SISTEMA EXPERTO PARA EL DIAGNOSTICO DE FALLAS

Autores: *Alejandro Loccicero, Guillermo Vázquez, José Aronson*
Coordinación y dirección: *Jorge Pluss, Manuel Molina*
Universidad Nacional de Rosario
Rosario - *Argentina*

RESUMEN

El SEDIF (Sistema Experto para el Diagnóstico de Fallas) es un Programa Inteligente que permite, entre otras cosas, detectar fallas en Sistemas Electrónicos mediante el suministro de datos relevantes sobre los síntomas que presentan cuando se observa un mal funcionamiento en los mismos.

La característica principal de este Sistema radica en el hecho de que su comportamiento se asemeja al de un experto humano en diagnóstico de fallas, logrado con el empleo de técnicas de la Inteligencia Artificial (IA), una disciplina formal de la Informática.

Así mismo se describe el diseño de un prototipo sobre el cual se realiza la prueba del Sistema Experto (SE), este es un Controlador de Acceso a un Recinto (CAR).

INTRODUCCION

El problema a resolver consiste en diseñar un Sistema Computable que manifieste las siguientes características:

1- Toma de Decisiones
Deberá llevar el control permanente durante la búsqueda de soluciones a través de su Sistema Ejecutivo, interrogando al usuario cuando no pueda continuar con los datos suministrados inicialmente, minimizando en lo posible el número de consultas a realizar al mismo, apelando a técnicas de control apropiadas. Esto es así puesto que entendemos que el máximo esfuerzo lo debe realizar SEDIF, lo que tiene más sentido si pensamos que el operador no conoce absolutamente nada sobre Electrónica.

2- Interacción Usuario-Sistema Clara y Simple
La forma en que presenta los resultados debe ser explícita, de manera que cualquier persona pueda interpretar a SEDIF. Asimismo, las respuestas que el usuario debe darle tendrán un formato sencillo.

3- Capacidad Explicativa
Será necesario proveer un mecanismo por el cual pueda dar explicaciones sobre los motivos que lo llevan a tomar una determinada decisión, es decir como hizo para inferir una

solución a partir del suministro de los síntomas manifiestos por el circuito.

4- **Posibilidad de Aprendizaje**
Puede ocurrir que la información suministrada en el momento del diseño, acerca del dominio en el que se debe manejar sea errónea ó incompleta, lo que será detectado cuando pretenda localizar una falla y no pueda lograrlo. En base a esta posibilidad deseamos que el Sistema pueda actualizar su conocimiento en forma permanente (Conocimiento Incremental).

5- **Adaptabilidad a diferentes circuitos**
Pretendemos dotar a SEDIF de una estructura que posibilite su aplicación sobre diversos prototipos, sin orientarlo a un circuito en particular. Esto implica que la misma deberá ser de características modulares, de modo que reemplazando un módulo de información por otro, pueda utilizarse el mismo Sistema Ejecutivo.

En general, lo que estamos requiriendo del Sistema Experto para el Diagnóstico de Fallas es que se comporte en una forma aproximada a la que manifiesta un Especislista Humano en Fallas.

Para otorgarle al Sistema los cinco atributos señalados, creamos tres módulos individuales que constituyen el Sistema Ejecutivo de SEDIF.

a) **Módulo de Diagnóstico.**

b) **Módulo de Enseñanza.**

c) **Módulo de Aprendizaje.**

La Figura 1 muestra el comportamiento elemental de cada Módulo.

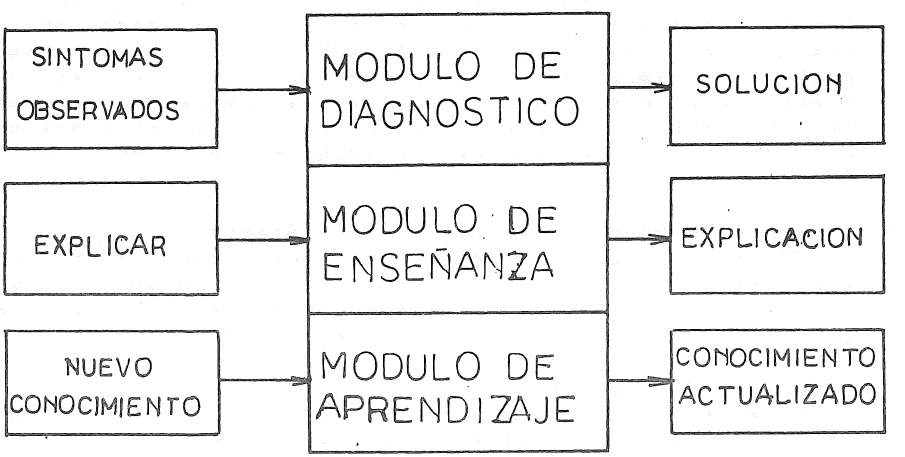


Figura 1 SISTEMA EJECUTIVO DEL SEDIF.

QUE SON LOS SISTEMAS EXPERTOS ?

Formalmente podríamos aceptar la siguiente definición de Sistema Experto:

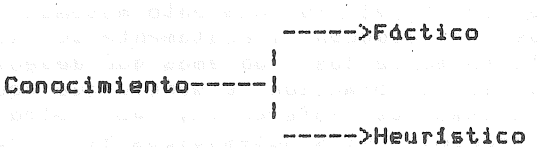
> "Se dice que un Programa de Cómputos es un Sistema Experto si al realizar una determinada tarea compleja , lo hace con tanta capacidad y eficiencia que nos parecería resuelto por un Experto Humano en esa materia."

Si analizamos rapidamente las cualidades pretendidas para SEDIF y observamos esta última definición, inferimos que el mismo no va a ser otra cosa que un Sistema Experto (en este caso el dominio de especialización será el Diagnóstico de Fallas en Circuitos Electrónicos).

Ahora bien, para simular el comportamiento de un especialista debemos hallar un modelo, aunque no sea más que una aproximación, de la estructura cognitiva del mismo, y a partir de este podremos visualizar como realiza sus actividades mentales. Escogido el modelo, el próximo paso consistirá en determinar cuales serán las herramientas computables más confiables para implementarlo. Pasemos a resolver el primer paso del problema planteado.

