13
2.ej.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

UTILERIA DE COMPUTO PARA LA IMPLEMENTACION RAPIDA DE PROTOTIPOS DE INTERFACES HOMBRE\MAQUINA

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A :

ALEJANDRO CAMPOS SERRANO

DIRECTORES DE TESIS:

M C. RAUL JACINTO MONTES

DR. RAMON MONTELLANO GARCIA

1991

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CAPITULO III

CRITERIOS GENERALES DE DISEÑO

CAPITULO IV

ESPECIFICACION DE LA UTILERIA

CAPITULO V

DISEÑO DE LA UTILERIA

CAPITULO VI

EVALUACION Y CONCLUSIONES

ANEXOS

CAPITULO I

INTRODUCCION

1.1 LA INTERFAZ HOMBRE/MAQUINA.

En el desarrollo de sistemas computacionales ha sido muy importante el avance tecnológico alcanzado para lograr que la computadora se convierta en una herramienta de uso común en las diversas actividades de trabajo del hombre, por lo que el sistema que modera la relación hombre/máquina cobra más importancia, necesitandose una interfaz hombre/máquina eficiente.

El término interfaz hombre/máquina es el "nombre colectivo que se les da a todos los componentes de una máquina a través de los cuales se realiza la comunicación entre el ser humano y la computadora" [HER89] (1).

Existe gran variedad de interfaces hombre/máquina que emplean un teclado especial, un monitor de tamaño específico, diferentes tipos de despliegue en pantalla, o distintos tipos de interacción con la computadora, ya sea usando ventanas, selección por menú, o interacción con preguntas y respuestas; en consecuencia la selección del tipo de interfaz adecuada depende del tipo de aplicación.

La presente tesis considera el diseño de interfaces gráficas empleando íconos, mímicos y despliegues numéricos con cambios de color, con la ventaja de aprovechar las características humanas de interpretar con mayor rapidez una imagen o símbolo respecto a un

texto. De igual forma una interfaz gráfica puede ayudar proporcionando múltiples vistas, donde cada vista está limitada a proporcionar información relevante a la tarea particular o problema determinado (2).

El uso de interfaces gráficas tiene la ventaja de permitir, representar y asociar el proceso (sistema o aplicación) al lenguaje del usuario. El objetivo de una representación gráfica es el de permitir representar la aplicación de una manera más simple y natural. Además de que tiene un fuerte poder descriptivo e interpretativo, permite (en ciertos casos) describir los aspectos funcionales del proceso. También se observa que una representación gráfica que visualiza el proceso o sistema habla más que mil palabras por su capacidad descriptiva.

Ahora bien, en los sistemas de supervisión y control de los procesos de una planta, el operador utiliza la interfaz para monitorear dichos procesos, esta interfaz debe cumplir ciertas características que permitan una fácil operación e incrementen la eficiencia del sistema.

La interfaz hombre/máquina es la parte que permite introducir y extraer información del sistema, y por lo tanto evaluar su eficiencia. Si se tiene una interfaz deficiente, aunque la parte de procesamiento sea efectiva, el sistema se considera deficiente, lo que significa que la interfaz es una parte susceptible de evaluación del sistema, de ahí su gran importancia.

1.2 PRESENTACION DEL PROBLEMA.

El problema básico detectado por el Instituto de Investigaciones Eléctricas (IIE) es el incremento del tiempo de desarrollo y costo de los sistemas de cómputo, originado en general por una definición incompleta y no detallada de los requerimientos

del sistema.

En el Instituto de Investigaciones Eléctricas (IIE), se han detectado varios problemas relacionados con el desarrollo de sistemas, en particular cuando se trata de modificaciones solicitadas por parte del usuario final en etapas avanzadas del desarrollo, estas modificaciones son muy costosas [GER85] (3), como se observa en la figura 1.2.1, ya que se tienen que realizar cambios en gran parte del sistema y se requiere mayor tiempo para realizar las modificaciones, algunas veces incluso implica redefinir o rediseñar el sistema.

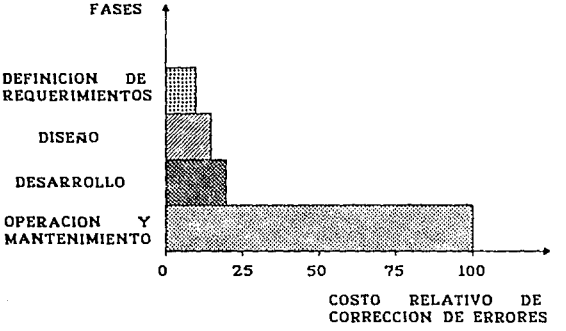


FIGURA 1.2.1 COSTO RELATIVO DE CORRECCION DE ERRORES EN LAS DIVERSAS FASES DEL PROCESO DE PROGRAMACION.

Muchos de los cambios solicitados al sistema final se deben a la mala definición de los requerimientos del usuario, o por una incorrecta interpretación de estos requerimientos, o simplemente por descuidos en el sistema de interacción hombre/máquina.

En muchas ocasiones la mala interpretación de estos

requerimientos por parte del diseñador de diálogo, se debe a la diferencia de términos manejados por el diseñador y el usuario. Se ha comprobado que cuando el usuario se incorpora al desarrollo del sistema, se minimiza la barrera entre usuario y diseñador, por lo tanto, existe el problema de incorporar al usuario al desarrollo del sistema, con el fin de conseguir el sistema final sin modificaciones significativas.

Por otra parte, uno de los aspectos más importantes dentro del desarrollo de sistemas de cómputo orientados al control y supervisión de procesos industriales es el diálogo hombre/máquina, aspecto que en ocasiones se descuida por otorgarle más atención a aspectos operativos del sistema en desarrollo, repercutiendo en la eficiencia y calidad de los resultados obtenidos en dichos sistemas, o en la poca aceptación del sistema, ocasionando modificaciones principalmente en la interfaz hombre/máquina.

"El diálogo hombre/máquina se define como el intercambio observable de símbolos y acciones entre el hombre y la computadora" [HAR89] (4), el cual es parte central del trabajo que propone una solución al problema planteado de acuerdo a la tesis que se presenta a continuación.

1.3 OBJETIVOS DE LA TESIS.

El objetivo principal de ésta tesis, es desarrollar una herramienta que permita la implantación rápida de prototipos de interfaces hombre/máquina, permitiendo así mejorar la definición de requerimientos del sistema.

Dentro de los proyectos desarrollados por el Instituto de Investigaciones Eléctricas (IIE), se encuentran los Sistemas de Adquisición de Datos y Registro de Eventos (SADRE's) [VIL84a], los

cuales están orientados principalmente a la operación de plantas generadoras de electricidad; las interfaces hombre/máquina de estos proyectos, involucran el empleo de diagramas mímicos, que reflejan el estado de operación de la planta que esta siendo supervisada. Estas interfaces requieren mejoras, que permitan una explotación más apropiada de la información generada por los SADRE's, y por otro lado se busca disminuir el tiempo de desarrollo de dichas interfaces.

Como objetivo particular se busca incorporar técnicas y tendencias actuales en el desarrollo de interfaces hombre/máquina, así como la generación rápida de prototipos que incluyan conceptos ergonómicos e independencia con respecto al diálogo.

Permitirá definir, generar, modificar y evaluar las características de la interfaz hombre/máquina de los nuevos sistemas a desarrollar, en particular de los SADRE's, incorporando técnicas de desarrollo rápido de prototipos y refinamiento iterativo.

Ayudará en la fase de definición de requerimientos en sus tres etapas :

1. Planteamiento de objetivos
2. Análisis
3. Especificación de requerimientos

Incorporará al usuario al desarrollo del sistema de manera "amigable".

1.4 JUSTIFICACION DE LA TESIS.

Como se indicó al inicio de este capítulo (Capítulo I, p. 2), la interfaz hombre/máquina es una parte importante de evaluación

del sistema, y donde en general surgen las modificaciones al sistema, por lo tanto, es necesario centrar la atención en el desarrollo de estas interfaces y buscar mecanismos que permitan generar resultados desde el inicio del sistema, logrando así una especificación completa de requerimientos.

Ahora bien, para aumentar el grado de aceptación del usuario final al proyecto, es necesario involucrarlo en todas las etapas de desarrollo del sistema, extrayendo el conocimiento que tenga del sistema y la tendencia que presenta al empleo y manipulación de la información para obtener resultados (algunos usuarios preferiran menúes, otros selección directa, ventanas, etc.), de tal forma que desde un principio el desarrollo se encamine en la dirección correcta, evitando malas interpretaciones o falta de especificación en etapas tempranas del proyecto.

El problema de involucrar al usuario en el sistema puede ser atacado utilizando diferentes técnicas como entrevistas, supervisión estricta y revisiones periódicas con el usuario, sin embargo, una de las técnicas más efectivas es el desarrollo rápido del prototipo del sistema, permitiendo al encargado del desarrollo y al usuario final, una evaluación preliminar de las características más sobresalientes del sistema, logrando una pronta realimentación del alcance funcional detallado del sistema.

La experiencia y la intuición demuestran que el grado de aceptación del sistema, por parte del usuario final, se incrementa en los casos en que es posible el desarrollo rápido de prototipos. La mayor aceptación del sistema se deriva de qué tanto interactuen el usuario final y el ingeniero de desarrollo, ya sea intercambiando opiniones o sugiriendo cambios al sistema aún antes de que el sistema entre a la fase de pruebas finales, evitando malos entendidos y desviaciones de la idea original.

El desarrollo de herramientas que permitan la elaboración de prototipos del sistema (estos prototipos no tienen que incluir todos los aspectos del sistema), es un paso importante en el desarrollo de sistemas, puesto que su impacto se deja sentir en todas las fases del proyecto como son :

- Definición de requerimientos,
- Diseño,
- Desarrollo,
- Operación y mantenimiento.

Basados en observaciones empíricas, Hartson e Hix concluyen que el desarrollo de interfaces hombre/máquina ocurre en ondas alternantes de dos clases de actividades complementarias, actividades relacionadas con el punto de vista del usuario (empíricas), y actividades de diseño y estructura [HAR89] (5).

Al analizar la gráfica de la figura 1.4.1 [GER85] (6), se puede observar que el mayor número de problemas en la detección de errores se encuentra en la fase de desarrollo, mientras que el porcentaje de generación de errores es bajo, es decir, aunque se generan pocos errores en esta fase son difíciles de detectar y representan un mayor problema. Por el contrario, en la fase de definición de requerimientos se generan alrededor del 55% de errores en un programa y representa poco problema detectarlos. De lo anterior podemos concluir que es mejor y más fácil centrar la atención en la fase de definición de requerimientos para mejorar la calidad de los sistemas.

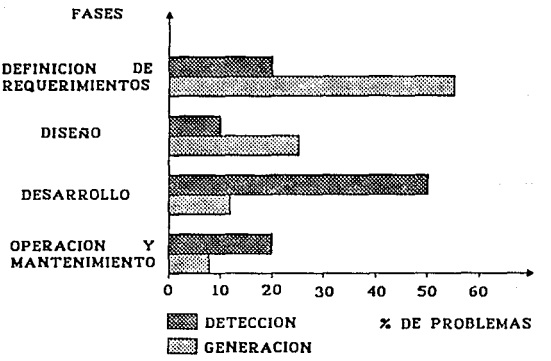


FIGURA 1.4.1 GENERACION Y DETECCION DE ERRORES EN LAS DIVERSAS FASES DEL PROCESO DE PROGRAMACION.

1.5 ESTADO DEL ARTE DE LA INTERFAZ HOMBRE/MAQUINA.

El diseño e implantación de diversos tipos de interfaces hombre/máquina ha evolucionado mucho desde el inicio de los sistemas de cómputo hasta la fecha.

Si retomamos la definición de interfaz hombre/máquina presentada al inicio de éste capítulo (Capítulo I, p. 1) podremos analizar y comprender estos cambios.

Los primeros componentes físicos de los sistemas de cómputo fueron cableados de tal suerte que sólo personal especializado y capacitado podía manejar. Más tarde surgen los dispositivos como lectoras de tarjetas que permitían interactuar al hombre con la computadora, hasta llegar a los modernos dispositivos de entrada y

salida como teclados especializados, lectores ópticos, monitores sofisticados, etc.

No sólo hubo avances en cuanto a hardware se refiere, también en el área del software o desarrollo de programas se mejoró notoriamente. Los desarrollos van desde diseños empíricos hasta la aplicación de modernas técnicas de diseño establecidas por la creciente ingeniería de software, y la aplicación de técnicas ergonómicas basadas en factores humanos en sistemas de cómputo [MAG85].

La combinación de un adecuado manejo de los avances logrados en las áreas de software y hardware, permite diseñar e implantar una interfaz hombre/máquina eficiente, pero no debemos olvidar que en las características de una interfaz también intervienen los diferentes factores humanos que la definen.

El diseño de una interfaz no se puede derivar de un conjunto de ecuaciones matemáticas, podrá ser rigurosamente evaluada y las alternativas podrán ser empíricamente probadas, pero un diseño no puede ser calculado de una teoría básica [DUM88].

La literatura dice que no se debe diseñar una interfaz si no se conoce la información sobre el lugar de trabajo y las tareas de quienes ahí van a laborar [RIJNS], pero también menciona que se pueden seguir ciertas directrices que permiten unificar ciertos criterios en el diseño de interfaces hombre/máquina [DUM88].

El estado actual de la ciencia de interacción hombre/máquina hace difícil definir las características de una interfaz en términos rigurosos, de aquí el uso del término "amigable al usuario" [DUM88], lo que nos lleva a concluir que el diseño e implantación de una interfaz hombre/máquina depende del tipo de aplicación desarrollada, empleando técnicas de ingeniería de software, apoyadas en técnicas ergonómicas y una serie de

directrices generales ampliamente analizadas en el capítulo 2 (Capítulo II, pp. 37-47).

1.6 CONTENIDO DE LA TESIS.

Hasta el momento se ha hablado de la importancia de la interfaz hombre/máquina en los sistemas de cómputo, de la misma manera del problema detectado y su relación con las interfaces y el cómo la tesis propuesta ayudará a solucionar dicho problema.

A continuación se presentan los capítulos con objeto de establecer los conceptos que aclaran el tema, definen y respaldan la tesis propuesta.

El capítulo dos presenta los antecedentes que se tienen sobre desarrollo de interfaces hombre/máquina, los conceptos y recomendaciones en diseños de interfaces hombre/máquina, así como las tendencias actuales en el desarrollo de estas interfaces.

En el capítulo tres se discuten los criterios generales básicos en diseño de interfaces hombre/máquina, algunos de ellos son : el perfil del usuario, las consideraciones del diálogo hombre/máquina, la independencia de diálogo y la generación rápida de prototipos. También se presenta el editor de mímicos que sirve de apoyo al desarrollo de prototipos de interfaces hombre/máquina, la selección de los criterios empleados y el alcance de la tesis.

El capítulo cuatro muestra la especificación de la utilería para desarrollo de prototipos de interfaces hombre/máquina, es decir, se muestra una descripción general de la utilería, un modelo conceptual y la descripción de las funciones disponibles.

El capítulo cinco retoma la documentación de las funciones de la utilería y las emplea para establecer el diseño de la misma, ésta comprende: la arquitectura del diseño, el diagrama de estructura o de procesos y la descripción de los procesos de la utilería.

En el capítulo seis se hace una evaluación de los beneficios obtenidos, comparando un desarrollo de interfaz hombre/máquina realizado con técnicas tradicionales y un desarrollo hecho empleando la utilería propuesta. También se presentan las conclusiones sobre el tema de interfaces hombre/máquina y la utilería desarrollada. Se discuten mejoras, se presenta la línea de investigación seguida, así como los temas propuestos para futuras investigaciones.

En la parte denominada Anexos se tiene un glosario de términos técnicos.

Finalmente se encuentra la Bibliografía, con las obras de mayor importancia y que respladan los términos y conceptos manejados en la tesis.

REFERENCIAS, CAPITULO I

(1) Victor Hugo Herrera López. "Diseño e implementación de un editor de mímicos para procesos industriales de alto riesgo." (Tesis de licenciatura, Universidad de las Américas, 1989) p. 5

(2) Luqui, P. D. Barnes and M. Zyda. "Graphical tool for computer-aided prototyping", (En : Information and software technology), 32:3 (april 1990) p. 199

(3) Victor Gerez Greiser [y otros]; Desarrollo y administración de programas de computadora (software). (D.F., México: Compañía editorial continental, S. A. de C. V., c1985) p. 48

(4) H. R. Hartson and D. Hix. "Human-Computer Interface Development", (En : ACM Computing Surveys), 21:1 (March 1989) p. 8

(5) Ibidem, p. 52

(6) Victor Gerez Greiser [y otros]; op. cit. p. 48

(7) Joseph S. Dumas. "Designing the interface". En : Designing user interfaces for software, Ed. Prentice Hall. (U.S.A., c1988) p. 49

CAPITULO II

ANTECEDENTES Y TENDENCIAS ACTUALES

2.1 INTRODUCCION.

Los sistemas de cómputo han sido diseñados e implantados con diversas técnicas que van desde las intuitivas hasta las establecidas por la ingeniería de software.

Un aspecto muy importante para la eficiencia de estos sistemas es la aceptación por parte del usuario, ya que su diseño o implantación debe estar orientado a ayudar a las tareas y solución de los problemas del usuario. Sin embargo, se siguen presentando problemas durante la puesta en marcha de estos sistemas, originados por especificaciones deficientes por parte del usuario, interpretaciones incorrectas por parte de los diseñadores, desconocimiento de técnicas ergonómicas y lentitud en el proceso de implantar un desarrollo completo.

Las técnicas modernas de ingeniería de software han logrado resolver parcialmente estos problemas a través de herramientas o utilerías que ayuden a solucionar dichos problemas, un ejemplo es la utilería desarrollada en esta tésis, en la cual se toma en cuenta el punto de vista del usuario, se busca una implantación rápida de prototipos en colaboración con el usuario, y se incorporan técnicas ergonómicas en el diseño que permiten acondicionar el medio de trabajo al usuario.

2.2 LA INTERFAZ HOMBRE/MAQUINA BASADA EN MIMICOS.

Los mímicos han sido utilizados ampliamente para la supervisión y operación de plantas industriales, específicamente plantas termoeléctricas, para control y mantenimiento de dichas plantas. Dichos sistemas son manejados por sus operadores, quienes tienen la responsabilidad del manejo de la planta.

La necesidad de conocer en forma rápida y confiable el estado de operación en que se encuentra la planta, dio lugar a la creación de sistemas computarizados que ayudan al operador de la planta a administrar los procesos de seguridad, vigilar los componentes del sistema, su operabilidad y a cumplir con los procedimientos de operación.

En el Instituto de Investigaciones Eléctricas (IIE) se han desarrollado dichos sistemas como parte del proyecto SADRE [DIA84] (Sistema de Adquisición de Datos y Registro de Eventos) dedicado básicamente a la operación de centrales termoeléctricas.

Una parte importante en el sistema SADRE es la interfaz hombre/máquina, la cual permite al usuario accesar información lo más rápido posible, los datos se presentan en forma clara y precisa y son fáciles de manipular, sin embargo en recientes investigaciones en el IIE se observa la necesidad de mejorar dichas interfaces para hacerlas más interactivas y amigables, esta necesidad dio origen al inicio de varias tesis de licenciatura, entre otras la de: "Utilería de cómputo para la implantación rápida de prototipos de interfaces hombre/máquina".

Como propuesta para mejorar la implantación de dichas interfaces, se plantea la colaboración directa del usuario en el diseño de la interfaz, para lo cual es necesario una utilería que permita su implantación rápida y eficiente, mejorando notoriamente la calidad de la especificación final del sistema y por

consiguiente una mayor aceptación por parte del usuario del sistema.

Con la colaboración del usuario se puede determinar la presentación de información en procesos de alto riesgo y la utilización de diagramas mímicos de la planta, en los cuales se resalta y refresca periódicamente la información relevante al proceso y relativo a la seguridad de operación de la planta. Se le hace notar al usuario que la información se complementa con cambios de color, alarmas audibles, gráficas de tendencias, y reconocimiento y manejo de alarmas.

Cabe aclarar que las funciones de cambio de color, alarmas audibles, gráficas de tendencias, reconocimiento y manejo de alarmas no forman parte del alcance de esta tesis, pero podrán tenerse en cuenta para futuros desarrollos.

Como solución se plantea la implantación rápida de prototipos de interfaces hombre/máquina, teniendo como antecedente un editor de mímicos, que permite crear la parte estática (gráfica) de la interfaz, siendo desarrollado dicho editor en el mismo IIE como parte del proyecto de mejoras a las interfaces hombre/máquina de los SADRE's, quedando por realizar la parte dinámica de la interfaz como es el refresco de la información presentada por cada mímico, las interrelaciones que tendrán los mímicos dentro de la interfaz, el diálogo que manejará la interfaz, etc. El diseño de la utilería planteada estará fundamentado en ciertos principios que se presentan a continuación.

