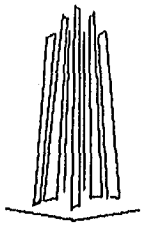
12
2y

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Escuela Nacional de Estudios Profesionales
Campus "ARAGÓN"



"Diseño e Implementación de una Interface Gráfica con Orientación a Objeto para la captura y edición de Cartas ASM"

TESIS PROFESIONAL
QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN
P R E S E N T A N:

OSWIN ALFREDO CORONA GARCÍA
EFRÉN LÓPEZ MARTÍNEZ

San Juan de Aragón, Edo. Méx. México, 1996

TESIS CON
FALLA DE ORIGEN

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
ARAGÓN
DIRECCIÓN

OSWIN ALFREDD CORONA GARCIA
P R E S E N T E .

En contestación a la solicitud de fecha 12 de octubre del año en curso, presentada por Efren López Martínez y usted, relativa a la autorización que se les debe conceder para que el señor profesor, Ing. ROBERTO BLANCO BAUTISTA pueda dirigirle el trabajo de Tesis denominado " DISEÑO E IMPLEMENTACION DE UNA INTERFACE GRAFICA CON ORIENTACION A OBJETOS PARA LA CAPTURA Y EDICION DE CARTAS ASM ", con fundamento en el punto 6 y siguientes, del Reglamento para Exámenes Profesionales en esta Escuela, y toda vez que la documentación presentada por usted reúne los requisitos que establece el precitado Reglamento; me permito comunicarle que ha sido aprobada su solicitud.

Aprovecho la ocasión para reiterarle mi distinguida consideración.

ATENTAMENTE
"POR MI RAZA HABLARA EL ESPIRITU"
San Juan de Aragón, Mex., 14 de octubre de 1994
EL DIRECTOR

M en I. CLAUDIO C. MERRIFIELD CASTRO

c c p Lic. Alberto Ibarra Rosas, Jefe de la Unidad Académica.
c c p Ing. Silvia Vega Muytoy, Jefe de la Carrera de Ingeniería en Computación.
c c p Ing. Roberto Blanco Bautista, Asesor de Tesis.

CCMC'AIR'lla.

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
CAMPUS ARAGON
JEFATURA DE CARRERA DE INGENIERIA EN COMPUTACION

LIC. ALBERTO IBARRA ROSAS
JEFE DE LA UNIDAD ACADEMICA
P R E S E N T E .

Por este conducto comunico a usted que, el alumno *OSWIN ALFREDO CORONA GARCIA*, con número de cuenta *8561372-2* de la carrera de *INGENIERIA EN COMPUTACION* desarrollará el trabajo de Tesis titulado *"DISEÑO E IMPLEMENTACION DE UNA INTERFACE GRAFICA, CON ORIENTACION A OBJETOS PARA LA CAPTURA Y EDICION DE CARTAS ASM"* conjuntamente con *EFREN LOPEZ MARTINEZ*, con número de cuenta *8561386-3*, asesorados por el *Ing. Roberto Blanco Bautista.*

Lo anterior es con el fin de que se les permita realizar los trámites correspondientes a la autorización del trabajo.

Sin otro particular, aprovecho la ocasión para enviarle un cordial saludo.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
San Juan de Aragón, Edo. de Méx., Octubre 12 de 1994.
JEFE DE CARRERA

ING. SILVIA VEGA MUYOY

SVM/gga.

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGON
JEFATURA DE CARRERA DE INGENIERIA EN COMPUTACION

ING. ROBERTO BLANCO BAUTISTA

ING. ERNESTO PEÑALOSA ROMERO

ING. JUAN RAUL REA PEREZ

ING. DAVID JAIME GONZALES MAXIMOS

ING. RICARDO TAPIA ARMAS

Informamos a ustedes de la autorización que se le concede al alumno **OSWIN ALFREDO CORONA GARCIA**, para que conjuntamente con **EFREN LOPEZ MARTINEZ** puedan desarrollar el trabajo de tesis **"DISEÑO E IMPLEMENTACION DE UNA INTERFACE GRAFICA CON ORIENTACION A OBJETOS PARA LA CAPTURA Y EDICION DE CARTAS ASE"** dirigida por el *Ing. Roberto Blanco Bautista*, solicitando a ustedes, sean tan amables de revisar el avance del mismo y hacer las observaciones que consideren pertinentes, o en su caso, indicar a los alumnos si dicha revisión se hará a la conclusión del trabajo de tesis.

Sin otro particular, aprovecho la ocasión para enviarles un cordial saludo.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
San Juan de Aragón Edo. de Méx., Noviembre 24 de 1994.
EL JEFE DE LA CARRERA

ING. ENRIQUE VERA MUÑOZ

SVM/pga.

San Juan de Aragón, Edo. de Mexico. a 5 de Septiembre de 1995

Escuela Nacional de Estudios Profesionales Aragón

C. Lic. Alberto Ibarra Rosas
Jefe de la Unidad Académica
PRESENTE.

Por este conducto me dirijo a Ud. para hacer de su conocimiento el término del trabajo de tesis, que con titulo "Diseño e Implementación de una Interface Gráfica con Orientación a Objetos para la captura y edición de Cartas ASM", realizó el alumno Oswin Alfredo Corona García con número de cuenta 8561372-2 de la carrera de Ing. Computación y que estuvo bajo mi dirección.

Así mismo manifiesto que no existe inconveniente alguno para continuar los trámites pertinentes.

Sin más por el momento me despido de Ud., esperando contar con su fina atención, quedo de Ud.

"Por mi raza hablará el espiritu"

Director de Tesis

Ing. Roberto Blanco Bautista

Vo. Bo.

Ing. Silvia Vega Muytoy
Jefe de Carrera

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
ARAGÓN
DIRECCIÓN

EFREN LOPEZ MARTINEZ
P R E S E N T E .

En contestación a la solicitud de fecha 12 de octubre del año en curso, presentada por Oswin Alfredo Corona García y usted, relativa a la autorización que se les debe conceder para que el señor profesor, Ing. ROBERTO BLANCO BAUTISTA pueda dirigirles el trabajo de Tesis denominado " DISEÑO E IMPLEMENTACION DE UNA INTERFACE GRAFICA CON ORIENTACION A OBJETOS PARA LA CAPTURA Y EDICION DE CARTAS ASM ", con fundamento en el punto 6 y siguientes, del Reglamento para Exámenes Profesionales en esta Escuela, y toda vez que la documentación presentada por usted reúne los requisitos que establece el precitado Reglamento; me permito comunicarle que ha sido aprobada su solicitud.

Aprovecho la ocasión para reiterarle mi distinguida consideración.

ATENTAMENTE
"POR MI RAZA HABLARA EL ESPIRITU"
San Juan de Aragón, Mex., 14 de octubre de 1994
EL DIRECTOR

M en I. CLAUDIO C. MERRIFIELD CASTRO

c c p Lic. Alberto Ibarra Rosas, Jefe de la Unidad Académica.
c c p Ing. Silvia Vega Muytoy, Jefe de la Carrera de Ingeniería en Computación.
c c p Ing. Roberto Blanco Bautista, Asesor de Tesis.

CCMC'AIR'11a.

UNIVERSIDAD NACIONAL AVENTMA DE MEXICO

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
CAMPUS ARAGON
JEFATURA DE CARRERA DE INGENIERIA EN COMPUTACION

LIC. ALBERTO IBARRA ROSAS
JEFE DE LA UNIDAD ACADEMICA
P R E S E N T E .

Por este conducto comunico a usted que, el alumno ***EFREN LOPEZ MARTINEZ***, con número de cuenta *8561386-* de la carrera de ***INGENIERIA EN COMPUTACION*** desarrollará el trabajo de Tesis titulado ***"DISEÑO E IMPLEMENTACION DE UNA INTERFACE GRAFICA, CON ORIENTACION A OBJETOS PARA LA CAPTURA Y EDICION DE CARTAS ASN"*** conjuntamente con ***OSWIN ALFREDO COROMA GARCIA***, con número de cuenta *8561372-2*, asesorados por el *Ing. Roberto Blanco Bautista.*

Lo anterior es con el fin de que se les permita realizar los trámites correspondientes a la autorización del trabajo.

Sin otro particular, aprovecho la ocasión para enviarle un cordial saludo.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
San Juan de Aragón, Méx., a 12 de Octubre 12 de 1994.
JEFE DE CARRERA

ING. [illegible] VEGA MOTTOY

SVM/gga.

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGON
JEFATURA DE CARRERA DE INGENIERIA EN COMPUTACION

ING. ROBERTO BLANCO BAUTISTA

ING. ERNESTO PEÑALOZA ROMERO

ING. JUAN RAUL REA PEREZ 19/11/94

ING. DAVID JAIME GONZALEZ MAXINEZ

ING. RICARDO TAPIA ARMAS 9/05/94

Informamos a ustedes de la autorización que se le concede al alumno *EFREN LOPEZ MARTINEZ* , para que conjuntamente con *OSWIN ALFREDO CORONA GARCIA* puedan desarrollar el trabajo de tesis *"DISEÑO E IMPLEMENTACION DE UNA INTERFACE GRAFICA CON ORIENTACION A OBJETOS PARA LA CAPTURA Y EDICION DE CARTAS ASM"* dirigida por el *Ing. Roberto Blanco Bautista*, solicitando a ustedes, sean tan amables de revisar el avance del mismo y hacer las observaciones que consideren pertinentes, o en su caso, indicar a los alumnos si dicha revisión se hará a la conclusión del trabajo de tesis.

Sin otro particular, aprovecho la ocasión para enviarles un cordial saludo.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
San Juan de Aragón Edo. de Méx., Noviembre 24 de 1994.
EL JEFE DE LA CARRERA

ING. SERGIO VEGA MUTTOY

SVM/gga.

San Juan de Aragon, Edo. de Mexico, a 5 de Septiembre de 1995

Escuela Nacional de Estudios Profesionales Aragon

C. Lic. Alberto Ibarra Rosas
Jefe de la Unidad Académica
PRESENTE.

Por este conducto me dirijo a Ud. para hacer de su conocimiento el término del trabajo de tesis, que con titulo "Diseño e Implementación de una Interface Gráfica con Orientación a Objetos para la captura y edición de Cartas ASM", realizó el alumno Efren López Martinez con número de cuenta 8561386-3 de la carrera de Ing. Computación y que estuvo bajo mi direccion.

Asi mismo manifiesto que no existe inconveniente alguno para continuar los trámites pertinentes.

Sin más por el momento me despido de Ud., esperando contar con su fina atención, quedo de Ud.

"Por mi raza hablara el espiritu"

Director de Tesis

Ing. Roberto Blanco Bautista

Vo. Bo.

Ing. Silvia Vega Muytoy
Jefe de Carrera

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
CAMPUS ARAGÓN

UNIDAD ACADÉMICA

Ing. SILVIA VEGA MUYTOY
Jefe de la Carrera de Ingeniería
en Computación,
P r e s e n t e .

En atención a la solicitud de fecha 3 de octubre del año en curso, por la que se comunica que los alumnos EFREN LÓPEZ MARTÍNEZ y OSWIN ALFREDO CORONA GARCÍA, de la carrera de Ingeniero en Computación, han concluido su trabajo de investigación intitulado "DISEÑO E IMPLEMENTACIÓN DE UNA INTERFACE GRÁFICA CON ORIENTACIÓN A OBJETOS PARA LA CAPTURA Y EDICIÓN DE CARTAS ASM", y como el mismo ha sido revisado y aprobado por usted se autoriza su impresión; así como la iniciación de los trámites correspondientes para la celebración del examen profesional.

Sin otro particular, le reitero las seguridades de mi distinguida consideración.

ATENTAMENTE
"POR MI RAZA HABLARA EL ESPÍRITU
San Juan de Aragón, México., 4 de octubre de 1995
EL JEFE DE LA UNIDAD

Lic. ALBERTO IBARRA ROSAS

c c p Asesor de Tesis.
c c p Interesado.

AIR'lla.

El árbol que llena los brazos de un hombre, crece de un pequeño tallo.

La torre de nueve pisos se hizo uniendo piedra tras piedra.

El viaje de mil Leguas comienza por un solo paso.

Lao Tse

Agradecimientos

A mi amada esposa Norma Migueles,

Luz perenne en mi camino.

A mi familia,

Mina y crisol de mi persona.

A mis compañeros y amigos,

Hermandad incondicional e invaluable.

Por su presencia y apoyo, que colman mis menesteres enalteciendo mi espíritu.

Oswin Alfredo Corona García.

Agradecimientos

A MIS ESCUELAS:

- Primaria "Adrián Zamora López", mis primeros conocimientos.
- "Escuela Secundaria Técnica No. 8", todo un reto haber estado ahí.
- "Escuela Secundaria y de Bachilleres Oficial Minatitlán", el pilar para mis posteriores estudios.
- "Universidad Nacional Autónoma de México", mi Alma Mater.

A MIS PROFESORES, por los conocimientos brindados, con dedicatoria especial a:

- Profesora Reyna Cárdenas Torres, por su cariño y los recuerdos.
- Ing. Odilón Huerta Rico por la amistad y las bases sólidas de matemáticas, "Con la comprensión del problema, lo demás es lo de menos".
- M. en C. David Jaimes González Maxinez, modelo a seguir.

A MIS AMIGOS Y COMPAÑEROS, por su amistad desinteresada y porque siempre estuvieron conmigo en los buenos y malos momentos.

A MIS HERMANOS: Angelina, Martín, Carmina, Gracieta, Efraín, Reyna y Araceli, por creer siempre en mi.

A MIS PADRES: Cesáreo López Martínez y Raymunda Martínez de López, por su amor, por su gran fe en mi y porque han sido, son y serán por siempre, la fuente de energía que me impulsa a continuar superándome hoy y siempre, ¡¡¡los amo!!!.

A DIOS, por la vida, la salud, por haber puesto en mi camino a todas esas maravillosas personas sin yo merecerlo y por los grandes momentos de felicidad que he recibido.

A todos mil gracias, espero no haber defraudado a nadie, con cariño:

Efrén López Martínez.

ÍNDICE

"DISEÑO E IMPLEMENTACIÓN DE UNA INTERFACE GRÁFICA CON ORIENTACIÓN A OBJETOS PARA LA CAPTURA Y EDICIÓN DE CARTAS ASM"

OBJETIVO PRINCIPAL:

- Crear una herramienta de software para el auxilio del diseño de controladores digitales.
  - Subobjetivo 1: Desarrollar un ambiente que fundamente la creación de un editor con elementos gráficos.
  - Subobjetivo 2: Utilizar el ambiente desarrollado para permitir el procesamiento de Cartas ASM.
  - Subobjetivo 3: Generar archivos cuya información describa la estructura de una Carta ASM.
  - Subobjetivo 4: Aplicación de los conceptos de orientación a objetos en el desarrollo de la aplicación.

INTRODUCCIÓN

Introducción

La acelerada integración de la computadora a la vida moderna ha provocado el crecimiento de una nueva industria en el escenario mundial, la del software de computadora. Encargado de acercar la máquina a la vida cotidiana, el software populariza el manejo contable automatizado, facilita la edición electrónica y educa a los alumnos de hoy. Pero no siempre fue así, aún tenemos presentes los días cuando se hablaba de la posible expansión de la tecnología informática sobre las actividades humanas y una de las razones que poderosamente ha influido para llegar a la actual realidad de esa expansión ha sido el desarrollo de aplicaciones que usan representación gráfica.

La comunicación gráfica es el más antiguo medio de comunicación que ha usado el hombre. Más expresivos que el lenguaje, los dibujos muestran una idea, una relación entre el autor, su realidad y su obra; que se transmite de un individuo a otro aprovechando una única base conceptual fácilmente disponible a toda una comunidad. Es por eso que, más allá de los límites que imponen las diferencias culturales de los pueblos, los conceptos universales son expresados gráficamente, símbolos comunes como los correspondientes a Alto, Curva, Peligro, Veneno, Paso Peatonal, etc., son comprensibles por todas las culturas que posean vías de comunicación, manejen sustancias peligrosas o convivan con material inflamable.

Puesto que las comunidades suelen dividirse en grupos especializados de trabajo, como lo son los contadores, los economistas, los arquitectos, diseñadores, abogados, etc., también pueden reconocerse conceptos generalmente exclusivos a ellos. Así, la balanza de pagos, la desviación estándar, el isométrico, la perspectiva, la ergonomía y la legislación son todos elementos que, si nos son mencionados, los relacionamos rápidamente con aquellas profesiones. Ahora, si nos encontramos ante la aplicación de la computadora a los distintos dominios del saber humano, podremos ver que las aplicaciones destinadas al usuario final han resuelto el problema de enfrentarse a la variedad conceptual por medio de símbolos que traducen un concepto abstracto del usuario en una o más acciones de las posibles que ejecutarán los sistemas.

Podemos decir entonces que la tendencia actual de las aplicaciones en una computadora es desarrollar sistemas bajo un ambiente gráfico, en forma modular, de manejo sencillo y de fácil comprensión para el usuario, para que éstos últimos vean mejorada su eficiencia por medio de herramientas que brindan un mayor potencial productivo en su campo de aplicación, mientras que del lado informático, más computadoras son vendidas y más aplicaciones son desarrolladas para cubrir nuevas necesidades.

En los últimos años han florecido un gran número de sistemas gráficos, donde la comunicación se hace mediante símbolos sencillos, muy semejantes al ámbito cotidiano, tanto en la entrada de datos

como en los resultados. A tales símbolos se les ha denominado iconos, en una analogía a las representaciones religiosas de la iglesia que muestran con una sola imagen toda una escena bíblica.