MODELO DEL ESPECIALISTA

Podemos realizar una división general de los Tipos de Conocimientos que normalmente maneja cualquier experto, tal como lo expresa el siguiente esquema:

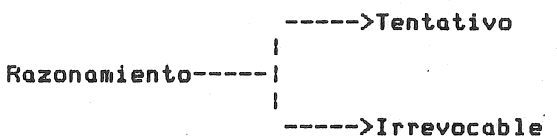


a) **Conocimiento Fáctico**
Es llamado habitualmante "Conocimiento Enciclopedista" ó de "libro de texto", necesita poca elaboración puesto que es el más obvio. Se puede expresar y representar con facilidad en una Computadora.
E.j. "Si la impedancia que existe entre el colector y emisor de un transistor es nula, entonces no funcionará correctamente."

b) **Conocimiento Heurístico**
Es más difícil de archivar en una Computadora, es la red de intuiciones, reglas de juicio, "teorías de entrecasa", experiencia y en general procedimientos de inferencia que en combinación con el Conocimiento Fáctico sobre un determinado tema, le permite al hombre exhibir comportamiento inteligente.
La heurística es una propiedad intrínseca que posibilita dirigir el pensamiento a lo largo de las rutas que más verosimilmente conducen a la meta, dejando sin explorar (en los mejores casos) las avenidas menos prometedoras cuando analiza situaciones donde el número de

alternativas a considerar para llegar a la solución es significativo. Cualquier Sistema que se intente sea un recurso sofisticado de Información Inteligente debe, de algún modo, incorporar este nivel superior de conocimiento.

Analicemos como razona un especialista, ó sea que métodos emplea para inferir información nueva a partir de la existente. Tambien los tipos de razonamiento en términos generales pueden ser dos (2) tal como se expresa a continuación:



a) **Razonamiento Tentativo ó no Monótono**
Es el tipo de razonamiento que más comunmente emplea un especialista al postular afirmaciones plausibles en algún momento, pudiendo retractarse de ellas regresando al punto de decisión original. Esto ocurre generalmente cuando toda la información disponible en un dado instante no es suficiente para inferir nuevos resultados con absoluta certeza.

b) **Razonamiento Irrevocable ó Monótono**
Tiene que ver con el razonamiento matemático formal, los resultados se deducen directamente de la información previa; la forma de los problemas que deseamos abordar no presentan la información de manera que podamos escoger este mecanismo de inferencia, por otra parte donde existen procedimientos matemáticos bien definidos para resolver un problema los Sistemas Expertos son innecesarios.

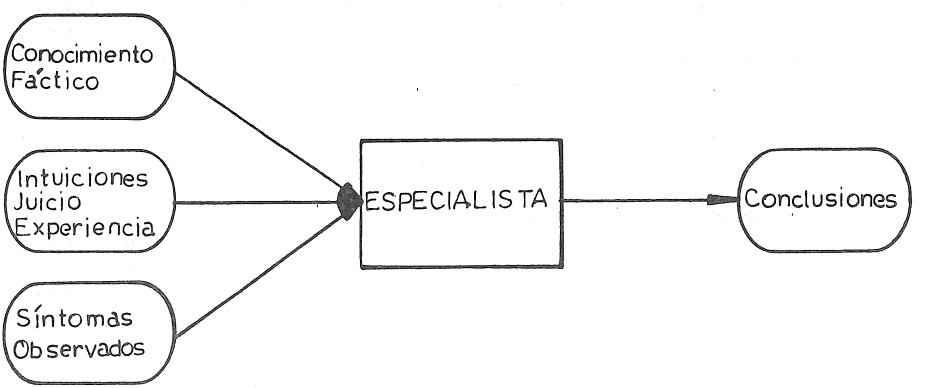


Figura 2 MODELO DEL ESPECIALISTA

La Figura 2 muestra el modelo escogido para representar globalmente la estructura cognitiva de un especialista en fallas,

quien a través de su Conocimiento Fáctico y Heurístico, de los Síntomas observados y con un mecanismo de inferencia adecuado, alcanza las conclusiones.

Un experto en fallas basicamente intentará reconocer las causas que originan el mal funcionamiento del Sistema Electrónico bajo análisis partiendo de las evidencias observadas, conociendo a priori cual es el funcionamiento correcto, aplicando eventualmente los criterios derivados del Conocimiento Heurístico.

REPRESENTACION DEL CONOCIMIENTO. RESOLUCION DE PROBLEMAS.

La situación planteada puede representarse como el intento de arribar a un **Estado Objetivo**, que es la solución del problema, a partir de un **Estado Inicial** al que podemos asociar con las evidencias manifiestas por el circuito (síntomas).

Consideremos que a partir del estado inicial pueden generarse nuevos estados aplicando las reglas de inferencia lícitas definidas por el problema; representémoslo mediante un estructura tipo árbol, donde se asocia a cada estado con un nodo del árbol y a las transiciones entre estados con las ramas del mismo tal como lo indica la Figura 3. (hemos escogido para SEDIF una de las estructuras posibles para representar el Conocimiento Fáctico, suelen ser útiles tambien los Frames y las Redes Semánticas.)

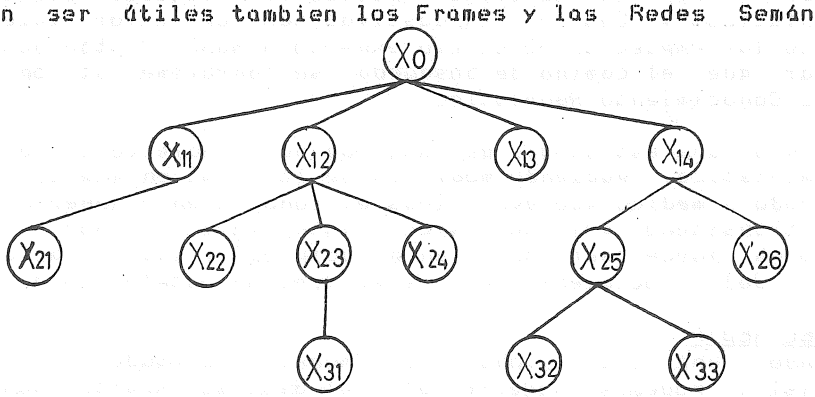


Figura 3 REPRESENTACION DEL CONOCIMIENTO FACTICO

El nodo indicado con X0 es llamado raíz del árbol y pertenece al nivel de decisión cero (0). A partir de este, se derivan X11, X12, X13 y X14 pertenecientes al nivel uno (1) y así, en forma sucesiva los nodos Xnm pertenecientes al nivel N. Tambien suele denominarse a la estructura anterior, "Espacio de Búsqueda" a partir de X0 aplicando la totalidad de las reglas permitidas. En este ejemplo; X13, X21, X22, X24, X26, X31, X32, y X33 son **nodos terminales**, o sea, aquellos en los cuales no pueden derivarse otros nuevos.