2.3 PRINCIPIOS DE DISEÑO DE UNA INTERFAZ.

Los principios de diseño de una interfaz efectiva [DUM88] (1), se muestran a continuación :

A. Colocar al usuario en el control de la interfaz.
B. Dirigir los niveles de técnica y experiencia del usuario.
C. Ser consistente en redacción, formatos y procedimientos.
D. Proteger al usuario de las configuraciones y estructura internas de hardware y software.
E. Proporcionar documentación junto con la aplicación en línea.
F. Minimizar la carga de la memoria del usuario.
G. Seguir los principios de diseño de gráficas de calidad.

A. Colocar al usuario en el control de la interfaz implica los siguientes principios:

1. Proporcionar ayuda en línea que informe al usuario acerca de la estructura y operación de la aplicación.

2. Proporcionar un mensaje de presencia ("prompt") efectivo y mensajes de estado que guíen al usuario a través de procedimientos y lo mantenga informado acerca del estado de la aplicación.

3. Proporcionar mensajes de error que permitan indicarle al usuario que estuvo mal y como recuperarse del error.

4. Proporcionar al usuario los medios para moverse libremente entre pantallas y la habilidad para moverse fácilmente a menúes principales o nodos y rápidamente salir de la aplicación.

5. Proporcionar consistencia en el uso de palabras, formatos y procedimientos.

B. Dirigir los niveles de técnica y experiencia del usuario.

El software creado debe ser dirigido al tipo de usuario de la interfaz; el porcentaje de gente que tiene experiencia con computadoras es pequeño y uno de los problemas más difíciles para el desarrollador de software es librar este obstáculo entre la habilidad del diseñador y la de los usuarios.

Si la aplicación desarrollada es utilizada por usuarios virtualmente sin experiencia en computadoras, el diseño debe favorecer a estos usuarios.

Las prácticas que contribuyen a este principio son:

1. Evitar los términos no familiares al usuario, diseñando la interfaz desde el punto de vista del usuario. El diseño debe estar sujeto a pruebas para estar seguro que el potencial de los usuarios entiende el significado de las palabras en los menúes, mensajes, textos de ayuda y tutores.

2. Usar procedimientos apropiados para transferencia de control :

   -- menúes,
   -- preguntas y respuestas,
   -- secuencia de comandos,
   -- teclas de función, etc.

3. Proporcionar diversos niveles de detalle para errores y mensajes de ayuda : mensajes de error o de ayuda que sólo recuerden que significan, o procedimientos paso a paso y ejemplos que instruyan al usuario en la operación de las aplicaciones.

C. Ser consistente en redacción, formatos y procedimientos.

La consistencia ayuda al usuario a aprender, a usar y recordar más fácilmente cuando existe un problema.

D. Proteger al usuario de las configuraciones y estructuras internas de hardware y software que están detrás de la interfaz.

Las prácticas que contribuyen a éste principio son :

1. Evitar los modismos, empleando plenamente el Español para que palabras y frases se refieran directamente al software que se está utilizando.

2. Evitar el despliegue de mensajes de estado que describan el trabajo interno del software, como : "LINK MAIN" o "FORTRAN END". Estos mensajes son términos que el usuario no tiene que interpretar.

3. Evitar pasar mensajes de error, generados por el software empleado como herramienta, directamente al usuario, es decir, traducir primero al Español y entonces desplegarlos al usuario (proteger los sistemas y manejo adecuado de los errores).

E. Proporcionar documentación junto con la aplicación en línea para ayudar al usuario a entender como operar la aplicación y recuperarse de los errores.

Un sistema de ayuda en línea extenso, bien diseñado y bien escrito es una meta de muchas aplicaciones importantes de software. Esa es una evidencia de que un sistema de ayuda bien diseñado provoca la productividad de los usuarios e incrementa su satisfacción con un producto [DUM88].

La documentación en línea que el usuario necesita es más que sólo mensajes en línea. Todos los mensajes que son desplegados son parte de la documentación en línea. Los mensajes de estado, los prompts y los mensajes de error son también componentes de una documentación en línea.

Estos mensajes informan al usuario acerca del estado de operación de la aplicación, y lo ayudan a conocer como proceder y como recuperarse de los errores.

Las directivas que contribuyen a este principio son las que describen el diseño de la ayuda, los mensajes de estado y error al igual que los prompts.

F. Minimizar la carga de la memoria del usuario.

La naturaleza humana es notoriamente pobre en recordar información detallada (2), pero excelente para reconocerla. Considerando este principio, la información que se presente en las pantallas de computadora debe de ser para reconocer y no para recordar.

Un buen diseño de una interfaz deberá minimizar la necesidad del usuario de memorizar y posteriormente de recordar información. Cuando es posible, los usuarios deberán ser capaces de seleccionar de una lista de opciones que permita utilizar su memoria de reconocimiento antes que su memoria de recuerdos. La amplia aceptación de los menúes como medio primario de control de las transferencias del usuario mediante software, es un ejemplo de una aplicación de este principio.

Existen otras prácticas que contribuyen a este principio:

1. Ser consistente en el uso de palabras, formatos y procedimientos. La consistencia reduce la necesidad del usuario de aprender y recordar nueva información. Por ejemplo, cuando el mismo procedimiento de salida es usado en todos los menúes, un usuario tiene que aprenderlo sólo una vez y es fácil de recordar.

2. Desplegar mensajes de estado que recuerden al usuario donde está en una aplicación y qué opciones tienen efecto. Un simple mensaje de estado tal como "Pantalla 2 de 3" y "F2 = Menú principal" reduce la necesidad del usuario de recordar dónde están dentro de una aplicación y qué operaciones pueden usar.

3. Proporcionar ayuda en línea que esté diseñada como una memoria adicional. No es solamente el nuevo usuario quien necesita recordar información. El usuario experimentado también necesita ocasionalmente recordar información, por ejemplo, los detalles del formato de los datos de entrada. La ayuda en línea que está diseñada como memoria adicional puede proporcionar esta información para que los usuarios experimentados puedan continuar sin tener que buscar a través de su memoria o de un manual.

4. Utilizar memoria corriente en prompts y leyenda de datos de entrada. Por ejemplo, decir a los usuarios cual es el formato de la fecha, como (mm/dd/yy). Muchos de los detalles de formatos y procedimientos carecen, simplemente, de importancia para el usuario de la información, tanto que rápidamente la olvidan. Pero estos detalles son importantes para el software que almacena y manipula la información. Una de las primeras cosas que el nuevo usuario aprende acerca de las computadoras, es que ellos deben ser cuidadosos con estos detalles. Un buen diseño de interfaz ayuda al usuario con estos detalles

para la inclusión de memoria corriente en mensajes que solicitan información.

G. Seguir los principios de diseño de gráficas de calidad mediante una correcta distribución de información en la pantalla.

En una interfaz efectiva, los despliegues deberán estar formateados para que los usuarios puedan encontrar lo que ellos esperan y leerlo fácilmente. El diseño de todas las pantallas deberá maximizar la probabilidad de que los ojos del usuario encuentren la información importante, fácil y rápidamente [DUM88] (3).

Las prácticas que contribuyen a éste principio son :

1. Utilizar técnicas de iluminación para enfatizar la importancia de la información sin distraer al usuario con parpadeos innecesarios.

2. Separar bloques de texto usando listas, pasos numerados y ejemplos específicos.

3. Colocar títulos en las pantallas y encabezados de listas.

4. Alinear y dejar espacio interlíneas en las listas de datos textuales y numéricos para facilitar la exploración.

2.4 CONCEPTOS BASICOS EN EL DISEÑO DE LA INTERFAZ HOMBRE/MAQUINA.

El diseño de una interfaz eficiente debe aprovechar los conceptos teóricos que han sido comprobados prácticamente, de aquí el manejo de éstos presentados a continuación.

2.4.1 ERGONOMIA.

La ergonomía es la ciencia que estudia las necesidades del ser humano y las condiciones bajo las cuales trabaja; analiza las características humanas más importantes y la forma en que se manifiestan, así como la forma en que éstas deben ser consideradas al momento de realizar el diseño de un sistema computarizado [CHA86].

Por lo tanto, el objetivo de un diseño ergonómico debe ser "la adaptación del ambiente de trabajo al ser humano". Esto no quiere decir que el ser humano no se pueda adaptar a las nuevas tecnologías, más bien se busca que los diseñadores de sistemas provean a los usuarios con una variedad de opciones a escoger. Como cada persona tiene características propias muy diferentes, un diseño ergonómico debe satisfacer una amplia gama de características humanas para ser eficiente.

Un gran problema que todavía existe es que las metodologías de diseño reflejan la posición prominente de los ingenieros, quienes escogen el grado y el tipo de automatización y especifican o desarrollan el hardware y el software, sin dar importancia a los factores humanos, los cuales se revisan en etapas en las que ya es muy difícil modificar el subsistema técnico. Esto implica que haya una mala aceptación por parte del usuario, con las consecuentes demoras, errores, malos entendidos e ineficiencias [RIJNS] (4).

Para que un sistema de control pueda estar libre de estos defectos, el diseñador debe de :

1. Tener conocimiento del usuario potencial del sistema [CHA86].

2. Diferenciar dentro del sistema de aspectos técnicos de los aspectos psico-sociales o referentes a las características humanas.

3. Balancear la carga de trabajo de un empleado. El ejemplo típico de una carga de trabajo mal balanceada lo tenemos en los procesos altamente automatizados, como en el caso de las modernas plantas generadoras de energía eléctrica, donde el trabajo normal consiste en esperar la ocurrencia de eventos anormales (99 % de aburrimiento y 1 % de terror), dando como resultado problemas psicológicos en los operadores. Además, es cuestionable si una persona que ha estado inactiva por mucho tiempo pueda reaccionar repentina y adecuadamente ante situaciones inesperadas (5).

En general, el aspecto que debe reforzar el diseñador del diálogo es el que se refiere a los factores humanos [MAG85].

"Los factores humanos podrían ser los determinantes más significativos de un diseño de interfaz de usuario exitoso" como lo indica Luqui [LUQ90] (6), por ejemplo, apunta que algunos "estudios han mostrado que el hombre tiene la capacidad de recordar 7 +/- 2 cosas a la vez" o si los controles están organizados en la misma secuencia que los usuarios estan acostumbrados, se elimina la necesidad de búsqueda y minimizan la cantidad de información que el usuario debe recordar. Conceptos como éstos generalmente el diseñador no los tiene en cuenta.

Cada sistema es diferente, como lo son los factores humanos, el estudio y aplicación de estos factores deberá basarse en el

sistema desarrollado.

2.4.2 INDEPENDENCIA DE DIALOGO.

La independencia de diálogo surge como un concepto análogo a la independencia de datos, en el que los investigadores y diseñadores de bases de datos encontraron un problema similar con la necesidad de modificar fácilmente los datos sin tener que cambiar los programas de aplicación correspondientes [HAR89] (7), como se ha observado en muchos desarrollos en los que se tiene el problema de que el diálogo está estrechamente ligado al sistema y cuando se tiene necesidad de realizar algún cambio en el diálogo, es necesario alterar gran parte del sistema en donde éste interviene.

En el diseño de una interfaz se requiere que exista independencia del diálogo y que esté basada en una definición formal para comunicación entre la interfaz hombre/máquina y los programas de computadora.

La independencia de diálogo es una aproximación en la cual las decisiones de diseño que afectan sólo al diálogo hombre/máquina, están aislados del diseño que afecta solamente a la estructura del sistema de aplicación y los programas de computadora [HAR89]. En la práctica, esto quiere decir que la apariencia de la interfaz para el usuario final y las opciones de estilo de interacción (ejemplos: lenguajes de comandos, menúes, formas de captura) acostumbrados para obtener entradas del usuario final, son transparentes para el software encargado de realizar el procesamiento de la información (sistema de aplicación) y que no interactúa con el usuario, de tal forma que esta independencia permite modificar el diálogo para adaptarlo al tipo de usuario del sistema, sin tener que modificar el sistema de aplicación que no interactúa con el usuario.

Por lo anteriormente expuesto se considera a la independencia de diálogo como crucial para una fácil modificación de la interfaz por refinamientos iterativos, como es el caso del mantenimiento.

La independencia de diálogo debe estar respaldada al momento de diseño en la separación del diálogo de los programas de computadora, permitiendo desarrollar módulos independientes apegándose a los principios del diseño estructurado y observándose en el sistema de aplicación un componente de diálogo, através del cual se realizan todas las operaciones de intercambio con el usuario, y un componente de cálculo con un mecanismo funcional de procesamiento del sistema de aplicación con el cual el usuario no interactúa.

Desde que la independencia de diálogo permitió más de un componente de diálogo para un sólo componente de cálculo, un sistema de aplicación puede tener dos o más interfaces de usuario diferentes. La independencia de diálogo es menos importante en el contexto de estuche de herramientas (toolkit), el cual es una librería de rutinas para la implantación de los rasgos de interfaz hombre/máquina [HAR89] (8).

La posibilidad de independencia de diálogo en los sistemas futuros es muy promisoria y ha abierto una fuente de estudio en el área, para especializarse en factores humanos, llegando a ser una parte importante en el desarrollo de sistemas, dichos especialistas son llamados desarrolladores de diálogo, ingenieros de diálogo, ingeniero de interfaces o diseñador de diálogo.

El desarrollador del diálogo es un especialista en factores humanos, interesado en el diseño, implantación, y evaluación de la forma, estilo, contenido y secuenciación de las interfaces hombre/máquina (9). La importancia que está teniendo el diseñador de diálogo es tal que en la mayoría de universidades de Computación e Informática existen cursos de especialización en el "diseño de

interfaces hombre/máquina" para graduados en ciencias de la computación. El enfoque de estos cursos permite al estudiante estar consciente de los problemas que involucran los factores humanos y los métodos de solución asociados con el diseño y uso de los sitemas de cómputo.

La independencia de diálogo ha sido además una importante fuerza de impulso para los desarrollos teóricos y de implantación de los Sistemas Manejadores de Diálogo (DMS) (10), pero al estar apoyado por un tipo de modelado, como se muestra a continuación, se fortalece aún más.

2.4.3 MODELADOS PARA INTERACCION HOMBRE/MAQUINA.

Hay muchos tipos de modelado aplicados a la interacción hombre/máquina, tres de los más sobresalientes son: para análisis de tareas, representación de interface y descripción estructural.

El modelado orientado a tareas es usado para analizar y describir los detalles de una actividad particular de un usuario, frecuentemente por descomposición jerárquica dentro de niveles de subactividades. Los modelos orientados a tareas son típicamente usados para dirigir el proceso de diseño para interfaces específicas y frecuentemente incluye una descripción del conocimiento que un usuario tiene o necesita acerca de la actividad y como realizarla.

Los modelos para representación de interfaces son esquemas de representación particular de instancias de la interacción hombre/máquina (11), es decir, se presentan casos particulares en los que se encuentre trabajando la interfaz y se observa su presentación al usuario, el tipo de diálogo que se requiere en ese momento y posteriormente formar una secuencia de todas estas instancias para formar la interacción hombre/máquina.

Los modelos estructurales de la interfaz hombre/máquina son descriptivos del proceso general de comunicación hombre/máquina; esto es, su teoría y generalidad describen la estructura de intercambio del usuario con la computadora, siendo de sumo interés para el tipo de interfaces que se pretenden obtener, de aquí la importancia de este modelado por lo que se amplia la información en los párrafos siguientes.

2.4.4 MODELADO ESTRUCTURAL DE LA INTERFAZ HOMBRE/MAQUINA.

El modelado estructural descriptivo de la interfaz hombre/máquina es un concepto fundamental en el manejo de interfaces, necesario para entender la naturaleza de la interacción hombre/máquina y por lo tanto necesario para el proceso de desarrollo de la interfaz [HAR89] (12), ya que a partir del modelado se puede obtener el diálogo necesario para la interfaz.

Los métodos para representación pueden estar basados en un modelo estructural y descriptivo. En tal caso el modelo estructural puede guiar al desarrollador durante el proceso de representación del diseño del diálogo.

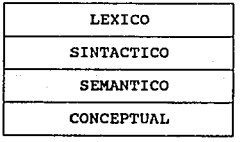
Los modelos lingüísticos y no lingüísticos como parte del modelo estructural nos permiten identificarlo y estudiarlo claramente.

2.4.4.1 MODELOS LINGÜISTICOS.

Son aquellos modelos en los que el diálogo se maneja como un lenguaje, ya que el diálogo hombre/máquina puede formalmente ser modelado como la gramática y vocabulario de un "lenguaje de interacción" (comunmente conocido como un lenguaje de comandos).

El concepto de modelo lingüístico se refiere al empleo de un punto de vista lingüístico, ya que se manejan los niveles conceptual, semántico, sintáctico y léxico.

El "nivel conceptual" es la colección de un sistema básico de metas y funciones que el usuario final debe entender. El "nivel semántico" abarca las operaciones de entrada y las técnicas de presentación de salida. El "nivel sintáctico" contiene las secuencias de los "tokens" específicos para invocar las acciones semánticas, también como la forma específica y el contenido de la salida. El "nivel léxico" define la estructura del "token" en términos de hardware (13). Como se observa este tipo de modelado aplica la técnica de diseño de un compilador, ya que se puede interpretar el diálogo hombre/máquina desde un punto de vista secuencial y con la estructura de un lenguaje.



Niveles lingüísticos

El principal objeto lingüístico es un "token", una abstracción representando la unidad más pequeña de una entrada de un usuario que puede tener formalmente un significado definido en términos de la aplicación o tarea (14).

El modelado de diálogo de transacción satisface un diálogo secuencial, el cual usualmente tiene una estructura lingüística

entre partes del diálogo relacionado con comandos, parámetros, selección de opciones, introducción de datos y valores, requiriendo "parsing" y/o validación. El modelo ha sido también aplicado a un diálogo (con estilo) de manipulación directa, el cual involucra entrada de valores de "token" por el usuario, pero no es típicamente receptivo para una estructuración lingüística.

2.4.4.2 MODELOS NO LINGÜISTICOS.

Podemos encontrar diferentes modelos no lingüísticos, pero de los más sobresalientes están: las celdas de diálogo y los eventos de interacción.

A) Celdas de diálogo.

Una "celda de diálogo" es un modelo no lingüístico para descripción y desarrollo secuencial de un diálogo hombre/máquina [BOR81] [BOR82].

Una "celda de diálogo" consta de cuatro elementos básicos, que definen la estructura del diálogo, estos son :

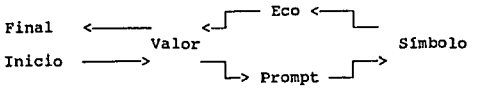
1. El "prompt" que prepara el sistema para las acciones de entrada del usuario e indica el tipo de dato que será introducido por el usuario.

2. El "símbolo" es la entrada del usuario; es una construcción para la especificación y valor de esa entrada.

3. El "eco" es el sistema de interpretación de símbolos introducidos por el usuario.

4. El "valor" es el mapeo del símbolo de entrada para la

utilización del dato por el programa de aplicación.



Ciclo de información en celdas de diálogo.

Una "celda de diálogo" es la unidad que describe la interacción de un diálogo secuencial con el usuario, incluye información del usuario, sus acciones de entrada, la evaluación de las entradas, el eco de la entrada, el mapeo desde la entrada hasta el valor y la entrega del valor resultante para el componente computacional.

Podemos considerar a una interfaz hombre/máquina como si estuviera estructurada dentro de celdas, y las celdas como si estuvieran estructuradas en "prompts", símbolos, ecos y valores. Las herramientas proporcionadas por un desarrollador de diálogo incluyen un sistema de entrada/salida para accesar los dispositivos gráficos y un lenguaje de diálogo para especificación de la estructura de datos y control del flujo. Esta aproximación puede ser usada conceptualmente para producir interfaces hombre/máquina.

B) Eventos de interacción.

Se ha propuesto considerar a un evento de interacción como las bases de otro modelo no lingüístico. La premisa básica es que el diálogo secuencial esté compuesto principalmente de una serie de "eventos de interacción". Tanto un evento está compuesto de un "prompt" del sistema, una

entrada de usuario, una acción de procesamiento del sistema, y un control del flujo para determinar el siguiente evento de interacción. A través del "prompt", el sistema indica al usuario que está esperando una entrada. El usuario entonces proporciona la entrada, y el sistema responde con una acción basada en la entrada. El control de flujo del sistema decide que evento de interacción deberá continuar.

Otras características del diálcgo incluyen: la "validación de entradas" para filtrar las entradas del usuario; una característica de "ayuda" invocable por el usuario; un mecanismo de "escape" que permite al usuario saltar la actual solicitud de entrada, y los "valores de default" que son respuestas asumidas por el sistema si el usuario no proporciona una entrada.

