado, para recorrer el texto dentro de ésta, se utilizan las flechas del cursor, las teclas de avance página o con el mouse se presionan los íconos de avance de página, mismos que aparecen en las esquinas superior e inferior derecha de la ventana. Para cerrar la ventana de ayuda se oprime el botón Cancelar o se da un click izquierdo en cualquier lugar fuera de la misma o se posiciona el cursor sobre el close dot (pequeña ícono que está en la esquina superior izquierda de cualquier ventana) y se da un click.

Todas las opciones están señaladas con íconos, títulos o ambos, por lo cual la identificación de estas es muy simple. Si no se cuenta con mouse, pueden usarse las teclas del cursor, para mover a éste dentro de la pantalla y se usa la tecla de ENTER como si fuera el click izquierdo del mouse. Nota: si se arranca el sistema sin mouse, el programa lo detecta y envía un mensaje que indica que se debe de pulsar el signo '+' del teclado numérico para usar el teclado en el manejo de la interfaz.

3.- ADMINISTRACION DE ARCHIVOS

3.1 CREACION DE TABLAS

Este módulo presenta una pantalla con el encabezado "CREACION DE TABLAS Y ARCHIVO DE CONOCIMIENTOS"; dicha pantalla consta de un editor, en el cual aparecen preguntas que deben ser contestadas por el usuario y que conciernen a la creación del Sistema Experto.

El primer dato solicitado es el nombre genérico que se dará a los archivos que se crearán, debe proporcionarse la ruta en la cual se salvarán dichos archivos; ésta ruta debe de seguir los lineamientos establecidos por el Sistema Operativo D.O.S. En caso de no darse la ruta los archivos serán almacenados en el directorio de trabajo actual. El nombre genérico no deberá de tener ninguna extensión.

Ejemplo: c:\mio\trabajo\nombre_archivo

En caso de cometer algún error al teclear puede usarse la tecla backspace para borrar y reescribir.

Como siguiente paso se solicita el número de niveles que tendrá el árbol, este valor debe de ser entre 2 y 20; el programa solicitará el número de nodos con los que cuenta el árbol indicando la cantidad mínima aceptada. Después se solicita el nombre del atributo para cada nivel, así como su tipo (numérico o no numérico); en el caso de cometer un error en el nombre se puede

corregir antes de teclear <ENTER>, y con el tipo, si no se da una selección válida (presentadas en pantalla) el sistema lo indica. Terminado esto se solicita el comentario o PROMPT que debe llevar cada nivel; estos datos son solicitados para los N-1 niveles del árbol, debido a que el último es el nivel de conclusiones y en ese caso sólo se pedirá el nombre del atributo del nivel.

Ahora el usuario debe proporcionar el número de nodos por nivel, el sistema no solicita el número de niveles del nodo final(conclusiones) ni del nodo inicial(1). Después debe proporcionar al sistema la conclusión para cada nodo final, dicha conclusión debe ser no nula, ya que de serlo el sistema marcará error.

El paso final es el barrido de todo el árbol, para determinar las conexiones y sus valores, por lo que el programa requerirá del usuario, las conexiones con las cuales cuenta cada nodo, comenzando desde el inicial hasta el N-1 (los nodos finales no se conectan hacia adelante con ningún nodo), es decir pregunta el nodo destino y solicita el valor de conexión.

3.2 CONSULTA DE ARCHIVOS

Al activar esta opción el Sistema mostrará una pantalla dividida en una ventana de edición, una ventana de mensajes, una línea de teclas activas y en la parte superior aparece el indicador del nombre del archivo que se está visualizando.

En caso de haber accesado esta opción por error se podrá salir de ella con sólo presionar la tecla <ESC>.