Se denomina **Estrategia de Búsqueda** a las técnicas empleadas por un especialista cuando, a partir de una situación inicial,

intenta arribar a la solución de un problema planteado. Este proceso tiene que ver directamente con la forma en que se irá moviendo a lo largo del árbol durante la búsqueda. Dada la naturaleza de los problemas que deberá resolver SEDIF, donde el estado inicial se conoce pero no así el estado final (solución), un mecanismo de preguntas y respuestas será el responsable de orientar al usuario hacia la meta. En general existe más de una alternativa para dirigirse a la solución a partir de los síntomas observados, en otras palabras hay varios caminos entre el nodo inicial y el nodo objetivo.

El **Camino Optimo** entre un nodo inicial y uno objetivo es aquel que contiene la menor cantidad de nodos intermedios, es el camino más directo ente los mismos.

En un espacio de búsqueda existen nodos que pertenecen al camino óptimo y nodos que no pertenecen; el propósito de las técnicas de búsqueda razonables es minimizar el número de este tipo de nodos a explorar para reducir al máximo el gasto empleado en arribar a la solución.

Existen técnicas o estrategias de búsqueda **exhaustivas** y no **exhaustivas**; las primeras, en general no apelan al conocimiento heurístico y se los denomina "Metodos de Búsqueda Ciega" ó "Deterministicos". Las no exhaustivas son de mayor interés, puesto que las emplea un verdadero experto humano; están basados en lograr que el camino de búsqueda se aproxime al óptimo, empleando Conocimiento Heurístico.

En general una estrategia de búsqueda no exhaustiva es de tipo no determinística, pudiendo modificarse la forma en que un árbol es explorado a medida que evoluciona el conocimiento general -el camino transitado por un especialista para arribar a sus conclusiones puede ser uno en un principio, y luego gracias al incremento del conocimiento (E.j. experiencia) adoptar otro-.

MODELO DEL SEDIF

Teniendo una idea global del modelo adoptado para el especialista, podémos sugerir el del Sistema Experto que se comporta en forma aproximada a como lo hace aquél. En ese sentido nos hemos apoyado en los denominados **Sistemas Basados en Reglas** quienes separan los dos componentes principales de la inteligencia: Razonamiento y Conocimiento. En principio esta no es una división rígida, puesto que los procesos de razonamiento se apoyan en la mayoría de los casos en el conocimiento, aunque aquí tenemos por un lado al mecanismo general de razonamiento ó motor de inferencias, tambien llamado el Intérprete del Sistema Inteligente. Por otro lado estará el conocimiento fáctico, conformando la **Base de Conocimiento**, la red de reglas que describen (en este caso) el funcionamiento del Circuito bajo Diagnóstico.

El intérprete manipulará el Conocimiento Heurístico, y será quien fije la estrategia de búsqueda a emplear.

La Figura 4 describe el modelo propuesto para SEDIF, quien en el modo de consulta recogerá los síntomas observados del circuito bajo ensayo a través del usuario y comenzará la búsqueda controlada por el mecanismo general de razonamiento apelando a los tipos de conocimiento vistos.

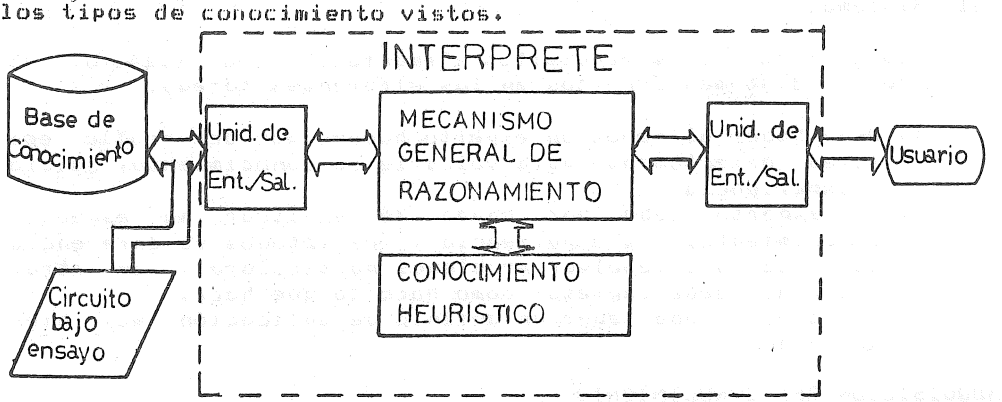


Figura 4 MODELO DEL SEDIF

Las unidades de Entrada-Salida del intérprete le permiten comunicarse con los componentes internos del SEDIF, así como con el operador.

Analicemos las ventajas de este modelo frente al de los programas tradicionales. La principal diferencia (que ya podemos visualizar) radica en la clara separación ente el Conocimiento Específico y el Mecanismo General de Razonamiento. Esta división, junto a la partición adicional del conocimiento general en reglas separadas, ofrece ventajas significativas que pasamos a enumerar:

1) Permite que la Base de Conocimiento pueda desarrollarse incrementalmente a lo largo del tiempo mediante un mejoramiento de las reglas presentes y el agregado de otras nuevas.
2) Un mismo Intérprete puede ser usado para aplicaciones diversas, mediante el recurso de cambiar un "juego" de reglas por otro.
3) Ofrece la posibilidad de que un mismo "paquete" de conocimiento pueda ser utilizado de modos diversos (incluyendo aplicaciones pedagógicos en enseñanza) adecuando el diseño del Sistema Experto. Digamos que esto último puede ser logrado con relativa facilidad tal como ocurre en el desarrollo del SEDIF.
4) El programa puede dar claras y simples explicaciones de su comportamiento describiendo las reglas aplicadas en la última sesión. Este recurso resulta de gran utilidad para encontrar reglas erróneas.
5) La posibilidad de desarrollar Sistemas que sean Instrospectivos (que puedan, por ejemplo, chequear la consisitencia de sus propias reglas para evitar contradicciones), y Evolutivos (que puedan modificar

automaticamente sus reglas, y aún aprender otras nuevas).