Los eventos de interacción, sin embargo, sólo son descriptivos de diálogos secuenciales. Además, la definición de un evento de interacción acompaña más que sólo el diálogo. El componente de acción y algunos de los componentes del control de flujo son realmente parte de los componentes computacionales y del control global del sistema, estrechando el dominio del modelo dentro de un sistema completo hombre/máquina, no sólo de la interfaz.

Algunos modelos son descriptivos de la arquitectura de la interfaz hombre/máquina, que indica como se relaciona con el resto de un sistema de aplicación, como es el caso del tipo de interfaz gráfica que emplea diagramas mímicos para representar su aplicación.

2.4.5 IMPLANTACION RAPIDA DE PROTOTIPOS.

En general la implantación de un sistema es extensa y requiere

mucho tiempo. Una alternativa es construir prototipos antes que sistemas completos. La habilidad de fabricación humana llama al desarrollo de herramientas de diálogo que produzcan rápidamente prototipos para permitir una pronta observación del comportamiento de la interfaz y facilitar la modificación de diseños. Esto es, **"la implantación rápida de prototipos"** es un concepto esencial en el diseño de interfaces hombre/máquina.

La implantación de prototipos es un acercamiento al desarrollo de sistemas que involucra la producción de al menos una primera versión del sistema de aplicación. Recordando que "la meta de un prototipo es diferente a la de un sistema de producción" [LUQ88] (15), entendemos que la meta de la implantación rápida de prototipos es una rápida comunicación con alternativas de diseño de interfaz para los desarrolladores, usuarios finales e implantadores. Otra meta será el permitir el proceso de refinamiento iterativo que permita iniciar el proceso de diseño.

Se debe estar consciente de que la implantación rápida de prototipos primeramente es una técnica, no una herramienta.

Teniendo en cuenta que se ha incrementado la demanda de grandes sistemas de alta calidad hasta el punto en que se necesita un cambio radical en la tecnología de software, y los prototipos rápidos son una solución prometedora a éste problema, además de que "los prototipos rápidos son particularmente efectivos para asegurar que los requerimientos reflejen las necesidades reales del usuario, incrementando la confiabilidad y reduciendo los costos por cambios en los requerimientos" [LUQ88] (16).

Los ingredientes clave de un acercamiento a la implantación rápida de prototipos incluye una pronta habilidad de observación, por parte del usuario, hacia el comportamiento del sistema, así como la utilización de escenarios, participación del usuario final y una evaluación y orientación para el desarrollo del sistema. Por

más de 15 años, la literatura ha intentado involucrar al usuario final en el diseño del sistema; la generación rápida de prototipos proporciona la forma de hacerlo efectiva y eficientemente [HAR89] (17).

Un prototipo reduce los cambios sorpresivos para el usuario final, ayuda a solucionar los problemas de la falta de habilidad del usuario final para dar una completa especificación al diseñador del sistema y da al usuario final un mayor e inmediato significado del sistema propuesto, por lo tanto, la implantación de prototipos facilita la comunicación entre el usuario final y el desarrollador.

Por claridad y conveniencia, podemos retomar las características del método de Luqui y Valdis [LUQ88] de la construcción rápida de prototipos de grandes sistemas de tiempo real:

1.- Los prototipos deberán satisfacer y ser dirigidos por sus requerimientos.

2.- El prototipo deberá ser fácil de modificar porque estará sujeto a muchas revisiones antes de que el usuario quede satisfecho con los requerimientos reflejados por el funcionamiento del prototipo.

3.- El código del prototipo deberá ser fácil de leer y analizar.

En el IIE se ha observado que el desarrollo de herramientas que permitan la elaboración de prototipos (estos prototipos no tienen que incluir todos los aspectos del sistema), es un paso importante en el desarrollo de sistemas puesto que su impacto se deja sentir en todas las fases del proyecto como son :

1. Definición de requerimientos
2. Diseño
3. Desarrollo
4. Operación y mantenimiento.

Basados en observaciones empíricas Hartson-Hix[HAR89], concluyen que el desarrollo de interfaces hombre/máquina ocurren en ondas alternantes de dos clases de actividades complementarias, actividades relacionadas con el punto de vista del usuario (empíricas) y aquellas actividades de diseño y estructura.

En estudios realizados por Alavi y Boehm (18) en 1984, se concluye que la implantación rápida de prototipos "fue efectiva como una aproximación al desarrollo de sistemas interactivos" (como en muchos sistemas con interfaces hombre/máquina), "los resultados también mostraron que los usuarios finales de sistemas desarrollados usando prototipos fueron más favorecidos por el sistema final que donde no se emplearon prototipos... Se encontró que el software desarrollado con implantación de prototipos, comparado con especificaciones completas a priori, tiene un equivalente en la ejecución pero con 40 % menos código y con 45 % menos de esfuerzo".

Un buen esquema que explica el ciclo de vida de un prototipo propuesto por Luqui y Valdis (19) se muestra en la figura 2.4.5.1, donde se observa el desarrollo del prototipo empleando un método iterativo o cíclico, es decir, se establecen los requerimientos iniciales, se construye un prototipo, si la ejecución del prototipo no satisface los requerimientos establecidos, se procede a ajustar los requerimientos y se repite nuevamente el ciclo hasta que se satisfagan las requerimientos y es entonces cuando se procede a desarrollar el sistema completo.

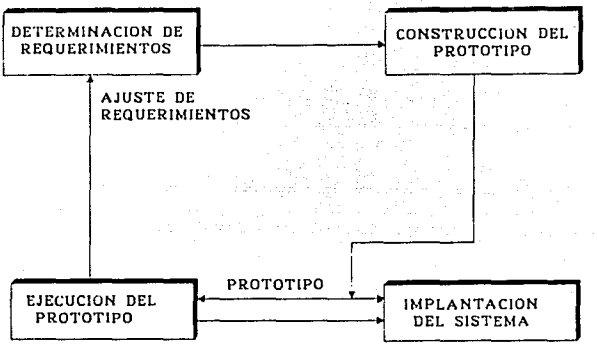


FIGURA 2.4.5.1 CICLO DE VIDA DEL PROTOTIPO Y LOS PASOS PARA ACTUALIZAR LOS REQUERIMIENTOS DEL SISTEMA.

Ventajas de los prototipos rápidos.

Mediante prototipos rápidos, el proceso de análisis del sistema se acelera tanto que muchas alternativas pueden ser evaluadas y los efectos de cada modificación pueden ser rápidamente observados.

Un prototipo rápido permite detectar y corregir errores en los requerimientos que son difíciles de identificar.

La técnica de prototipos rápidos proporciona al diseñador una forma de escribir especificaciones y utilizar componentes de software reutilizables, además de una herramienta de análisis con la que puede decidir cuál es la información importante y discriminar la que no (20).

El desarrollo rápido de prototipos promueve la metodología iterativa que beneficia a los grandes sistemas en donde la dificultad radica en la gran variedad de eventos a analizar, tal es el caso de un sistema de síntesis de alarmas como lo apunta Maizener [MAI90] (21).

También la técnica de prototipos rápidos del sistema permite mejorar la especificación del sistema, incluyendo la incorporación del usuario al desarrollo del sistema final, dirigiendo la idea original desde sus inicios con el mínimo de modificaciones o realizadas a tiempo.

APROXIMACIONES A LA IMPLANTACION RAPIDA DE PROTOTIPOS.

Existen propuestas de implantación de prototipos para interfaces representadas por diagramas de transición de estado, gramática estilo BNF y mecanismos basados en eventos [HAR89], entre los que se encuentran los siguientes productos:

1. RApid Prototypes of Interactive Dialogues (RAPID/USE), soporta la construcción de prototipos y sistemas de información interactivos.

2. Intérprete de Diagramas de Transición (TDI, por sus siglas en inglés) ejecuta la representación de código y simula la interfaz.

3. Sistema Manejador de Diálogo (DMS) en el cual la implantación rápida de prototipos esta hecha por un subsistema llamado el "Demostrador de comportamiento".

4. El sistema FLAIR de TRW que permite al diseñador de sistemas construir interfaces hombre/máquina extremadamente gráficas.

La implantación de prototipos, basada en la gramática estilo BNF, para representación de interfaces es muy similar a los prototipos basados en diagramas de transición. La gramática está interpretada con mecanismos que son basicamente máquinas de estado finito. El tipo diferente de representación lleva con él una escasa diferencia en énfasis, tratando la interfaz en una forma más orientada al lenguaje con menor énfasis directo en la estructura de control y la secuenciación.

De igual manera existen los sistemas basados en eventos, donde el desarrollador del diálogo puede definir objetos asociados con lugares en la pantalla y conocidos como botones de luz, haciendo los objetos seleccionables por el usuario, con acciones tales como oprimir el boton del "ratón" (mouse).

El desarrollador del diálogo también define una respuesta para cada evento. Una librería de íconos, muchos todavía con semántica adherida, facilitan gratamente el desarrollo de la interfaz.

2.5 DIRECTRICES PARA EL DISEÑO DE LA INTERFAZ.

Las directrices descritas a continuación son ampliamente recomendadas como apoyo o consulta durante el diseño de interfaces hombre/máquina, estas directrices apoyan gratamente el diseño del control de transferencias de la interfaz, el despliegue de información, las recomendaciones para la introducción de datos y la comunicación fluída con documentación en línea [DUM88].

2.5.1 PARA EL CONTROL DE TRANSFERENCIAS :

El control de transferencias se refiere al manejo de los cambios de un sistema de interacción a otro, por ejemplo, la transferencia de un menú a la opción seleccionada y su tipo de despliegue, o también al orden de las pantallas que serán desplegadas junto con las opciones proporcionadas por el sistema para cambiar de pantalla o despliegue. Las directrices son las siguientes:

1. Seleccionar el método de transición que sea apropiado para el nivel de experiencia del usuario y para los objetivos de ejecución de la aplicación.

2. Colocar un título representativo en la cabecera de todo menú.

3. Para menúes que ocupen una pantalla, proporcionar un balance simétrico centrando el título y el menú de opciones al centro de la pantalla.

4. Seleccionar un principio de organización para el menú de opciones.

5. Facilitar la exploración del menú, colocar líneas en blanco entre los grupos lógicos del menú de opciones y después de cada cinco opciones en una lista larga.

6. Limitar el número de opciones del menú a una pantalla.

7. Usar un método de selección de opciones que sea consistente con la tecnología disponible en la estación de trabajo del usuario y el tamaño de la aplicación existente diseñado.

8. Proporcionar un camino para que el usuario abandone el menú sin seleccionar ninguna opción.

9. Usar palabras para el menú de opciones, que, de manera clara y específica, describan lo que el usuario está seleccionando; de fácil uso y con verbos de acción para describir las opciones del menú.

10. Utilizar íconos que no sean ambiguos e identifíquen el significado de la opción del usuario.

11. Minimizar la iluminación usada en un menú.

12. No exigir al usuario introducir primero o al final blancos o ceros, y no incluir valor de default en un menú.

13. Desplegar el menú de opciones combinando letras mayúsculas y minúsculas.

14. Organizar un menú jerárquico de acuerdo a las tareas que el usuario deberá realizar, antes que la estructrura de los módulos de software.

15. Usar teclas de función moderadamente para acelerar la ejecución de operaciones frecuentemente usadas.

16. Estar seguro que cualquier tecla de función que utilice deberá operar correctamente en todos los teclados que el usuario tenga.

17. Utilizar comúnmente palabras cortas que describan claramente la acción que el comando o instrucción deberá llevar a cabo; escojer palabras que sean distintivas unas de otras.

18. Permitir al usuario acortar comandos o instrucciones utilizando reglas consistentes de abreviación.

19. Permitir al usuario apilar una serie de comandos o

instrucciones en una línea.

20. Permitir al usuario desplegar una lista de comandos o instrucciones.

21. Ser consistente en el uso de formatos de menúes, procedimientos y en la redacción; el mapeo de teclas de función; el nombramiento y abreviación de comandos o instrucciones; y el diseño del "prompt" de preguntas y respuestas.

2.5.2 PARA EL DESPLIEGUE DE INFORMACION :

El despliegue de información indica los diferentes métodos o técnicas para mostrar la información de la manera más eficiente.

1. Colocar un título en todas las pantallas de despliegue que describan clara y específicamente el contenido de la pantalla.

2. Desplegar solamente información que el usuario necesite conocer.

3. Desplegar datos a el usuario en forma directamente utilizable.

4. Proporcionar un balance simétrico en los despliegues, centrando títulos, encabezados y colocando información en ambos lados del eje central de la pantalla.

5. Todos los despliegues deberan indicar como salir de la pantalla.

6. Cuando un despliegue continue en múltiples pantallas, la pantalla deberá indicar en dónde se encuentra el usuario en relación al despliegue.

7. Considerar la habilidad de los usuarios y la información que ellos deberán manipular cuando se despliegue información en múltiples ventanas.

8. Desplegar texto convenientemente mezclado, con letras mayúsculas y minúsculas con la puntuación apropiada. Colocar una línea en blanco en cada uno de los párrafos y doble espacio en el texto cuando se pueda.

9. Justificar el texto a la izquierda pero dejar un margen derecho.

10. Permitir una separación con guión de palabras entre líneas.

11. Utilizar abreviación y acrónimos solamente cuando sea significativamente más corto que el texto completo y cuando pueda ser entendido sin ambigüedad por el usuario.

12. Colocar una etiqueta significativa, con letras mayúsculas en las columnas, y si es apropiado, los renglones de tablas y listas. Continuar las etiquetas cuando una tabla o lista se extienda sobre más de una pantalla.

13. Ordenar los puntos en una tabla o lista para facilitar la búsqueda.

14. Colocar los puntos en una lista de múltiples columnas con columnas verticales y que se lea de izquierda a derecha de la pantalla.

15. Justificación a la izquierda en las columnas de datos alfabéticos; justificación a la derecha en las columnas con datos numéricos o alinearlos por el punto decimal u otro delimitador.

16. Insertar una línea en blanco después de cada cinco renglones en una columna muy extensa.

17. Colocar al menos dos espacios en blanco entre el punto final de una columna y el inicio de la siguiente columna.

18. Iniciar con un "1" y no con un "0", cuando los puntos de una lista estén etiquetados por números.

19. Separar cadenas de caracteres muy largas en pequeños grupos de tres o cuatro caracteres cada una.

20. Utilizar la iluminación o brillantez solamente para enfatizar la importancia de la información. No sobreutilizar esta directriz.

21. Utilizar el parpadeo y tonos audibles solamente para resaltar información crítica que requiera una respuesta inmediata por parte del usuario. Apagarla tan pronto como el usuario tenga su respuesta.

22. Seleccionar un método de iluminación que sea apropiado para el nivel de importancia de la información enfatizada y para la distribución en la pantalla en la cual será desplegada.

23. Cuando el usuario deba leer la información que está resaltada, no utilizar un método de iluminación que reduzca la legibilidad de la información enfatizada.

24. Seleccionar colores de la parte central del espectro de colores. Seleccionar combinaciones de color que se complementen.

25. Ser consistente en la distribución de despliegues y en la forma en que se resalte la información.

2.5.3 PARA LA INTRODUCCION DE DATOS :

La introducción de datos es muy importante debido a que esta parte es susceptible de fallas en el sistema, siendo muy importante orientar al usuario de los tipos de datos requeridos y realizar un filtrado de esta información.

1. Cuando el usuario deba transcribir datos diréctamente de una forma a la pantalla, la distribución de la pantalla deberá ser similar a la distribución de la forma.

2. Agrupar campos de datos en categorías lógicas en la pantalla; proporcionar un encabezado que describa el contenido de cada categoría.

3. Hacer áreas de la pantalla, que no se necesiten para la introducción de datos o de comandos, que sean inaccesibles para los usuarios.

4. Nunca exigir al usuario que introduzca información que ya esté disponible para el software o que pueda ser calculada por él.

5. No exigir al usuario que introduzca unidades dimensionales.

6. Permitir al usuario introducir datos para remplazo de caracteres.

7. Colocar una leyenda que describa los datos que son introducidos junto a cada campo de datos; incorporar memoria corriente dentro de la leyenda.

8. Justificar la entrada de datos automáticamente.

9. Desplegar valores de default en los campos de datos cuando sea apropiado.

10. Proporcionar ayuda de acuerdo al contexto, para los campos de datos de entrada.

11. Permitir al usuario moverse libremente a través de los campos en una pantalla de captura.

12. Permitir al usuario trabajar en toda la pantalla de captura antes de transmitir los datos al programa de aplicación.

13. Permitir al usuario salir de la pantalla de captura sin el llenado de cualquiera de los datos.

14. Cada tecla de función utilizada en una pantalla de captura deberá tener una y sólo una función.

15. Proveer a los usuarios con un mensaje de estado en la pantalla de captura o mensajes de ayuda que muestren el mapeo de las teclas de funciones.

16. Ser consistente en el diseño de las pantallas de captura, campos y procedimientos.

2.5.4 COMUNICACION FLUIDA CON DOCUMENTACION EN LINEA :

La comunicación fluída con documentación en línea se refiere básicamente a la ayuda proporcionada al usuario cuando interactua con la computadora, esta ayuda son todos los mensajes mostrados al usuario para indicar el manejo de la interfaz, así como los textos de ayuda por petición del usuario. Para lograr el objetivo mencionado se recomienda:

1. Escribir documentación en línea totalmente en el idioma del usuario.

2. Utilizar sólo verbos de acción para describir procedimientos. No utilizar sustantivos para reemplazar pronombres, verbos o adjetivos.

3. Describir procedimientos en orden lógico.

4. Utilizar palabras en el lenguaje especializado del usuario, y no términos sofisticados de computación.

5. Permitir el humor en la documentación en línea.

6. Utilizar mensajes de estado para decirle al usuario lo que está haciendo el software, dónde se encuentra el usuario dentro de una secuencia, y qué opciones tiene el usuario seleccionadas o qué opciones tiene disponibles.

7. Utilizar prompts para preguntar al usuario que opción escoge o introducir un dato o comando. Ser tan específico como sea posible.

8. Cuando los valores de default sean permitidos con los prompts, indicar claramente que valor de default deberá ser inicializado.

9. Diseñar el software para checar los errores obvios.

10. Ser tan específicos como sea posible en la descripción de la causa de un error. No utilizar códigos de error.

11. No asignarle la culpa al usuario o al softwre en un mensaje de error. Utilizar un tono neutral.

12. Siempre que sea posible, el mensaje de error deberá indicar que acción correctiva deberá tomar el usuario.

13. Considere la descripción de mensajes de error en más de un nivel de detalle.

14. Ser consistentes en el formato, redacción y colocación de los mensajes.

15. Planear, bosquejar y evaluar el texto de ayuda en línea.

16. Escribir en pocas palabras oraciones completas, puntualizando cada una de las oraciones.

17. Escribir oraciones en forma afirmativa o de manera negativa simple. Permitir la voz pasiva y no utilizar la doble negación.

18. Escribir párrafos cortos.

19. Utilizar listas numeradas y tablas que facilitan encontrar la información más importante. Dejar suficiente espacio abierto.

20. Utilizar ejemplos que muestren a los usuarios lo que deben escribir y los resultados que obtendrán.

21. No esperar que el usuario lea más de tres pantallas de ayuda a un mismo tiempo.

22. Utilizar ayuda en línea para explicar conceptos, procedimientos, mensajes, opciones de menúes, comandos, palabras, teclas de función y formatos.

23. Proveer a los usuarios con una orientación de la estructura del software.

24. Siempre que sea posible, desplegar textos de ayuda en las pantallas junto con la aplicación.

25. Proporcionar una ruta directa de regreso a la tarea de aplicación.

26. Ser consistente en el formato y en la redacción del texto de ayuda.

27. Seguir las directrices para el despliegue de texto en los mensajes de ayuda.

(12) Idem,

(13) Ibidem, p. 19

(14) Ibiden, p. 20

(15) Luqui and Valdis Berzins. "Rapidly Prototyping Real-Time Systems", (En: IEEE Software), (September 1988) p. 25

(16) Idem,

(17) H. R. Hartson and D. Hix. op. cit. p.46

(18) Idem,

(19) Luqui and Valdis Berzins. op. cit. p. 30

(20) Luqui, P.D. Barnes and M. Zyda, op. cit. p. 199

(21) A. Maizener, D. Thonon, "Un prototype pour la spécification des alarmes de synthèse au futur dispatching national", (En: Bulletin de la direction des études et recherches) Serie: B, Num.: 1 ; p. 33

(12) Idem,

(13) Ibidem, p. 19

(14) Ibidem, p. 20

(15) Luqui and Valdis Berzins. "Rapidly Prototyping Real-Time Systems", (En: IEEE Software), (September 1988) p. 25

(16) Idem,

(17) H. R. Hartson and D. Hix. op. cit. p.46

(18) Idem,

(19) Luqui and Valdis Berzins. op. cit. p. 30

(20) Luqui, P.D. Barnes and M. Zyda, op. cit. p. 199

(21) A. Maizener, D. Thonon, "Un prototype pour la spécification des alarmes de synthèse au futur dispatching national", (En: Bulletin de la direction des études et recherches) Serie: B, Num.: 1 ; p. 33

CAPITULO III

CRITERIOS GENERALES DE DISEÑO

3.1 CONSIDERACIONES.

En el presente capítulo se muestran los criterios empleados para implantar el diseño de la utilería, manifestando los antecedentes que llevan a tomar las decisiones de diseño de la utilería, para esto, es necesario evaluar los conceptos vistos en el capítulo anterior y acoplar aquellos que fundamenten el desarrollo de la utilería planteada en esta tesis.