Ejemplos de aplicaciones bajo ambiente gráfico son los paquetes para microcomputadora: MS-Windows, AutoCad, Ventura Publisher, etc. Sin embargo durante los últimos quince años las aplicaciones gráficas eran excepcionales y hasta hace poco, sumamente especializadas en tareas científicas. No debemos olvidar que las computadoras cumplen ya veinticinco años en el ámbito comercial y que el porqué del surgimiento tardío de los ambientes gráficos en computadora, tiene que ver con la tecnología electrónica en constante perfeccionamiento y que hasta hace relativamente pocos años a otorga la posibilidad gráfica.

Esta evolución de la capacidad gráfica de las computadoras interactúa estrechamente con el desarrollo de las aplicaciones. Así, el aumento de la capacidad del hardware produce un refinamiento en las técnicas del diseño de las aplicaciones que aprovechen las nuevas facilidades, lo que en términos de relación nivel/complejidad del software, ha llevado a los programadores a generar códigos de miles de líneas, en que la mayoría de éstas se dedica a facilitar el entendimiento y manejo del mismo.

Pero la electrónica, en especial la digital, no sólo forma parte de la tecnología informática, sino que así como en aquella, se ha ido incorporando a la vida moderna haciendo extensivo el uso de dispositivos cada vez más pequeños que realizan tareas especializadas dentro de la industria y el hogar y de una forma tan eficiente y económica que ya es difícil pensar la vida moderna sin ellos.

Dentro de la electrónica digital, los controladores son los elementos que determinan el flujo del proceso, son conjuntos de dispositivos interconectados para lograr, con su sinergia, responder a distintas situaciones con determinadas acciones. Son de hecho elementos independientes que pueden a su vez formar parte de sistemas controladores mayores que realicen tareas proporcionalmente complejas. Cada uno de esos controladores deberá ser diseñado para realizar su labor sin error y de la forma más eficiente posible para obtener en consecuencia un sistema controlador con los mismos atributos.

El proceso de desarrollo de circuitos controladores requiere de un método para su diseño, siendo parte de ellos los métodos basados en la Síntesis de Cartas ASM, que llaman la atención por hacer uso de una herramienta especial, la Carta ASM, una descripción gráfica o diagrama del algoritmo que caracterizará al controlador a diseñar. Dicha descripción conjuga elementos bien definidos en una estructura semejante a la usada por los diagramas de flujo, herramienta de descripción de procesos utilizada en el diseño de software. Además, ya qua esos métodos proporcionan una forma de construir un equivalente tabular -que no proposicional como en la programación- del diagrama, para luego

aplicar cualquier técnica de interpretación ya establecida que nos lleva al resultado final, son también susceptibles de ser procesados en una computadora por una aplicación de software.

La idea del desarrollo de un editor gráfico nace en las aulas y laboratorios de la carrera de Ingeniería en Computación de la UNAM Campus ENEP-Aragón, con la intención de aplicar la práctica informática a la intensa preparación en electrónica digital que llena uno de los perfiles de la carrera.

Desde un punto de vista práctico, la existencia de un sistema semejante otorgaría muchos beneficios a alguien experimentado en algún método y que tuviera que diseñar un controlador con una Carta ASM reducida, pero mejor aún, para casos más complejos o para diseñadores novicios, un sistema así resultaría en un razonable ahorro de tiempo o en un auxiliar contra posibles errores. La necesidad pues de un sistema procesador de Cartas ASM será proporcional a la complejidad del problema a enfrentar.

Ya que la función primordial del sistema consistirá en facilitar el procesamiento de la información derivada de una carta ASM, a fin de obtener la solución al problema por medio de una técnica aplicable a los controladores digitales, debemos pensar en la forma de lograr este objetivo. Como resultado de ello y dados los actuales requerimientos para la presentación de este trabajo, el equipo de desarrollo decidió dividir el proyecto en dos partes, ambas presentables en dos distintos pero muy relacionados trabajos de tesis, uno un editor gráfico de Cartas ASM, que es el que nos ocupa y otro, el proceso de solución o síntesis de Cartas ASM por medio del editor.

El planteamiento original del equipo de desarrollo contempla un objetivo principal único para el proyecto y una serie de subobjetivos particulares para esta parte del proyecto:

Objetivo principal: Crear una herramienta de software para el auxilio del diseño de controladores digitales.

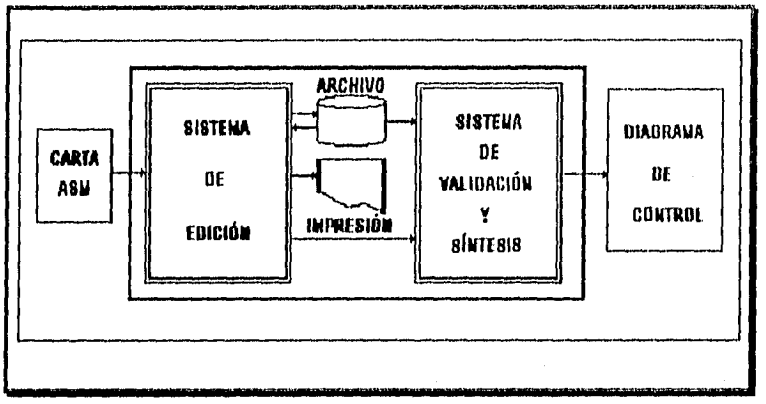
Subobjetivo 1: Desarrollar un ambiente que fundamente la creación de un editor con elementos gráficos.

Subobjetivo 2: Utilizar el ambiente desarrollado para permitir el procesamiento de Cartas ASM.

Subobjetivo 3: Generar archivos cuya información describa la estructura de una Carta ASM.

Subobjetivo 4: Aplicación de los conceptos de orientación a objetos en el desarrollo de la aplicación.

El sistema planteado tendrá una estructura general como la que se muestra a continuación:



Para el subobjetivo 1, dentro de la realización de la primera parte del proyecto, la generación de la aplicación, que por la naturaleza de los diagramas de Cartas ASM, es necesariamente bajo el ambiente gráfico de la computadora IBM PC compatible, fueron evaluadas distintas tecnologías disponibles en el mercado. En atención al subobjetivo 4, algunos lenguajes de programación utilizados por la comunidad universitaria como son el Pascal y el "C" fueron contemplados inicialmente, así como el SmallTalk V, el lenguaje orientado a objetos por antonomasia, sin embargo, un lenguaje híbrido, el C++ -en la versión de Borland Turbo C++-, fue elegido por su similitud al popular lenguaje "C" y por presentar la oportunidad de ejemplificar la nueva tecnología de la orientación a objetos en un caso particularmente propicio como lo es éste, en que el sistema se compone de elementos gráficos genéricos y donde dicha tecnología puede hacer un mejor modelado del problema a la vez que facilita el desarrollo inicial y el mantenimiento ulterior.

Esta primera parte del desarrollo será la encargada, como apuntamos en el subobjetivo 2, de la función de captura de la carta ASM en forma general (SISTEMA DE EDICIÓN), es decir, sin importar la arquitectura con la cual se sintetizará. Obedeciendo al subobjetivo 3 el sistema generará básicamente dos archivos: el primero describiendo el flujo de la carta, registrando la interconexión entre sus elementos, y el segundo conteniendo la información acerca de las salidas de aquella, en los puntos donde existan. Así mismo, se tiene la posibilidad de imprimir dicha carta.

Los dos archivos anteriores forman la base del desarrollo complementario del sistema, dejando el resto del proyecto para una segunda parte (SISTEMA DE VALIDACIÓN Y SÍNTESIS CON CONTROL MICROPROGRAMADO), en donde se desarrollará el circuito del controlador con la arquitectura microprogramada conocida como MICA I. Esta segunda parte proporcionará los resultados finales del sistema propuesto, que incluyen: tablas particulares de la arquitectura, tablas de resultados de la carta

a sintetizar, tablas donde se muestra el microprograma en formato binario y hexadecimal, y el diagrama del circuito controlador.

En cumplimiento del objetivo del proyecto, la segunda parte efectúa el proceso completo de síntesis, utilizando una de las posibles arquitecturas de diseño y deja abierta la posibilidad de realizar posteriores trabajos para la implementación con diferentes arquitecturas (como sería el diseño con contadores, con MICA II, etc.), utilizando como base los resultados del presente trabajo.

Puesto que este trabajo pretende ser un material didáctico, hemos identificado los aspectos tecnológicos que serán comprendidos:

Análisis y Programación de sistemas: Transportar un esquema de trabajo definido a su representación informática (algoritmos y datos) mediante modelado del problema orientado a objetos y ejemplificar los concepto de la orientación a objetos.

Diseño de controladores digitales: Automatizar la abstracción propia de los métodos de diseño basados en síntesis de Cartas ASM.

La siguiente es la lista de los capítulos que constituyen este trabajo:

Capítulo I. Ambiente gráfico en la computadora. Donde se describirán el manejo y características que comprende el modo gráfico en una microcomputadora IBM PC o compatible.

Capítulo II. Tópicos de orientación a objetos. Donde se muestran los conceptos básicos de la filosofía de la programación orientada a objetos.

Capítulo III. Cartas ASM (Algorithm State Machine). Teoría de Cartas ASM necesaria para la comprensión de los lineamientos del diseño.

Capítulo IV. Desarrollo de la aplicación. Donde se muestra el análisis y diseño de los elementos de la aplicación, utilizando una herramienta que maneja los conceptos de la Orientación a Objetos.

Conclusiones: Donde se evalúan los resultados obtenidos y se menciona la posibilidad de extender los alcances del sistema.

CAPÍTULO I

AMBIENTE GRÁFICO EN LA COMPUTADORA

CAPÍTULO I

AMBIENTE GRÁFICO EN LA COMPUTADORA

Se dice que una imagen vale más que mil palabras. Aunque ésta es una frase muy trillada, es cierta, y la computación gráfica la expresa en forma moderna. Actualmente es posible crear y modificar ilustraciones utilizando sistemas modernos de gráficos por computadora.

Aunque los ingenieros y científicos han empleado ilustraciones durante cientos de años para representar y transmitir ideas, y han utilizado computadoras por lo menos durante 30 años, no ha sido sino hasta los últimos 5 a 15 años que los ambientes gráficos por computadora han dejado de ser costosos y difíciles de usar.

Los ambientes gráficos por computadora parecen estar sintonizados con la manera como piensa la gente, es decir, con el diseño funcional del cerebro. Los ambientes gráficos muestran imágenes de objetos reales o acciones en pantalla a las que el usuario puede apuntar o seleccionar. Estas imágenes se llaman iconos, un ejemplo de éstos sería la figura de un lápiz con goma de borrar para indicar la acción de desechar algo o borrarlo. Otro ejemplo, sería un disquete para representar el almacenamiento o recuperación de información. Cada icono permite al usuario pensar en términos reales acerca de los elementos e información electrónicos. Esto implica que se requiere muy poca instrucción para utilizar satisfactoriamente un sistema diseñado de esta manera.

Algunas aplicaciones gráficas requieren alta capacidad interactiva entre el usuario y la computadora, y virtualmente demandan un "Ratón" ("Mouse") como dispositivo primario de entrada. En muchos casos, los gráficos han sido el medio de interacción preferido contra la opción del manejo por texto; es por eso que la programación básica de estas aplicaciones cobra una gran importancia en el ámbito comercial.

En su contraparte técnica, la evolución en tas aplicaciones gráficas implica un mayor conocimiento interno de la computadora y de cómo disponer de ella. Para programar una computadora en forma eficiente, es indispensable un conocimiento firme del lenguaje de programación que se va a emplear, y la programación gráfica no es la excepción. Los programas de gráficos utilizan las mismas proposiciones que emplean los demás programas, así como otras especiales para gráficos. Los elementos de lenguajes específicos para gráficos, que normalmente no se mencionan en los cursos convencionales de programación se cubrirán -sin entrar en aspectos particulares de algún lenguaje de programación específico- en los siguientes párrafos.

El presente capítulo contiene un marco teórico general para el desarrollo de aplicaciones e interfaces gráficas, que permite reflejar los requisitos necesarios para todo aquel interesado en el tema.

Se hablará de los elementos con que deben contar los lenguajes que se usan en la programación en modo gráfico. Cada elemento se presentará junto con una explicación general de la forma como se emplea y sus limitaciones.

I.1 Ventajas

Las ventajas de las diferentes aplicaciones de los gráficos por computadora son evidentes en la productividad, creatividad y rapidez. No hay duda que los sistemas de computación gráfica tienen un costo apreciable, tanto económico como en esfuerzo de desarrollo. Sin embargo, los beneficios que pueden obtenerse compensarán con creces la inversión, si consiguen liberar a las personas de trabajos repetitivos, volviéndolas así más creativas y, por tanto, más productivas. Además, como las computadoras y sus sistemas de gráficos permiten examinar eficientemente muchas alternativas y escoger la mejor basados en los requerimientos especificados para cada problema en particular, reducirán el número y la magnitud de los errores que cuestan tiempo y dinero.

I.2 Antecedentes

Cuando las primeras microcomputadoras aparecieron en el mercado, una Unidad Central de Procesamiento (CPU, por sus siglas en inglés, Central Processing Unit) de 8 bits y arquitectura de bus de 16 KiloByte (KB) de Memoria de Acceso Aleatorio (RAM, en inglés: Random Access Memory), fueron el estándar; las velocidades de reloj de 2 MHz fueron consideradas rápidas y las pantallas de vídeo con 25 líneas de 80 caracteres cada una, fueron los límites absolutos. Si se requería desplegado gráfico, un conjunto de caracteres gráficos definidos en la Memoria de Sólo Lectura (ROM: Read Only Memory) fue la única alternativa al conjunto de caracteres ASCII, usados hasta ahora, para representar en un arreglo matricial de 8 x 9 bits patrones de caracteres. Se ofrecían pocos medios para acceso directo a la memoria de vídeo de la máquina y las limitaciones del sistema de memoria y de la CPU lo hacían difícil.

Este inicio del manejo de la computadora se dividió en dos partes: por un lado, sistemas concentrados en el uso eficiente de la memoria y de la velocidad de procesamiento de la computadora, y por el otro, sistemas "orientados a juegos", que sacrificaban muchas de las capacidades de la computadora en favor de la elaboración de gráficos monocromáticos o en color.

Con el mejoramiento de los componente de la computadora, es decir, con CPU's y sistemas operativos más poderosos, y velocidades superiores, fueron posibles mayores facilidades de uso, tanto del hardware como del software de la computadora. Al usar el Z-100 (el sistema operativo DOS inicial) por ejemplo, tres bancos de memoria de vídeo estuvieron disponibles, cada uno con 32 ó 64 KB de RAM, para controlar los colores rojo, azul y verde (hasta 192 KB en total). Si el Z-100 contaba sólo con uno de tres bancos (por omisión el banco del color verde), se limitaba al desplegado monocromático o,

si se tenían los tres bancos de RAM de vídeo pero un monitor monocromo, el color se simulaba con "tonos de gris". Con un monitor de alta resolución a color se presentaba la opción de 8 colores (negro, azul, violeta, rojo, naranja, verde, amarillo y blanco), con la combinación de los tres originales.

Algo relevante fue el hecho de tener memoria de vídeo de pixel (de la contracción de Picture Element: Elemento de imagen más pequeño) direccionables, con rango de despliegue de 640 pixeles horizontales por 225 pixeles verticales. Recordando que para texto, el desplegado fue de 25 líneas (renglones) de 80 caracteres cada una, pero no habla distinción necesaria aun entre los modos texto y gráfico; ambos eran lo mismo.

Se podía también, escribir (y leer) pixeles individuales en la memoria RAM de vídeo, entremezclando éstos con caracteres estándares sin cambiar de modo de presentación o deshabilitar un tipo de desplegado en favor de otro.

Con la arquitectura de la computadora propuesta por IBM -usando esencialmente la misma CPU, con bus de datos de 16 bits-, pero con 4 KB de RAM dedicada al vídeo limitada para desplegar únicamente texto, resultó una configuración de hardware reducida. Si se requerían capacidades gráficas, estaba disponible el Adaptador Gráfico a Color (CGA, Color Graphic Adapter) como un añadido, que provela gráficos de alta resolución (640 x 200 pixeles) con dos colores o de baja resolución (320 x 200 pixeles) con pantalla de cuatro colores previamente seleccionados. Con esto se introduce el concepto de **sistema de vídeo**, que separaba a la memoria de vídeo y su manejo, de la tarjeta principal, facilitando su posterior mejoramiento independiente -del sistema de vídeo-.

Con esto quedó establecida la diferencia al manejar los dos modos posibles de la computadora: **modo texto y modo gráfico.** El desarrollo de aplicaciones bajo el ambiente gráfico, sólo dependió del tiempo transcurrido en mejorar los componentes electrónicos de entonces, lo que permitió el florecimiento de áreas de trabajo como producto de la integración de la computación en otras ramas del conocimiento; así, tenemos por ejemplo el CAD/CAM (Diseño/Manufactura Asistido por Computadora - Computer Aided Design/Manufacture-) aplicado a la arquitectura, diseño industrial, electrónico, metal-mecánico y estructural.

Las aplicaciones gráficas actuales se desarrollan alrededor de los requerimientos específicos de cada área del conocimiento, pero independientemente, las interfaces gráficas de usuarios se diseñan para que sean de fácil uso, de agradable presentación y con alta capacidad de ayuda del sistema, haciendo uso de diversos elementos gráficos como son: iconos, ventanas de presentación, cajas de diálogos de usuario, barras de menús, paneles de configuración, cursores de dispositivo apuntador, etc.