La **Modularidad** ofrecida por el modelo propuesto permite realizar con muy poco esfuerzo, el mantenimiento y actualización del Sistema.

Podemos citar tres requisitos elementales para asegurar el éxito de los Sistemas Expertos en las diferentes tareas:

1) Debe haber al menos un experto humano que desarrolle esa tarea correctamente para volcarle su conocimiento, juicio y experiencia.
2) El experto debe ser capáz de explicar su especial conocimiento, su experiencia y los métodos de inferencia que usa para resolver problemas particulares, en otras palabras, debe expresar como hace lo que hace.
3) La tarea debe tener un dominio de aplicación muy bien definido.

ADQUISICION DEL CONOCIMIENTO

Uno de los "cuellos de botella" que afronta la construcción de Sistemas Expertos es el elevado tiempo que se necesita para interrogar a los especialistas y representar su conocimiento especial en forma de reglas (como ocurre en nuestro caso). Este es el problema de la adquisición del conocimiento; más precisamente, la Base de Conocimiento se construye usando los mecanismos provistos por la **Ingeniería del Conocimiento,** basado en el diálogo entre el profesional y el experto en un dado campo. Las etapas involucradas en este proceso son dos (2).

ETAPA 1

a) Identificación de los conceptos y características claves del problema.
b) Conceptualización: clasificación de los conceptos y sus relaciones con el terreno del conocimiento.

ETAPA 2

c) Formalización: elección de una estructura adecuada para representar el conocimiento y el método de inferencia a ser usado.
d) Implementación: Formalización de todas las reglas específicas y de la heurística que conduce a la solución del problema.
e) Validación de las reglas y heurísticos a ser implementados.

A pesar de los esfuerzos realizados, la implementación de la adquisición del conocimiento sigue siendo uno de los problemas más difíciles. El desarrollo de un Sistema que contenga algunos pocos cientos de reglas suele llevar varios meses de trabajo a los expertos, y aún más al constructor del programa.

ANALISIS DEL CIRCUITO BAJO ENSAYO

A continuación describiremos el comportamiento del circuito que se diseñó a fin de ser utilizado como ejemplo de test para

analizar la performace de SEDIF.

Lo denominamos Controlador de Acceso a un Recinto (CAR), y su función es custodiar el ingreso en un sitio cualquiera que deba estar protegido, permitiendo solo el acceso a aquellas personas que dispongan una tarjeta con un código autorizado, e impidiéndolo a las que no lo posean. Además el CAR brinda la posibilidad de grabar ó borrar códigos que se pretendan memorizar ó eliminar respectivamente.

CAR consta de tres (3) entradas:

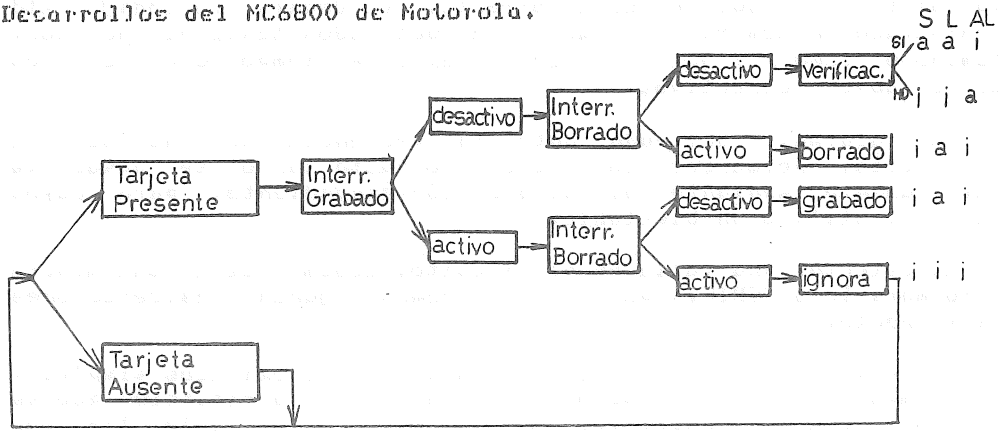
Unidad lectora de códigos.
Pulsador de borrado.
Pulsador de apagado.

y de tres (3) salidas:

Indicador luminoso.
Solenoide electromagnético.
Alarma.

En la Figura 5 podemos visualizar esquematicamente el comportamiento de este dispositivo. Observamos que si algún intruso pretende ingresar con un código no registrado en la memoria, el CAR no la verificará accionando la alarma. Cabe destacar que si por error se accionaran simultaneamente los interruptores de grabado y borrado, el Sistema ignorará la órden hasta que alguno sea desactivado.

A través de la Figura 6 apreciamos el esquema circuital del dispositivo. El mismo fué implementado en Base al Kit de Desarrollos del MC6800 de Motorola.