3.2 EVALUACION.

3.2.1 EL PERFIL DEL USUARIO.

El principio de diseño ergonómico indica "la adaptación del trabajo al hombre" [HER89], por lo que es necesario considerar el tipo de usuario de la interfaz, para así poder tomar las decisiones necesarias que permitan realizar la adaptación de la interfaz de manera eficiente.

El tipo de interfaz que se pretende diseñar con la utilería proporcionada por esta tesis, está enfocada al control y supervisión de plantas industriales, en donde se emplean diagramas

mímicos para dicho propósito; los usuarios potenciales de esta utilería son: el operador de la planta, el diseñador de la interfaz, los jefes de turno y el personal de mantenimiento.

Para llegar a una adaptación de la interfaz a estos usuarios, se tendrá que considerar el lugar de trabajo y las tareas de quienes ahí van a laborar [HER89], para poder definir las características funcionales y el alcance de los diálogos de la interfaz.

Algunos ejemplos de las tareas que debe realizar el operador [VIL84b] y que debe de tener conocimiento el diseñador de la interfaz, son :

1. formulación de objetivos,
2. recolección de valores (datos),
3. tratamiento y manipulación de datos,
4. vigilancia,
5. acciones de control,
6. comando secuencial,
7. optimización,
8. comunicación con otros operadores,
9. operaciones manuales,
10. detección de fallas,
11. diagnóstico de fallas,
12. corrección de fallas,
13. prevención de accidentes y
14. acciones de seguridad en caso de accidentes.

Tomando en cuenta las necesidades del operador, la utilería debe permitir el desarrollo de una interfaz que accese la información de acuerdo a los principios :

1. De manera periódica.- Repaso y actualización de imágenes [VIL84b].

2. A la demanda del operador.- Selección de variables de poca importancia [VIL84b].

3. De manera automática.- los eventos anormales provocan el despliegue de informaciones, por ejemplo, diálogos de alármas y monitor de secuencia de eventos [VIL84b].

De acuerdo a los anteriores principios se pretende realizar el refrescamiento dinámico de la información que muestre el estado de operación de la planta, este despliegue será numérico y con cambios de color del mismo; por lo tanto, se realizará el diseño de la base de datos que guardará dicha información, tratando de tener un manejo eficiente de la información y que sea susceptible de mantenimiento para posibles mejoras a esta versión.

Como se ha venido observando, al usuario que más énfasis se le ha dado, es al operador, y mediante el concepto de implantación rápida de prototipos, independencia de diálogo y de diseño ergonómico (básicamente), se pretende incorporar al operador en la etapa de diseño y especificación, ya que este usuario finalmente es quien evalúa el funcionamiento de la interfaz y por lo tanto de la operación del sistema en general.

Se pretende que el manejo de la utilería y de los prototipos de interfaces generadas con ésta, utilicen mecanismos de acceso que de acuerdo a Villavicencio [VIL84b] sean:

1. Sencillos de utilizar,
2. No obliguen al usuario a memorizar códigos informáticos u ocasionen búsquedas tediosas de teclas, y sean,
3. Rápidos, donde el tiempo requerido para obtener información sea del orden de 2 segundos.

La experiencia de investigadores del IIE en otras interfaces informatizadas, señala que si no se cumplen estos requisitos, en general, los operadores tienden a no utilizar los diálogos, es decir, no consultan informaciones importantes y, por lo mismo, no realizan las maniobras o correcciones adecuadas, repercutiendo en la eficiencia de los procesos y de la vida útil de los equipos.

3.2.2 CONSIDERACIONES DEL DIALOGO HOMBRE/MAQUINA.

Como se comentó anteriormente, los usuarios de esta utilería son los diseñadores y los operadores, por lo que se debe de considerar el tipo de diálogo que requieren estos usuarios para el manejo de la utilería y los diálogos que será capaz de diseñar mediante la utilería.

En general, estos diálogos deben de estar basados en los principios de diseño de una interfaz presentados en el capítulo dos (Capítulo II, pp. 15-21), porque son puntos necesarios que guían el logro de una interfaz de buena calidad, la lógica nos indica que todos estos puntos tienen cavida en el diseño ya que si no tuviéramos estas guías, podríamos omitir algún concepto importante y demeritar nuestro trabajo.

Existen dos tipos de diálogos que se deben de considerar, el diálogo que empleará la utilería para interactuar con el diseñador del diálogo, y el diálogo que proporcionará la utilería para que se incorpore a la interfaz.

El diálogo de la utilería estará enfocado principalmente al diseñador, mientras que el diálogo que es capaz de generar la utilería, para la interfaz, debe de estar enfocada al operador.

Para la implantación de los diálogos se emplearán los siguientes criterios [HER89] :

1. Se buscará una definición, tamaño y distribución de los elementos que constituyan las imágenes de los diálogos.

2. Se evitará la sobrecarga mental, para aumentar la precisión y la velocidad de respuesta del usuario, disminuyendo la frecuencia de errores.

3. Se evitará que el usuario experto se aburra con preguntas simples, ayudando al usuario novato a aprender el proceso de operación.

4. No se pedirá información que ya se encuentra en el sistema.

5. Se estructurará de forma tal, que nunca sorprenda al usuario.

6. Se permitirá que el usuario cometa errores, dándole la oportunidad de corregirlos en lugar de abortar al programa.

Una característica que se ha detectado en los operadores, es el agrado por las teclas de función o dedicadas, ya que les evitan estar escribiendo comandos o funciones, por lo que se tendrá que crear un ambiente que facilite la creación o asignación de funciones a un grupo de teclas. También se ha observado que el uso de un ratón o "mouse" no es del agrado de los operadores [VIL84b], por lo que se deberá de recomendar no utilizarlo para la aplicación final, pero sin embargo, este dispositivo es de suma importancia para el diseñador, ya que es de gran utilidad cuando está elaborando los diagramas mímicos, o creando algún nuevo ícono.

Diálogo para operadores.

El diálogo que estará dirigido a los operadores debe de contemplar las características de éstos, como son los diversos estados psicológicos y de tensión, los factores ambientales como

temperatura, ruido, etc., además de las diversas necesidades de información que requiere el operador, como son el acceso a las variables del sistema, las diversas formas de presentación que requiere : diagramas mímicos, lista de variables, gráficas en pantalla e impresión de diversos tipos de información.

Para efectos de presentación y actualización, la información se estructurará en forma jerárquica, de acuerdo con principios de origen material: sistemas, subsistemas, grupos y variables.

También se ha observado que una interfaz que interactúa con el usuario por medio de íconos y menúes es más aceptada que una que utiliza comandos alfanuméricos, ya que el tecleo de cadenas alfanuméricas ocasiona la ocurrencia de errores por parte del usuario, obligándolo a recordar muchos comandos y el significado de cada uno de ellos, además de que puede cometer errores al momento de teclear, esto se evita en gran medida utilizando íconos y menúes, por ser éstos más naturales para el ser humano; por ejemplo, los íconos permiten al usuario aprender en poco tiempo su significado y ayudan a disminuir el número de errores, sólo hay que estar concientes de que éstos íconos deben estar bien diseñados para evitar ambigüedades.

En general se debe de considerar la experiencia adquirida en el desarrollo de éstas interfaces, como el presentar a los usuarios informaciones relativas a las tareas que debe de realizar el operador, ya que la experiencia ha mostrado que éste método es el más eficaz [VIL84b].

Es importante considerar una de las recomendaciones que hace Villavicencio [VIL84b] sobre tener cuidado en las interpretaciones y sugerencias de los futuros usuarios, ya que algunas experiencias han mostrado que estas sugerencias se basan en evaluaciones subjetivas y no siempre son las mejores desde el punto de vista ergonómico. En estos casos, será mejor realizar algunas imágenes de

prueba y evaluarlas con una muestra de futuros usuarios con el propósito de determinar la mejor manera de presentar la información; con esto se observa la importancia que tiene la generación rápida de prototipos, ya que permite realizar fácil y rápidamente estos cambios.

3.2.3 MODELO PARA LA INTERFAZ HOMBRE/MAQUINA.

Para seleccionar el tipo de modelo necesario en el desarrollo de la tesis, se analizaron cada tipo de modelado presentado en el capitulo dos (Capítulo II, pp. 26-31) : modelo para análisis de tareas, modelo con descripción estructural y el modelo de representación de interfaz; se elimina la posibilidad de emplear un modelo para análisis de tareas, debido a que este método es usado para analizar y describir los detalles de una actividad particular de un usuario, y como la utilería será manejada por diferentes usuarios, no se pueden saber las tareas que deberá realizar cada usuario y mucho menos analizarlas para conocer el tipo de interfaz que se requiere de acuerdo a las necesidades del usuario en particular.

El modelado por representación de la interfaz no puede aplicarse aquí, porque indica que deben tenerse esquemas para representación particular de instancias de la interacción hombre/máquina, y en base a cada instancia, se observa el tipo de diálogo necesario, y conjuntando éstos diálogos se forma la secuencia de interacción hombre/máquina, y como la utilería desarrollada no está ligada a ningún sistema en particular, no pueden preveerse las instancias particulares necesarias para desarrollar la interfaz.

La utilería planteada pretende ser una aplicación muy general dentro del contexto de desarrollo de interfaces hombre/máquina basadas en despliegues de diagramas mímicos, por lo que no debe de

depender de ningún sistema en particular.

Se observa la necesidad de emplear un modelo estructural para desarrollo de la interfaz. Se ha seleccionado un modelo estructural no ligüístico debido a que las interfaces desarrolladas estarán basadas en diálogos formados por diagramas mímicos, en general, y como se tiene la interrelación que guardan estos diagramas, sin importar las particularidades de cada diagrama, se puede establecer el modelo que servirá para los diálogos del sistema de interacción con el usuario.

Ahora, de acuerdo a lo expuesto en las consideraciones del diálogo hombre/máquina (Capítulo III, pp. 53-56), se concluye que el modelo no lingüístico es el recomendable para basarse y guiarse en el desarrollo del sistema de interacción; se eliminó el empleo de un lenguaje de comandos, el cual es una aproximación a la estructura de un lenguaje, por lo que no se tiene la necesidad de manejar un modelo lingüístico.

Concluyendo, se observa la necesidad del empleo de un modelo, el cual servirá de guía en el desarrollo de la utilería y del tipo de diálogo que permitirá desarrollar, como modelo a seguir será un modelo estructural no lingüístico basado en eventos de interacción, ya que se aplica a diálogos secuenciales, en donde cada evento estará compuesto por un "prompt", una entrada del usuario, una acción de procesamiento del sistema y un control del flujo para determinar el siguiente evento de interacción, aunque en este caso la accion de procesamiento del sistema sea independiente, la comunicación con la interfaz hombre/máquina se hará mediante una base de datos que actualizará el sistema de procesamiento o de cálculo del sistema en general.

3.2.4 INDEPENDENCIA DE DIALOGO.

Como se mencionó en el capítulo anterior (Capítulo II, pp. 24-26), la independencia de diálogo es un concepto de suma importancia en el desarrollo actual de software, debido a que permite una independencia entre sistemas de cálculo y procesamiento de los sistemas de interacción hombre/máquina, y en general de un sistema o subsistema con otro.

Este concepto será aplicado tanto al desarrollo e implantación de esta utilería como a las interfaces desarrolladas por la misma, ya que evitan estar ligados en forma estricta a un sistema en particular, permitiendo modificaciones de la manera más flexible, ya que los cambios realizados, tanto a la utilería como a la interfaz en desarrollo, no repercuten en la codificación del sistema de cálculo de aplicación para el cual se está desarrollando, siendo esto muy importante ya que permite desarrollos más económicos y en menor tiempo, tanto en su implantación como en su mantenimiento, siendo esto un punto muy importante desde el punto de vista de la ingeniería.

El concepto de independencia de diálogo es básicamente importante en el desarrollo de interfaces, debido a que se ha observado que éstas tienen una mayor aceptación cuando el usuario participa en su diseño, y para que éste diseño sea el óptimo, es necesario emplear un método de refinamiento iterativo, logrando con esto desarrollar una interfaz "al gusto del cliente" y sin tener que estar modificando el sistema de procesamiento o de cálculo, ya que éste tiene un "diálogo independiente".

3.2.5 GENERACION RAPIDA DE PROTOTIPOS.

La generación rápida de prototipos es un concepto ampliamente discutido a lo largo de esta tesis, y la importancia de su

aplicación en esta tesis radica básicamente en la incorporación del usuario al diseño de la interfaz, logrando una mayor especificación del sistema requerido, a la vez que se está trabajando con el desarrollo de la interfaz.

Otro punto a favor de la incorporación de éste concepto es la aceleración de los desarrollos de interfaces, siendo de gran ventaja en este tipo de proyectos, debido a que su implantación llega a ser muy tardada, como por ejemplo en los diseños realizados para la termoeléctrica de Tula (5 años) o en la de Manzanillo, donde todo el sistema se llevo mucho tiempo y en donde la interfaz hombre/máquina tuvo mucho que ver.

Ahora el concepto innovador de realizar este tipo de interfaces, de manera rápida y directamente sobre los requerimientos del usuario permite una mayor aceptación por parte de éste, ya que se le permite aportar su punto de vista, y hacer refinamiento iterativo de la interfaz solicitada en menor tiempo que los sistemas tradicionales.

3.2.6 HERRAMIENTAS DE APOYO.

En cuanto al manejo de los gráficos desarrollados en ésta tesis, los mímicos generados por el editor son manejados de acuerdo al sistema GKS [GKSa] [GKSb] (Graphic Kernel System), ya que en la actualidad es el lenguaje de gráficas que se utiliza en E.U. y a nivel internacional, además de que este sistema ha sido adoptado como norma de software de gráficos por la "International Standards Organization" (ISO) y por varias organizaciones de normas como el "American National Standards Institute" (ANSI). Este sistema cuenta con gran variedad de dispositivos gráficos y permite una independencia del dispositivo que se emplee, además de que cuenta con primitivas básicas de graficación que nos permiten ahorrar tiempo en el desarrollo de nuestras rutinas, evitándonos parte del

trabajo laborioso como son las primitivas básicas de salida de graficación.

3.2.7 EL EDITOR DE MIMICOS.

Como punto de partida se desarrolló, en una tesis previa, un editor de mímicos, el cual realiza la creación de la parte gráfica de la interfaz (pero solamente en su parte estática, es decir, sin muestrear la base de datos para mostrar el estado de operación de la planta), sin mostrar el manejo de la interacción existente de un mímico con otro, como lo requiere todo el diálogo de interacción de la interfaz y que será el centro de trabajo de esta tesis [HER89].

El editor de mímicos es la herramienta capáz de efectuar la edición de diagramas mímicos, los cuales podrán ser reutilizables y adaptables a las necesidades específicas de otras plantas.

Este es un editor de mímicos para interfaces gráficas hombre/máquina para la presentación de información en tiempo real en procesos de alto riesgo. El editor de mímicos es amigable, fácil de usar, rápido y eficiente, evita al usuario el trabajo tedioso, minimiza la posibilidad de errores y el continuo despliegue de mensajes de error.

Las características que el editor tiene son :

- Selección de comandos.
- Manejo de opciones por medio del teclado y/o un ratón.
- Dibujos de elementos gráficos sencillos (círculos, rectángulos, etc.).
- Llenado de polígonos.
- Borrado de figuras.
- Líneas elásticas (rubber banding).

- Permite que el usuario especifique figuras o áreas de la pantalla para manipularlas.
- Traslado de figuras.
- Rotación de figuras.
- Copiado de figuras.
- Administración de una biblioteca de imágenes.
- Especificación de ventanas para gráficas de tendencias.
- Especificación de variables para monitoreo.

El editor produce únicamente íconos del tipo dato, ya que éstos sólo son de referencia al mundo físico y no se pretende que su identificación provoque acción alguna. Para la selección de comandos y el manejo de opciones se usaron menúes y apuntadores (teclado y/o ratón) [HER89].

El sistema proporciona al usuario mecanismos de ayuda en el caso de que cometa errores o que eviten que el usuario los pueda llegar a cometer. Estos mecanismos son despliegues de mensajes que indican la naturaleza del error cometido o mensajes en los que se pide al usuario la confirmación para realizar alguna acción (al borrar archivos, por ejemplo).

3.3 SELECCION FINAL.

La utilería planteada en esta tesis deberá de producir una herramienta de cómputo práctica, que incorpore los conceptos actuales para el diseño de interfaces hombre/máquina, es decir, consideraciones de los factores humanos, que la interfaz sea amigable con el usuario, eficiente, fácil de usar y que produzca resultados rápidos (prototipo de la interfaz); para alcanzar los objetivos mencionados se tomaron en cuenta las siguientes consideraciones:

Se tiene la necesidad de emplear gráficos en los prototipos a diseñar, pero las facilidades que proporcionan las versiones turbo de los lenguajes como PASCAL, C, PROLOG, BASIC, etc., no se emplearon, debido a la decisión de emplear GKS como paquete de graficación (como se explica más adelante), por el mismo motivo se tuvo que elegir entre los lenguajes FORTRAN Y C que tienen la capacidad de manejar las rutinas de GKS para realizar la graficación.

Para la implantación de los programas de computadora se opta por desarrollarlos en el lenguaje C, en su versión "Lattice C" [LATCa] [LATCb], ya que fue el lenguaje con el que se implementó el editor de mímicos, ahora las modificaciones necesarias para adaptar el editor a nuestras necesidades se harán en ese mismo lenguaje.

La versión de "Lattice C" fue seleccionada debido a que es la que mejor soporta el manejo de las rutinas de GKS, ya que GKS fue implantado con "Lattice C", a diferencia de algunos problemas que se tuvieron con TURBO C al tratar de ligarlo con GKS, además de que "Lattice C" es uno de los compiladores con los que cuenta el Instituto de Investigaciones Eléctricas y que es compatible con la versión de GKS disponible.

Se decide continuar manejando el sistema GKS en esta tesis para el manejo de las gráficas y despliegues que muestren el estado de operación de la planta, continuando con el mismo criterio del editor de mímicos, debido a las ventajas que presenta respecto a la capacidad de adaptar nuevos manejadores ("drivers") y a las características mencionadas en HERRAMIENTAS DE APOYO (Capítulo III, p. 59).

Se decide realizar la implantación de esta utilería en computadoras personales del tipo PC y compatibles, debido a la facilidad de acceso de este tipo de recursos, a diferencia de otros diseños de interfaz hombre/máquina realizados por el IIE en equipos

grandes.

Actualmente se descarta la implantación de este sistema en equipos "workstation" que planea adquirir el IIE debido a que su llegada será muy tardada y no conviene esperar tanto tiempo para iniciar el desarrollo, sin embargo se cuenta con suficiente equipo tipo PC para lograr observar las ventajas que presenta esta utilería, además de la transportabilidad que proporcionará en computadoras personales; también se selecciona la computadora personal PC debido a la generalidad de aplicación de esta utilería, ya que no está diseñada para estar ligada a un sistema en particular y sin embargo podría ser utilizada fácilmente en diferentes aplicaciones mediante computadoras PC.

En general se aplicarán la mayor parte de los conceptos expuestos en el capítulo anterior sobre diseño de interfaces hombre/máquina, además de los conceptos modernos de implantación rápida de prototipos e indepencia de diálogo.

3.4 ALCANCE DE LA TESIS.

El desarrollo de la utilería de cómputo para la implantación rápida de prototipos de interfaces hombre/máquina abarca los siguientes puntos :

1. Generación del sistema de interacción (diálogos) que tiene lugar entre el hombre y la computadora, con el propósito de realizar actividades de supervisión y monitoreo dentro de una planta industrial, principalmente de la industria eléctrica.

   Será un sistema amigable y fácil de usar, mediante el manejo de menúes, teclas de función, ventanas, textos de ayuda, mensajes de error y estado de operación de la utilería.