I.3 Modos de trabajo en la computadora personal

Como ahora sabemos, las computadoras personales tienen dos modos de trabajar para visualizar información en pantalla, conocidos como modos de vídeo: modo texto y modo gráfico. Tanto en el modo texto como en el modo gráfico existen diferentes opciones de manipular la cantidad de información presentada en pantalla, dependiendo esto, de la resolución de trabajo seleccionada para un adaptador en particular, entendiendo por resolución al producto de caracteres ASCII por columna y renglones para el modo texto y al producto de un número determinado de pixeles verticales por otro número de pixeles horizontales para el modo gráfico. A continuación mencionaremos como funcionan estos modos de trabajo para darnos una idea clara de la diferencia entre ellos.

I.3.1 Modo texto

Cuando se emplea el modo texto de la computadora personal, puede desplegarse cualquiera de los 256 caracteres ASCII estándares. Estos caracteres residen permanentemente en la ROM de la computadora.

Como se mencionó anteriormente, para desplegar información, la computadora usa un adaptador de vídeo, el cual es una tarjeta sobre la que van montados circuitos, que conecta a la computadora con el monitor. La mayoría de las computadoras personales presentaban comúnmente los siguientes adaptadores: monocromáticos, gráficos a colores (CGA) y los gráficos mejorados (EGA, en inglés, Enhaced Graphic Adapter). En la actualidad los adaptadores de arreglo gráfico de vídeo (VGA, Video Graphic Array) y Super VGA son los más populares. Este capítulo se centra en los adaptadores monocromos y de gráficos en color, pero los conceptos que se ven también se pueden aplicar a los demás adaptadores de vídeo.

De esta forma, cuando un carácter es introducido por medio del teclado, el BIOS (el Sistema de Entrada/Salida Básico de la computadora personal -Basic Input/Output System-) coloca la representación de su código en la memoria de vídeo, donde a su vez, el controlador de vídeo se encarga de desplegarlo en la pantalla.

En una opción particular de presentación de información del modo texto, el monitor puede desplegar hasta 80 caracteres horizontales por 25 verticales, lo que da un total de 2000 caracteres en una pantalla completa. Cada carácter tiene su forma (foreground) y su fondo (background). El color de la forma es el color del carácter en sí mismo, el color del fondo es el color del espacio alrededor del carácter.

La computadora almacena forma y fondo en la **Memoria de Vídeo**. Ésta le dice a la computadora cuales son las características de la forma y del fondo del carácter a desplegar. El primer byte en la memoria de vídeo, contiene el primer carácter que aparecerá en la esquina superior izquierda del monitor. Así, si el primer byte contiene el valor 41h (el valor ASCII en hexadecimal para la letra "A"), la

controladora dibuja en el monitor la letra "A" en la esquina superior izquierda; el segundo byte es el byte de atributos para el primer carácter, contiene el color y otras informaciones de despliegue.

Este patrón -carácter, atributo, carácter, atributo- se repite para todos los 2000 caracteres que aparecerán en el monitor. Así, el contenido de una sola pantalla en este modo en particular, requiere de 4000 bytes de memoria de vídeo. A continuación veremos en forma detallada como está estructurado el byte de atributos para los caracteres ASCII.

I.3.1.1 El byte de atributos

Una computadora con adaptador gráfico a color es capaz de desplegar hasta 16 diferentes colores, cada uno de los cuales puede estar formado de hasta 4 elementos: azul, rojo, verde y brillo. El color de desplegado del carácter, depende de la combinación particular de estos elementos. Por ejemplo, el color negro no combina ningún elemento, mientras que el cyan claro combina el azul, el verde y el brillo. Los adaptadores de vídeo también soportan un elemento que no tiene efecto sobre el color, pero que hace que el carácter parpadee.

El byte de atributos almacena las características del fondo y de la forma para el byte de carácter que le precede. La Fig. I.1, muestra cómo contribuye cada bit en el color del carácter y de su fondo.

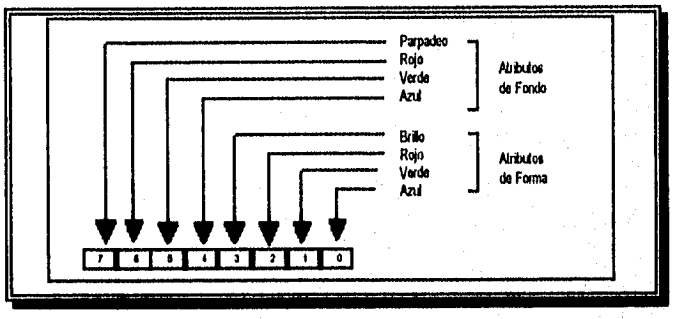


Fig. I.1.- Mapa del byte de atributos.

Los primeros tres bits (0,1,2) en el byte de atributos, controlan la forma del carácter, mientras que el cuarto bit (bit 3) añade brillo al color. Se pueden combinar estos cuatro elementos para crear 16 colores para la forma del carácter. Los bits 4, 5 y 6 determinan el color de fondo del carácter, y el bit 7, cuando está activado, hace que el carácter parpadee. Debido a que el fondo no tiene el elemento brillo, únicamente 6 colores están disponibles para él.

En un monitor monocromático, el byte de atributos puede crear únicamente 5 formatos de desplegado: oculto, normal, subrayado, brillo e inverso. Así, como con los adaptadores gráficos a colores, el brillo está controlada por el bit 3 y el parpadeo es controlado por el bit 7. Los caracteres están ocultos cuando el fondo y la forma son del mismo color. El brillo y el parpadeo no tienen efecto con caracteres ocultos.

Para desplegar caracteres en vídeo inverso (caracteres obscuros sobre un fondo iluminado), se pone el fondo a luz gris (blanco sin brillo) y la forma a negro. Los caracteres en vídeo inverso no tienen brillo, pero pueden ser parpadeantes.

El subrayado es una especialidad del adaptador monocromático. Un carácter subrayado, sólo se puede visualizar cuando el bit 0 (el color azul) está activo, puede ser mostrado brillante y ser parpadeante, pero no puede ser mostrado en vídeo inverso.

I.3.1.2 Alternativas del modo texto de despliegue de la computadora personal

Las computadoras personales soportan 16 diferentes modos de trabajo de desplegado y 5 de éstos son en modo texto. Estos modos, listados en la tabla I.1, controlan el tamaño de los caracteres y el color en el desplegado. El modo 0, por ejemplo, despliega caracteres alargados (40 por línea) en tonos de gris.

Los cuatro primeros modos de despliegue, del 0 al 3, trabajan únicamente con CGA y EGA, la mayoría de los cuales pueden conmutar entre los cuatro modos. El modo 7, es usado únicamente por adaptadores monocromáticos.

MODO	TAMAÑO	COLORES	ADAPTADOR
0	40 x 25	Tonos de gris	CGA, EGA
1	40 x 25	Colores	CGA, EGA,
2	80 x 25	Tonos de gris	CGA, EGA
3	80 x 25	Colores	CGA, EGA
7	80 x 25	Blanco y Negro	Monocromático

Tabla I.1. Modos Texto de despliegue de la computadora personal.

El monitor de la computadora personal consta generalmente de 25 líneas de 80 caracteres, dependiendo estos datos de la interface de vídeo instalada y del modo seleccionado. La línea superior es el renglón 1 y la posición más a la izquierda dentro de éste, es la columna 1. Las posiciones dentro de la pantalla se refieren comúnmente como X y Y, donde X es la posición de la columna y Y es la posición del renglón (las coordenadas son indicadas en formato X,Y).

I.3.2 Modo gráfico

La capacidad del manejo de la computadora personal en modo gráfico, es una de sus características más relevantes. Usando gráficos se pueden crear dibujos, diagramas, texto multifuente o cualquier cosa que pueda ser dibujada. Se pueden desarrollar métodos para dibujar líneas o caracteres. En la actualidad hay más de 10 adaptadores gráficos disponibles en el mercado, comprendiendo más de 2 modos de resolución de gráficos diferentes cada uno.

Comparado con el modo texto, el modo gráfico de la PC requiere mayor detalle en su manejo. Desplegar información en modo texto en la pantalla, es tan simple como poner caracteres ASCII en localidades especificas de memoria, y como se vio anteriormente, la controladora realiza el despliegue de éstos en la pantalla.

El modo gráfico requiere una orientación completamente diferente, en vez de caracteres y bytes de atributo se tienen pixeles, los elementos de imagen más pequeños en la computadora. Un carácter sencillo en la pantalla de la computadora, está hecho de muchos pixeles arreglados en un patrón que le da forma. En el modo gráfico, se pueden iluminar pixeles en lugares específicos de la pantalla.

La pantalla de gráficos consiste de pixeles ordenados en líneas horizontales y verticales. Esto es verdad en todos los modos gráficos de la computadora; la mayor diferencia es el tamaño del pixel. La resolución depende de la interface de vídeo instalada, siendo las más usuales:

CGA (Color Graphics Adapter) con:

16 colores en modo texto y 2, 4 ó 16 colores en modo gráfico.

Resolución de la pantalla:

640 x 200 pixeles, resolución alta y 2 colores.

320 x 200 pixeles, resolución media y 4 colores.

160 x 200 pixeles, resolución baja y 16 colores.

HERC con:

Resolución de la pantalla:

720 X 348 pixeles, resolución alta.

EGA (Enhanced Graphics Adapter) con:

4 ó 16 Colores dependiendo del modo de video.

Resolución de la pantalla:

640 x 350 pixeles, resolución alta y 16 colores.

640 x 200 pixeles, resolución media y 16 colores.

320 x 200 pixeles, resolución baja y 16 colores

640 x 350 pixeles, resolución alta y 4 colores (EGA64).

640 x 350 pixeles, resolución alta y 16 colores (EGA64).

PGA (Professional Graphics Adapter) con:

Resolución de la pantalla:

640 x 480 pixeles, 256 colores.

MCGA (Memory Controller Gate Array) con:

2 ó 256 colores en modo gráfico.

Resolución de la pantalla:

640 x 480 pixeles, resolución alta y 2 colores.

320 x 200 pixeles, resolución baja y 256 colores.

VGA (Video Graphics Array) con:

2, 16 ó 256 colores en modo gráfico.

Resolución de la pantalla:

640 x 480 pixeles, resolución alta y 16 colores.

640 x 480 pixeles, resolución alta y 2 colores.

320 x 200 pixeles, resolución baja y 256 colores.

VGAMONO (Video Graphics Array Monocromo) con:

Resolución de la pantalla:

640 x 350 pixeles.

SVGA (Super Video Graphics Array) con:

Resolución de la pantalla:

800 x 600 pixeles, resolución alta y 16 colores.

IBM8514/A con:

16 ó 256 colores en modo gráfico.

Resolución de la pantalla:

1024 x 768 pixeles, resolución alta y 256 colores

1024 x 768 pixeles, resolución alta y 16 colores

640 x 480 pixeles, resolución baja y 256 colores

XGA (Extended Graphics Array) con:

256 ó 65,536 colores en modo gráfico.

Resolución de la pantalla:

1024 x 768 pixeles, resolución alta y 256 colores.

640 x 480 pixeles, resolución media y 65,536 colores.

640 x 480 pixeles, resolución media y 256 colores.

ATT400 con:

Resolución de la pantalla:

640 x 400 pixeles, resolución alta y 2 colores.

640 x 200 pixeles, resolución media y 2 colores.

320 x 200 pixeles, resolución baja (los colores depende de la paleta gráfica definida).

PC3270 con:

Resolución de la pantalla:

720 x 350 pixeles, resolución alta y 2 colores.

Cada adaptador tiene uno o más modos de resolución de gráficos y cada modo de resolución tiene un sistema coordenado asociado. Tal como se muestra en la Figura I.2, las coordenadas gráficas comienzan en la posición (0,0), la cual está localizada en la esquina superior izquierda de la pantalla. La esquina inferior derecha de la pantalla está definida por las coordenadas máximas del modo de trabajo activo.

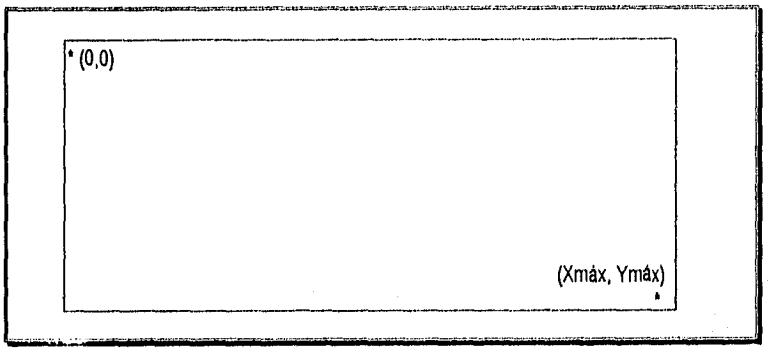


Figura I.2. Sistema Coordenado.

De lo anterior, se puede deducir que, es muy importante conocer qué adaptador gráfico está instalado y el modo de resolución en que se trabaja, pues si se desea iluminar el pixel de la esquina inferior derecha de la pantalla, se necesita conocer el sistema de coordenadas con que se trabaja.

I.3.2.1 Texto en modo gráfico

En modo gráfico, las clásicas pantallas de texto ya no tienen ninguna validez y, tanto los rótulos, como la información textual, sólo podrán representarse a través de terminales gráficos.

Los gráficos sin texto no tendrían mucha utilidad. La posibilidad de combinar texto y gráficos, es una de las herramientas más poderosas de los lenguajes que soportan el modo gráfico, y es la base de la industria de la autoedición. Este tipo de lenguajes cuenta por lo regular con diferentes fuentes de letras (fonts), por ejemplo: una fuente de carácter en mapa de bits y otras fuentes definidas mediante algoritmo.

En modo gráfico, las pantallas de texto gráfico operan de una forma absolutamente diferente a las pantallas de texto convencionales. Por ejemplo, las posiciones de los caracteres en las pantallas clásicas (coordenadas de fila y columna) ya no son aplicables; ahora, los caracteres pueden aparecer en cualquier posición de la pantalla. Harán falta nuevas funciones para marcar con exactitud las posiciones de visualización de texto gráfico y para escribirlo sobre la pantalla.

Dada la gran variedad de tamaños de caracteres, el entorno gráfico ofrece diferentes justificaciones, tanto horizontales como verticales. Además, puesto que el texto puede mostrarse tanto en sentido vertical como horizontal, habrá una gran confusión en los desplazamientos de las líneas y en las posiciones de visualización.

Una fuente de mapa de bits es definida por una matriz de pixeles, cuando ésta es agrandada, la matriz es multiplicada por un factor de escala y el carácter pierde definición. En contraparte, debido a que los algoritmos son insensibles a la escala, las fuentes definidas mediante algoritmo no son deformadas al agrandarlas.

Para **seleccionar una fuente**, se usa un procedimiento integrado del lenguaje que soporta el modo gráfico, el cual requiere de tres parámetros. El primer parámetro es el tipo de letra, el segundo es la orientación en que el texto es escrito (horizontal o vertical), el tercero controla el tamaño del carácter (estos pueden ser incrementados por lo regular hasta 10 veces el tamaño normal).

Para **desplegar texto**, se utiliza otro procedimiento integrado del lenguaje, que requiere de tres parámetros; los dos primeros definen la posición del pixel donde empieza el texto (en formato X:Y), el tercero define el texto en sí.

Para textos estándar, la fuente por mapa de bits es suficiente, pero para textos con diferentes estilos, es mejor utilizar las fuentes definidas mediante algoritmo. Como por lo regular, en el modo gráfico no se cuenta con funciones que realicen la tarea de leer de teclado algún texto y después desplegarlo en el monitor, se tiene que implementar una función que tome en cuenta los dos procedimientos anteriores así como otras consideraciones que quedan fuera del alcance de este tema presentarlas.

I.3.2.2 Algunas funciones del modo gráfico

A continuación presentaremos las funciones más relevantes del modo gráfico, las cuales abarcan las utilizadas por el sistema a desarrollar. Todas o la gran mayoría de estas funciones casi siempre están presentes en el desarrollo de cualquier aplicación en modo gráfico, siendo el presente trabajo uno de ellos.

Las funciones de programación, entre ellas las gráficas, hacen uso de las rutinas del BIOS, que atienden entre otras cosas, la administración del sistema de video, donde la tarjeta controladora es reconocida como un área de memoria. El utilizar directamente las funciones gráficas hace transparente el trabajo del BIOS para el usuario de lenguajes de alto nivel.

Para mostrar algunas de las funciones gráficas existentes, podemos agruparlas de acuerdo a la tarea que desempeñan. Aquí es conveniente mencionar que la presentación de dichas funciones no será basada en algún lenguaje de programación en particular, sino más bien se tratarán en un formato más comprensible de ellas, así tenemos:

- Funciones relativas a configuración.
- Funciones relativas a coordenadas.
- Funciones para obtener o poner atributos.
- Funciones para dibujar y visualizar imágenes.
- Funciones para visualizar texto.
- Funciones para transferir imágenes (animación).

I.3.2.2.1 Funciones relativas a configuración

Dentro de esta clasificación, se abarcan funciones tanto para determinar la configuración para inicializar el sistema gráfico, como para finalizarlo.