S : Solenoide
L : Luz
AL: Alarma

a: activo
i: inactivo

Figura 5 DESCRIPCION FUNCIONAL DEL CAR

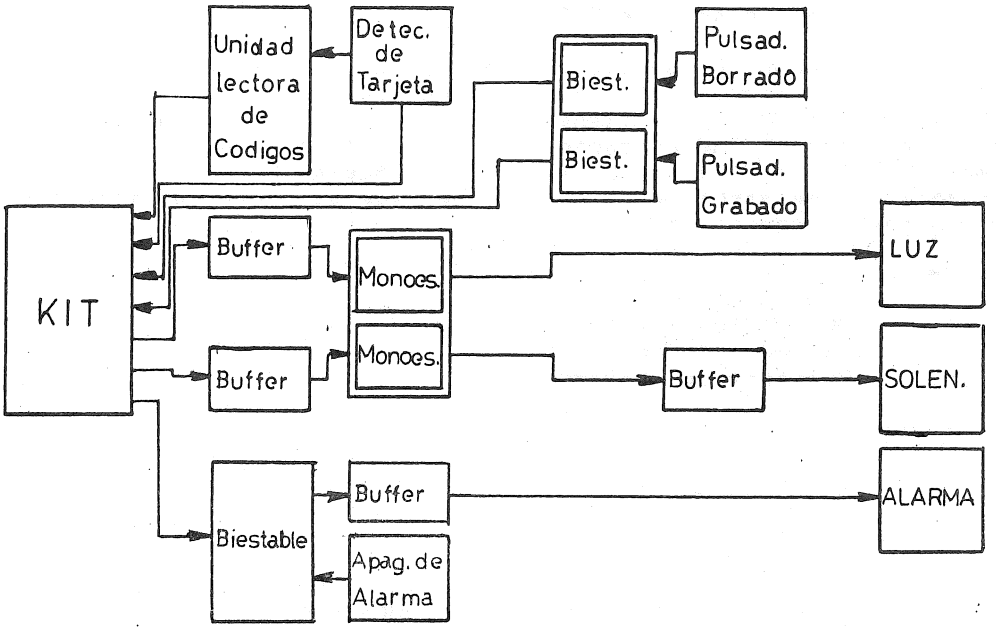


Figura 6 ANALISIS DEL CIRCUITO EN BLOQUES

Cuando se introduce una tarjeta, su presencia es acusada por un detector, el cual se encarga de informarle esta situación al Microprocesador y energizar la Unidad lectora de códigos. Los dos pulsadores envían información a través de sendos biestables que evitan aleatorios por rebotes en los contactos de los mismos. Si la tarjeta introducida fuera correcta, es decir, su código está registrado en memoria, permanecerán activados tanto el indicador luminoso como el solenide por un lapso de tiempo dado por los monoestables respectivos.

La alarma, activada ante la presencia de una tarjeta incorrecta, permanece en ese estado hasta tanto se desactive externamente mediante el pulsador correspondiente, para cuyos fines se utilizó un biestable.

Al detectar una falla en el circuito, debemos de alguna manera informárselo a SEDIF, por lo que haremos algunas consideraciones al respecto.

En general habrá dos tipos de tests, los denominados simples y los compuestos. Los primeros están dados por la situación que se presenta cuando analizamos las salidas con un solo tipo de tarjeta (correctas e incorectas en forma separada). Por ejemplo, verificar el estado de la alarma con una tarjeta correcta. Los tests compuestos se conforman analizando tanto la situación de las salidas para una tarjeta correcta como para una tarjeta incorrecta.

Cada una de esas situaciones representará un síntoma posible, los que a su vez se convertirán en los nodos iniciales ó raices de los "subárboles" que conforman la Base de Conocimiento.

REPRESENTACION DEL CONOCIMIENTO

Veamos sintéticamente como representar una "porción" de conocimiento del CAR. Como se observa en la Figura 7, es una Estructura tipo árbol donde un síntoma está expresado en términos de la raíz, y cuyos nodos teminales serán los posibles resultados del mal funcionamiento del Circuito que manifiesta ese síntoma. Si uno lo desea, puede seguir derivando estos últimos hasta resolver al nivel de definición que se pretenda (bloque ó componenete). En paricular, el árbol tomado a título de ejemplo representa lo que acontece en un bloque de la interfase de salida pero no resuelve un inconveniente que pueda presentarse en el Kit, indicado en este caso como el bloque "logica", lo que no significa que el SEDIF no puede estar capacitado para diagnosticar inconvenientes en el mismo; todo lo contrario, el usuario puede "colgar" nuevos subárboles ó ramas del nodo en cuestión y resolver fallas en los componentes del Kit. Esto es importantísimo ya que le permite al SEDIF poseer una propiedad natural del hombre , el **Conocimiento Incremental**, adquiriendo más información sin eliminar la existente.

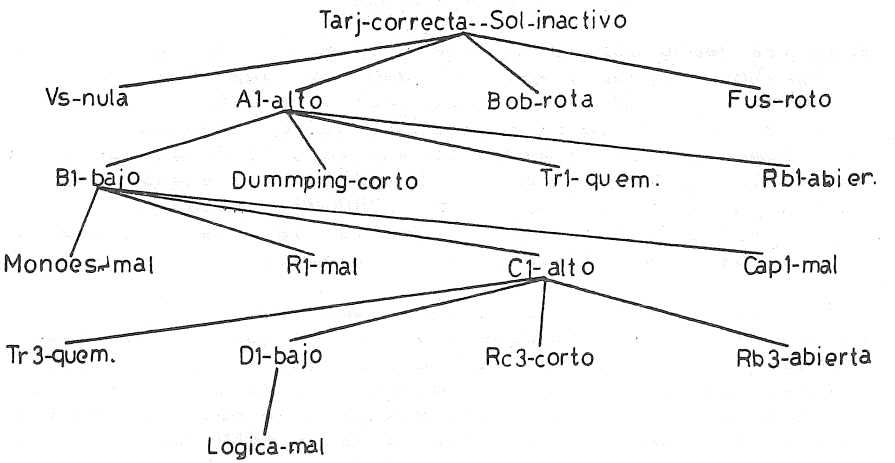


Figura 7 ESPACIO DE BUSQUEDA

Los nodos del árbol en este caso son **Nodos OR**, es decir que existe independencia entre los estados de un mismo nivel de decisión, lo que en algunas ocasiones puede no ser así. En estos casos la solución está en incorporar a ese nivel nuevas ramas que contemplen tales posibilidades. La Base de Conocimineto del SEDIF contiene 69 Reglas.

LA NECESIDAD DE LOS LENGUAJES DECLARATIVOS

Escoger el lenguaje que permita implementar cualquier proceso en un Computadora, significa fundamentalmente analizar la

adaptabilidad de las Estructuras de Control y de Datos, ofrecidas por las versiones disponibles, para ejecutar y representar dicho proceso. En este caso buscamos codificar el modelo propuesto para el Sistema Experto representado en la Figura 4.