2. Edición de diagramas mímicos que formarán parte del diálogo hombre/máquina.

   Permitirá la edición de mímicos de manera amigable, utilizando técnicas de gran utilidad para el diseñador, como son las diversas transformaciones : escalamiento, rotación, copiado, etc., y la edición de diagramas e íconos ya existentes.

3. Permitirá la implantación rápida de prototipos de interfaces hombre/máquina basadas en mímicos, con el objeto de observar rápidamente el funcionamiento de la interfaz y poder realizar los cambios o ajustes necesarios con la prontitud necesaria para encaminar el desarrollo del sistema final de acuerdo a los verdaderos requerimientos del sistema.

4. Manipulación de diagramas mímicos para formar el diálogo o secuencia de diagramas dentro del diálogo.

   Deberá de permitir la manipulación de diagramas mímicos para establecer la interfaz con el usuario, definiendo la jerarquía que existe entre los diagramas, indicando qué diagramas son de vista general y cuáles son de detalle, e indicando la secuencia de presentación de los diagramas.

5. Manipulación de la información de refresco o de actualización temporal sobre los diagramas mímicos mediante una rutina de simulación que emplea un generador de eventos en substitución del sistema de aplicación real.

6. Generador de los casos de prueba o generador de eventos.

   Deberá de permitir observar el funcionamiento de la interfaz y permitir modificarla si es necesario; esto se realizará mediante la generación de casos de prueba que permitan

observar el funcionamiento de la interfaz y así poder evaluar a tiempo los ajustes que requiera el sistema, pudiendose corregir rápidamente debido a su fácil manejo.

7. Diseño de una base de datos que enlace la parte estática de la interfaz (diagramas mímicos) con la información de actualización temporal, coordinando de manera eficiente el refresco de la información dinámica dentro de la interfaz.

8. Permitirá la participación del usuario (los operadores principalmente) en la implantación de la interfaz, logrando incorporar al usuario en la etapa de especificación y diseño del sistema, permitiendo encaminar al proyecto hacia el sistema deseado y logrando una mayor aceptación del usuario hacia el sistema final.

9. Será un sistema portable a computadoras personales del tipo PC y compatibles, con la observación de que deberán de soportar el manejo de gráficos estos equipos.

CAPITULO IV

ESPECIFICACION DE LA UTILERIA

4.1 ESPECIFICACION DE LA UTILERIA.

La especificación de la utilería nos muestra las funciones del sistema, así como los métodos de solución en un lenguaje especializado, familiar a los analistas de sistemas.

Es aquí donde se analiza con mucho detalle la interacción del sistema con el exterior, en particular con el usuario (interfaz hombre/máquina).

La especificación de la utilería consta de cuatro partes: generalidades de la especificación, descripción general, modelo conceptual y descripción de las funciones.

4.2 GENERALIDADES DE LA ESPECIFICACION.

4.2.1 PROPOSITO.

Esta sección describe de manera general el software y las funciones que deberá realizar la utilería desarrollada.

Las especificaciones están enfocadas al diseñador de diálogo, para mostrar de manera general la capacidad del sistema, así como

proporcionar un panorama claro al personal que decida modificar el sistema.

4.2.2 ALCANCE.

La utilería de cómputo a desarrollar, será un sistema que permitirá realizar de manera rápida, la interfaz hombre/máquina que necesitan los sistemas de supervisión y control de plantas industriales.

La utilería permitirá realizar prototipos de interfaces de manera sistemática, creando los diagramas mímicos que formarán parte del diálogo de la interfaz mediante un editor de mímicos, estos diagramas podrán corregirse o modificarse hasta que satisfagan las necesidades del usuario.

Por medio de la utilería se establecerá la jerarquía o secuencia de los diagramas mímicos dentro de la interfaz, es decir, se programará el diálogo gráfico de la interfaz creandose el prototipo de la misma. También se podrán realizar modificaciones a un prototipo ya definido.

Por último, la utilería permitirá observar el funcionamiento de la interfaz mediante un generador de eventos que simule cambios en las variables de la base de datos, y se reflejen en el diagrama mímico que le corresponda, el cambio de valor de las variables se observará en pantalla junto con un cambio de color del despliegue de la variable, de acuerdo a las normas de la Comisión Federal de Electricidad (CFE).

4.3 DESCRIPCION GENERAL.

Esta descripción abarca las funciones, el hardware y el software como parte de la especificación del sistema.

4.3.1 FUNCIONES DEL SISTEMA.

La utilería a desarrollar estará formada en primer lugar por un editor de mímicos, que permitirá la edición tanto de diagramas mímicos como de los íconos que lo forman, ya sea creando o modificando alguno que ya exista.

Se podrán realizar las siguientes funciones para íconos y para diagramas :

1) Hacer figura
2) Modificar tamaño
3) Copiar
4) Mover
5) Rotar
6) Borrar
7) Leer ícono o diagrama
8) Poner texto y
9) Almacenar ícono o diagrama.

Algunas funciones particulares para diagramas serán :

1) Hacer bit-map,
2) obtener impresión,
3) crear variable,
4) modificar variable y,
5) borrar variable.

En donde el hacer bit-map es una de las formas de almacenar diagramas mímicos en la biblioteca, la otra forma es en diagramas de edición. La forma de bit-map es la forma definitiva del diagrama que será utilizado por la aplicación, y tiene como fin el despliege

del diagrama en el menor tiempo posible.

Otra función de la utilería será la edición del prototipo, donde se indique la jerarquía de los diagramas mímicos, es decir, se definirá la interdependencia de los diagramas, utilizando una estructura de árbol donde el nodo raíz represente una vista general de la planta y los nodos hijos o de nivel superior serán representaciones de diferentes niveles de detalle.

Las funciones de edición del prototipo son:

1. Crear prototipo
2. Modificar prototipo
3. Leer prototipo
4. Almacenar prototipo
5. Borrar prototipo

También existirá una parte encargada de probar el prototipo diseñado mediante una función para generar eventos, que actualice la base datos del sistema y una función que la monitoré, posteriormente se actualizarán los valores de las variables en los diagramas correspondientes, con el fin de observar el funcionamiento del prototipo, desplegando el valor numérico de esa variable con cambio de color, mostrando así el estado del proceso en la planta.

4.3.2 HARDWARE Y SOFTWARE DEL SISTEMA.

El sistema que se desarrolla en esta tesis será implantado en el lenguaje de programación C, en la versión Lattice C 3.0, y el paquete de graficación utilizado será el GKS de GSS, cuyas librerías se pueden ligar con la versión de Lattice C utilizada. Para manejo de ventanas y diferentes tipos de despliegue de menúes, se empleará el paquete Pforce [PFORCE].

El sistema se desarrollará en una microcomputadora Olivetti PC-XT, con 1 drive y disco duro, monitor a color y bajo la versión 3.2 del sistema operativo DOS. El sistema terminado deberá correr en microcomputadoras PC XT y AT compatibles con IBM, bajo el sistema operativo DOS.

4.4 MODELO CONCEPTUAL.

Como se mencionó en el capítulo tres (Capítulo III, pp. 56-57), el modelo a seguir es un modelo estructural no lingüístico, basado en eventos de interacción, en el cual la premisa básica es que el diálogo secuencial esté compuesto de un "prompt" del sistema, una entrada del usuario, una acción de procesamiento por parte del sistema y un control de flujo para determinar el siguiente evento de interacción [HAR89]. El ciclo completo se muestra en los párrafos siguientes.

En el modelo indicado se tendrá que manejar un "prompt" para cada evento, al igual que el filtrado de las entradas de datos del usuario, la información adicional de ayuda y los mensajes de error serán particulares de cada evento.

El modelo conceptual empleado en el desarrollo de esta tesis es el siguiente :

I Prompt
II Entrada (obtenida del usuario o del valor de default)
III Salida o escape : Si la entrada = "escape", entonces
    III.A Actualizar el indicador del siguiente evento.
    III.B Fin del ciclo.
IV Ayuda : Si la entrada = "ayuda", entonces
    IV.A Desplegar información adicional.
    IV.B Fin del ciclo.
V Chequeo o revisión : Aplicar revisión o filtrado a las

entradas. Si hay error, entonces

V.A Reportar error.

V.B Fin del ciclo.

VI Acción : invocar al procesamiento relacionado.

VII Control del flujo : Actualizar el indicador de siguiente evento.

Los eventos que se consideran para esta utilería son los siguientes :

I Edición de mímicos.

I.A Despliegue de menúes.

I.B Teclas de función o dedicadas.

I.C Preguntas y respuestas.

II Edición de prototipos.

II.A Programación del diálogo gráfico mediante preguntas y respuestas.

II.B Generación de casos de prueba.

II.B.1 Teclas de función.

II.B.2 Despliegue de ayudas.

II.B.3 Despliegue de menúes.

4.5 DESCRIPCION DE LAS FUNCIONES.

Las funciones están agrupadas en bloques de acuerdo al subsistema que las maneja, en total son 31 funciones. Estos subsistemas son : I) el editor de mímicos, II) el editor de prototipos y III) el simulador de prototipos.

La descripción de las funciones se presenta en el siguiente orden: parámetros de entrada, parámetros de salida y proceso que realiza la función.

I. FUNCIONES DEL EDITOR DE MIMICOS:

El editor de mímicos consta de 18 funciones, el total de estas funciones cumple con las características descritas en la sección 3.2.7 (Capítulo III, PP. 60-61), éstas son:

I.A HACER FIGURA.

Entradas

- Figura a dibujar : POLILINEA, CIRCULO, ARCO, POLIGONO, REBANADA y RECTANGULO.
- Atributos geométricos: posición, radio, lados, etc.
- Atributos gráficos: ancho y tipo de línea, patrón de llenado, color, etc.

Salida

- La figura escogida con los atributos indicados.

Proceso

- Identificar la figura y dibujarla con sus atributos geométricos y gráficos correspondientes.

I.B MODIFICAR TAMAÑO.

Entradas

- Figura.
- Atributo geométrico: nuevo factor de escala.

Salida.

- Figura con el nuevo tamaño elegido.

Proceso.

- Identificar la figura, calcular el nuevo escalamiento y modificarlo.

I.C COPIAR.

Entradas.

- Figura de la pantalla a copiar.
- Lugar o lugares donde dibujar una o más copias de la figura escogida.

Salida.

- Copia de la figura seleccionada en el lugar indicado.

Proceso.

- Identificar el objeto seleccionado y copiarlo tantas veces como quiera el usuario.

I.D MOVER.

Entradas.

- Figura a mover.
- Nueva posición de la figura.

Salida.

- Cambio de lugar de la figura escogida.

Proceso.

- Identificar el objeto seleccionado, calcular su desplazamiento y dibujarlo en la posición indicada.

I.E ROTAR.

Entradas.

- Figura en la pantalla a rotar.
- Atributos geométricos: punto y ángulo de rotación.

Salida.

- Figura rotada conforme a las especificaciones.

Proceso.

- Identificar la figura y cambiar su atributo de rotación.

I.F BORRAR.

Entrada.

- Figura a borrar.

Salida.

- Borrado en la pantalla de la figura indicada.

Proceso.

- Identificar la figura y borrarla.

I.G LEER ICONO.

Entradas.

- Icono a leer.
- Atributo geométrico: posición en la cual dibujarlo.

Salida.

- Despliege del ícono en la posición elegida.

Proceso.

- Identificar el ícono a cargar y dibujarlo en la posición indicada.

I.H PONER TEXTO.

Entradas.

- Texto a escribir.
- Posición en la cual desplegarlo.
- Atributos gráficos: foreground, espaciamiento, expansión y font.

Salida.

- Texto desplegado en la posición indicada con los atributos indicados.

Proceso.

- Desplegar el texto indicado en la posición seleccionada usando los atributos escogidos.

I.I LEER DIAGRAMA.

Entrada.

- Archivo del cual se va a leer el diagrama.

Salida.

- Despliegue del diagrama indicado o de un mensaje de error en el caso de que no exista el archivo o de que no sea un archivo de diagrama.

Proceso.

- Búsqueda en disco del archivo a leer, ejecución de los segmentos que componen el diagrama del archivo

especificado o de un mensaje de error si no existe el archivo o no es un archivo de diagrama.

I.J ALMACENAR ICONO.

Entrada.

- Identificador para el ícono.

Salida.

- Ninguna o un mensaje de error en caso de que ya exista un ícono con el mismo identificador.

Proceso.

- Busqueda del identificador indicado, si existe desplegar un mensaje de error, si no existe se almacena en disco.

I.K BORRAR ICONO.

Entrada.

- Identificador del ícono a borrar.

Salida.

- Ninguna o un mensaje de error en caso de que no exista un ícono con el identificador indicado.

Proceso.

- Búsqueda del identificador indicado, si existe borrar el ícono del disco, si no existe desplegar un mensaje de error.

I.L ALMACENAR DIAGRAMA.

Entrada.

- Nombre con el que se va almacenar el diagrama.

Salida.

- Ninguna o un mensaje de error en caso de que ya exista un archivo con el mismo nombre.

Proceso.

- Búsqueda del archivo indicado, si existe desplegar un

mensaje de error, si no existe se almacena en disco.

I.M BORRAR DIAGRAMA.

Entrada.

- Nombre del diagrama a borrar.

Salida.

- Ninguna o un mensaje de error en caso de que no exista un archivo con el nombre indicado.

Proceso.

- Búsqueda del archivo indicado, si existe borrar el archivo de disco, si no existe desplegar un mensaje de error.

I.N HACER BIT MAP.

Entrada.

- Nombre del archivo del que se va ha generar el bit map.

Salida.

- Ninguna o un mensaje de error en caso de que no exista un archivo con el nombre indicado, un mensaje de error en caso de que ya exista un bit map de ese archivo, el cual será destruido, o un mensaje de error en caso de que el archivo elegido no sea un archivo de diagrama.

Proceso.

- Se convierte el diagrama del archivo a bits del raster y se almacena en disco.

I.O IMPRESION.

Entrada.

- Diagrama a imprimir.

Salida.

- Mensaje del estado de la impresión.

Proceso.

- Se manda a la impresora el hardcopy del diagrama en pantalla.

I.P CREAR VARIABLE.

Entradas.

- Nombre del archivo de diagrama asociado.
- Variable: su nombre, posición y tipo de variable.

Salida.

- Ninguna o un mensaje de error en caso de que no exista el archivo de diagrama asociado.

Proceso.

- Captura de la variable, su posición, tipo y almacenamiento en disco.

I.Q MODIFICAR VARIABLE.

Entradas.

- Nombre del archivo de variables.
- Variable a modificar, su posición y tipo de variable.

Salida.

- Archivo con las modificaciones solicitadas.

Proceso.

- Alta, baja o modificación de la variable, su posición y tipo de variable.

I.R BORRAR ARCHIVO DE VARIABLES.

Entrada.

- Nombre del archivo de variables a borrar.

Salida.

- Ninguna o un mensaje de error en caso de que no exista un archivo con el nombre indicado.

Proceso.

- Búsqueda del archivo indicado, si existe borrar el

archivo de disco, si no existe desplegar un mensaje de error.

II. FUNCIONES DEL EDITOR DE PROTOTIPOS.

II.A DESPLEGAR MENU.

Entrada.

- Número de opciones.

Salida.

- Opción elegida.

Proceso.

- Desplegar el menú de opciones de acuerdo al evento de interacción de que se trate, solicitar el ingreso de una opción y aceptar únicamente una opción válida.

II.B AYUDA DE LA UTILERIA.

Entradas.

- Nombre del evento activo.
- Nombre del archivo de ayuda del evento.

Salida.

- Texto de ayuda desplegado en la pantalla o un mensaje de error si el archivo de ayuda no existe.

Proceso.

- De acuerdo con el evento de interacción que se esté atendiendo, se desplegará el texto de ayuda del archivo indicado. Si no existe el archivo, se desplegará un mensaje de error.

II.C MENSAJES DE ERROR.

Entrada.

- Número de mensaje de error.

Salida.

- Despliegue en pantalla del mensaje de error.

Proceso.

- Selección del mensaje de error de acuerdo al error indicado y despliegue en pantalla en la línea de estado del mensaje correspondiente.

II.D CREAR PROTOTIPO.

Entrada.

- Nombre del prototipo.

Salida.

- Tablas que definen la estructura del prototipo.

Proceso.

- Captura de los nombres de los diagramas que definen al prototipo, interrelación de diagramas y definición de las tablas para manipular el prototipo.

II.E MODIFICAR PROTOTIPO.

Entrada.

- Tablas que definen la estructura del prototipo.

Salida.

- Tablas actualizadas que definen al prototipo.

Proceso.

- Se solicita el nombre de un diagrama, al cual se podrá cambiar de nivel de detalle, el diagrama que lo referencia (padre) y a los que hace referencia (hijos), así como las ligas de diagrama anterior y siguiente.

II.F LEER PROTOTIPO.

Entrada.

- Nombre del prototipo a leer.

Salida.

- Tablas que definen la estructura del prototipo.

Proceso.

- Apertura y lectura de los archivos que definen al prototipo, mensaje de error si no existe el prototipo especificado.

II.G BORRAR PROTOTIPO.

Entrada.

- Nombre del prototipo a borrar.

Salida.

- Despliegue en pantalla del prototipo borrado.

Proceso.

- Borrado en disco de los archivos que definen al prototipo.

II.H ALMACENAR PROTOTIPO.

Entrada.

- Nombre del prototipo a almacenar en disco.

Salida.

- Archivos en disco que definen al prototipo o mensajes de aviso si ya existen esos archivos.

Proceso.

- Escritura en disco a los archivos que definen al prototipo, si ya existen los archivos se pide una confirmación para escribir sobre ellos.

III. FUNCIONES DEL SIMULADOR DE PROTOTIPOS.

III.A RASTREAR TECLADO.

Entrada.

- Ninguna.

Salida.

- Bandera indicadora de tecla presionada y en su caso la tecla presionada.

Proceso.

- Monitorear el teclado, atendiendo y filtrando las entradas del usuario; cuando no se presiona el teclado se indica que las rutinas: generar eventos y refresco de informacion se debe realizar, de lo contrario, se indica que hay un cambio de estado.

III.B GENERAR EVENTOS.

Entrada.

- Tablas de variables analógicas y digitales y sus atributos.

Salida.

- Tablas de variables con valores actualizados.

Proceso.

- Modificar las variables de acuerdo a su tipo de variable, las variables analógicas se incrementan o decrementan hasta llegar a sus límites superior e inferior respectivamente, y las variables digitales sólo cambian de estado normal al de alarma y viceversa.

III.C DESPLEGAR DIAGRAMA.

Entrada.

- Nombre del archivo del cual se va a leer el diagrama.

Salida.

- Despliegue del diagrama indicado o de un mensaje de error si no existe el archivo o no es archivo de diagrama.

Proceso.

- Búsqueda en disco del archivo a leer, ejecución de los segmentos que componen el diagrama, o despliegue de un mensaje de error si no existe el archivo o no es archivo de diagrama.

III.D ACTUALIZAR POR VALOR.

Entradas.

- Nombre de la variable.
- Posición en pantalla de la variable.
- Diagrama al que pertenece la variable.
- Límites de tolerancia.
- Valor actual de la variable.

Salidas.

- Despliegue en pantalla con el valor actualizado en la posición indicada si está activo el diagrama al que pertenece.
- Mensaje de error si la variable no tiene asignada una posición y/o diagrama mímico.

Proceso.

- Actualizar el valor de la variable en la pantalla si está activo el diagrama mímico al que pertenece. El valor a actualizar es directamente el valor actual de la variable y en la posición donde se declaró esa variable. El color del despliegue de la variable depende del estado de la variable: verde-normal, amarillo-precrítico y rojo-crítico.

III.E TRANSFERENCIA DE NIVEL.

Entradas.

- Número y nombre del nivel de detalle actual.
- Número y nombre del nivel de detalle superior.
- Número y nombre de los niveles de detalle inferiores.

Salida.

- Mímico del nivel solicitado o mensaje de aviso si no existen niveles a donde cambiarse.

Proceso.

- Se presentan los niveles de detalle a donde se puede cambiar, se captura la opción y se realiza el cambio de nivel junto con el despliegue del diagrama solicitado.

CAPITULO V

DISEÑO DE LA UTILERIA

5.1 DISEÑO DE LA UTILERIA PARA INTERFACES H/M.

El diseño de la utilería retoma la documentación de las funciones establecidas en el capítulo anterior, la especificación de la utilería se realizó mediante un lenguaje natural y ahora el diseño se encargará de transformarlo en un lenguaje propicio para su codificación en un lenguaje de computadora.

La fase de diseño de la utilería está dividida en tres etapas: arquitectura, diagrama de estructura y descripción de procesos.

5.2 ARQUITECTURA.

La arquitectura del sistema de utilería es la primera etapa de la fase de Diseño. Aquí se muestran los diferentes niveles de abstracción del sistema que permiten tener una visión global del sistema e iniciar un análisis más detallado subdividiendo en módulos el sistema.