Det_Tarj(). *Detecta existencia de tarjeta gráfica en el sistema, reportando el tipo adecuado de controlador o un código de error en caso contrario.*

Modo_Video(Modo). *Selecciona la modalidad de video apropiada para el adaptador de video instalado en la computadora personal. El parámetro* **Modo** *determina cuál de los modos de video disponible se usará.*

Activa_Pág(Pág). *Para configuraciones con suficiente memoria, que soportan múltiples páginas de video,* ***Activa_Pág()*** *especifica el área de memoria donde son almacenados los resultados gráficos de la ejecución del programa. El parámetro* **Pág** *selecciona la página activa en un instante determinado (por omisión es la página 0).*

Visualiza_Pág(Pág). *Selecciona la página de video a visualizar. Mientras tanto, el programa puede continuar e ir almacenando resultados gráficos en otra página activa. El parámetro Pág especifica la página a visualizar, siendo la 0 por omisión.*

Máx_Mod_Gráf(). *Obtiene el máximo modo gráfico de la tarjeta.*

Ini_Parám_Gráf(CtrGráf,ModoGráf,RutaCtrldr). *Esta función permite activar determinados parámetros gráficos con valores, cargar los controladores gráficos y poner el sistema en el modo gráfico adecuado.*

Reinic_Parám(). *Restablece parámetros de inicialización gráfica a valores por omisión.*

Cam_Mod_Txt(). *Conmuta al modo texto.*

Fin_Gráf(). *Cierra el modo gráfico del sistema para prepararlo a modo texto.*

I.3.2.2.2 Funciones relativas a coordenadas

En esta clasificación entran todas aquellas funciones que proporcionan información o establecen ubicación de los elementos en pantalla.

Máx_Coord(). *Obtiene coordenadas máximas del modo de resolución de sistema establecido.*

Cambia_Origen_Coord(X,Y). *Establece el origen lógico*[1] *(0,0) de las coordenadas, en el punto físico definido por los parámetros (X, Y).*

> *Un sistema de coordenadas lógicas es creado al mover el origen a una posición determinada; a partir de ese instante, el origen (0,0) estará en la posición indicada, por lo que el resto de las funciones gráficas referirán los valores de coordenadas empleados a este nuevo punto. Los valores de X y Y, mantienen su orientación.*

Coord_Pix(). *Obtener coordenadas de pixel en X, Y. Algunos sistema proporcionan funciones separadas para obtener a X y Y.*

Traslada_Coordenadas_Físicas(X,Y). *Traslada las coordenadas físicas (X,Y) a coordenadas lógicas y devuelve los valores correspondientes en un arreglo.*

Traslada_Coordenadas_Lógicas(X,Y). *Traslada las coordenadas lógicas (X,Y) a coordenadas físicas y devuelve los valores correspondientes en un arreglo.*

Limita_Área(X_1,Y_1,X_2,Y_2). *Limita el área de visualización de la pantalla al rectángulo definido por los parámetros (X_1, Y_1) y (X_2,Y_2).*

Limita_Área_Cambia_Origen(X_1,Y_1,X_2,Y_2). *Limita el área de visualización de la pantalla a un rectángulo definido por (X_1,Y_1), como esquina superior izquierda, y por (X_2,Y_2), como esquina inferior derecha, y cambia al sistema de coordenadas lógicas situando el origen en el punto físico (X_1,Y_1).*

I.3.2.2.3 Funciones para obtener o poner atributos

Clasificación de las funciones para establecer configuración adecuada del sistema gráfico como son: parámetros de color, tipo de líneas, ubicación de ventanas, patrones de relleno, etc.

Muestra_Cursor(Parám). *Determina si el cursor en modo gráfico se hace visible o no. El parámetro Parám puede ser: visible/no visible.*

Fija_Color_Fondo(ColorFdo). *Establece el color de fondo, a el indicado por el parámetro ColorFdo.*

---

1 El sistema de coordenadas físicas es el establecido por omisión. Tiene su origen en el punto (0,0). Los valores de este sistema son siempre positivos. El valor de X aumenta de izquierda a derecha y el valor de Y lo hace de arriba hacia abajo. Los valores X y Y, se expresan en puntos o pixeles.

Determina_Color_Fondo(). *Devuelve como resultado el número del color actual de fondo. Por omisión este valor es 0.*

Fija_Color(Color). *Pone como color del primer plano, el indicado por el parámetro* **Color**. *Por omisión, este valor es el más alto de la paleta con la que estemos trabajando.*

Determina_Color(). *Devuelve como resultado el número del color actual del primer plano. Por omisión, éste es el valor más alto de la paleta con la que estemos trabajando.*

Fija_Tipo_Línea(Másc). *Fija el tipo de líneas a dibujar por otras funciones que se verán más adelante. El tipo de línea es fijado por una máscara de 16 bits. Cada bit representa un punto de línea. Si un bit es 1, se dibuja un punto y si es 0 no se dibuja. En términos más técnicos diríamos, si el bit es 1, el punto correspondiente se pone al color de la línea y si el bit es 0, el punto correspondiente se deja como está. Por omisión, el valor de la máscara es el valor hexadecimal FFFF que producirá una línea continua.*

Ejemplo:		
Máscara:	FFFF	B3C9
Valor en Binario:	111111111111	1011001111001001
Tipo de Línea:	————————	_ __ ____ _ _

Determina_Tipo_Línea(). *Devuelve el número correspondiente al valor actual de la Máscara que se está utilizando para trazar líneas por otras funciones.*

Fija_Área_Máscara(Másc). *Recubre un área, de acuerdo a una máscara formada por un arreglo de 8 x 8 bits, donde cada bit representa un punto. Si el bit es 1, el punto correspondiente se pone al color actual y si el bit es 0, el punto correspondiente se deja como está, no cambia. Por omisión el parámetro* **Másc** *es un arreglo de ceros.*

Determina_Área_Máscara(Másc). *Devuelve el valor actual de la máscara formada por un arreglo de 8 x 8 bits, utilizada para recubrir áreas.*

Rellena(X,Y,Color_Borde). *Rellena un área utilizando el color y máscara actuales. Los argumentos (X, Y), corresponden a las coordenadas de un punto; si el punto está dentro de una figura, se recubre su interior y si está fuera se recubre su exterior. El punto no debe estar sobre el borde -que debe ser una línea continua- de la figura. El parámetro Color_Borde es una expresión numérica que identifica el color que define el borde de la figura. El área a rellenar queda limitada por este borde.*

I.3.2.2.4 Funciones para dibujar y visualizar imágenes

Clasificación de aquellas funciones que permiten dibujar figuras mediante primitivas gráficas como son: dibujar puntos en pantalla, líneas, arco, limpiar parte o el total de pantalla, etc.

Limpia(Área). *Borra el área de pantalla especificada. El parámetro Área puede ser una constante de las siguientes:*

CONSTANTE	ACCIÓN
Pantalla	Limpia toda la pantalla.
Ventana_Gráficos	Limpia la ventana activa para gráficos.
Ventana_Texto	Limpia la ventana activa para texto.

Mueve_A(X,Y). *Mueve la posición actual, al punto determinado por los parámetros (X,Y). No dibuja.*

Dibuja_Línea(X,Y). *Dibuja una línea desde la posición actual, hasta el punto determinado por los parámetros (X,Y).*

Dibuja_Rectángulo(Control, X_1,Y_1,X_2,Y_2). *Dibuja un rectángulo. Los parámetros (X_1,Y_1) y (X_2,Y_2), corresponden a las esquinas superior izquierda e inferior derecha respectivamente. El parámetro* **Control** *es una de las siguientes constantes:*

CONSTANTES	ACCIÓN
Borde	Dibuja el borde del rectángulo.
Relleno	Rellena el rectángulo con el color actual.

Dibuja_Arco(X_1,Y_1,X_2,Y_2,X_3,Y_3,X_4,Y_4). *Dibuja un arco. El centro del arco es el centro del rectángulo definido por los puntos (X_1,Y_1) y (X_2,Y_2). El arco comienza en el punto de intersección con el vector definido por (X_3,Y_3) y finaliza en el punto de intersección con el vector definido por (X_4,Y_4). El arco es dibujado en sentido contrario al de las agujas del reloj.*

Dibuja_Punto(X,Y). *Dibuja un punto en la posición determinada por los parámetros (X,Y) con el color actual.*

Determina_Color_Punto(X,Y). *Da como resultado el número correspondiente al color del punto (X,Y).*

I.3.2.2.5 Funciones para visualizar texto

Clasifica a aquellas funciones para controlar la presentación de texto en pantalla.

Fija_Color_Texto(Const). *Fija color para texto. El parámetro* **Const** *es un valor de 0 a 31. Los valores del 0 al 15 corresponden a los colores normales, y los valores del 16 al 31 corresponden a los mismos colores, pero hacen que el texto parpadee. El color para el texto por omisión es el valor más alto de los colores normales.*

Fija_Posición_Texto(Ren,Col). *Sitúa el cursor en la fila y columna indicada por los parámetros Ren, Col. La posterior salida de texto será colocada a partir de ese punto.*

Muestra_Texto(Texto). *Visualiza el texto indicado por el parámetro* **Texto** *en la posición actual del cursor.*

Determina_Posición_Texto(). *Da como resultado la fila y columna de la posición actual del cursor sobre el texto. El resultado es devuelto en un arreglo.*

Fija_Ventana_Texto(X_1,Y_1,X_2,Y_2). *Especifica la ventana donde va a ser visualizado todo el texto. Los parámetros (X_1,Y_1) corresponden a la fila y columna de la esquina superior izquierda de la ventana, y los parámetros (X_2,Y_2) especifican la fila y columna de la esquina inferior derecha de la ventana.*

I.3.2.2.6 Funciones para transferir imágenes (animación)

Clasificación de aquellas funciones que nos permiten determinar la cantidad de espacio en bytes que ocupa una imagen, almacenar la imagen, recuperarla y manipularla en pantalla.

Almacena_Imagen(X_1,Y_1,X_2,Y_2,Loc). *Almacena en el área de memoria apuntada por el parámetro Loc, la figura de la pantalla encerrada por los puntos definidos con los parámetros (X_1,Y_1) y (X_2,Y_2). El área de memoria debe ser lo suficientemente grande para contener la figura.*

Tamaño_Imagen(X_1,Y_1,X_2,Y_2). *Da como resultado el número de bytes necesario para almacenar la figura definida dentro del rectángulo especificado por la coordenadas (X_1,Y_1) y (X_2,Y_2).*

Coloca_Imagen(X,Y,Loc,Acción). *Transfiere a la pantalla, la figura almacenada en la zona de memoria apuntada por* **Loc**, *colocando la esquina superior izquierda del rectángulo que contiene dicha figura, en el punto (X,Y). El parámetro* **Acción** *es utilizado para superponer o transformar imágenes, con otras imágenes ya en pantalla, y puede ser cualquiera de los siguientes:*

CONSTANTES	SIGNIFICADO
COLOCA	Da lugar a una copia exacta de la imagen almacenada en memoria.
INVERSO	Es igual al parámetro anterior, excepto que produce una imagen en vídeo inverso.
SUSTITUYE	Se usa para transferir una imagen almacenada, sobre una ya existente en pantalla.
ENCIMA	Se usa para sobreponer o combinar una figura almacenada sobre otra ya desplegada en pantalla.

I.4 Funciones para el manejo del ratón

Con el incremento en la popularidad de las aplicaciones gráficas, los dispositivos apuntadores (ratón, lápiz óptico, ratón estacionario, palanca de mando) han tomado cada vez mayor importancia, puesto que permiten un mejor desenvolvimiento dentro de éstas. El dispositivo de entrada de datos asociado a una pantalla gráfica (excepto en los gráficos de texto) suele ser el ratón, como mecanismo de selección principal y la teclas del cursor como dispositivo secundario.

En si, un dispositivo apuntador es un periférico de entrada que permite la interacción en línea con la computadora, cuando la entrada por teclado resulta menos eficiente. Existen dos alternativas al uso de un ratón: primera, leer los códigos de salida del ratón e identificarlos con las teclas del cursor y segunda, incorporar directamente una interface de ratón en el programa e interpretar los sucesos provocados por éste directamente.

En cuanto al primer método, utiliza la interface por omisión del ratón (MOUSE.COM o el controlador MOUSE.SYS) que se suministra con el hardware de éste y no necesita una atención especial por parte del programador. Aquí cualquier evento del ratón proporciona códigos de eventos previamente asignados por el manejador del ratón. Sin embargo, con muchos ratones, estos códigos de respuesta pueden reasignarse o incluso interpretarse como instrucciones de tipo macro; en este caso es difícil asegurar que los valores devueltos por el software del ratón coincidan con las respuestas apropiadas o al menos con las deseadas.

Para la mayoría de las aplicaciones gráficas, la interpretación indirecta de la entrada generada por un ratón (tratando el ratón de manera similar al teclado) es imposible o insatisfactoria por diversos motivos: el desplazamiento del ratón y las respuestas de las teclas puede que no estén asignados correctamente; la información devuelta es menos completa que la leída directamente desde el ratón; el control ejercido sobre los parámetros del ratón es pequeño o nulo; por último, el ciclo temporizador del programa puede llegar a producir demasiado retardo en respuesta al desplazamiento del ratón.

Por estos motivos, la mayoría de las aplicaciones gráficas buscan una interface directa con el dispositivo ratón que permita al programa cargar sus parámetros y leer directamente los sucesos y la posición de éste.

Pero para asumir el control directo y la comunicación con el ratón, la aproximación óptima consiste en crear un objeto ratón (profundizaremos el concepto "objeto" en el capítulo dedicado a tópicos de la orientación a objetos) que proporcione los procedimientos para:

- ✓ Restringir el desplazamiento del ratón a una zona determinada en la pantalla.
- ✓ Leer directamente la posición de pantalla en la que se encuentra el ratón.
- ✓ Cambiar los factores de respuesta del ratón y ajustar el desplazamiento vertical y horizontal de éste a los factores de desplazamiento de la pantalla.
- ✓ Leer las teclas del ratón para generar sucesos y para interrumpirlos, y examinar el estado de cada pulsación individual.
- ✓ Seleccionar los cursores del ratón gráfico o crear cursores nuevos.
- ✓ Ocultar o mostrar de forma selectiva el cursor del ratón.

Estas propiedades se muestran en el objeto Ratón creado para la aplicación.

I.5 Impresión gráfica

En la práctica de la impresión gráfica, el desplegado de éstas en el video de una computadora tiene el problema de que, si se desea su imagen en papel, la información deberá reprocesarse para lograrlo. Existen en el mercado dispositivos especializados para la obtención de gráficas (plotters de mesa, de tambor, de rodillo y electrostático), sin embargo, éstos por lo regular resultan costosos y de difícil mantenimiento, así se ha desarrollado la capacidad gráfica en los dispositivos más económicos y comunes que son las impresoras; en este caso se dispone de una cabeza de impresión con arreglo de agujas, las cuales son impulsadas electromagnéticamente sobre la cinta entintada y el papel, imprimiendo puntos sobre éste.

La densidad de impresión, define la calidad con la que se lograrán las imágenes sobre el papel, este parámetro depende directamente de la cantidad de agujas en la cabeza de impresión, que forman un arreglo matricial de 9 ó de 24. Este tipo de dispositivos pueden imprimir gráficos en dos formas: la primera, por GRÁFICOS DE BLOQUES Y LÍNEA -usa el conjunto de caracteres semi-gráficos definidos en código ASCII, para formar recuadros y líneas rectas-, el segundo por GRÁFICOS DE PUNTO -permite por medio de programación, especificar exactamente donde será impreso cada punto-.

En una cabeza de nueve agujas, estas están alineadas formando un arreglo vertical. La sincronización entre el movimiento horizontal de la cabeza y vertical del rodillo, da por resultado la producción de caracteres, y en general, puntos de tinta donde se requieren a lo largo de la hoja.

Existen dos estándares básicos para las impresoras de matriz de puntos: el EPSON y el IBM.

En el caso de las gráficas por bloque, el estándar EPSON maneja el conjunto de códigos ASCII del 169 al 223, como caracteres gráficos de bloque, con una altura de 7 puntos de alto, teniendo que ser impresos con un avance de línea de 7/72 de pulgada.

Para la IBM, los caracteres de bloque son de 12 puntos y comprenden del código ASCII 169 al 223, y no requieren de cambio alguno en el avance.

Para la impresión de gráficas por puntos, el manejo en ambos estándares es muy similar. Aquí, la longitud de la línea y el espaciamiento entre éstas no es fijo como en el caso de los caracteres predefinidos. Para usar las gráficas de puntos deben indicarse tres parámetros al controlador del dispositivo.

I.5.1 Qué tan cerca se espaciará cada columna de puntos (densidad de gráficos)

Para esto se requiere de los comandos gráficos de impresión; existen diferentes comandos que proporcionan distintas densidades gráficas. La densidad determina cuán cerca serán impresos horizontalmente los puntos que conforman un gráfico. Existen, básicamente, cuatro densidades: Sencilla, Doble, Doble Alta velocidad, y Cuádruple densidad.

I.5.2 Cantidad de columnas por línea

Si consideramos que una columna de impresión es de un punto de ancho, a 60 puntos por pulgada (ppp), por ejemplo, se podrán imprimir 480 columnas en una línea de ocho pulgadas de largo y para el caso de 240 ppp se imprimirán 1920 columnas. Debido a que se requiere transmitir al dispositivo la cantidad de columnas gráficas a imprimir y de que el número máximo representado por los ocho bits de comunicación es 255, se ha dispuesto de dos parámetros (n_1 y n_2) para aquellos casos en que se manejan más de 255 columnas gráficas de impresión, siendo el primer número (n_1) la cantidad de columnas de 0 a 255, y el segundo (n_2) el número de veces en que se añade un conjunto de 256 líneas al primer grupo.