Lo primero que se debe hacer para poder visualizar un archivo es presionar la tecla <F10>, a continuación se debe introducir la ruta en la cual se encuentra el archivo que se desea visualizar, así como el nombre del archivo con su extensión, en caso de cometer algún error y si todavía no se presiona la tecla <ENTER> se podrá utilizar la tecla <BACKSPACE> para borrar el texto erróneo y después escribir el dato correcto, o se puede presionar la tecla <ESC> para regresar al menú principal. Una vez que el dato solicitado es dado correctamente el sistema buscará el archivo, en caso de no encontrarlo emitirá un mensaje de error y solicitará nuevamente la ruta y el nombre del archivo; en este punto el usuario puede introducir nuevamente el dato o presionar <ESC> para regresar al menú principal; en caso de encontrar el archivo este se mostrará en pantalla, y con ayuda de las teclas de manejo del cursor se podrá ir recorriendo dicho texto a lo largo de la pantalla.

En caso de querer salir y regresar al menú principal simplemente se debe presionar la tecla de <ESC>, o si se desea visualizar otro archivo sólo es necesario presionar <F10> e

introducir nuevamente el nombre del archivo como se indicó anteriormente.

3.3 CAMBIAR UNIDAD Y/O DIRECTORIO

Esta opción del menú permite al usuario cambiar la unidad de trabajo actual a una diferente, siempre y cuando ésta esté dada de alta en el sistema. Así mismo el usuario puede cambiar el subdirectorio de trabajo actual a uno diferente dentro de la unidad lógica actual; el subdirectorio deberá existir.

Al accesar esta opción se le presentará al usuario una ventana que estará sobrepuesta a la ventana del menú principal (al igual que esta ventana); esta ventana es accesible por teclado (flechas del cursor) o por el ratón. La ventana presenta dos opciones:

1.- Cambiar Unidad
2.- Cambiar Directorio

para elegir cualquiera de las dos opciones sólo es necesario resaltarla con ayuda del cursor, al estar resaltada de esta forma la opción queda activada y sólo es necesario presionar <ENTER> ó el botón izquierdo del mouse. Al hacer esto, se le presenta una segunda ventana la cual pide el dato correspondiente al usuario.

Si se eligió cambiar unidad, el dato a teclear debe ser la letra de la unidad lógica seguida de dos puntos (:).

Si se eligió cambiar subdirectorio se debe teclear al inicio el Backslash (\) y después el nombre del subdirectorio deseado. Si la ruta del subdirectorio continúa, después de cada nombre se debe de poner el (\) hasta terminar la ruta deseada. Cabe aclarar que deben de ser un máximo de 40 caracteres incluyendo los (\) necesarios.

Si se desea salir de la opción sólo es necesario posicionar el cursor fuera de la ventana y dar un click o un <ENTER>.

Si la unidad ó la ruta de subdirectorio no existen, el sistema da un mensaje al usuario y retorna al menú principal sin cambiar el lugar de trabajo.

3.4 CONSULTAR CONTENIDO DEL DIRECTORIO

Esta opción permite ver en forma total o específica, el contenido del directorio actual de trabajo.

Su manejo es muy simple y sólo es necesario elegir, en una ventana que presenta un menú pop-up con las diferentes tipos de archivos que crea la interfaz, la opción deseada.

Una vez hecho lo anterior se presenta en pantalla el listado por orden alfabético, de los archivos seleccionados.

Si se decide no ver ningún directorio sólo es necesario dar un click fuera de la ventana de menú o elegir el botón de cancelar.

Una vez terminada la consulta de los archivos para cerrar la ventana se puede dar un click fuera de ella o elegir el botón cancelar.

3.5 ELIMINACION DE ARCHIVOS

Esta utilería presenta una pantalla con el encabezado "UTILERIA PARA EL BORRADO DE ARCHIVOS DE TABLAS", la cual cuenta con tres ventanas: de captura de datos, de mensajes y de teclas activas.

El primer dato que se solicita es el nombre del archivo genérico sin incluir la extensión, junto con la ruta completa del mismo o sin ella en caso de que los archivos a borrar se encuentren en el directorio por default.