La arquitectura contiene la descripción de los conjuntos de información y un diagrama del flujo de información.

5.2.1 CONJUNTOS DE INFORMACION.

El análisis de los conjuntos de información se divide en tres partes: las entradas, las salidas y la base de datos del sistema.

I ENTRADAS.

Las entradas que requiere la utilería, son tomadas de :

-- teclado y
-- base de datos

Mediante el teclado el usuario seleccionará opciones de menúes, responderá a preguntas sencillas, oprimirá teclas de función o dedicadas, realizará el diseño de los diagramas que forman parte de la interfaz h/m, y posicionará el cursor en la pantalla para indicar las diferentes posiciones de íconos, diagramas, variables y textos, y dando nombres de archivos para que sean leídos o creados.

De la base de datos serán leídos los nombres de diagramas y prototipos existentes, así como la información particular de estos diagramas y prototipos. Además será leída la información con las características de las variables que tengan que ser monitoreadas junto con los diagramas que se requiera desplegar.

II SALIDAS.

Las salidas de los diversos conjuntos de información producida por el sistema serán enviados a :

-- pantalla y
-- base de datos

En pantalla se desplegarán los diversos mensajes que guían la operación de la utilería como son: menúes, mensajes de aviso y error en la línea de estado, textos de ayuda y se desplegarán los diagramas e íconos leídos de disco o diseñados en el momento.

En la base de datos se guardarán los resultados parciales o totales de los diseños realizados con íconos o diagramas y los nombres que contienen la estructura lógica del diseño de la interfaz, así como de los diversos archivos propios del sistema.

III BASE DE DATOS.

La base de datos es el conjunto de archivos predefinidos que guardarán la información que sea reutilizable como son diagramas, variables y la definición de prototipos.

III.A DISEÑO CONCEPTUAL.

Conceptualmente se requiere almacenar la información generada por la utilería, en archivos con extensión predefinida como se indica a continuación :

A. En el archivo PROTOTIPOS.PRO se almacenarán los nombres de los prototipos definidos.

B. En archivos .DIP se almacenarán los nombres de diagramas que pertenezcan a un prototipo, y para cada diagrama se tendrá el nombre del diagrama padre si existe, el nombre del diagrama adjunto izquierdo, el nombre del diagrama adjunto derecho y el nivel de detalle al que pertenece el diagrama. El nombre de estos archivos estará formado por el nombre del prototipo pero con extensión .DIP (DIagramas del Prototipo).

C. En archivos .VAP se almacenarán los nombres de las variables y sus atributos, como nombre del diagrama propietario, tipo de variable, nombre de la variable asociada del sistema de aplicación, las coordenadas de su posición en la pantalla y el tipo de variable; el nombre de éste archivo es el mismo del prototipo diseñado pero con extensión .VAP (Variables y Atributos del Prototipo).

D. Existirá otro archivo .VAC (VAriables de Campo), el cual tendrá los nombres de las variables que serán sensadas por un sistema conectado a la planta monitoreada, y que estará actualizando el "valor actùal" del sistema, en la utilería esta actualización es simulada por un generador de eventos, otro tipo de información que tiene este archivo son los límites de operación de la variable dentro del sistema y el valor sensado.

E. También existirán archivos en formato bit-map en los que la aplicación irá actualizando las variables de su base de datos en la posición que les corresponde en los diagramas mímicos.

F. Además, habrá otros archivos para la operación del sistema: metarchivos donde están los íconos y diagramas en formato GKS.

G. Un archivo con extensión .BIB donde se encuentran los nombres de los íconos definidos por el usuario, así como las coordenadas del centro del MER (rectángulo de área mínima que encierra una figura en su totalidad) de cada ícono.

H. Un archivo .DIA donde están los nombres de los diagramas elaborados.

I. Archivos .CNT donde están las coordenadas de los centros de los MER de todas las figuras que componen un diagrama en particular.

III.B DISEÑO LOGICO.

Los metarchivos tendrán la estructura propia de GKS y los archivos bit-map no tendrán ningún formato en particular.

La definición lógica de los archivos que forman la base de datos para la utilería diseñada requiere de la siguiente definición de archivos:

```
PROTOTIPOS.PRO = record
     Nombre_prototipo     :    cadena de caracteres

archivo .DIP = record
     Nombre_diagrama      :    cadena de caracteres
     Diagrama_padre       :    cadena de caracteres
     Diagrama_anterior    :    entero
     Diagrama_siguiente   :    entero
     Nivel_de_detalle     :    entero

archivo .VAP = record
     Nombre_variable      :    cadena de caracteres
     Diag_propietario     :    cadena de caracteres
     Tipo_variable        :    carácter
     Var_sistema_asociada:     cadena de caracteres
     Abscisa              :    real
     Ordenada             :    real

archivo .BIB = record
     Nombre_ícono         :    cadena de caracteres
     Abscisa              :    real
```

```
    Ordenda              :    real

archivo .DIA = record
    Nombre_diagrama      :    cadena de caracteres

archivos .CNT = record
    Nombre_figura        :    cadena de caracteres
    Abscisa              :    real
    Ordenada             :    real
```

También existirá un archivo .VAC con la lista de variables de campo, límites de tolerancia superior e inferior, y el valor actual sensado, en este caso el valor sensado será simulado, pero se recuerda que este archivo deberá ser actualizado por el sistema de aplicación real.

```
archivo .VAC = record
    Nombre_variable      :    cadena de caracteres
    Lím_precrítico_bajo  :    real
    Lím_precrítico_alto  :    real
    Lím_crítico_bajo     :    real
    Lím_crítico_alto     :    real
    Valor_actual         :    real
```

5.2.2 DIAGRAMA DE FLUJO DE INFORMACION.

El diagrama de flujo de información muestra de manera global el recorrido y la transformación de la información a través del sistema. Este diagrama permite visualizar rápidamente los procesos del sistema así como los efectos que tienen sobre la información.

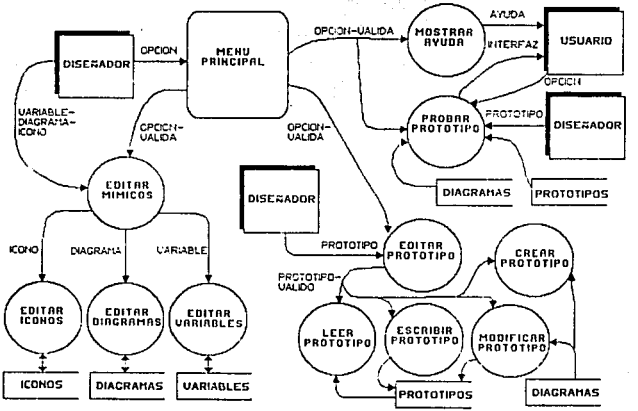
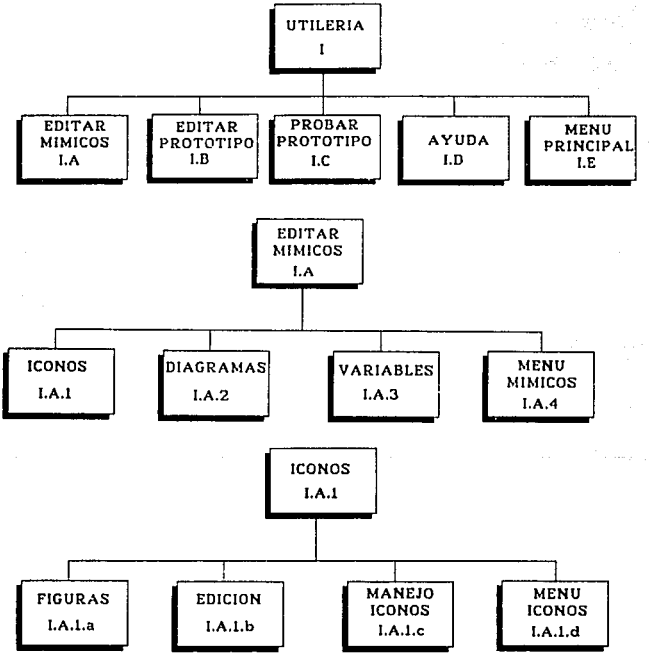
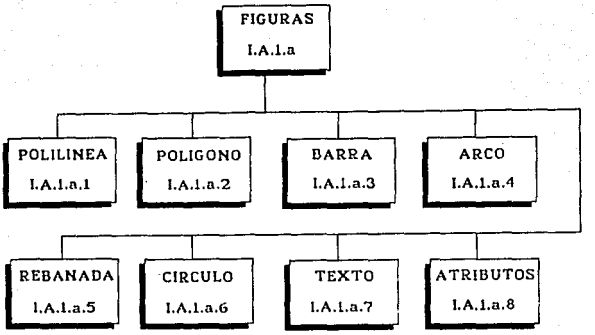


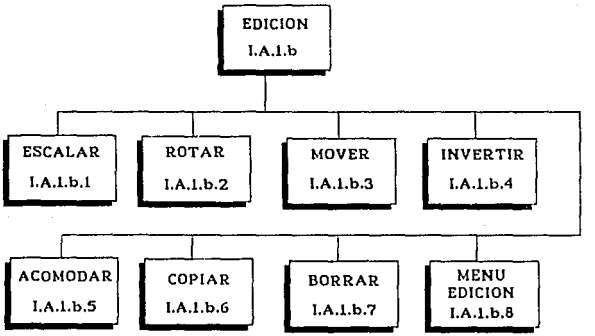
FIGURA 5.2.2.1 DIAGRAMA DE FLUJO DE INFORMACION

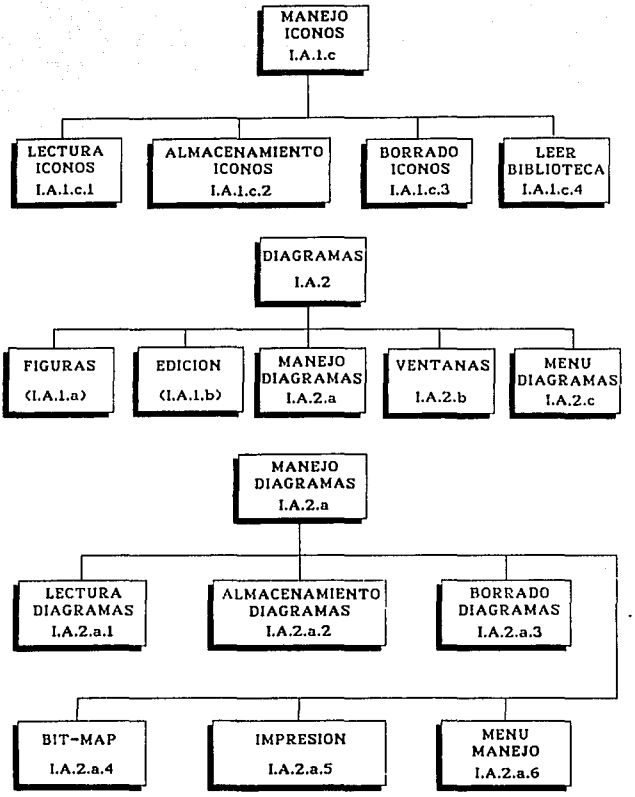
5.3 DIAGRAMA DE ESTRUCTURA.

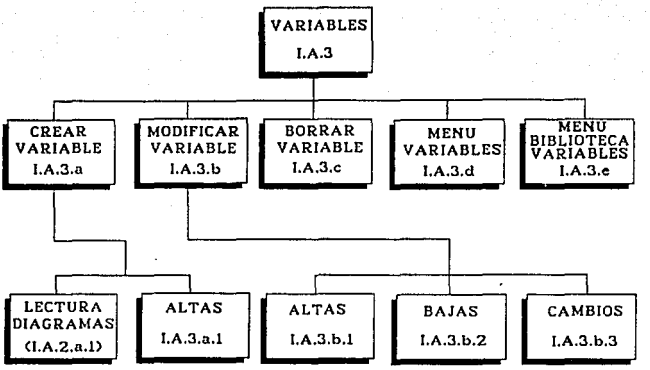
La técnica de dividir el programa de aplicación se emplea en la tesis, y se representa mediante un diagrama de estructura en donde cada parte del diagrama se llama módulo. Este diagrama nos permite observar la jerarquía de cada uno de los módulos para comprender al sistema rápidamente.

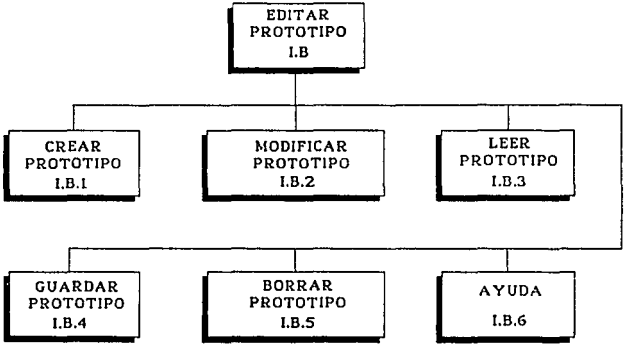


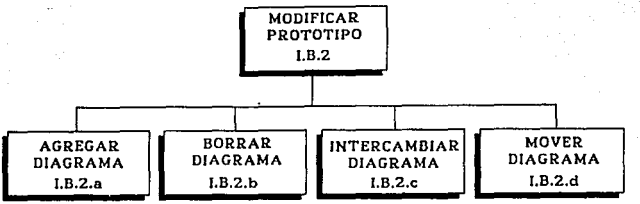

FIGURAS
I.A.1.a
POLILINEA
I.A.1.a.1
POLIGONO
I.A.1.a.2
BARRA
I.A.1.a.3
ARCO
I.A.1.a.4
REBANADA
I.A.1.a.5
CIRCULO
I.A.1.a.6
TEXTO
I.A.1.a.7
ATRIBUTOS
I.A.1.a.8

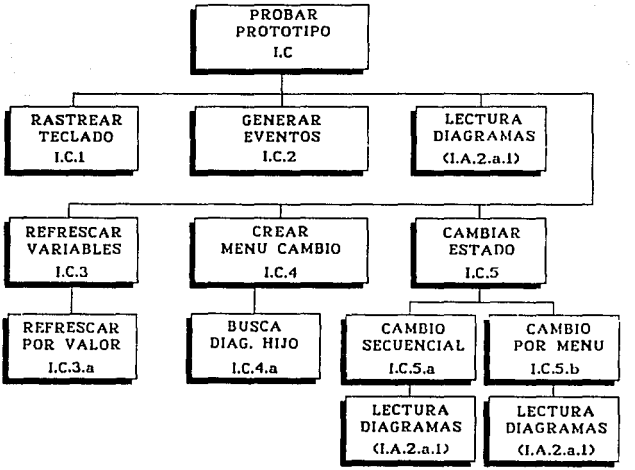

EDICION
I.A.1.b
ESCALAR
I.A.1.b.1
ROTAR
I.A.1.b.2
MOVER
I.A.1.b.3
INVERTIR
I.A.1.b.4
ACOMODAR
I.A.1.b.5
COPIAR
I.A.1.b.6
BORRAR
I.A.1.b.7
MENU EDICION
I.A.1.b.8


MANEJO ICONOS
I.A.1.c
LECTURA ICONOS
I.A.1.c.1
ALMACENAMIENTO ICONOS
I.A.1.c.2
BORRADO ICONOS
I.A.1.c.3
LEER BIBLIOTECA
I.A.1.c.4
DIAGRAMAS
I.A.2
FIGURAS
(I.A.1.a)
EDICION
(I.A.1.b)
MANEJO DIAGRAMAS
I.A.2.a
VENTANAS
I.A.2.b
MENU DIAGRAMAS
I.A.2.c
MANEJO DIAGRAMAS
I.A.2.a
LECTURA DIAGRAMAS
I.A.2.a.1
ALMACENAMIENTO DIAGRAMAS
I.A.2.a.2
BORRADO DIAGRAMAS
I.A.2.a.3
BIT-MAP
I.A.2.a.4
IMPRESION
I.A.2.a.5
MENU MANEJO
I.A.2.a.6


VARIABLES
I.A.3
CREAR VARIABLE
I.A.3.a
MODIFICAR VARIABLE
I.A.3.b
BORRAR VARIABLE
I.A.3.c
MENU VARIABLES
I.A.3.d
MENU BIBLIOTECA VARIABLES
I.A.3.e
LECTURA DIAGRAMAS
(I.A.2.a.1)
ALTAS
I.A.3.a.1
ALTAS
I.A.3.b.1
BAJAS
I.A.3.b.2
CAMBIOS
I.A.3.b.3


EDITAR PROTOTIPO
I.B
CREAR PROTOTIPO
I.B.1
MODIFICAR PROTOTIPO
I.B.2
LEER PROTOTIPO
I.B.3
GUARDAR PROTOTIPO
I.B.4
BORRAR PROTOTIPO
I.B.5
AYUDA
I.B.6


MODIFICAR PROTOTIPO
I.B.2
AGREGAR DIAGRAMA
I.B.2.a
BORRAR DIAGRAMA
I.B.2.b
INTERCAMBIAR DIAGRAMA
I.B.2.c
MOVER DIAGRAMA
I.B.2.d


PROBAR PROTOTIPO
I.C
RASTREAR TECLADO
I.C.1
GENERAR EVENTOS
I.C.2
LECTURA DIAGRAMAS
(I.A.2.a.1)
REFRESCAR VARIABLES
I.C.3
CREAR MENU CAMBIO
I.C.4
CAMBIAR ESTADO
I.C.5
REFRESCAR POR VALOR
I.C.3.a
BUSCA DIAG. HIJO
I.C.4.a
CAMBIO SECUENCIAL
I.C.5.a
CAMBIO POR MENU
I.C.5.b
LECTURA DIAGRAMAS
(I.A.2.a.1)
LECTURA DIAGRAMAS
(I.A.2.a.1)

5.4 DESCRIPCION DEL PROCESO DEL PROGRAMA.

Esta sección muestra el pseudocódigo de las rutinas principales del sistema con el propósito de que sea claro su funcionamiento y los desarrollos posteriores basados en la utilería sean fáciles de incorporar.

5.4.1 MODULOS DEL PROGRAMA.

I UTILERIA.

Reseña del proceso.

Programa principal de la utilería para desarrollo de prototipos de interfaces hombre/máquina, encargado de realizar las inicializaciones necesarias del sistema, desplegar el menú principal, capturar la opción seleccionada por el usuario y llamar la rutina correspondiente.

Detalle del proceso.

```
UTILERIA()
INICIO
     S. Inicializar variables del sistema
     S. Desplegar menú de opciones
     S. Leer opción filtrando F1 a F4 y F10
     MIENTRAS (opción no es F10-salir a DOS) HACER
          SI (opción es de ayuda) ENTONCES
               S. Llamar la rutina AYUDA()
          DE OTRA FORMA SI (opción es editar mímicos) ENTONCES
               S. Llamar la rutina MIMICOS()
          DE OTRA FORMA SI (opcion es editar el prototipo) ENTONCES
```

```
            S. Llamar la rutina EDITAR_PROTOTIPO()
        DE OTRA FORMA SI (la opción es probar el prototipo)
        ENTONCES
            S. Llamar la rutina PROBAR_PROTOTIPO()
        DE OTRA FORMA
            S. Desplegar mensaje de error 1 : "Opción
              inválida, seleccionar nuevamente"
        FIN DE SI
        S. Desplegar menú de opciones
        S. Leer opción filtrando F1 a F4 y F10
    FIN DE MIENTRAS
FIN DE UTILERIA()
```

I.A EDITAR MIMICOS.

Referirse a la tesis del editor de mímicos [HER89] para análisis de sus algoritmos.

I.B EDITAR PROTOTIPO

Reseña del proceso.

Programa encargado de controlar la edición del prototipo, llama a la rutina correspondiente de acuerdo al grado de avance del prototipo, las rutinas podrán crear, modificar, leer, guardar o borrar la estructura del prototipo creado.

Detalle del proceso.

```
EDITAR_PROTOTIPO()
INICIO
    S. Limpiar pantalla
    S. Inicializar variables
```

```
    S. Desplegar menú de opciones
    S. Leer opción, filtrando de acuerdo a la existencia del
       prototipo
    MIENTRAS (la opción no es regresar al menú principal) HACER
        SI (opcion es ayuda) ENTONCES
            S. Llamar la rutina AYUDA()
        DE OTRA FORMA SI (opción es crear prototipo) ENTONCES
            S. Llamar la rutina CREAR_PROTOTIPO()
        DE OTRA FORMA SI (opción es leer un prototipo) ENTONCES
            S. Llamar la rutina  LEER_PROTOTIPO()
        DE OTRA FORMA SI (opción es modificar prototipo) ENTONCES
            S. Llamar la rutina MODIFICAR_PROTOTIPO()
        DE OTRA FORMA SI (opción es guardar prototipo) ENTONCES
            S. Llamar la rutina GUARDAR_PROTOTIPO()
        DE OTRA FORMA SI (opción es borrar prototipo) ENTONCES
            S. Llamar la rutina BORRAR_PROTOTIPO()
        DE OTRA FORMA
            S. Desplegar mensaje de error 1: "Opción inválida,
               seleccionar nuevamente"
        FIN DE SI
        S. Desplegar menú de opciones
        S. Leer opción, filtrando de acuerdo a la existencia
           del prototipo
    FIN DE MIENTRAS
    S. Limpiar pantalla
FIN DE EDITAR_PROTOTIPO()
```

I.B.1 CREAR PROTOTIPO

Reseña del proceso.