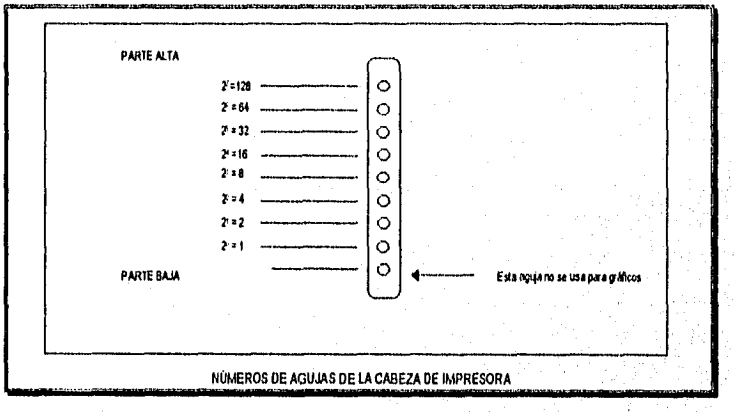
I.5.3 Cuantas agujas activar en cada columna vertical impresa

En su modo gráfico principal, la EPSON imprime una columna de hasta 8 puntos por cada código recibido, usando las ocho agujas superiores de la cabeza de impresión (el puerto paralelo sólo envía 8 bits de información a la vez). Así, el programa deberá enviar a la impresora códigos (uno por cada

columna) de modelos de puntos, donde indique cuáles de las ocho agujas disponibles serán usadas. Para imprimir columnas de más de ocho puntos, la cabeza de impresión debe realizar más de un recorrido, imprimiendo una línea (de 8 puntos de ancho) en cada uno, y avanzando después el papel para repetir esta operación cuantas veces sea necesario.

Para evitar que la cabeza de impresión deje espacios entre líneas gráficas, como lo hace con el texto, debe ser modificado el parámetro de espaciamiento entre líneas. Se pueden imprimir gráficos de líneas horizontales adyacentes separadas no más de 8/72 de pulgada.

Si tenemos ocho agujas, tendremos también $2^8 = 256$ posibles combinaciones entre ellas. Cada aguja se identificará, para su selección, con un número determinado por la posición de ésta, como se muestra en la Fig. siguiente.



Así, de acuerdo a esto para activar la aguja superior se envía el numero 128, para la inferior el 1, y para ambas el 129 que es la suma de los dos códigos. Cada carácter se envía individualmente al puerto de impresión, comenzando desde el de la columna de la izquierda.

I.6 ¿Cómo preparar al sistema para modo gráfico?

Como último tema del presente capítulo mencionaremos las directrices con que debe contar un lenguaje que maneje el modo gráfico, sin entrar en particularidades de algún lenguaje específico.

Los sistemas de cómputo, capaces de producir gráficos, están diseñados para trabajar tanto en modo de texto como en modo gráfico, pero no en ambos a la vez. Cuando los sistemas de cómputo

están en modo texto, escriben caracteres en la pantalla, pero no trazan líneas; cuando están en modo gráfico, dibujan líneas, pero no escriben caracteres (a menos que se les programe especialmente para hacerlo). Los caracteres que pueden escribirse estando en modo gráfico, son muchas veces de forma o tamaño diferente de los que se escriben en modo texto.

Las proposiciones que se requieren para poner a un sistema en modo gráfico, varían de un sistema a otro, pero generalmente deben hacer lo siguiente:

1) **Detectar el tipo de tarjeta gráfica;**
2) **establecer el sistema en modo de gráficos;**
3) **especificar el lenguaje gráfico que se va a emplear;**
4) **describir el tamaño del área de trabajo, es decir, la resolución de la pantalla;**
5) **definir los colores de fondo (pantalla) y trazo, si el sistema puede trabajar con colores o con tonos de gris;**
6) **y en algunos sistemas, definir la parte de la pantalla que se va a usar (puede considerarse la pantalla completa como opción por omisión).**

Algunas de estas tareas, pueden llevarse a cabo en forma automática en algunos sistemas. Para dibujos en blanco y negro, lo más recomendable es elegir el color negro (BLACK) o ningún color (NONE), para el fondo, y blanco (WHITE) para la forma.

Al principio, estas proposiciones pueden parecer abrumadoras, pero casi siempre son las mismas para cualquier programa de gráficos en un sistema dado, excepto que las listas de parámetros pueden ser diferentes, pueden copiarse de un programa a otro.

CAPÍTULO II

TÓPICOS DE ORIENTACIÓN A OBJETOS

CAPÍTULO II

TÓPICOS DE ORIENTACIÓN A OBJETOS

II.1 Contexto histórico y enfoque paradigmático

Desde sus inicios en 1950 y hasta los 70´s, la tecnología de la información[1] consistía en el proceso de datos, que veía a los sistemas como entidades compuestas de dos partes: información a procesar (datos) y una secuencia de pasos para lograr un fin (proceso), para dar solución a la búsqueda de reducción de costos y tiempo que las empresas solicitaban. Rápidamente los sistemas llegaron a formar parte importante de la planeación estratégica básica de las empresas que adoptaron esa tecnología y una meta dentro de las que no.

El esfuerzo dedicado a poner la tecnología al servicio de los procesos productivos dio por fruto el nacimiento de los lenguajes de alto nivel, y con ellos las técnicas de programación. Comenzaron a desarrollarse los profesionales en el área y fue apareciendo una extensa bibliografía para explicar y facilitar el acceso al innovador uso de las computadoras y sus posibilidades aplicativas.

Conforme se fueron automatizando, las empresas trasladaron el poder de cómputo empleado en sus procesos internos, a la impartición de servicios y productos para sus clientes, en esta traslación, los sistemas cambiaron la base sobre la que eran diseñados, puesto que el usuario final ya no era un especialista, sino una persona que exigía del sistema una facilidad de operación que, en términos del proceso de datos se antojaba impensable.

Durante la década de los 70´s, -bajo la ahora llamada primera era de la informática- los sistemas que se mantenían y desarrollaban en cada instalación fueron creciendo en volumen; cada vez que un nuevo proceso se incluía en el universo local de cómputo, se realizaban y modificaban los parámetros de interacción entre los distintos sistemas ya existentes. La programación se tornó tan compleja, que no era tan fácil hacer software a la medida, pues cada pieza era diseñada y producida para satisfacer las necesidades específicas del usuario y de los propios sistemas; un software hecho por especialistas que intentaban utilizar al máximo las herramientas de que disponían para cumplir con el tiempo requerido, dejando de lado toda facilidad posible con tal de no incrementar las largas listas de instrucciones, escritas línea por línea para interpretar los requerimientos básicos. Para entonces la ingeniería del

[1] La tecnología de la información es el conjunto de procesos y técnicas que, conjuntamente con la tecnología electrónica especializada han permitido la incorporación creciente del manejo electrónico de datos a los procesos productivos, incrementando la eficiencia, velocidad y productividad. En la práctica, la tecnología de la información se traduce en el análisis de requerimientos, selección del equipo adecuado y todo el proceso de desarrollo de sistemas que traen por resultado una herramienta software-hardware altamente especializada y que el usuario final del sistema podría usar.

software[2], comenzaba a imponer orden en la planeación y control del proceso de desarrollo de los sistemas.

En la década de los 80's, los sistemas se fueron comprometiendo en procesos cada vez mayores y, aún con la ingeniería aplicada, la vasta mayoría de los programas seguían teniendo características de arte; complejos y a menudo grandes, su desarrollo exigía mucho tiempo, codificándose de acuerdo a un conjunto específico de requerimientos y partiendo de cero para su construcción; haciendo uso de técnicas que ni son mensurables ni pueden repetirse consistentemente. El autor de cada programa se convertía así en su único intérprete y para el mantenimiento ulterior, podía resultar mejor opción un nuevo desarrollo a desenmarañar las líneas de un estilo de programación ajeno. Es un aprovechamiento preindustrial, haciendo una analogía a la revolución industrial del siglo XVIII.

En la práctica, el proceso de desarrollo de software cada vez requería de más tiempo, recursos de cómputo y dinero, la planeación y organización no eran ya suficientes para asegurar la creación de software efectivo dentro del tiempo y presupuesto adecuados. Los sistemas se hacinaban, crecían, se volvían obsoletos, y se remendaban dentro de los sistemas de almacenamiento. Las herramientas CASE (Ingeniería de Software Auxiliada por Computadora), pensadas para minimizar esa tendencia no encontraban eco en los nuevos desarrollos, pues su aprovechamiento requería un nuevo software capaz de asimilarlas.

Para finales de la década, las amistosas interfaces gráficas de usuario (GUI -Graphical User Interface-, por sus siglas en inglés) vinieron a poner un nuevo requerimiento al software; iniciando una tendencia que ha popularizado el uso de la computadora en sectores de la sociedad que se oponían al manejo de CUI (Interface de usuario basada en caracteres).

Una interface de usuario es un ambiente que pone a disposición del operador un medio para comunicarse con la computadora; desde sus inicios, la programación y operación de los equipos de cómputo han contado con un conjunto de instrucciones, operadores y comandos codificados, que interactúan con un hardware diseñado para poder "leer" los códigos correspondientes a los botones que conforman un teclado; por otra parte la capacidad de vídeo de los equipos extendió el concepto de las CUI's en términos de menús, ventanas de texto, barras de mensaje y líneas de comando. Los ambientes CUI's dominaban la escena, haciendo menester aprender conjuntos de instrucciones, comandos especiales y técnicas de manejo de la información especializadas, hasta el momento en que fue técnicamente posible el empleo de elementos gráficos en los monitores de vídeo.

Fue así como las instrucciones pudieron traducirse en iconos -pequeños dibujos representando la acción buscada-, llenando la pantalla de figuras familiares que podían ser manejadas con un dispositivo

[2] La aplicación de un aprovechamiento sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software.

apuntador que, análogo al dedo índice, permitía señalar acciones, en lugar de deletrearlas usando el teclado. Con la nueva facilidad, las personas se acercaron con menos temor, y hasta con entusiasmo a las computadoras; la demanda de las GUI's creció rápidamente, empujando a los programadores de vanguardia un poco más hacia el enfoque de una mayor facilidad de uso, lo que para un Sistema Orientado al Proceso (SOP), significó mayor complejidad en los algoritmos, herramientas de programación e interacción con el hardware (Fig. II.1).

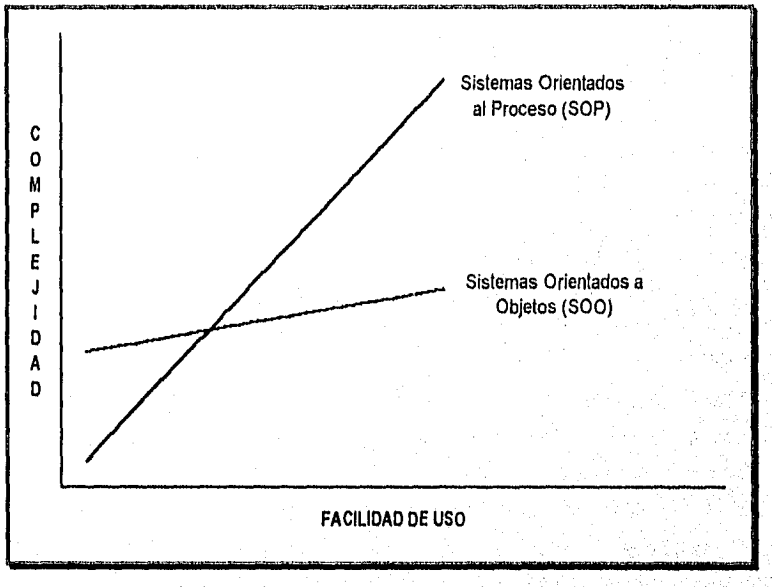


Fig. II.1.- Gráfica facilidad de uso/complejidad del software.

Adicionalmente a la problemática de orientar los nuevos desarrollos al usuario final, se presentó el problema de actualizar los sistemas ya existentes, los que al estar ubicados en un solo centro de cómputo, centralizaban también su actualización, cosa que facilita por una parte el mantenimiento pero que generaliza problemas que de otra forma serían vistos como locales. De ese modo la creciente complejidad de los sistemas comenzó a provocar costosos retardos en la entrega del producto final, la cancelación de proyectos o hasta su rediseño completo, siendo precisamente este último punto el que obligó a la comunidad informática a identificar los elementos que aparecían como aristas comunes a todos los desarrollos de sistemas basados en un punto de vista procedimental (SOP) y que influían grandemente en la decisión de un rediseño:

1.- Existe una limitada reusabilidad y portabilidad del software.

2.- El mantenimiento es engorroso y caro.

3.- Los sistemas forman islas de tecnología.

4.- La curva del aprendizaje del sistema es asintótica.

5.- Existe una dependencia con la plataforma de desarrollo.

6.- Sólo los especialistas pueden crear el software.

7.- El software se enfoca al proceso.

Este conjunto de características conforman la problemática producida por una forma de reflejar la realidad dentro de los sistemas, un esquema que permite comprender el mundo que nos rodea, un marco de referencia o un modelo, es decir, un **paradigma**. Para el caso que nos ocupa, ésta "Crisis del Proceso de Datos" conformó el ambiente que dio origen a nuevos enfoques alternativos (ver Tabla "Identificación de Paradigmas"), que de acuerdo a las nuevas características de los problemas, replantearon las necesidades y su solución, constituyéndose en nuevos estándares que más tarde se reconocerían como nuevos paradigmas, entre los que se encuentra la Orientación a Objetos (OO) que, como muestra la Fig. II.1 tienden a disminuir la componente de la complejidad propia de los sistemas altamente especializados o mayormente orientados al usuario.

IDENTIFICACIÓN DE PARADIGMAS EN LA CRISIS DEL PROCESO DE SAICS		
ASPECTO	PARADIGMA ANTERIOR	NUEVO PARADIGMA
Medios de desarrollo de software	El tiempo, tamaño y costo de desarrollo de software está en función de las habilidades y la creatividad de personal que lo desarrolla	Los desarrolladores usan y reutilizan módulos o partes de módulos previamente estandarizados y que trabajan bien. * Paradigma de la orientación a objetos
Diseño y construcción de interfaces de usuario	La CUI (Character User Interface) emplea las instrucciones disponibles en forma de tablas de codigos, para utilizar y administrar el sistema	En la GUI (Graphics User Interface) el usuario manipula simultáneamente imágenes y objetos directamente relacionados con acciones. * Paradigma de la interfaz gráfica de usuario.
Diseño de aplicaciones Comerciales	Se diseña la aplicación particular con sus propios estándares y facilidades. Se generan sistemas aislados	Las aplicaciones comparten facilidades y estandarizan sus características funcionales. * Paradigma de la Manufactura Integrada del Software.
Redes y sistemas distribuidos	El proceso productivo se centraliza junto con la administración de la empresa en una computadora central o Host.	El proceso se distribuye a lo largo de las distintas entidades informáticas de la empresa. Arquitectura Cliente-Servidor * Paradigma del proceso distribuido.
Multimedia	Cada presentación de la información se trata por separado con su propia tecnología.	La digitalización generalizada y los estándares acordados permiten el manejo simultáneo dentro de las aplicaciones. * Paradigma de la integración de medios

IDENTIFICACIÓN DE PARADIGMAS EN LA CRISIS DEL PROCESO DE DATOS		
ASPECTO	PARADIGMA ANTERIOR	NUEVO PARADIGMA
Método del desarrollo de software	La calidad, tiempo y costo de desarrollo del software está en función de las habilidades y la creatividad del profesional que lo desarrolla.	Los desarrolladores usan y reutilizan módulos o partes de módulos previamente estandarizados y que trabajan juntas. * Paradigma de la orientación a objetos
Diseño y construcción de Interface de usuario	La CUI (Character User Interface) encripta las instrucciones disponibles en forma de tablas de códigos, para utilizar y administrar el sistema.	En la GUI (Graphical User Interface) el usuario manipula intuitivamente imágenes o iconos directamente relacionados con acciones. * Paradigma de la interface gráfica de usuario.
Diseño de aplicaciones Comerciales	Se diseña la aplicación particular con sus propios estándares y facilidades. Se generan sistemas aislados.	Las aplicaciones comparten facilidades y estandarizan sus características funcionales. * Paradigma de la Manufactura Integrada del Software.
Redes y sistemas distribuidos	El proceso productivo se centraliza junto con la administración de la empresa en una computadora central o Host.	El proceso se distribuye a lo largo de las distintas entidades informáticas de la empresa. Arquitectura Cliente-Servidor. * Paradigma del proceso distribuido.
Multimedia	Cada presentación de la información se trata por separado con su propia tecnología.	La digitalización generalizada y los estándares acordados permiten el manejo simultáneo dentro de las aplicaciones. * Paradigma de la inlegración de medios.

II.2 El paradigma de la orientación a objetos

En un sistema informático, los elementos del universo de un problema intentan ser reflejados o "modelados" como entidades internas mediante técnicas que, dentro de los métodos de desarrollo del software, se presentan en tendencias claramente distinguibles entre sí y que reciben su nombre de acuerdo a la forma de conceptualizar al objeto de estudio. Históricamente el método llamado Procedural o Procedimental, que se enfoca a los procedimientos, ha dominado por sobre el resto, ya que sigue los preceptos tradicionales para la resolución de problemas de la ingeniería, suponiendo a un proceso formado de varios subprocesos que al analizarse facilitan el conocimiento del problema y permiten visualizar la solución como una interconexión de ellos (orientación al proceso). Para este enfoque, el "modelo" y su solución son integrados finalmente, después del proceso de desarrollo específico, en un conjunto de archivos que contendrán la información y programas o procedimientos encargados de manipularla.

La actual inclinación de la tecnología por el paradigma de la OO puede interpretarse como una preferencia a observar los problemas de una forma en la que se facilite la reutilización de elementos previamente desarrollados, derivando su poder de modelado de la funcionalidad y manipulación subyacentes, en el concepto abstracto de objeto.