Si analizamos las estructuras de control de los diferentes lenguajes, nos encontraremos con dos grupos generales bien definidos. Por un lado los que se basan en el modelo de la máquina propuesta por Von Neumman, donde un programa es una colección de órdenes ó instrucciones a ejecutar en forma secuencial, salvo excepciones, tal como lo hace la Unidad Central de Procesamiento quien ejecuta el trabajo al más bajo nivel. Si bien, gracias a los compiladores la programación en lenguajes de alto nivel no se torna tediosa, desde el punto de vista de la "abstracción" no hemos avanzado demasiado ya que permanece intacta la idea "procedimental" de Von Neumman; por esta razón a los lenguajes que se ajustan a este modelo rígido se los denomina **Lenguajes de Procedimiento** (Procedural Languages) ó **Imperativos**. BASIC y FORTRAN integran este grupo. Al ser la Estructura de Control bastante rígida, vinculada a la forma en que operan las partes de una Computadora, cuando queremos codificar procesos que no se ajustan a este modelo, ya que se hace necesario plantear uno abstracto, (como ocurre con los procedimientos inteligentes) resulta muy difícil ó casi imposible hacerlo.

Los **Lenguajes Declarativos** en cambio intentan "declarar" que es lo que se pretende hacer más que como hay que hacerlo, la estructura semántica de los mismos permite representar un modelo abstracto (en el sentido de la relación ente los objetos de ese modelo y los elementos que normalmente manipula la Computadora) y es precisamente la "interfase" entre el modelo y la máquina. Los más difundidos en este grupo son LISP y PROLOG, basados en dos disciplinas formales como son la **Teoría de las funciones recursivas** y la **Lógica matemática** respectivamente.

Cuando se analiza el tipo de datos a emplear para representar la información de cualquier procedimiento computable, se busca (en lo posible), estrechar las diferencias que puedan existir entre la forma adoptada por la representación natural (la que manipula el hombre) y la forma representativa de esa información que debe manejar la Computadora. Esto es así puesto que conviene que el programador dedique su mayor esfuerzo a interpretar el funcionamiento general del Sistema sin entrar en los detalles que no hacen la comportamiento mismo de este (no por ello menos trascendentes en el diseño de un programa); esto, naturalmente tiene mayor sentido en los problemas complejos.

Los elementos que deberá manejar SEDIF, más que números, serán símbolos, por lo que los datos tendrán que ser de características simbólicas, en particular nos interesarán aquellos que permitan representar las reglas del tipo SI-ENTONCES. Por ejemplo:

> "Si el punto A1 del circuito permanece en nivel lógico alto, ó la tensión Vs es nula, ó la bobina está rota, ó el fusible está cortado, **entonces** al colocar una tarjeta

correcta el solenoide no actuará."

El texto anterior expresa una porción de conocimiento de SEDIF sobre el CAR, es decir es una parte de la Base de Conocimiento. Podemos decir lo mismo de la siguiente forma:

(tarj-correcta--solenoide-inactivo (Vs-nula Bob-rota Fus-cort A1-alto))

La regla ha sido expresada en términos de una lista cuyo primer elemento (conclusión de la regla) es un átomo y el segundo (las premisas) es otra lista de átomos. (Un átomo es una cadena de símbolos alfanuméricos y/o ciertos caracteres especiales). Observamos que el implicante lógico y la disjunción lógica están implícitos en la estructura adoptada. Precisamos un formalismo que permita tratar con este tipo de datos, es decir manejar listas. **LISP(LISt Programming)** ofrece recursos poderosos para esto último, es el lenguaje más difundido en el ámbito de la IA y pertenece al grupo de los **Lenguajes Funcionales**, en los que la programación consiste en definir funciones. (los lenguajes funcionales integran un grupo más general, el ya comentado, los lenguajes declarativos.

Una de las razones principales que nos impulsaron a elegir LISP, se basó en el análisis de la Estructura de Datos necesaria para almacenar el conocimiento del Sistema Experto ó sea representar la Base de Conocimiento en alguna forma que se facilite su creación, acceso y mantenimiento. La versión de LISP adoptada, permite asociarle a cualquier variable una **lista de propiedades**, ó sea que cada variable podrá tener asignado un conjunto de propiedades así como una persona tiene determinados atributos que lo identifican (sexo, número de documento, edad y otros).

La forma que toma la estructura mencionada es la siguiente:

	PROPIEDAD 1	PROPIEDAD 2	- - - - - - -	PROPIEDAD N
VARIABLE	Valor1	Valor2	- - - - - - -	ValorN

	CONCLUSION	PREMISAS
REGLA 1	tarjeta-correcta--sol-inactivo	(Vs-nula A1-alto Bob-rota Fus-roto)

Figura 8 LISTAS DE PROPIEDADES

En la Figura 8 se muestra tambien, la forma adoptada para una regla, observamos que los valores que pueden tener asignados para

las diferentes propiedades serán, ó bien átomos, ó bien listas. Un atributo saliente de LISP es que absorve la recursión, una propiedad matemática que les permite a las funciones definirlas en términos de ellas mismas. Ej (potencia, factorial).

$$M^n = M * M^{(n-1)}$$

$$N! = N * (N - 1)!$$

La utilidad que le reporta la recursión a LISP está manifiesta en que el concepto matemático se extiende al simbólico, permitiéndole el uso de ella para hacer lo que mejor conoce LISP, manejar listas. Una persona sabrá programar en LISP cuando domine este simple y poderoso recurso.

Una gran cantidad de fenómenos naturales, incluso algunos procedimientos mentales son de naturaleza recursiva, lo que nos permite codificarlos sin necesidad de simularlos. En conclusión, analizando la Estructura de Control y de Datos, el manejo simbólico y la recursividad que ofrece LISP, decimos que este es un lenguaje adecuado para implementar el Sistema de las características ya señaladas.