Programa encargado de capturar la información necesaria para crear las tablas que definen la estructura de un prototipo.

Detalle del proceso.

```
CREAR_PROTOTIPO()
INICIO
     S. Abrir ventana para crear prototipo
     S. Inicializar tablas: la de variables analógicas y la de
        diagramas
     S. Capturar el nombre del prototipo en nom_prototipo
     S. Capturar el número de niveles de detalle en niv_detalle
     S. Inicializar variable: cont_niv=1
     MIENTRAS (cont_niv es menor o igual que niv_detalle) HACER
          S. Desplegar aviso 10: "F1-Ayuda   F10-Sig. nivel"
          S. Capturar el nombre del diagrama o tecla de función
          MIENTRAS (la entrada es diferente de F10) HACER
               SI (entrada es F1 - Ayuda) ENTONCES
                    S. Llamar rutina AYUDA()
               DE OTRA FORMA SI (es diagrama válido y aún no forma
               parte del prototipo ) ENTONCES
                    S. Guardar el nombre del diagrama y cont_niv
                       en la estructura de datos del prototipo
               DE OTRA FORMA SI (es nombre de diagrama inválido)
               ENTONCES
                    S. Desplegar mensaje de error 3: "Nombre de
                       diagrama inválido, verificar con la ayuda"
               DE OTRA FORMA SI (es diagrama de un nivel anterior)
               ENTONCES
                    S. Desplegar mensaje de error 4: "El diagrama
                       pertence a un nivel anterior"
               FIN DE SI
               S. Desplegar aviso 10: "F1-Ayuda   F10-Sig. nivel"
               S. Capturar nombre del diagrama o tecla de función
          FIN DE MIENTRAS
          S. Incrementar contador de niveles de detalle cont_niv
     FIN DE MIENTRAS
     S. Inicializar contador: cont_niv=1
```

```
MIENTRAS (cont_niv es menor que niv_detalle) HACER
    S. Encontrar el primer diagrama del nivel cont_niv
    MIENTRAS (hay diagramas del nivel cont_niv) HACER
        S. Preguntar si el diagrama correspondiente tiene
           diagramas subordinados (hijos)
        SI (hay diagramas subordinados) ENTONCES
            S. Desplegar lista de diagramas del siguiente
               nivel que aún no tienen diagrama
               subordinador (padre)
            S. Establecer y guardar el diagrama
               subordinador (padre) de los diagramas
               seleccionados
        FIN DE SI
        S. Encontrar el siguiente diagrama que corresponda
           al nivel cont_niv
    FIN DE MIENTRAS
    S. Incrementar en uno el contador de niveles cont_niv
FIN DE MIENTRAS
S. Establecer en la estructura de datos correspondiente y
   para cada diagrama la liga con el diagrama siguiente y el
   diagrama anterior, formando una lista circular con los
   diagramas del mismo nivel de detalle
S. Integrar información referente al manejo de variables en
   la estructura de datos que corresponde al archivo .VAP
S. Prender bandera de prototipo cargado en memoria
S. Cerrar ventana para crear prototipo
FIN DE CREAR_PROTOTIPO()
```

I.B.2 MODIFICAR PROTOTIPO

Reseña del proceso.

Programa encargado de controlar las modificaciones al prototipo solicitadas por el usuario, estas modificaciones pueden

ser : agregar, borrar, mover y cambiar un diagrama.

Detalle del proceso.

```
MODIFICAR_PROTOTIPO()
INICIO
     S. Abrir ventana para modificaciones
     S. Desplegar menú de opciones
     S. Leer opción, filtrando las opciones válidas
     MIENTRAS (entrada es diferente de F10-Al menú anterior) HACER

          SI (entrada es F1-Ayuda) ENTONCES
               S. Llamar la rutina AYUDA()
          DE OTRA FORMA SI (opción es agregar diagrama) ENTONCES
               S. Llamar la rutina AGREGAR_DIAGRAMA()
          DE OTRA FORMA SI (opción es borrar diagrama) ENTONCES
               S. Llamar la rutina BORRAR_DIAGRAMA()
          DE OTRA FORMA SI (opción es mover diagrama) ENTONCES
               S. Llamar la rutina MOVER_DIAGRAMA()
          DE OTRA FORMA
               S. Llamar la rutina CAMBIAR_DIAGRAMA()
          FIN DE SI
          S. Desplegar menú de opciones
          S. Leer opción, filtrando las opciones válidas
     FIN DE MIENTRAS
     S. Cerrar ventana de modificaciones
FIN DE MODIFICAR_PROTOTIPO()
```

I.B.2.a AGREGAR DIAGRAMA

Reseña del proceso.

Programa que agrega un diagrama solicitado por el usuario al prototipo de interfaz hombre/máquina.

Detalle del proceso.

```
AGREGAR_DIAGRAMA()
INICIO
     S. Leer de archivo los nombres de diagramas existentes
     SI (No hay diagramas) ENTONCES
          S. Desplegar aviso 16: "No hay diagramas creados"
     DE_OTRA_FORMA
          S. Abrir ventana para agregar diagramas
          S. Desplegar lista de diagramas existentes
          MIENTRAS (no se oprima F10-Salida) HACER
               S. Capturar el nombre del diagrama para agregar
               S. Capturar el nombre del diagrama padre
               S. Actualizar tabla que define al prototipo
          FIN_DE_MIENTRAS
          S. Cerrar ventana para agregar diagramas
     FIN_DE_SI
FIN_DE_AGREGAR_DIAGRAMA()
```

I.B.2.b BORRAR DIAGRAMA

Reseña del proceso.

Programa que borra algún diagrama solicitado de la definición del prototipo de interfaz hombre/máquina.

Detalle del proceso.

```
BORRAR_DIAGRAMA()
INICIO
     S. Abrir ventana para borrar diagramas
     S. Desplegar lista de diagramas que forman el prototipo
     S. Capturar nombre del diagrama a borrar
     SI (diagrama tiene hijos) ENTONCES
```

```
        S. Quita el diagrama junto con sus hijos
    DE OTRA FORMA
        S. Quita el diagrama solicitado
    FIN DE SI
    S. Actualiza la tabla que define al prototipo
    S. Cerrar ventana para borrar diagramas
FIN DE BORRAR_DIAGRAMA()
```

I.B.2.c INTERCAMBIAR DIAGRAMA

Reseña del proceso.

Programa que intercambia de lugar, dentro de la estructura arborecente del prototipo, dos diagramas; sin mover sus hijos.

Detalle del proceso.

```
INTERCAMBIAR_DIAGRAMA()
INICIO
    S. Abrir dos ventanas para intercambio
    S. Desplegar en la primer ventana la lista de diagramas del
       prototipo
    S. Seleccionar el diagrama a intercambiar
    SI (No hay cancelación de intercambio) ENTONCES
        S. Desplegar en la segunda ventana la lista de
           diagramas del prototipo
        S. Seleccionar el diagrama por el que se intercambia
        SI (No hay cancelación de intercambio) ENTONCES
            S. Realiza el intercambio en el prototipo
            S. Actualiza los diagramas hijos con su nuevo
               padre
        FIN DE SI
    FIN DE SI
    S. Cierrar ventanas para intercambio
```

FIN_DE_INTERCAMBIAR_DIAGRAMA()

I.B.2.d MOVER DIAGRAMA

Reseña del proceso.

Programa que permite mover un diagrama dentro de la estructura arborecente del prototipo, se puede mover con toda su "rama del árbol" o sólo el diagrama indicado.

Detalle del proceso.

```
MOVER_DIAGRAMA()
INICIO
    S. Abrir dos ventanas para mover diagramas
    S. Desplegar en la primer ventana los diagramas del
       prototipo y seleccionar el diagrama a mover
    SI (No se cancela el movimiento) ENTONCES
        SI (Tiene hijos el diagrama seleccionado) ENTONCES
            S. Preguntar si el movimiento es con toda la rama
               del árbol
        FIN_DE_SI
        S. Desplegar en la segunda ventana los posibles
           diagramas padres y seleccionar alguno válido
        SI (No se cancela el movimiento) ENTONCES
            SI (El movimiento es junto con una "rama del
            árbol") ENTONCES
                S. Analizar el tipo de movimiento: al mismo
                   nivel del árbol, con hijos a diferente
                   nivel del árbol o sin hijos a diferente
                   nivel del árbol
                S. Realizar el movimiento de acuerdo al
                   análisis anterior y actualizar el
                   prototipo
```

```
                FIN DE SI
            FIN DE SI
        FIN DE SI
        S. Cerrar ventanas para mover diagramas
FIN DE MOVER_DIAGRAMA
```

I.B.3 LEER PROTOTIPO

Reseña del proceso.

Programa encargado de leer de la base de datos y cargar en memoria la estructura que define un prototipo ya creado.

Detalle del proceso.

```
LEER_PROTOTIPO()
INICIO
    S. Abrir ventana de lectura de prototipos
    S. Inicializar variable nom_prototipo y estructuras de datos
       correspondientes
    S. Desplegar aviso 11: "F1-Ayuda      F10-Menú anterior"
    S. Capturar el nombre del prototipo en nom_proto o tecla de
       función
    MIENTRAS (entrada es diferente de F10-Menú anterior) HACER
        SI (entrada es F1-Ayuda) ENTONCES
            S. Llamar la rutina AYUDA()
        DE OTRA FORMA SI (es nombre de prototipo válido) ENTONCES
            S. Leer los archivos .DIP y .VAP con nombres igual
               al del prototipo y desplegar mensaje de aviso
               7: "Leyendo especificación del prototipo"
            S. Prender bandera de prototipo cargado en memoria
            S. Desplegar aviso 9: "Prototipo cargado en
               memoria"
        DE OTRA FORMA
```

```
            S. Desplegar mensaje de error 5: "Nombre de
               prototipo inexistente, verificar con la ayuda"
        FIN DE SI
        S. Inicializar variable nom_prototipo
        S. Desplegar aviso 11: "F1-Ayuda      F10-Menú anterior"
        S. Capturar el nombre del prototipo en nom_proto o
           tecla de función
    FIN DE MIENTRAS
    S. Cerrar ventana de lectura de prototipos
FIN DE LEER_PROTOTIPO()
```

I.B.4 GUARDAR PROTOTIPO

Reseña del proceso.

Programa encargado de actualizar la base de datos de acuerdo con la estructura de datos que define al prototipo creado.

Detalle del proceso.

```
GUARDAR_PROTOTIPO()
INICIO
    S. Abrir ventana para guardar prototipo
    S. Capturar el nombre del prototipo
    SI (entrada es RETURN) ENTONCES
        S. Asignar el valor de default: nom_proto no cambia
    DE OTRA FORMA
        S. Asignar entrada a nom_proto
    FIN DE SI
    S. Revisar si el nombre del prototipo ya existe
    SI (el prototipo ya existe) ENTONCES
        S. Pedir una confirmación para escribir en ese archivo
        SI (se confirma la escritura) ENTONCES
            S. Grabar información que define al prototipo en
```

```
                la base de datos: archivos .DIP y .VAP
          FIN_DE_SI
     DE_OTRA_FORMA
          S. Grabar información que define al prototipo en la
             base de datos: archivos .DIP y .VAP
     FIN_DE_SI
     S. Cerrar ventana para guardar prototipo
FIN_DE_GUARDAR_PROTOTIPO()
```

I.B.5 BORRAR PROTOTIPO.

Reseña del proceso.

Programa que borra los archivos que definen la estructura del prototipo especificado.

Detalle del proceso.

```
BORRAR_PROTOTIPO()
INICIO
     S. Abrir ventana para borrar prototipo
     S. Desplegar aviso 11: "F1-Ayuda    F10-Menú anterior"
     S. Capturar el nombre del prototipo en nom_proto o tecla de
        función
     MIENTRAS (entrada es diferente de F10-Menú anterior) HACER
          SI (entrada es F1-Ayuda) ENTONCES
               S. Llamar la rutina AYUDA2()
          DE_OTRA_FORMA SI (es nombre de prototipo válido) ENTONCES
               S. Pedir confirmación para borrar prototipo
               SI (confirma el borrado) ENTONCES
                    S. Borrar los archivos que definen al
                       prototipo
               FIN_DE_SI
          DE_OTRA_FORMA
```

S. Desplegar mensaje de error 5: "Nombre de prototipo inexistente, verificar con la ayuda"

FIN DE SI

S. Desplegar aviso 11: "F1-Ayuda F10-Menú anterior"

S. Capturar el nombre del prototipo en nom_proto o tecla de función

FIN DE MIENTRAS

S: Cerrar ventana para borrar prototipo

FIN DE BORRAR_PROTOTIPO()

I.C PROBAR PROTOTIPO

Reseña del proceso.

Programa encargado de controlar la simulación del prototipo de interfaz hombre/máquina previamente definido, haciendo el llamado de las rutinas correspondientes.

Detalle del proceso.

PROBAR_PROTOTIPO()

INICIO

S. Limpiar pantalla

S. Cargar definición del prototipo

S. Llamar la rutina FORMAR_MENU_CAMBIO()

S. Inicializar variables: diagrama_presenteactivo y opción

S. Calcular incrementos de las variables

S. Llamar la rutina DESPLEGAR_DIAGRAMA() para el diagrama inicial

S. Desplegar mensaje aviso 13: "F2-Cambio por menú Flechas-Al diagrama anterior, siguiente o 'padre'"

S. Llamar rutina RASTREAR_TECLADO()

MIENTRAS (opción es diferente de F10 - terminar prueba) HACER

SI (entrada es flecha, cambiar de diagrama) ENTONCES

```
            S. Llamar la rutina CAMBIAR_ESTADO()
            S. Activar el diagrama correspondiente
            S. Llamar a la rutina DESPLEGAR_DIAGRAMA()
        DE OTRA FORMA
            S. Llamar la rutina GENERAR_EVENTOS()
            S. Llamar la rutina REFRESCAR_VARIABLES()
        FIN DE SI
        S. Llamar la rutina RASTREAR_TECLADO()
    FIN DE MIENTRAS
    S. Limpiar pantalla
FIN DE PROBAR_PROTOTIPO()
```

I.B.1 RASTREAR TECLADO

Reseña del proceso.

Programa encargado de verificar si se oprimió una tecla, validar y aceptar unicamente las teclas permitidas.

Detalle del proceso.

```
RASTREAR_TECLADO()
INICIO
    S. Inicializar variables contadores
    MIENTRAS (contador de revisiones al teclado es menor que el
    límite) HACER
        S. Verificar si hay una tecla oprimida
        S. Incrementar contador de revisiones al teclado
        SI (se oprimió una tecla) ENTONCES
            SI (tecla es de movimiento o F2)  ENTONCES
                S. Hacer opción igual a CAMBIO DE ESTADO
                S. Hacer contador de revisiones igual al
                   límite
            DE OTRA FORMA
```

```
                    S. Hacer opción igual a GENERAR EVENTOS
                    SI (tecla no es F10-Terminar prueba) ENTONCES
                        S. Desplegar mensaje de error 1: "Opción
                           inválida, seleccionar nuevamente"
                    FIN DE SI
                FIN DE SI
            FIN DE SI
        FIN DE MIENTRAS
FIN DE RASTREAR TECLADO()
```

I.C.2 GENERAR EVENTOS

Reseña del proceso

Programa encargado de simular el sistema que monitorea las varibles del proceso y actualiza los valores de campo en la base de datos.

Detalle del proceso.

```
GENERAR_EVENTOS()
INICIO
    S. Inicializar variables
    S. Apuntar a la primer variable analógica
    MIENTRAS (existan variables analógicas) HACER
        S. Definir el signo del incremento de la variable
        S. Hacer valor actual igual a la suma algebraica del
           valor actual mas el incremento de acuerdo al signo
           del incremento
        S. Apuntar a la siguiente variable analógica.
    FIN DE MIENTRAS
    S. Apuntar a la primer variable digital
    MIENTRAS (existan variables digitales) HACER
        SI (estado de operacion es normal) ENTONCES
```

S. Hacer estado de operación con ALARMA
DE OTRA FORMA
S. Hacer estado de operación NORMAL
FIN DE SI
S. Apuntar a la siguiente variable digital
FIN DE MIENTRAS
FIN DE GENERAR EVENTOS()

(I.A.2.a.1) LECTURA DIAGRAMAS

Reseña del proceso.

Programa encargado de desplegar el diagrama que se solicite.
Detalle del proceso.

LECTURA_DIAGRAMAS()
INICIO
S. Inicializar variables: segmento e ícono inicial y final, y apuntadores de íconos y segmentos
S. Ocultar el diagrama que está presente
MIENTRAS (apuntador de segmento apunta a un segmento del diagrama presente) HACER
S. Hacer invisible el segmento correspondiente
S. Apuntar al siguiente segmento
FIN DE MIENTRAS
MIENTRAS (apuntador de icono apunta a un ícono del diagrama presente) HACER
S. Hacer invisible el ícono correspondiente
S. Apuntar al siguiente ícono
FIN DE MIENTRAS
S. Apuntar al segmento e ícono iniciales del diagrama a desplegar
MIENTRAS (apuntador de segmento apunta a un segmento del diagrama a desplegar) HACER

```
            S. Hacer visible el segmento apuntado
            S. Apuntar al siguiente segmento
        FIN DE MIENTRAS
        MIENTRAS (apuntador de ícono apunta a un ícono del diagrama a
        desplegar) HACER
            S. Hacer visible el ícono apuntado
            S. Apuntar al siguiente ícono
        FIN DE MIENTRAS
FIN DE LECTURA_DIAGRAMAS()
```

I.C.3 REFRESCAR VARIABLES

Reseña del proceso.

Programa que controla el tipo de actualización del valor en pantalla de cada una de las variables. Las variables analógicas tendrán un refresco por valor.

Detalle del proceso.

```
REFRESCAR_VARIABLES()
INICIO
    S. Apuntar a las primeras variables analógica y digital
    MIENTRAS (existan variables analógicas) HACER
        SI (la variable es del diagrama presente) ENTONCES
            S. Llamar a la rutina REFRES_VALOR()
        FIN DE SI
        S. Apuntar a la siguiente variable analógica
    FIN DE MIENTRAS
    MIENTRAS (existan variables digitales) HACER
        SI (la variable es del diagrama presente) ENTONCES
            S. Llamar a la rutina REFRES_COLOR()
        FIN DE SI
        S. Apuntar a la siguiente variable digital
```

FIN DE MIENTRAS

S. Recuperar los atributos de default para el despliegue

FIN DE REFRESCAR_VARIABLES()

I.C.3.a REFRES VALOR

Reseña del proceso.

Programa encargado de actualizar en pantalla las variables analógicas, el despliegue será numérico y de acuerdo al rango en que esté: normal, precrítico o crítico, se desplegará en color verde, amarillo o rojo respectivamente.

Detalle del proceso.

```
REFRES_VALOR()
INICIO
    SI (valor de actual es precrítico) ENTONCES
        S. Asignar color amarillo (precrítico) al atributo de
           despliegue de texto
    DE OTRA FORMA SI (valor actual es crítico) ENTONCES
        S. Asignar color rojo (crítico) al atributo de
           despliegue de texto
        S. Desplegar mensaje de aviso 7: "Existe estado de
           alarma"
    DE OTRA FORMA
        S. Asignar color verde (normal) al atributo de
           despliegue de texto
    FIN DE SI
    S. Desplegar valor actual en las coordenadas en que se
       definió la variable y de acuerdo al formato especificado
       por el diseñador del prototipo
FIN DE REFRES_VALOR()
```

I.C.4 CREAR MENU CAMBIO

Reseña del proceso.

Programa que forma el menú jerárquico con los diagramas del prototipo, el menú permite visualizar la interrelación de los diagramas.

Detalle del proceso.

FORMAR_MENU_CAMBIO()
INICIO
- S. Hacer niv_detalle=1
- S. Hacer que el diagrama general con niv_detalle=1, sea el primero de la lista de diagramas
- S. Guardar en lista_menu el diagrama general y su respectivo niv_detalle
- S. Llamar rutina BUSCA_DIAG_HIJO() con parámetros apuntador de diagrama padre, nombre del diagrama padre y niv_detalle

FIN FORMAR_MENU_CAMBIO()

I.C.4.a BUSCA DIAG. HIJO

Reseña del proceso.