Un SOO, en oposición a la descomposición funcional -base de la orientación al proceso- modela al sistema como una colección de "prototipos" de los objetos que componen la realidad. El análisis y diseño se efectúa sobre los prototipos, vistos como objetos o unidades atómicas encapsuladas que contienen a la vez datos y su proceso, y sobre la manera en que interactúan entre sí mediante "mensajes", en la forma en que son agrupados en colecciones, en la forma en que éstas serán manipuladas y en su jerarquía estructural. Su diseño detallado evoluciona y se difiere hasta después, en el proceso de desarrollo.

El paradigma de la OO observa, como el paradigma procedural, técnicas para el análisis, el diseño y la programación y, como aquel, recomienda su aplicación de manera uniforme a lo largo del proceso de desarrollo de un sistema para lograr mayor efectividad. En la práctica, la OO, no garantiza la realización del mejor software, si no que ofrece una herramienta para tener mayor control en su producción y mantenimiento acelerados.

Los beneficios de la utilización de este paradigma se reconocen mayormente al aplicar su técnica a sistemas masivos, adaptándose rápidamente a los cambios significativos en la aplicación, lo que empleando el paradigma procedural, necesitaría más rigor en el control del desarrollo de módulos, documentación, estandarización, mantenimiento y manejo de errores. La siguiente es una tabla comparativa de los paradigmas identificados por Henderson-Seller, B:

TIPO DE PARADIGMA	CARACTERÍSTICAS
Procedural	El lenguaje es esencialmente imperativo. Se dispone de estructuras de control como la secuencia, la iteración y la selección.
	El diseño basado en este paradigma y en la descomposición funcional es la base del método de desarrollo y su aprovechamiento.
Lógico	Se enfoca a las reglas y relaciones de inferencia implícitas en el cálculo proposicional. Se caracteriza por su manipulación de listas. La recursión usualmente es utilizada y resulta evidente en su aplicación. También se le conoce como programación declarativa (Prolog).
Funcional	Existe un conjunto de primitivas predefinidas por el lenguaje. Se reconocen un conjunto de formas funcionales disponibles. (Miranda y Lisp).
Orientación a Objetos	El sistema se descompone en entidades (objetos) jerarquizadas y clasificadas que corresponden a elementos específicos. El sistema se enfoca a los elementos del negocio y a su interacción, restándole importancia a los procedimientos y enfatizando el encapsulamiento.

II.3 Historia de los sistemas orientados a objetos

La historia de los SOO incluye las estructuras OO que han ido apareciendo en la ciencia, así como el pensamiento OO relacionado a la computación. Inicialmente la ciencia buscó identificar los distintos elementos que conviven en la resolución de un problema, obteniendo dos elementos; el primero es el proceso de búsqueda de la solución, que puede ser inductivo (de lo particular a lo general), deductivo (de lo general a lo particular) o eductivo (derivando nuevos elementos de las características previas o potenciales). Este proceso requiere en primer lugar de los requerimientos del problema (enunciado inicial), luego un procedimiento aplicado a la búsqueda de relaciones, experiencias y conexiones para

llegar finalmente a una especificación (enunciado solución), que es el segundo elemento y puede ser representado por su "función", codificando sus características (basado en el proceso o función) o por su "forma", clasificando sus elementos (basado en clases).

La especificación de una solución por "forma", se reduce a reconocer las distintas entidades que componen el universo del problema, llamadas objetos, junto con sus características funcionales e informativas, para luego someterlas a una clasificación, todos los elementos que componen la clasificación conforman el universo donde nuestra solución tomará forma a partir de la interacción de todas y cada una de ellas.

La unión de objetos y software inició a principios de la década de los 60's, comenzó a experimentarse en "inteligencia artificial" introduciendo la clasificación y la metáfora[3] de objeto a los sistemas de representación del conocimiento.

En 1967, Simula, un lenguaje desarrollado por Ole'Johan Dahl y Kristen Nygaard en Noruega, incluyó estos conceptos; Simula nunca se hizo popular entre la comunidad de sistemas, lo que hizo que, aunque era un lenguaje de propósito general, su uso se suscribiera solamente al modelado de problemas y simulación, y sus características de objetos eran conocidas sólo por unos cuantos investigadores.

La idea de objetos como elemento de construcción de software de propósito general se atribuye a Alan Kay, un graduado de la Universidad de Utah a finales de los 60's. Kay concibió la idea de verdadera computadora personal, a la que llamó Dynabook. La Dynabook seria una computadora del tamaño de un cuaderno que podría ser utilizada de distintas formas, esta idea por sí misma era una demostración de que la metáfora de objeto podía extenderse más allá de las estructuras de programación.

En 1970, un grupo de la investigadores, también de la Universidad de Utah en EUA, Dan Ingalls y Adele Goldberg, unieron su trabajo al de Alan Kay en el Palo Alto Research Center (PARC) de Xerox, formulando los razonamientos básicos para el desarrollo de SmallTalk y su ambiente, permaneciendo como herramienta de investigación hasta mediados de los 80's cuando las subsidiaria de Xerox, ParcPlace Systems convirtió a SmallTalk en un producto comercializable donde los objetos gráficos podían ser manipulados por un dispositivo apuntador o Ratón.

Un equipo de diseño de Apple Computer visitó PARC y sus instalaciones a inicios de los 80's y salieron con nuevos conceptos radicales de diseño para sistemas operativos y métodos de interacción con el usuario. Apple inicialmente contrató a Dan Ingalls, y sus ideas que formaban la investigación en

[3] Metáfora: Recurso literario mediante el cual se identifican, sin comparación expresa, dos objetos que guardan entre sí una relación o semejanza. En este contexto, hace referencia al modelo obtenido mediante la aplicación del enfoque de la OO.

Xerox, se convirtieron en la base del sistema operativo de la Apple Macintosh, que popularizó inicialmente la interface gráfica de usuario.

El creciente interés en el lenguaje C, a inicios de los 80's, llevó a programadores de lenguajes, como Bjarne Stroustrup de los laboratorios Bell AT&T, a comenzar a trabajar en una versión de C orientado a objetos, conocida como C++, o como a Brad Cox de StepStone Corp. y Objective C, al Departamento de Defensa de EUA con ADA orientado a objetos o, en otros casos desarrollar un nuevo ambiente basado totalmente en el concepto, como Bertrand Meyer con Eiffel, C. Shaffert con Trellis/Owl y Ungar-Smith con Self.

Se comenzaron a notar diferencias de fondo entre los lenguajes especializados, pues se denominó "BASADOS EN OBJETOS" a aquellos que solamente los utilizan, sin posibilidades de modificación o de creación, mientras que se llamó "ORIENTADOS A OBJETOS" a los que además de utilizarlos permiten la definición de las clases y de su herencia.

Desde finales de los 80's, los conceptos afines a la OO, fueron integrándose al diseño de aplicaciones como en el Diseño Asistido por Computadora (CAD), dedicado al manejo de texto y gráficos. Mientras tanto la potencia de hardware en las computadoras personales se elevó hasta un nivel en el que soportaba cabalmente la operación bajo los nuevos requerimientos gráficos de aplicaciones como aquellas.

Las herramientas OO para desarrollar rápidamente sistemas de manipulación gráfica comenzaron a surgir a inicios de los 90's, así como la formalización de las distintas metodologías de OO (Booch-1990, Wirfs-Brock-1990, Coad y Yourdon-1990). Los sistemas operativos modernos han tomado las ideas de la OO, y actualmente (1996), se han desarrollado nuevos productos que en su mayoría presentan una interface gráfica de usuario con una fuerte relación entre objetos gráficos de pantalla y construcciones internas del sistema operativo, simplificando su uso, organizando su complejidad y facilitando su extensión en forma de nuevos objetos. Microsoft Windows NT, IBM OS/2 v2.1, PenPoint y NextStep son algunos ejemplos de ello.

Por otra parte, los sistemas de objetos distribuidos están conformando también las características del futuro cercano del proceso distribuido. Soportado bajo los modernos ambientes de red de área local (LAN) y de área amplia (WAN), la computación distribuida complementa los sistemas centralizados, llevando a cabo procesos específicos bajo la arquitectura Cliente-Servidor. La transición hacia los SOO consiste en utilizar los objetos como una herramienta para manejar la complejidad de los sistemas. Los objetos con su combinación natural de código y datos, y su estricta separación de interface, constituyen un útil instrumento para distribuir datos entre usuarios finales.

Empresas como IBM, con su System Object Model (SOM/DSOM) y el Component Object Model (COM) de Microsoft intentan resolver los problemas de un alto acoplamiento binario entre una aplicación

y sus objetos subyacentes. Por su parte el Object Management Group, fundado en 1989 por 11 compañías entre las que se incluyen Digital, Hewlett Packard, Hyperdesk, NCR y SunSoft, adoptaron el estándar desarrollado por ObjectDesign, el Common Object Request Broker Architecture (CORBA) - liberado inicialmente en 1991 y actualmente desde 1992 en su versión 1.1- para regular la interoperabilidad entre objetos y aplicaciones siguiendo la arquitectura llamada Object Management Architecture (OMA).

La utilización de la Tecnología de Orientación a Objetos (TOO.- Conjunto de técnicas, métodos y herramientas OO) ha ido creciendo a ritmo acelerado, impulsada en gran medida por la habilidad de la industria del hardware de producir CPU's más pequeños, más poderosos y más baratos que soporten la sobrecarga inherente a su manejo y, sobre todo el de los ambientes integrales que, sin duda irán emergiendo para beneficio de un mejor y más eficiente manejo de la información.

II.4 Elementos de los sistemas orientados a objetos

El principal objetivo para cualquier sistema de software, será siempre el presentar un modelo de la realidad lo más cercano posible a nuestra percepción. Bajo este contexto resulta interesante el hecho de que el software sea un elemento intangible que se presenta como una poderosa herramienta para el modelado de la realidad. Para nosotros el universo está compuesto de "cosas" que nos rodean y que se manifiestan en cuanto se ponen al alcance de nuestros sentidos, a éstos "objetos", nuestra experiencia indica que se llaman de algún modo, que hacen actividades que los identifican y que tienen propiedades que los caracterizan.

Nuestra percepción entonces se inclina por la clasificación sistemática de todo, identificando la existencia de caracteres comunes entre perros, autos, edificios, personas, pájaros, piedras, etc. La clasificación de los elementos u objetos de un problema es una herramienta que utilizamos cotidianamente; agrupando a los objetos de acuerdo a una o más de sus características, reducimos la definición del problema a la interpretación de las necesidades de una clase de objeto en particular. El reflejar este esquema de pensamiento en una herramienta abstracta como lo es el software nos permite interactuar más "naturalmente" con los problemas, pues sus elementos se convierten en algo familiar y que por experiencia podemos conocer.

El potencial de la OO reside en la capacidad de modelado derivada de la funcionalidad y sinergia[4] de tres conceptos: encapsulamiento y/u ocultamiento de información (cuya unidad atómica es el objeto), abstracción por clasificación (que administra colecciones de objetos), y polimorfismo (logrado a través de herencia), que estructura colecciones de clases (ver Fig. II.2).

---

[4] Sinergia: Producto notable resultado de la acción conjunta de dos o más entidades que por sí solas producen a su vez un efecto distinto.

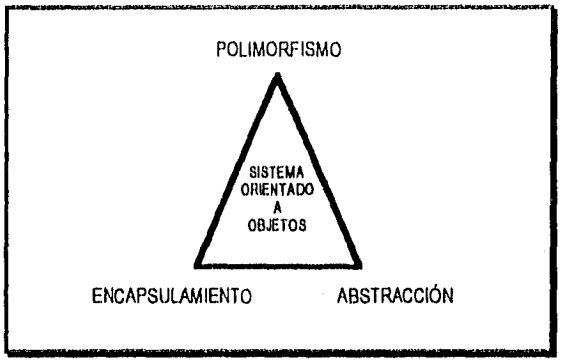


Fig. II.2.- Triángulo de la Orientación a Objetos.

Las actuales tendencias de los sistemas hacia los tipos de datos complejos (manejo de imágenes y datos) y en sistemas integrados donde se comunican e interrelacionan los sistemas, ha favorecido al modelo de objetos sobre los métodos convencionales de análisis y diseño.

II.4.1 Objeto, interface y *métodos*

Los SOO aprovechan nuestro conocimiento de la realidad para llevar a cabo su tarea de **modelado**, observan al problema como un conjunto de entidades que cumplen una función y que están subordinadas a su contexto, comunicándose entre sí. Un **objeto** es la analogía del software a un objeto del mundo real. Es una entidad **autocontenida**, es decir que contiene tanto código como los datos sobre los que opera. Cada objeto tiene la habilidad de recibir mensajes de otros objetos, almacenar la información de su estado actual y de efectuar un número limitado de operaciones basadas en los datos.

En el dominio del software un **objeto** es un elemento de información *autónomo que contiene* una estructura de datos **privada** y procesos llamados **operaciones** o **métodos**, que son los únicos que pueden transformar los valores de sus variables; las operaciones contienen las construcciones procedimentales que pueden ser llamadas por un **mensaje** (una petición al objeto para que realice una de sus operaciones), o ser utilizadas por el objeto mismo para realizar sus funciones (**métodos locales**). El **objeto** es también una unidad que presta sus **servicios** a todo objeto que los solicite, para ello cuenta, como ya se dijo, con **métodos** claramente especificados que determinan su comportamiento. Los **métodos** dedicados exclusivamente para uso propio del objeto son denominados como métodos pertenecientes a su **parte privada**, mientras que aquellos que se ponen a disposición de otros objetos se denominan como pertenecientes a la **parte pública** o como constituyentes de la **interface** del **objeto**.

La **interface** se convierte entonces en el único medio por el que la información del objeto podrá tener contacto con el exterior o con la interface de otros objetos y, ya que se presenta como una lista de métodos necesarios para establecer una comunicación interobjetos, dándole una fuerte protección a la parte privada, se le denomina también como el **protocolo de comunicación** del objeto.

Los objetos se relacionan entre sí al enviar y recibir "**mensajes**" a través de su **interface**. Un **mensaje** podría definirse como un requerimiento para efectuar una operación. Un **objeto** responde a un mensaje al **elegir** el método apropiado, ejecutar su procedimiento y regresar el control a su "cliente". Sin embargo, dentro del contexto de los SOO, un mensaje puede ser:

- Informativo.- presenta el estado local resultante de una operación.
- Interrogativo.- Solicitando información sobre el estado actual.
- Imperativo.- Requiriendo la ejecución de una operación.

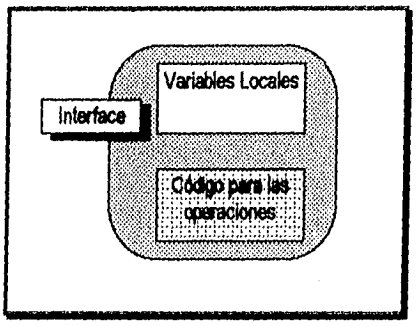
II.4.2 Encapsulamiento y ocultamiento de información

Sí como se ha dicho, un objeto es una unidad atómica consistente en datos y métodos, ningún objeto podría considerarse definido sin hallarse descritos el qué es, y el cómo es. De ésta forma un objeto empaca la información acerca de lo que es (estructura) y de cómo es (funcionalidad). A lo que se denomina **encapsulamiento de la información.**

Los elementos internos de un objeto mantienen un **estado local**, compuesto por variables de uso permitido solamente a él mismo. Las operaciones de un objeto pueden ser **locales** (que sólo pueden ser llamadas por otra operación local o por una de interface de él mismo) y **no locales o de interface** (sólo pueden ser llamadas por medio de un mensaje originado por un objeto externo). Ambas, comparten el estado local del objeto, de forma que los cambios de estado realizados por una operación pueden ser apreciados por otra; sin embargo, cuando un objeto externo trate de accesar a la información del estado actual, sólo podrá hacerlo mediante los procesos no locales (o de **interface**), que proporcionarán la información solicitada o ejecutarán el procedimiento requerido; la **interface** es una lista que muestra los servicios que un objeto puede proporcionar a otro. De ésta forma, para el sistema, un objeto es una caja negra a la que sólo se tiene acceso por su **interface**, y tanto el código como las variables internas son invisibles. A esta propiedad se le denomina **ocultamiento de información.**

El **encapsulamiento** reúne los elementos de un objeto dentro de él mismo (autocontención) y no a lo largo del sistema, como en los SOP; lo que permite ver al objeto como un sistema cerrado (Fig. II.3) que se comunica con los demás por su interface-protocolo. Podemos asumir que el objeto "sabe" como hacer sus funciones, solamente indicándole qué método, de entre los que nos presenta, queremos que haga. Esto no significa que al definir el objeto no hace falta indicarle cómo hacer su trabajo, sino que

necesita hacerse solamente una vez, evitando actualizaciones constantes y personalización de los métodos.



Al encapsular, permitimos que el objeto sea un elemento funcional dentro de un esquema Cliente-Servidor, ocultando los datos locales a la vista del Cliente y respondiendo sólo a los requerimientos de su protocolo. También permite que el objeto sea un elemento altamente modularizado, que separa efectivamente al usuario del elemento de software de su autor.

Fig. II.3.- Encapsulamiento.

Es importante mencionar que **el encapsulamiento no garantiza ocultamiento de información**, de tal suerte que algunas técnicas de programación recurren al encapsulamiento en la forma de modularización de procesos sin evitar accesos no planeados al conjunto de datos encapsulados. En la OO, esto no sucede, pues el objeto aparece como una entidad monolítica y atómica que maximiza el encapsulado.

A diferencia de las entidades de un SOP -que presentan la autonomía al nivel de los procedimientos y mantiene su interacción a través de variables no locales- las de un SOO sacrifican la autonomía de sus procesos y su reusabilidad para llevarlos a un nivel superior de organización, donde cada objeto puede ser visto como una caja negra y ser usado dondequiera que sus propiedades sean útiles. Sin tener que redefinir nuevos objetos para cada sistema, podemos usar nuevamente un objeto predefinido y a lo sumo, definir nuevos métodos; estaremos entonces reusando nuestro objeto.