Se pueden corregir errores con la tecla <BACKSPACE> o <SUPR> antes de oprimir la tecla <ENTER>.

Un vez que el dato es correcto, aparece una ventana de selección, que muestra las extensiones de los archivos que pueden ser borrados (recuérdese que por eso sólo se da el nombre genérico como dato); en ésta ventana se navega a través de ella por medio de las flechas de control del cursor, se ubica el cursor en el archivo deseado y se presiona la tecla <SUPR> para que el archivo quede marcado y se proceda a la búsqueda del mismo dentro del directorio indicado y de ésta forma se borre; en caso de no encontrarse el archivo, el Sistema lo indicará por medio de un mensaje. Esta tarea continúa hasta que se han borrado todos los archivos correspondientes al nombre genérico o hasta que el usuario desee salirse por medio del uso de la tecla <ESC>, que lo regresará a la ventana que permite cargar otro nombre de archivo genérico (con su ruta), si no se quiere borrar otro archivo se da la tecla <ESC> para regresar al menú principal.

3.6 IMPRESION DE ARCHIVOS

Esta opción permite al usuario la impresión de dos tipos de archivos: los de tablas o los archivos de ayuda de un item específico ó toda la ayuda; la selección la hace el usuario desde un menú pup-up.

La primera opción se lleva a cabo como se indica a continuación:

Se presenta una pantalla con el encabezado "UTILERIA PARA LA IMPRESION DE ARCHIVOS DE TABLAS", la cual cuenta con tres ventanas: de captura de datos, de mensajes y de teclas activas.

El primer dato que se solicita es el nombre del archivo genérico sin incluir la extensión, junto con la ruta completa del mismo o sin ella en caso de que los archivos a imprimir se encuentren en el directorio por default.

Se pueden corregir errores antes de oprimir la tecla de <ENTER>, con la ayuda de la tecla <BACKSPACE> ó <SUPR>.

Un vez que el dato es correcto, aparece una ventana de selección, que muestra las extensiones de los archivos que pueden ser impresos (recuérdese que por eso sólo se da el nombre genérico como dato); en ésta ventana se navega a través de ella por medio de las flechas de control del cursor, se ubica el cursor en el archivo deseado y se presiona la tecla <SUPR> para que el archivo quede marcado y se proceda a la búsqueda del mismo dentro del directorio indicado y de ésta forma se imprima; en caso de no encontrarse el archivo, el Sistema lo indicará por medio de un mensaje. Esta tarea continúa hasta que se han impreso todos los archivos correspondientes al nombre genérico o hasta que el usuario desee salirse por medio del uso de la tecla <ESC>.

La segunda opción permite al usuario imprimir todo el archivo de ayuda ó imprimir un archivo determinado.

Para esto se presenta un segundo menú pop-up que presenta en pantalla el nombre de cada uno de los archivos de ayuda presentes en el sistema y además se presenta la opción para imprimir todos los textos de ayuda. Para seleccionar cualquiera de las opciones sólo es necesario resaltar con el cursor la opción deseada y presionar <ENTER> ó el click izquierdo del mouse.

4.- GRAFICACION DEL ARBOL DE CONOCIMIENTOS

Este módulo nos permite obtener el gráfico, en forma de árbol de algún archivo de Base de Conocimientos que se desee, una vez que se han creado sus reglas y sus archivos correspondientes.

Como presentación, se observa una primera pantalla que es la de petición de nombre genérico del archivo que se podrá dar en alguna de estas formas:

- NOMBRE.- Se dará el nombre del archivo sin especificar ruta ni extensión ya que ésta es tomada por default y se busca en la unidad que se está trabajando.

- NOMBRE Y RUTA.- Se da el nombre del archivo especificando la ruta que debe seguir para localizarlo; o sea la unidad y directorio en

la que se encuentra almacenado éste.