Programa recursivo que busca los diagramas subordinados (hijos) y va formando el menú jerárquico de los diagramas del prototipo.

Detalle del proceso.

BUSCA_DIAG_HIJO(ap_diag, diag_padre, niv_detalle)
INICIO

```
    MIENTRAS (ap_diag apunte a un diagrama) HACER
        S. Apuntar al diagrama siguiente con ap_diag
        S. Incrementar niv_detalle
        MIENTRAS (ap_diag no sea hijo de diag_padre y ap_diag
        apunte a un diagrama) HACER
            S. Apuntar al diagrama siguiente con ap_diag
        FIN_DE_MIENTRAS
        SI (ap_diag es hijo de diag_padre) ENTONCES
            S. Guardar en lista_menu ap_diag y niv_detalle
            S. Llamar rutina BUSCA_DIAG_HIJO() con parámetros
               ap_diag, nombre de diagrama apuntado por
               ap_diag y niv_detalle
        FIN_DE_SI
        S. Decrementar niv_detalle
    FIN_DE_MIENTRAS
FIN_DE_BUSCA_DIAG_HIJO()
```

I.C.5 CAMBIAR ESTADO

Reseña del proceso.

Programa encargado de cambiar o moverse de un diagrama a otro mediante las teclas de movimiento del cursor (flechas), las teclas de movimiento lateral nos cambian al diagrama anterior o siguiente pero dentro del mismo nivel de detalle, la tecla de movimiento hacia arriba nos cambia al nivel anterior pero al diagrama designado padre, y la tecla de movimiento hacia abajo nos cambia al siguiente nivel de detalle si es que existe.

Detalle del proceso.

```
CAMBIAR_ESTADO()
INICIO
    SI (tecla es de movimiento lateral o hacia arriba) ENTONCES
```

```
        S. Llamar la rutina SECUENCIAL_CAMBIO()
    DE_OTRA_FORMA SI (tecla es de movimiento hacia abajo o F2-
    cambio por menú) ENTONCES
        S. Llamar la rutina de MENU_DE_CAMBIO()
    FIN DE SI
FIN DE CAMBIAR_ESTADO()
```

I.C.5.a SECUENCIAL CAMBIO

Reseña del proceso.

Programa que realiza el cambio al siguiente o al anterior diagrama que pertenece al mismo nivel de detalle, o al diagrama del nivel superior o diagrama padre.

Detalle del proceso.

```
SECUENCIAL_CAMBIO()
INICIO
    SI (tecla es de movimiento a la izquierda) ENTONCES
        S. Apuntar al diagrama que se definió como anterior
        S. Actualizar la variable diagrama_presente
    DE_OTRA_FORMA SI (tecla es de movimiento a la derecha)
    ENTONCES
        S. Apuntar al diagrama que se definió como siguiente
        S. Actualizar la variable diagrama_presente
    DE_OTRA_FORMA SI (tecla es de movimiento hacia arriba)
    ENTONCES
        S. Apuntar al diagrama superior o diagrama padre
        S. Actualizar la variable diagrama_presente
    FIN DE SI
    S. Llamar la rutina DESPLEGAR_DIAGRAMA()
FIN DE SECUENCIAL_CAMBIO()
```

I.C.5.b MENU DE CAMBIO

Reseña del proceso.

Programa que despliega un menú jerárquico de los diagramas del prototipo, de donde el usuario selecciona el diagrama que desea observar.

Detalle del proceso.

```
MENU_DE_CAMBIO()
INICIO
     S. Limpiar pantalla y activar ventana para el menú
     S. Inicializar el num_pags de acuerdo al total de diagramas
     S. Desplegar página uno
     S. Desplegar aviso de página siguiente o anterior de acuerdo
        a num_pags y a la pag_actual
     S. Leer teclado, filtrando RePag, AvPag, F10, flechas o
        RETURN
     MIENTRAS (tecla es diferente de F10-Regresar al diagrama y
     diferente de RETURN) HACER
          SI (tecla es flecha) ENTONCES
               S. Resaltar el diagrama indicado por el cursor
          DE OTRA FORMA SI (tecla es RePag) ENTONCES
               S. Decrementar una unidad pag_actual
          DE OTRA FORMA SI (tecla es AvPag) ENTONCES
               S. Incrementar una unidad pag_actual
          FIN DE SI
          S. Desplegar pag_actual
          S. Desplegar aviso de página siguiente o anterior de
             acuerdo a num_pags y a la pag_actual
          S. Leer teclado, filtrando RePag, AvPag, F10, flechas o
             RETURN
     FIN DE MIENTRAS
     SI (tecla es RETURN) ENTONCES
```

```
            S. Actualizar diagrama_presente con el diagrama
               seleccionado
            S. Llamar la rutina DESPLEGAR_DIAGRAMA()
        FIN_DE_SI
FIN_DE_MENU_DE_CAMBIO()
```

CAPITULO VI

EVALUACION Y CONCLUSIONES

En este capítulo se hace una evaluación de los resultados obtenidos, comparando un desarrollo de interfaz hombre/máquina realizado con técnicas tradicionales y uno empleando la utilería propuesta.

También se presentan las conclusiones sobre el tema de interfaces hombre/máquina y la utilería desarrollada, se discuten mejoras, se presenta la línea de investigación seguida, así como los temas propuestos para futuras investigaciones.

6.1 DISEÑO DE UNA INTERFAZ UTILIZANDO TECNICAS TRADICIONALES.

La técnica tradicional indica la elaboración de un prototipo de interfaz mediante la codificación directa de instrucciones para la elaboración de los diagramas mímicos que la componen, dibujando lineas, rellenando figuras, y actualizando información; esta técnica genera resultados después de mucho tiempo, dependiendo del tamaño de la aplicación, ya que se tienen que realizar muchas pruebas hasta alcanzar la interfaz deseada.

Al realizar alguna modificación, empleando la técnica tradicional, se tiene que revisar nuevamente el código programado, pero existe el inconveniente de ser muy lento este proceso de modificación, además de que lo debe de realizar la persona que programó inicialmente la aplicación, de lo contrario el tiempo de

revisión del código existente demora aún más la modificación.

6.2 DISEÑO DE UNA INTERFAZ EMPLEANDO LA UTILERIA PROPUESTA.

El diseño de un prototipo de interfaz mediante la utilería permite la elaboración de los diagramas mediante un editor específico para este tipo de aplicaciones (editor de mímicos), no se tiene la necesidad de programar dichos diagramas.

Con el empleo de la utilería se toman los diagramas elaborados con el editor de mímicos y únicamente se indica la interdependencia entre ellos, se tiene la capacidad de observar la simulación de la interfaz en operación mediante un generador de eventos, permitiendo una pronta evaluación del prototipo.

Ahora, si se desea realizar algún cambio ya sea en la apariencia de un diagrama, se puede hacer rápida y fácilmente mediante el editor de mímicos, pero si el cambio es en la estructura de la interfaz, es decir, en la interdependencia de los diagramas también es fácil realizarlo. Todo tipo de cambio lo puede realizar cualquier persona relacionada con el proyecto, no necesita saber como se elaboró un diagrama anterior para poder realizar sus modificaciones.

6.3 COMPARACION DE LA TECNICA TRADICIONAL Y EL USO DE LA UTILERIA.

Como se puede observar al comparar dos diseños elaborados con distintas técnicas, el tiempo de elaboración es mayor usando técnicas tradicionales, una modificación involucra mayor cantidad de tiempo y dificultad si se emplea la técnica tradicional.

Al emplear la técnica tradicional el diseñador realiza la interfaz de acuerdo al documento de especificación del usuario, pero, generalmente, no se tiene contemplada una incorrecta interpretación de estos requerimientos, o una mala especificación, estas limitantes se pueden solucionar en gran parte si se involucra al usuario en el desarrollo del sistema, pero, naturalmente, ésto no es posible cuando se emplea la técnica tradicional y sí lo es si se ocupa la utilería planteada en la tesis.

6.4 BENEFICIOS OBTENIDOS.

Los resultados obtenidos son los siguientes :

1. Utilería capaz de generar íconos, diagramas mímicos, y prototipos de interfaz reutilizables, es decir, que se pueden emplear y modificar en otros diseños de prototipos.

2. Generación rápida de prototipos de interfaces hombre/máquina.

3. Incorporación del usuario al desarrollo del sistema.

4. Herramienta de ayuda para una mejor definición de los requerimientos del usuario al incorporarse al desarrollo del sistema.

5. Capacidad de evaluar y modificar el diseño del prototipo realizado.

6. Reducción del tiempo de desarrollo del prototipo de la interfaz.

6.5 LOGRO DE OBJETIVOS.

Analizando los resultados obtenidos junto con los objetivos planteados, se observa que se cumplió satisfactoriamente la meta propuesta con el tema de tesis.

Se desarrolló una utilería de cómputo que permite desarrollar prototipos rápidos de interfaces hombre/máquina, el prototipo generado tiene sus limitantes debido a que podría manejar otras funciones que apoyarían al usuario a una evaluación más acertada, pero también muestra las ventajas que tiene el desarrollo de un prototipo rápido, ganando tiempo que puede ser aprovechado para la elaboración del sistema final.

Con el manejo de la técnica de prototipos rápidos se puede incorporar el usuario al desarrollo del sistema, dando su opinión e interactuando con el diseñador al momento de realizar el prototipo. Si existe algún cambio se puede realizar rápidamente en ese momento, siendo de mucha importancia esto último ya que la definición de requerimientos será definitiva evitando los costosos cambios que existían en etapas avanzadas del proyecto final.

Con la participación del usuario en el desarrollo del sistema aumenta su grado de aceptación, ya que pudo expresar realmente sus necesidades y permitirá un incremento en la eficiencia del sistema debido a que la interfaz representa las verdaderas necesidades del usuario.

Permitirá definir, generar, modificar y evaluar las características de la interfaz hombre/máquina de los nuevos sistemas a desarrollar, en particular de los SADRE's, incorporando técnicas de desarrollo rápido de prototipos y refinamiento iterativo.

Con el empleo de técnicas ergonómicas se pudo realizar una utilería que fue de agrado a los usuarios de la misma, es decir, tiene la característica de ser "amigable" con el usuario.

El empleo de técnicas y tendencias actuales da la pauta a la realización de mejoras a la utilería no permitiendo la obsolecencia de dicha utilería.

En la medida que se logre una correcta definición de requerimientos, las modificaciones al sistema final serán menores y, por lo tanto, se encaminará el sistema hacia su versión original desde el inicio del proyecto, mejorando la aceptación del usuario para con el sistema final, y, algo muy importante, la reducción del tiempo de desarrollo y en consecuencia el costo del mismo.

6.6 LINEA DE INVESTIGACION.

Con el empleo de técnicas y tendencias actuales como es la generación rápida de prototipos, método de refinamiento iterativo y la aplicación de técnicas ergonómicas se observa la originalidad de la tesis. Existen algunas herramientas con características similares a la tesis desarrollada, pero no para el área de la industria eléctrica: plantas termoeléctricas o nucleoeléctricas que necesitan una interfaz hombre/máquina basada en diagramas mímicos como medio de comunicación con el personal de supervisión.

Como línea de investigación generada por la tesis se plantea la aplicación de técnicas de inteligencia artificial para la generación de prototipos, es decir, algunas técnicas que permitan asignar conocimiento a la elaboración validada de las conexiones de los elementos de un diagrama mímico.

Otra línea de investigación posible es la generación del sistema de aplicación en tiempo real a partir del prototipo, es decir, la generación del código que forma el sistema de aplicación y que es capaz de generar resultados reales junto con el sistema de procesamiento real.

También se puede plantear una línea de investigación de control, en la que además de ser posible la generación de la interfaz hombre/máquina para monitoreo de los procesos de una planta, se tenga la posibilidad de realizar acciones de control desde el mismo tablero de supervisión, permitiendo una respuesta más rápida y precisa a las condiciones anormales de la planta.

6.7 RECOMENDACIONES.

Existen algunas recomendaciones que se suguieren si se toma alguna de las líneas de investigación.

Si se tomara la decisión de generar el sistema de aplicación en tiempo real, es recomendable buscar otro estandar de graficación diferente al GKS, ya que es lento el despliegue si se trata de una aplicación con muchos elementos, debido a los pasos de despliegue en coordenadas del mundo, intentando desplegar información aunque no este dentro del rango del dispositivo.

Para una aplicación en tiempo real se requieren tiempos de respuesta muy elevados, solamente se recomienda seguir manejando GKS si se emplea en diseños donde el tiempo de respuesta no es tan crítico.

Respecto a la aplicación actual podrían agregarse algunas funciones de gráficas de estado, ya sea que sean instantáneas o que sean históricas, lo mismo se puede decir para las diversas gráficas

de tendencia que existen, permitiendo un análisis más completo del funcionamiento de la interfaz.

Finalmente, se sugiere hacer la transferencia de la utilería a computadoras workstation, agregando nuevas funciones que se consideren necesarias, como son las gráficas de tendencia y la generación de reportes entre otros. La utilería en un ambiente de workstation mejorará el tiempo de respuesta de los despliegues y tal vez no sea necesario cambiar el estandar de graficación GKS, sino continuar agregando funciones de importancia para el sistema de interacción hombre/máquina.

ANEXOS

GLOSARIO

Acrónimo: Palabra formada con las letras o sílabas provenientes de un grupo de palabras. Tales letras forman una "palabra" pronunciable. Por ejemplo, VEPS, es decir, último en entrar, primero en salir; GIGO ("garbage-in, garbage-out"), o sea, entra basura, sale basura; y MAD corresponde a memoria de acceso directo.

ANSI: Siglas de "American National Standards Institute". Esta organización organiza comites formados por usuarios de computadoras, fabricantes, etc., para desarrollo y publicidad de estándares de la industria, ejemplo, ANSI FORTRAN, ANSI Código Estándar para Identificación Periódica, etc. Nombres anteriores: "American Standards Association" (ASA) y "United States of America Standards Institute" (USASI).

BNF: Abreviación de "Backus normal form", que es una estructura de lenguaje formal para decodificación sintáctica utilizado en el diseño de ALGOL-60.

DMS: De las siglas en inglés: "Data Management System". Es un sistema de programas diseñado para proporcionar al

operador humano la capacidad de preguntar, aumentar y manipular grandes bases de datos almacenadas en computadora, en un lenguaje natural.

Driver : Usualmente se refiere a un módulo del sistema operativo que controla un periférico específico de entrada-salida; ejemplo: el manejador ("driver") es llamado por el ejecutivo del sistema operativo en respuesta a una llamada de entrada/salida del programa de usuario. En muchos sistemas, cada tipo de periférico tiene un único manejador en el sistema operativo.

Hardware: Es el material magnético, mecánico, electrónico y eléctrico que componen una computadora.

Interfaz hombre/ma´quina : Nombre colectivo que se les da a todos los componentes de una máquina a través de los cuales se realiza la comunicación entre el ser humano y la computadora.

Icono : Es una representación gráfica, usualmente en una pantalla de computadora, de un objeto del mundo real. Los íconos son escencialmente imágenes de cosas usadas para hacer más visual la operación de computadoras y que sean fáciles de entender para el novato.

ISO : De las siglas en inglés: International Organization for Standarization. ISO es la agencia internacional especializada en la estandarización, actualmente comprende a los cuerpos nacionales estándares de 86 países. Sólo una organización de cada país participante puede ser miembro: el cuerpo miembro de los Estados Unidos es ANSI. Los miembros participantes contribuyen al trabajo de los comites técnicos y tienen el poder de

votar a favor o en contra para aprobar el desarrollo de estándares. Miembros observantes pueden presenciar los encuentros pero no votar. En suma, organizaciones como CCITT participan como miembros de enlace.

Mapeo: Es una transformación de un conjunto a otro conjunto. Es una correspondencia. Comunmente ocupado el término de "mapeo de memoria" que significa : Modo opcional de operación de computadora en donde los 8 bits más significativos de una dirección virtual mayor de 15 son reemplazados por un valor alternativo, esto es proporcionado por relocalización dinámica de programas (algunas veces computadoras).

Mímico : Es un diagrama que muestra en forma condensada el estado operacional de un proceso, incluyendo los valores de sus componentes, flujos, conexiones, etc.

Parsing: Acción de efectuar el "parse" o búsqueda lexicográfica. Rastrear una línea de entrada carácter por carácter a fin de obtener un significado sintáctico y gramatical a partir de ella. Todos los compiladores, ensambladores e intérpretes utilizan rutinas de búsqueda lexicográfica para convertir las líneas de programa fuente en una forma que la computadora pueda entender.

Prompt : Mensaje de presencia del sistema o de un programa.

Software: Término que contrasta con el hardware de un sistema de computadora. El término software se refiere a los programas, lenguajes y procedimientos de un sistema de computadora.

Token : Una unidad detectable en una secuencia de carácteres.

BIBLIOGRAFIA

[DALLI] Dallimonti, Renzo; "Principles of design for man-machine interfaces in process control". En: Proceedings of the Sixth Annual Advanced Control Conference, W. Lafayette, Indiana: 1980 pp. 13-34

[DIA84] Diamant, Aarón M. [y otros]. "El sistema de adquisición de datos del SADRE: principios, funciones y alcances". En: Boletín del IIE, 8:3 (mayo/junio 1984) pp. 137-141

[DUM88] Dumas, Joseph S.; "Designing the interface". En: Designing user interfaces for software. pp. 45-168, U.S.A.: Prentice Hall, c1988

[GER85] Gerez Greiser [y otros]; Desarrollo y administración de programas de computadora (software). 2nd. ed. (D.F., México: Compañia editorial continental, S. A. de C. V., 1985) 299 p.

[HAR89] Hartson, H. R. [and] Hix, D.; "Human computer interface development". En: ACM Computing Surveys, 21:1 (March 1989) pp. 6-92

[HER89] Herrera López, Victor Hugo; "Diseño e implementación de un editor de mímicos para procesos industriales de alto

riesgo" Mexico, Puebla: Universidad de las Américas, 1989. 80 p., (Tesis de Licenciatura)

[HUM82] Human factors of user-computer interfaces/ presented by Computer Graphics Consultants, Inc. Washington, D.C.: c1982. [103 p.]

[CHA86] Charwat, Hans-Jürgen. Man machine communications in production plants. Siemens Power Engineering & Automation. 8:1 (1986)

[GKSa] Graphics Software System, Inc. GSS-TOOLKIT KERNEL SYSTEM, PROGRAMMER'S GUIDE. Wilsonville, Oregon, c1985, 454 p. [p.v.]

[GKSb] Graphics Software System, Inc. GSS-TOOLKIT KERNEL SYSTEM, C LANGUAGE BINDING. Wilsonville, Oregon, c1985, 44 p.

[LATCa] Lattice, Incorporated. LATTICE C COMPILER for MS-DOS, PROGRAMMER'S REFERENCE MANUAL. Version 3. Volumen 1. Glen Ellyn, U.S.A., c1986, 186 p. [p.v.]

[LATCb] Lattice, Incorporated. LATTICE C COMPILER for MS-DOS, PROGRAMMER'S REFERENCE MANUAL. Version 3. Volumen 2. Glen Ellyn, U.S.A., c1986, 467 p. [p.v.]

[LUQ88] Luqui and Valdis Berzins, "Rapidly Prototyping Real-Time Systems". En: IEEE Software, (September 1988) [pp. 25-35]

[LUQ90] Luqui; Barnes, P. D. and Zyda, M. "Graphical tool for computer-aided prototyping". En: Information and software technology, 32:3 (april 1990) pp. 199-206

[MAG85] Maguire, M. C., "A review of human factors guidlines and techniques for the design of graphical human-computer interfaces". En: Computer & Graphics, 9:3 (1985), pp. 221-235

[MAI90] Maizener, A. [y] Thonon, D.; "Un prototype pour la spécification des alarmes de synthèse au futur dispatching national". En: Bulletin de la direction des etudes et recherches, serie B, Num. 1, 1990, pp. 33-39

[PFORCE] Phoenix; PFORCE, Reference Guide. 557 p. [p.v.]

[RIJNS] Rinjsdorp, John E.; "A general review of the field of man/machine interfaces in terms of the psychological-social effects involved; treating needs, the results of recent projects, and future projections for the field". Twente University of Technology, Enschede, Netherlands. pp. 1-12

[VIL84a] Villavicencio R., Alejandro. "Sistema de adquisición de datos y registro de eventos (SADRE)". En: Boletín del IIE, 8:2 (marzo/abril 1984) pp. 69-80

[VIL84b] Villavicencio R., Alejandro; "La interfaz hombre/máquina del SADRE". En: Boletín del IIE, 8:4 (julio/agosto 1984) pp. 186-197