La **reusabilidad** es la habilidad que tiene un elemento del sistema para ser aprovechado en la construcción de nuevos sistemas, entendiéndose como elemento del sistema a cualquier definición de objeto, jerarquía de clase, diseño o análisis propios de un problema. La reusabilidad es consecuencia directa del encapsulamiento y ocultamiento de información, es la manifestación de la autonomía de un elemento, que incide directamente en el ciclo de desarrollo del sistema, es el objetivo principal del análisis, el elemento del diseño y la herramienta en la programación.

II.4.3 Clasificación, clase y abstracción

Si agrupamos a los objetos de acuerdo a alguna característica que nos sea de interés y luego particularizamos las características que los diferencian entre sí, obtendremos una jerarquía progresiva de clasificación en la cual podremos encontrar un lugar donde nuestro objeto encaje, mostrándonos el panorama de su situación dentro de la clasificación. A este conjunto de características que definen exactamente a un objeto, diferenciándolo del resto se le denomina **clase**.

La clasificación siempre ha sido una herramienta útil en la resolución de problemas, pues nos permite observar sus elementos objetivamente, yendo desde una descripción en común que se aplica a muchas clases especializadas, hasta una descripción específica para una sola clase; es un medio descriptivo que define a cualquier ordenación de objetos, realizado para facilitar su análisis.

El clasificar objetos puede obedecer a alguno de los siguientes métodos:

- ✓ Diferenciar objetos y sus atributos, separando el concepto (p. ej., Árbol) de sus características (altura, grosor, etc.).
- ✓ Clasificar objetos y componentes (observando la distinción entre el árbol y sus ramas).
- ✓ Agrupar conjuntos y familias afines (conformar clases de árboles en función de su forma, origen, tipo, etc.).

Durante el análisis y diseño de un SOO, pueden reconocerse claramente la aplicación de algunos -y a veces de la totalidad- de los métodos aquí indicados. Si bien es cierto que el proceso de clasificación es una forma intuitiva de descomposición del problema, proporciona un esquema fácilmente adaptable y -sobre todo- descriptivo, sobre el que pueden bosquejarse los elementos de todo un sistema.

Al especificar tanto los atributos conceptuales como las características distintivas de cada elemento, explícita e independientemente de la realización de los objetos mismos, la clasificación introduce el concepto de **abstracción**, que permite la apreciación de los caracteres de un elemento, destacando aspectos de interés y discriminando los detalles más profundos o menos evidentes.

El pensamiento abstracto complementa al concreto (su antónimo funcional), en un proceso que utilizamos cotidianamente en nuestra comunicación social, de tal suerte que la "percepción" de un objeto dado (abstracción) se convierte en la "descripción" de un elemento (concretado). La **abstracción** es una propiedad del proceso de definición de una clasificación, e interviene en la conceptualización de cada uno de sus niveles.

Así, para clasificar un conjunto de objetos inicialmente "abstraemos" su estado actual a lo más obvio, estaremos entonces en el primer nivel de abstracción y a medida que avancemos en la descripción, nuestra percepción del objeto será menos abstracta, hasta el grado de puntualizar todas y

cada una de sus características. Sin embargo, el nivel de abstracción depende de la necesidad de detalle en la información requerida.

Para un SOO la clasificación culmina en la definición de clases de objetos que componen al sistema. Cada **clase** es un conjunto de enunciados que justifican su separación del nivel anterior. La clase de un objeto determina:

- ✓ La denominación del tipo de objeto.
- ✓ La información asociada al objeto.
- ✓ Las funciones que pueden operar en el objeto.
- ✓ La clase de la que es derivado el objeto.

II.4.4 Instancia, clase base, clase derivada y tipo abstracto de datos

Una clase no puede efectuar operación alguna, puesto que no es una entidad concreta, en su lugar, la clase proporciona una plantilla que define las características que tendrán los objetos creados en base a ella; cuando esto ocurre se dice que el nuevo objeto creado es una **instancia** de la clase y ésta se denomina **clase base**.

El instanciamiento de una clase produce por resultado una entidad que comparte con su clase base el conjunto entero de su definición. Dentro de un sistema, cuando se instancia una clase para obtener un nuevo elemento, se dice que ocurre una definición dinámica (dynamic binding), puesto que ocurre siempre en tiempo de ejecución.

Como un ejemplo supongamos que se ha identificado dentro de nuestro universo del problema un elemento poseedor de una característica A, podemos comenzar definiendo una clase que, a falta de una denominación en particular, llamaremos Clase A. Si entrando en más detalle, reconocemos a un mismo nivel dos características *B* y *C*, podremos "derivar" de nuestra Clase A, un nivel inferior pero con mayor detalle, compuesto de dos clases derivadas (que llamaremos Clase derivada B, que reúne las características de la Clase Base A con las propias y la Clase derivada C, que hace lo propio con las características de la Clase Base A y las de ella misma). Ahora, si decidimos definir un objeto del tipo de la Clase B, instanciaremos la clase en la forma de un Elemento X, que presentará las características que ya definimos como propias de la Clase B, que se convierte en la clase base del objeto X. Nuestro objeto recibe los atributos de su clase base, y se le conoce como un objeto de nombre X del tipo B (Fig. II.4).

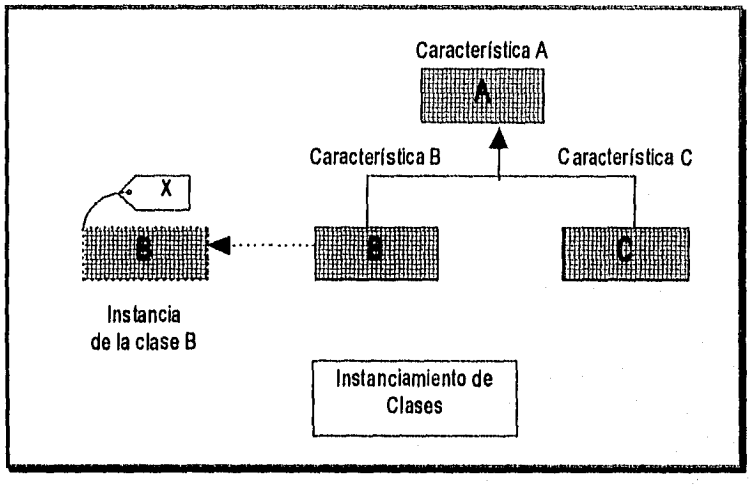


Fig. II.4.- Instanciamiento de Clase e Instancia de un Elemento de una Clase.

En términos de programación, una clase de objetos puede verse también como un tipo de objetos y ser usado como cualquiera otro tipo de datos del sistema; en esas condiciones la clase se relaciona fuertemente con el concepto del **tipo abstracto de datos** creado por el usuario o tipo de datos definido por el usuario. En un sistema procedural, si pudiésemos definir un tipo de datos distinto al entero, real, punto flotante o alfanumérico que tenemos disponibles en un lenguaje, determinando cada una de sus características y operaciones afines, estaríamos creando un tipo de datos a partir de la abstracción hecha al imaginar cómo sería. Dentro de un SOO existen las facilidades para realizar esta tarea.

II.4.5 Herencia

En Taxonomía, ciencia que clasifica a los seres vivos, se denomina **"Taxón"** al conjunto de especificaciones que configuran un sistema jerárquico, y que encuadra la condición que un ser mantiene dentro de su clasificación. Para un SOO, una clase es el producto del detallado sucesivo de un objeto, que destaca su importancia cuando se le enmarca bajo su jerarquía clasificatoria respectiva; para llegar a ella el **proceso de abstracción** ha pasado por la definición de varios niveles desde la clase más básica hasta su nivel jerarquico. En cada nivel el taxón de la clase acumula las características de los niveles superiores y las de la clase misma.

Se denomina, entonces, **herencia de clase** a las características provenientes de los niveles jerárquicos superiores que influyen directamente en el comportamiento de nuestro objeto, y **definición de clase** a las especificaciones distintivas de su clase. La herencia es el análogo a la herencia

taxonómica, con la diferencia de que aquí la herencia puede recibirse de más de dos ancestros o inclusive de uno solo y no de dos.

La **herencia** es un nuevo concepto que, en contraparte al encapsulamiento y a la clasificación, solamente está disponible en los SOO. Es una propiedad de las clases, que complementa su funcionalidad al hacer uso de las características definidas para su clase base en la ejecución de algunas o todas sus operaciones. La **herencia** emerge del hecho de que el objeto se compone de datos y sus funciones conservados dentro de un ámbito local o **contexto** bien definidos. Ya que la derivación de una clase se efectúa sobre el contexto de la clase base, la clase derivada presentará un nuevo conjunto de datos y funciones, extendiendo su ámbito más allá del de la clase base, que se convierte en un subconjunto de ésta. Es por este concepto que no se utiliza en este trabajo el término de subclase, pues supone un subconjunto de la clase inmediata superior, lo que resultaría contradictorio con el significado de herencia aquí expuesto. Sin embargo, autores como James Martin -en su libro Object Oriented Design- prefieren hacer una clara división de los conceptos asociados a la definición contextual. Para Martin, la generalización lleva al concepto de un supertipo de objeto, el cual incluye a uno o más tipos o contextos, y cuya definición es más general que aquellos a los que contiene, mientras que define a un subtipo de objeto como aquel cuyos miembros están contenidos en otro tipo de objeto, cuya definición es más especializada cada vez, es decir utiliza un supertipo de objeto para definir herencia y al subtipo para la especialización.

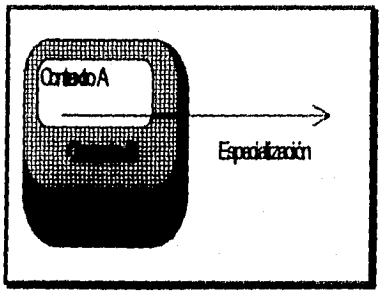


Fig. II.5.- Figura de "Lattice".

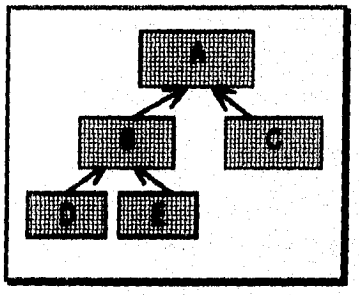


Fig. II.6.- Estructura de árbol para la herencia.

Como se muestra en la Fig. II.5 (llamada figura de "Lattice"), el contexto de la clase D se construye complementando la herencia obtenida de la clase base, que consta del contexto B que a su vez ha heredado el contexto A. Podríamos decir que la herencia de la clase D está compuesta por las características de clase de B y de A juntas y el Taxón se definiría como la suma de las propiedades de D y su herencia.

La herencia es estrictamente progresiva, es decir, sólo puede ir de un nivel de especialización inferior a uno superior o, en términos de abstracción, del menor al mayor (Fig. II.6). De ésta forma provee un modo de relacionar a las clases y de compartir la definición de cada nivel reflejando la especialización como atributos adicionales a los proporcionados por las clases. En el dominio del software la herencia también evita la duplicidad de código al compartir sus métodos con toda su "descendencia".

II.4.6 Polimorfismo

Del griego Poly (muchos) y morphos (forma): Múltiples Formas; en el ámbito de los SOO, puede definirse como la capacidad que tiene un **mensaje** de ser adoptado por la interface de más de un objeto y ser interpretado distintamente, según sea el tipo de objeto receptor. Es decir, es la propiedad del proceso de ejecución de una operación de ser sensitivo al tipo de objeto al que es aplicada.

Según Booch, "Es un concepto en el cual un nombre puede denotar a objetos de muy distinta clase relacionados por una clase base común. Así, cualquier objeto denotado por este nombre es capaz de responder al mismo conjunto de operaciones de distinto modo".

Mediante el **polimorfismo**, las abstracciones comparten elementos de su interface y presentan una homologación de operaciones que facilita al usuario el uso intensivo de los objetos obviando la discriminación de las características implícitas de la herencia. Es decir, se utiliza a la herencia en sustitución del análisis selectivo necesario para determinar si un objeto es candidato a recibir un mensaje.

El **polimorfismo** está fuertemente ligado con el concepto de construcción dinámica, ya que mediante ese mecanismo se permite la creación de objetos en tiempo de ejecución que reflejen el mismo conjunto de operaciones heredadas de su clase base, pero que también pueden implementar sus propias operaciones.

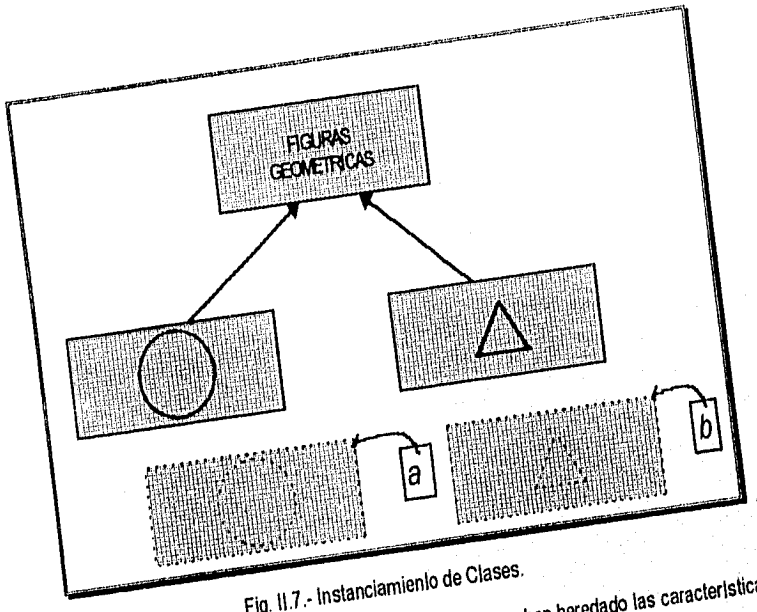


Fig. II.7.- Instanciamiento de Clases.

En la Fig. II.7, se presentan dos instancias, *a* y *b*, las cuales han heredado las características de la clase base "Figuras Geométricas", y comparten el conjunto de características de su herencia, pero cuando se les solicite una función que ambas deben ejecutar, por ejemplo la operación "dibujarse a sí mismo" llevará al objeto *b* a dibujar un triángulo, mientras que *a* dibujará una circunferencia.

II.5 Ejemplo de modelado orientado a objetos

Ubiquemos un "antiguo" reloj Rólex de pulso y cuerda, dentro del contexto de nuestra realidad como un elemento perteneciente al conjunto de los Relojes (dispositivos que nos permiten conocer la evolución cuantitativa del tiempo). Nuestro objeto constará de distintas características:

- Aquellas que comparte con todos los objetos Reloj (mostrar la hora, resolución en segundos, es modificable o ajustable) y que cualquier reloj, por definición posee. (léase reloj atómico, de cuarzo, de pared, despertador, de mesa, de pulso, de arena, de agua, etc.).
- Aquellas compartidas con los objetos Reloj Analógico (manecillas, segundero, carátula, numerales, indicadores de minutos), como son los de pared, de péndulo, de cucú, el Big Ben, algunos despertadores y algunos de pulso.

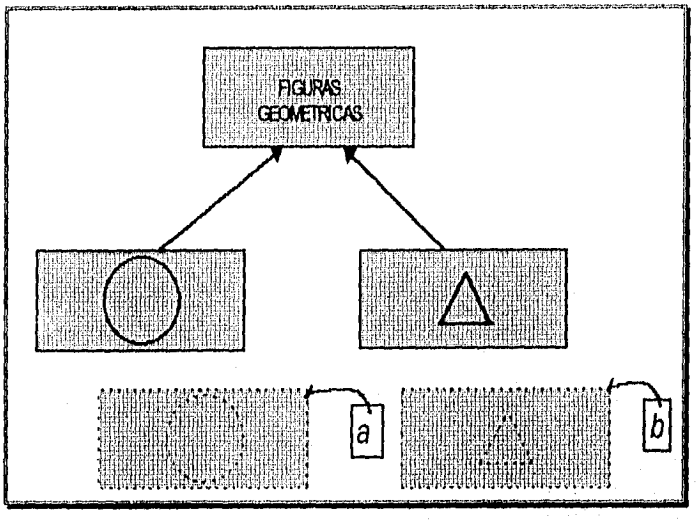


Fig. II.7.- Instanciamiento de Clases.

En la Fig. II.7, se presentan dos instancias, *a* y *b*, las cuales han heredado las características de la clase base "Figuras Geométricas", y comparten el conjunto de características de su herencia, pero cuando se les solicite una función que ambas deben ejecutar, por ejemplo la operación "dibujarse a sí mismo" llevará al objeto *b* a dibujar un triángulo, mientras que *a* dibujará una circunferencia.

II.5 Ejemplo de modelado orientado a objetos

Ubiquemos un "antiguo" reloj Rólex de pulso y cuerda, dentro del contexto de nuestra realidad como un elemento perteneciente al conjunto de los Relojes (dispositivos que nos permiten conocer la evolución cuantitativa del tiempo). Nuestro objeto constará de distintas características:

- Aquellas que comparte con todos los objetos Reloj (mostrar la hora, resolución en segundos, es modificable o ajustable) y que cualquier reloj, por definición posee. (léase reloj atómico, de cuarzo, de pared, despertador, de mesa, de pulso, de arena, de agua, etc.).
- Aquellas compartidas con los objetos Reloj Analógico (manecillas, segundero, carátula, numerales, indicadores de minutos), como son los de pared, de péndulo, de cucú, el Big Ben, algunos despertadores y algunos de pulso.