- <ESC>.- Esta tecla al ser presionada estando en la primera ventana aborta el proceso de visualizar gráfico y nos regresa al menú principal sin realizar ningún cambio.

Una vez entrando a la función gráfica se visualizará la pantalla que contiene el gráfico del árbol. Este gráfico es de gran utilidad porque en él se pueden observar todas las rutas que se siguen para llegar a las conclusiones. Se recomienda ver el apartado concerniente a Creación de Arboles de Conocimiento.

La pantalla es de uso interactivo, ya que permite varias utilerías que pueden ser seleccionadas por medio del teclado o por medio del Mouse.

En cuanto al uso del Ratón (Mouse), se menciona que en caso de no detectarse dicho periférico conectado a la computadora éste será emulado por el teclado. Para seleccionar alguna tarea con la ayuda del Ratón basta con resaltar o apuntar a la opción deseada y presionar el botón izquierdo del Ratón que es el único botón activo de este.

Cuando se emula el Ratón con el teclado será necesario utilizar las teclas que se describen a continuación :

- FLECHAS DEL CURSOR.- Estas serán las encargadas de desplazar el indicador a través de la pantalla.

- TECLAS "+" y "-".- Mediante estas teclas se podrá incrementar o disminuir el factor de desplazamiento del Ratón.

- INICIO, FIN, AVANZA Y RETROCEDE PAGINA.- Con estas teclas se puede desplazar el cursor de forma diagonal.

- INSERT.- Esta tecla emula la función del botón izquierdo del Ratón.

La primera de estas utilerías es un ícono que permite la consulta de tres tablas creadas con el sistema y que permiten el fácil entendimiento del gráfico. La selección de cualquiera de estas tablas se hace con la ayuda de un menú pop-up. Las tablas son:

Tabla de Conexiones.- permite ver el valor de las conexiones existentes entre los nodos del árbol de conocimientos y que concluyen un objetivo.

Tabla de Niveles.- permite consultar el nombre de cada uno de los niveles de que consta el árbol (atributos).

Tabla de Conclusiones.- permite ver la conclusión de una ruta

en específico.

Cabe aclarar que todas estas tablas se leen con la ayuda de los números que aparecen en los nodos del árbol.

Para seleccionar esta opción se usa el cursor del mouse sobre el ícono o se utiliza la primer letra del nombre seleccionado. Esto es igual para cualquier utilería.

Se tiene una ventana superior derecha que es en donde aparecerán los textos de las tablas. Esta ventana contiene 2 íconos más, una flecha hacia arriba y una hacia abajo, estos íconos se utilizan para recorrer el contenido de la tabla por la ventana. Si se usa teclado se utiliza la letra U para ver los datos anteriores, o la letra D para ver los datos siguientes.

La segunda utilería permite la impresión del gráfico con el fin de que el usuario pueda tenerlo para consultas posteriores. Esta opción detecta automáticamente la habilitación de la impresora, si ésta se halla inactiva se presentará un mensaje de error. En este caso se deberá activar la impresora y presionar <ENTER> si se desea imprimir, si ya no se desea imprimir presionar <ESC>.

La última opción permite la salida del modo gráfico a la pantalla inicial de captura, para solicitar un nuevo nombre genérico; como ya se dijo, si se desea regresar al menú principal presionar <ESC>.

5.- EJECUCION DE LA BASE DE CONOCIMIENTOS

Esta es la parte fundamental de éste software, y es en donde todo el trabajo anteriormente hecho presenta resultados tangibles. En esta parte se corre la mecanismo de Inferencia:

Microexpert

Microexpert interpreta archivos con la extensión .KB - Knowledge Base,(Base de Conocimientos).