DESCRIPCION FUNCIONAL DEL INTERPRETE

Al observar un funcionamiento incorrecto en el circuito llamamos a SEDIF y le indicamos cuáles son los síntomas observados. Por ejemplo.

```
tarj-correcta--sal-inactivo
```

A partir de este punto comienza a explorar el árbol de la Figura 7, y el objetivo es verificar si el problema está en alguno de los elementos de la lista **objetivos**, que en este punto valdrá:

```
objetivos = (A1-alto Vs-nula Bob-rota Fus-roto)
```

Digamos que la forma en que el intérprete explora el árbol dependerá de los **factores de decisión** asociados a cada nodo, teniendo mayor prioridad los de valores numéricos superiores. Este factor de decisión (D), depende directamente del número de veces que exploró dicho nodo en el pasado (se pondera la experiencia). Por otro lado será inversamente proporcional al nivel de profundidad (P) asociado a dicho nodo, el que se toma como el número de nodos, a partir de él, a recorrer para llegar al nodo terminal más alejado. Un nodo será el terminal más alejado de otro, si el primero cumple con estas condiciones: ó es el segundo tomado como referencia, ó se deriva del segundo, ó se deriva de otro que se deriva del segundo (notar el procedimiento recursivo de la última condición).

El último criterio permite realizar eventualmente la " vuelta hacia atrás" más corta cuando el camino escogido no conduce a la solución; es el "por las dudas". Veamos los niveles de profundidad asociados a cada elemento de la lista objetivos.

Nodo	Profundidad (P)
A1-alto	5
Vs-nula	1
Bob-rota	1
Fus-roto	1

Si pensamos que el número de veces que se transitó por esos nodos en el pasado es el mismo, convendrá escoger un terminal y nó el no terminal ya que si así no fuera (y la solución fuese alguno de los terminales), deberíamos volver hacia atrás con el consiguiente costo asociado. En esta situación SEDIF elegirá uno terminal; cuando los factores de decisión de dos ó más nodos sean iguales tendrán mayor prioridad los que se encuentren mas a la izquierda en la lista objetivos.

El criterio de búsqueda adoptado, si bien está relacionado con las ideas generales que emplea un experto, no deja de ser empírico; la forma de validar a los heurísticos es mediante la prueba misma del Sistema analizando si los resultados arrojados son satifactorios, obviamente no existe un "paquete estandarizado" de heurísticos que se adapte a cualquier clase de problemas.

El hecho de que la estrategia escogida sea tentativa implica la necesidad de dotar al intérprete de un mecanismo apropiado para regresar al punto original si la solución no está en el camino elegido en primer término, dicho mecanismo es el **BackTracking** y es muy común en los Sistemas basados en IA. El diseño del intérprete consistió en definir 40 funciones relacionadas como lo muestra la figura 9.

El intéprete además tiene en cuenta situaciones particulares como ser los problemas en el correcto suministro de energía, ó de interpretación errónea por parte del usuario del tipo de tarjeta que está utilizando el el test; en cuyos casos intenta en primer término verificar este tipo de problemas graves ó groseros y que están más a la vista. Esto se condice con el comportamiento del especialista cuando dá un "vistazo" al circuito bajo análisis, para verificar en función de ciertas evidencias si el problema puede detectarse a simple vista, antes de comenzar la búsqueda "fina" ó más detallada.

Volviendo al principio de este apartado, cuando el intérprete encuentra que el nodo cuyo factor de decisión más grande corresponde a uno terminal, busca el resto de los nodos terminales que pertenecen al mismo nivel preguntando (en el caso de que queden en la lista objetivos otros elementos) si alguno de ellos es la solución ó infiriéndola en el caso de que sean los únicos elementos de objetivos. Supongamos que el nodo de mayor factor de decisión es A1-alto, entonces debe seguir explorando el árbol hacia abajo derivando nuevos nodos, quedando objetivos de la siguiente forma:

objetivos = (Tr1-quem Dummping-corto B1-bajo Rb1-abier (Vs-nula Bob-rota Fus-roto))