Al acceder a ésta opción aparece una pantalla interactiva con el usuario en la cual se pide el nombre del archivo que contiene la Base de Conocimientos, la cual fue previamente creada con la ayuda de la Interfaz en la parte de Crea Arbol. A continuación, la base será analizada con el fin de daterminar si está correctamente escrita. En caso de que haya algún error (de sintaxis) será necesario hacer la corrección de la base, por medio de algún editor de textos que tenga la posibilidad de editar archivos ASCII. Si no hay error, el sistema pregunta por el objetivo de la consulta, que será aquel para lo que fue creada la base. Una vez dado el objetivo podemos entonces utilizar los comandos de Microexpert, o empezar a

responder las preguntas que el mecanismo de inferencia nos irá haciendo, dependiendo de qué 'prompt' se haya capturado al momento de crear la Base. Se recuerda que el prompt es el texto explicativo o pregunta con la cual de infiere sobre la Base de conocimientos. Para comprender esto se recomienda consultar la parte de Sintaxis.

A una pregunta del Mecanismo de Inferencia, podemos dar <ENTER> y nos mostrará en pantalla cuáles son las opciones válidas.

Nuestra respuesta debe ser una opción válida o bien un comando. En caso de que las respuestas del usuario no lleven a ninguna conclusión, el Sistema Experto nos lo indicará.

Se concluye, cuando el sistema muestra todos los hechos que se han validado para llegar a una conclusión.

Si se desea continuar con la consulta de la misma Base o de otra cualquiera para validar otro objetivo, sólo es necesario presionar <RETORNO> y dar el nombre de la Base deseada.

Para terminar la sesión, dar <QUIT>.

BIBLIOGRAFIA

- Sholom Weiss and Casimir Kulikowski
  A Practical Guide to Design Expert System
  Publicación: New Jersey, U.S.A., 1984
  Editorial: Publishers Rowman & Allanheld

- Daniel V. Hunt
  Artificiall Inteligence &
  Expert System Source Book
  Publicación: Londres, Inglaterra, 1986
  Editorial: Chapman & Hall

- Dieter Nebendhal
  Sistemas Expertos
  Introducción a la Técnica y Aplicación
  Publicación: Barcelona, España 1988
  Editorial: Marcombo Boixareu

- John G. Burch, Gary Grudnitski
  Diseño de Sistemas de Información
  Publicación: Mexico, D.F. 1992
  Editorial: Grupo Noriega

- José P. Sánchez y Beltrán
  Sistemas Expertos una Metodología de Programación
  Publicación: México, D.F. 1990
  Editorial: Macrobit

- Joseph Dumas
  Designing User Interfaces for Software
  Publicación: New York, U.S.A. 1988
  Editorial: Prentice - Hall

- L. Galbraith, M. Al-Najjar, M. Babu
  Expert Systems in Engineering
  Publicación: 1988
  Editorial: IEEE-AES Magazine

- Roger S. Pressman
  Ingeniería de Software
  Publicación: México, D. F. 1988
  Editorial: McGraw Hill

- Edward Yourdon
  Modern Structured Analysis
  Publicación: México, D. F. 1989
  Editorial: Yourdon Press

- Richard Forsyth
Expert Systems: Principles and Case Studies
Publicación: New York, U. S. A. 1984
Editorial: London Press

- M. Hayes, R. Waterman, D. Lenat
Building Expert Systems
Publicación: New York, U. S. A. 1983
Edotorial: Adison Wesley

- Raymundo Rangel Gutierrez
Apuntes de Programación Estructurada
Publicación: México, D. F. 1985
Editorial: U. N. A. M. - Facultad de Ingeniería

- Borland International Inc.
Turbo C Reference Guide
Publicación: U. S. A. 1988
Editorial: Borland International Inc.

- Stephen Kippur
Graphics programming using Turbo C
Publicación: U. S. A. 1989
Editorial: John Wiley & Sons, Inc.

- Kernigan / Ritchie
El lenguaje de Programación C
Publicación: México, D. F. 1986
Editorial: Prentice-Hall

- Herbert Schildt
Programación en Turbo C
Publicación: Madrid, España
Editorial: McGraw-Hill / Interamericana de España S. A.