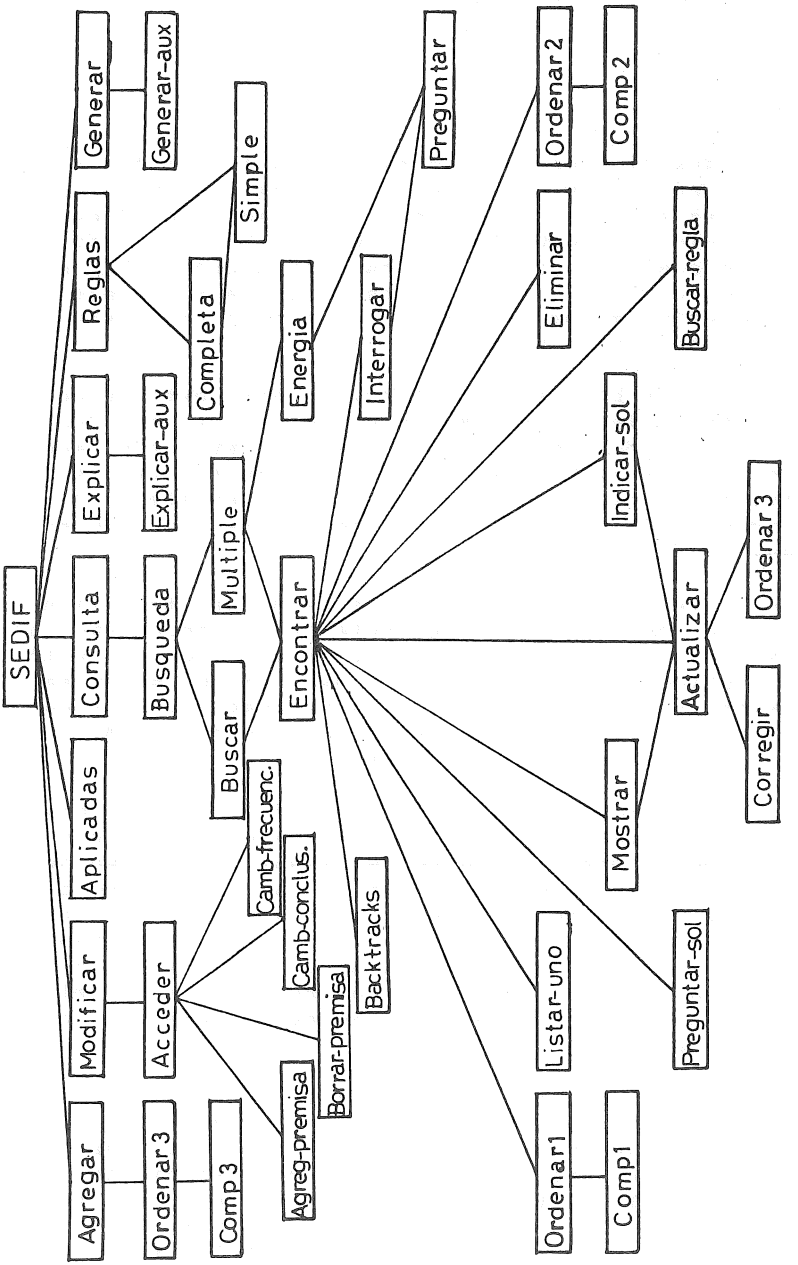


Figura 9 FUNCIONES DEL SEDIF

Los nodos que serán analizados en última instancia quedarán en las posiciones más profundas de la lista. Si el mayor factor de decisión fuese ahora (para los nodos menos profundos) el correspondiente a Tr1-quem el intérprete preguntará:

(VERFIQUE SI ALGUNA DE ESTAS SITUACIONES SE PRESENTA 'SI/NO')
(TR1-QUEM DUMMPING-CORTO RB1-ABIER)

A continuación, el usuario, haciendo uso de los elementos apropiados (tester, punta lógica, osciloscopio, etc.) podrá responder al interrogante planteado. Supongamos que la respuesta sea NO, luego objetivos pasa a valer:

objetivos = (C1-alto Monoes-mal R1-o-Cap1-mal (() (Vs-nula Bob-rota Fus-roto))

La presencia de la lista vacía en el segundo nivel, expresa que en el mismo nivel del árbol no quedan alternativas por explorar. El proceso de agregar nuevos elementos en la lista objetivos continúa hasta que, ó bien el usuario responde afirmativamente una pregunta indicando cual, de la lista que eventualmente indique SEDIF, es la solución del problema, ó bien llega a un nodo de profundidad 2 (D1-bajo) donde comienza el backtracking si las consultas que realiza el Sistema son respondidas negativamente. Así como la expansión hacia abajo consistía en agregar términos en la cabeza de la lista objetivos, el backtracking o retroceso hacia arriba consistirá en eliminar dichos elementos comenzando con los ubicados en los niveles más bajos de la lista en cuestión. El proceso de Backtracking no continúa necesariamente hasta la raíz, pudiendo recomenzar la búsqueda abajo (escogiendo otro camino) en cualquier nivel del árbol. La búsqueda hacia abajo, se realiza bajo el control de la función encontrar (el corazón del intérprete); mientras que el retroceso lo efectúa a través de backtracks.

Una vez que SEDIF resuelve un problema alcanzando a un nodo terminal que es la solución, comienza, antes de salir, un proceso de actualización incrementando en uno (1) a las variables (que indican el número de veces que el intérprete pasó por allí) de los nodos pertenecientes al Camino Optimo actualizando los factores de decisión de dichos nodos. Ademas realiza lo propio con los valores de la propiedad frecuencia de las reglas aplicadas, permitiendo que en las sucesivas sesiones sean consideradas con mayor prioridad que antes.

Con estas ideas explicamos brevemente cual es el espíritu de la búsqueda del SEDIF. Puede llegar a ocurrir que se explore todo el árbol y la solución no sea encontrada, en otras palabras que la lista objetivos sea finalmente la lista vacía. En tales casos el Sistema Experto le comunica al usuario que con la información actual no puede diagnosticar la falla, debiendo averiguarlo por otros medios y cuando lo haya conseguido puede invocar a SEDIF informándolo sobre las razones que provocan el mal funcionamiento del circuito, aprovechando la facilidad de mantener su Base de Conocimiento.

Cuando el Sistema infiere que la solución está entre un grupo de posibles causas y no puede distinguirlas le responde con una lista de ellas de izquierda a derecha según los valores de certeza decrecientes.

CONCLUSIONES

Hemos logrado que SEDIF manifieste las características apuntadas al principio de este trabajo, mediante el empleo de recursos de programación no convencionales que van desde el modelo en sí mismo hasta el propio lenguaje en el que fué implementado SEDIF. En conclusión, el Sistema Experto para el Diagnóstico de Fallas:

- posee un razonamiento no determinístico, modificando el sentido de la búsqueda según los resultados del pasado. Pudimos observar que al principio realiza un número significativo de preguntas, el que disminuye a medida que aumenta el número de consultas. Esto último es exactamente lo que acontece con un especialista que comienza a resolver un nuevo problema en su campo.
- controla la búsqueda y orienta al usuario asistiéndolo en forma permanente.
- puede explicar como hizo para llegar a la solución a partir de los datos iniciales suministrados por el usuario.
- puede actualizar su conocimiento con facilidad gracias a la estructura de datos (listas de propiedades) ofrecida por LISP.
- se adapta a diversos circuitos debido a que el modelo propuesto separa los componentes principales de la inteligencia: el razonamiento y el conocimiento.

Los Sistemas Expertos están en una etapa de crecimiento contínuo y van avanzando progresivamente en el ámbito de la Informática y de las otras disciplinas. Surgiendo de la Inteligencia Artificial, y siendo cada día más poderosos, los Sistemas Expertos realizan un aporte valioso en la mayoría de las Ciencias, no por remplazar a los especialistas, sino por ser una herramienta complementaria de alto valor científico que los asiste en su campo.

BIBLIOGRAFIA

Nillson, N. : "Principles of Artificial Intelligence"; Springer Verlag (1982).

Winston, P. : "Artificial Intelligence"; Mc Graw Hill (1978).

Hassemmer, T. : "Looking at LISP"; Microcomputer Books (1983).

Furtado, A : "Paradigmas de Linguagens de Programaçao"; IEABI Campinas (Brasil) 1986.

Piaget, J : "Sicologia de la Inteligencia"; Psique (1979).

ROSARIO, Julio de 1986.