- Las que son comunes a los objetos Relojes Analógicos Mecánicos (engranes, cuerda, tornillos, resortes, balancines, ejes, etc.), que agrupa a elementos como el Big Ben, relojes de pared y cuerda y, actualmente algunas piezas de relojería artesanal.
- Aquellas específicas de los Relojes Analógicos Mecánicos de Pulso (extensible, hebilla, pernos, caja, cristal de reloj, etc.) que comprenden a todos los relojes de las marcas conocidas que manejen o manejaron esa línea.
- Las que todos los relojes Rólex de ese tipo presentan (distintivo, logotipo, chapa de oro, mecanismo de diseño exclusivo).

Si continuamos nuestro proceso de identificación podremos concretar aún varios niveles más de especialización para nuestro objeto. Si decidimos, por el contrario omitir el resto de las posibilidades estaremos abstrayendo nuestro objeto hasta el nivel actual.

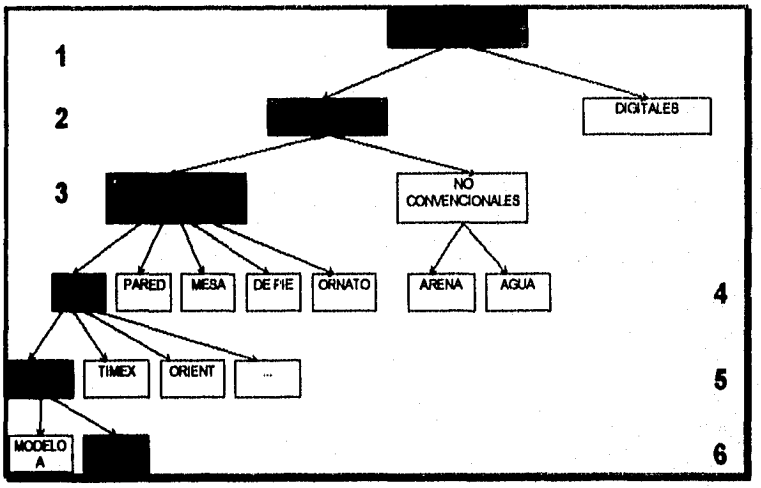


Fig. II.8.- Ejemplo de modelado orientado a objetos.

La abstracción nos permite omitir detalles de nuestra definición que no nos interesan. Así, si construimos una clasificación de las características que conocemos de los relojes, podemos armar una jerarquía que nos recuerda la taxonomía (ver Fig. II.8); donde, a menor especificación dentro de la clase corresponde una mayor abstracción. En nuestro caso, si decimos que tenemos simplemente un reloj estamos en el nivel máximo de abstracción para nuestro objeto, omitiendo los detalles más profundos, y

si por otra parte, damos toda la información de nuestro objeto -propia de la máxima profundidad en nuestra clasificación- reflejamos el menor nivel de abstracción.

Nuestro objeto ha quedado ubicado dentro de una clasificación, es un objeto de clase Reloj y, más específicamente, Reloj Analógico Mecánico de Pulso Marca Rólex Modelo B (ver Fig. II.8 parte sombreada) y podemos decir que nuestro reloj pertenece a esta clase en particular o que es una instancia de ella. Una representación de nuestro objeto podría ser el diagrama de entidad mostrado en la Fig. II.9, empleado como un auxiliar en el diseño de SOO.

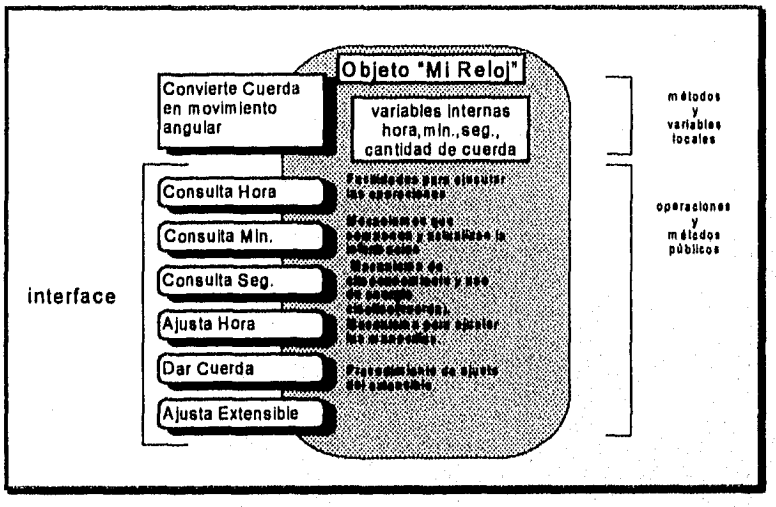


Fig. II.9.- Diagrama de Entidad del Objeto "Mi Reloj".

El objeto contiene variables que reflejan su estado dinámico que se mantiene encapsulado y oculto a nuestra vista, en este caso el tiempo, solamente disponible cada vez que lo consultamos -a través de alguna de las operaciones o métodos de su protocolo- que es cuando nuestro conocimiento de su estado se actualiza.

Las operaciones que podemos hacer con nuestro objeto son identificables fácilmente porque las conocemos, las hemos hecho intuitiva y hasta inconscientemente. Estas operaciones se agrupan en una interface que controla el acceso al estado actual, protegiendo los datos; mientras que el estado funciona como una memoria que afecta al estado futuro. Algunas de las operaciones son evidentes al tratar con la clase Reloj, esto se debe a que la clase especifica el conjunto de mensajes aceptados por los objetos que la componen, define una interface de encapsulamiento común a todos los relojes, sea el

tipo que fuere. La clase también proporciona una plantilla, un molde para crear objetos con sus características predefinidas; cuando esto ocurre decimos que el nuevo objeto es una instancia creada en términos de esa clase base.

Cuando se creó nuestro objeto, todas las características definidas para los niveles que le antecedieron dentro de la jerarquía de clasificación se acumulan para formar su contexto, acompañándose con las características particulares, así nuestro objeto heredó la lista de operaciones válidas de la clase base y copió las de la clase propia.

El Polimorfismo en nuestro ejemplo queda manifiesto dentro de la jerarquía de clase del objeto, al comprobar que una operación, Consulta Minutos puede ser atendida por cada clase de reloj de distinta forma, un reloj de arena al mostrar un nivel de su contenido, uno de agua al mostrar la cantidad de agua desplazada, uno digital al cambiar los numerales y uno analógico por medio del movimiento del minutero sobre la carátula.

II.6 Elementos de beneficio en la OO

La adopción de la TOO permite observar ventajas inmediatas sobre los sistemas así desarrollados. La mayoría de esas ventajas no son un producto exclusivo de las TOO, sino que son el resultado de la materialización de tendencias de diseño y de características de la programación, que muchos desarrolladores encontraban convenientes o que facilitaban su trabajo en las tecnologías anteriores y que fueron conceptualizadas para interactuar estrechamente.

Reusabilidad.- Si se aplica adecuadamente, los mecanismos de la OO, tales como el encapsulamiento, herencia, polimorfismo y la definición dinámica, obvian las barreras técnicas para reusar el código de un programa, lo que adicionalmente propone la reutilización de los modelos de diseño o del marco de desarrollo y de los modelos de análisis para dominios relevantes de un problema. En el nivel de análisis y diseño, la reusabilidad puede tomar dos formas básicas: reuso de los componentes de un diseño previamente desarrollado y reuso de abstracciones de componentes de un programa preexistente. Con la reusabilidad, el análisis y diseño consumen más recursos que la programación y, como consecuencia principal, el presupuesto de desarrollo puede ser reducido tras un estudio de los elementos del sistema susceptibles de usar partes de un diseño previo. El mantenimiento del sistema por cambios extensivos se facilita al zonificar el impacto del cambio sobre unos cuantos objetos.

Modularidad.- La modularidad de los objetos separa como ya se ha mencionado, al usuario de una aplicación, de su autor, de forma en que no existe dependencia de un estilo de programación específico; favoreciendo al mantenimiento en la medida en que los módulos sean independientes.

Persistencia.- Otra característica es que la información que portan los objetos "persiste" a lo largo de la operación de un sistema haciendo accesible la "historia" del sistema a través de la información de sus objetos componentes. Cada objeto de un sistema conserva así su estado de operación aún después de ser utilizado, lo que le permite accesar a los datos anteriores en el desarrollo de sus actividades presentes para determinar su acción futura.

Estandarización.- La presencia de una estructura definida de inicio para la creación de los elementos del sistema hace que los objetos y clases presenten una apariencia homogénea que facilita la identificación de los elementos de las clases y la construcción de nuevos elementos basados en el conjunto preexistente.

Cabe hacer mención que existen aplicaciones que, sin ser totalmente OO, hacen referencia a conceptos OO. Para distinguirlas, se dice que un sistema es OO cuando su operación explota las características de creación dinámica, clasificación, herencia y polimorfismo, y por otra parte se dice que es basado en objetos, cuando su ambiente simula la presencia de objetos. Un SOO presentará verdaderamente las ventajas aquí mencionadas de una manera "natural", sin recurrir a manipulaciones especiales o simulación, mientras que un sistema basado en objetos intentará la convivencia simultánea de las ideas de la OO con las facilidades obtenidas en otras tecnologías, hecho que confunde fácilmente al usuario final; pero un sencillo análisis de su operación concluirá en la determinación exacta de su condición.

II.7 Análisis orientado al objeto (AOO)

La evolución de las metodologías modernas, comienza a finales de los 60's con el concepto del ciclo de vida del desarrollo de sistemas o SDLC (por sus siglas en inglés). El incremento en la eficiencia del hardware y la adopción de lenguajes de alto nivel, permitió construir sistemas más grandes y complicados y el SDLC trajo orden al proceso de desarrollo, que estaba excediendo los métodos del control de proyecto de entonces, descomponiendo al proceso en fases discretas de proyecto que entregan documentos formales a la siguiente fase. Una metodología de desarrollo de sistemas, combina herramienta y técnicas para guiar el proceso de desarrollo a gran escala.

El concepto del SDLC dio a los desarrolladores una medida de control, pero con poca ayuda para mejorar la productividad y calidad de análisis y diseño. Al inicio de los 70's se desarrollaron las metodologías estructuradas para promover un análisis más efectivo y diseños más estables y mantenibles. éstas eran orientadas al proceso con un énfasis menor en el modelado de entidades y datos. Su orientación al proceso aparecía natural, dados los lenguajes de programación procedurales y las aplicaciones batch basadas en archivos. Los autores más representativos son: Yourdon y Constantine, De Marco, Ward y Mellor.

Si clasificamos las innovaciones tecnológicas como Incrementales, donde se introducen cambios relativamente menores a un proceso establecido, reforzando la tendencia actual y Radicales, que se basan en un diferente conjunto de principios estructurales y se desarrolla en un nuevo marco técnico y de solución de problemas, podemos decir que la etapa de análisis del paradigma de la OO representa un cambio radical sobre las metodologías orientadas al proceso del paradigma procedural, como la de De Marco, pero sólo representa un cambio incremental para las matodologías orientadas al dato como la ingeniería de la información de Martin.

Como en el análisis tradicional, la meta principal del AOO es la representación más completa y cercana al dominio del problema. También es meta del análisis el poner orden a nuestra percepción del mundo real para producir un modelado adecuado para el proceso de diseño. Debe simplificar el modelado de forma que se domine la complejidad al reformular el problema, removiendo el ruido y la sobreespecificación, encontrando inconsistencias, posponiendo la implementación, particionando el espacio del problema y documentándolo.

Sin embargo, como el mismo Yourdon señala, dentro de los autores que se han dedicado a estudiar los SOO, se dividen en un conjunto que se inclina por la visión sintesista, que ve a la OO como una simple acumulación de los principios de la ingeniería de software fácilmente adaptables a las metodologías existentes, mientras que otros, los revolucionarios, creen en la OO como un cambio radical que deja a la metodología y diseño convencionales en la obsolescencia.

Los participantes del sintesismo, como Wasserman, Pircher y Muller, toman la posición de que los métodos que utilizan son una elaboración de los métodos estructurados y consideran la terminología apta para el ambiente OO, de forma que declaran: *"El problema es que la OO ha sido ampliamente señalada como un enfoque revolucionario, una ruptura total con el pasado. Esto sería fascinante si fuera verdad, pero no lo es. Como muchos desarrollos en ingeniería, el enfoque de la OO es el refinamiento de algunas de las mejores ideas de la ingeniería del software del pasado"*.

Por su parte, los revolucionarios como Booch indican: *"No hay lugar a dudas de que el DOO es fundamentalmente diferente del enfoque tradicional del diseño estructurado, pues requiere una forma distinta de pensar acerca de la descomposición y produce arquitecturas de software separadas grandemente de la cultura del diseño estructurado"*.

A lo que Yourdon añade: *"No dudamos que podríamos llegar al mismo resultado (como el producido por el análisis de Coad y Yourdon), usando diferentes métodos; pero ha sido también parte de nuestra experiencia que el proceso de pensamiento, de descubrimiento, y la comunicación entre el usuario y el analista es fundamentalmente diferente con un AOO al producido por un método de análisis estructurado"*.

Ciertamente, a simple vista se aprecia que la OO aprovecha algunas de las características de los sistemas estructurados, como por ejemplo la abstracción, el encapsulamiento, la modularidad y la clasificación, haciéndolas evolucionar junto con nuevos conceptos de apreciación de los problemas como lo son la herencia, el polimorfismo y, como apuntamos anteriormente, conformando un nuevo paradigma que no puede ser comparado si no es en el contexto de solución. De entre las metodologías, expondremos aquí tres de ellas que se consideran las mejor documentadas: Coad-Yourdon, Bailin y Shaeler-Mellor.

II.7.1 Especificación de requerimientos orientado a objetos de Bailin

En respuesta a la aparente incompatibilidad entre el análisis estructurado y el DOO, Bailin desarrolló su Especificación de Requerimientos Orientado a Objetos, donde la descomposición del sistema es efectuada usando la notación de diagrama de flujo. En el análisis estructurado esta notación implica agrupar las funciones sólo si son constituidas por pasos que llevan una función de mayor nivel. En cambio, Bailin propone agrupar las funciones sólo si operan en la misma abstracción de datos; en otras palabras, las funciones no pueden existir separadas de sus datos, pero deben subordinarse a una sola entidad. Esta noción es usada para promover el encapsulamiento de funciones y datos.

Existen dos aspectos importantes en el AOO de Bailin:

Hay una distinción entre entidades, poseedoras de un estado que puede persistir a lo largo de repetidos ciclos de ejecución, y las funciones, que existen solamente para transformar las entradas en salidas sin estados de persistencia. Las entidades pueden ser descompuestas en subentidades o funciones, mientras que las funciones sólo pueden ser divididas en subfunciones.

Bailin distingue dos clase de entidades: activas y pasivas. Las activas efectúan operaciones (sobre ellas mismas o en otras entidades) lo suficientemente importantes para ser consideradas en detalle durante la fase de análisis, mientras que las pasivas son de menor importancia y pueden ser tratadas como caja negra hasta la fase de diseño. Esta distinción es importante, pues las entidades activas, las pasivas y las funciones son modeladas de diferente forma durante el proceso de análisis.

Esta metodología consiste de un proceso de siete pasos:

1) Identificar las entidades del dominio del problema. Dibujar un diagrama de flujo para designar objetos que aparecen en nombre de proceso como entidades candidatas.

2) Distinguir las entidades pasivas de las activas. Distinguir entre las entidades con operaciones significativas en los términos de la descripción de los requerimientos del sistema (activas), contra las que pueden diferir su especificación hasta el diseño (pasivas), y construir un diagrama de entidad-relación.

3) Establecer flujos de datos entre entidades activas. Construir el diagrama de entidad-flujo (EDFD) de nivel cero, designando cada entidad activa como un nodo de proceso y cada entidad pasiva como un flujo de datos o información almacenada.

4) Descomponer las entidades (o funciones) en subentidades y/o funciones. Iterativamente, junto con los pasos 5 y 6, se determina si cada entidad de nivel 0 puede estar o no compuesta por entidades de nivel inferior, considerando que cada entidad hace y designa su propias operaciones como funciones. Para cada una de las subentidades identificadas, crear un nuevo EDFD y continuar el proceso de descomposición.

5) Revisar por nuevas entidades. A cada fase de la descomposición, considérese que cuando se implica a nuevas entidades, nuevas funciones se introducen y se le añaden, modificando el EDFD.

6) Agrupar funciones bajo nuevas entidades. Identificar todas las funciones ejecutadas por o en nuevas entidades.

7) Asignar a las entidades un dominio propio. Asignar a cada entidad un dominio de aplicación y crear un conjunto de DER's (Diagrama de Entidad Relación), uno para cada dominio.

El resultado del método de Bailin es un diagrama de entidad-relación, junto con una jerarquía de diagramas entidad-flujo de datos. La metodología conforma los principios elementales de la OO, a pesar de que no se utiliza terminología de la OO. Los diagramas de entidad-relación capturan la clasificación de objetos y su oportunidades de herencia, mientras que las funciones coinciden con el concepto de encapsulamiento.

II.7.2 Análisis orientado al objeto de Coad-Yourdon

El AOO es visto por los autores como "la construcción a partir de los mejores conceptos del modelado de la información, Lenguajes Orientados a Objetos (LOO) y sistemas basados en conocimientos".

El modelo es construido a partir de un proceso de cinco pasos:

1) Definir Objetos y Clases. Buscar estructuras, otros sistemas, dispositivos, eventos, procedimientos operacionales y unidades organizacionales. La estrategia de búsqueda puede hacerse de manera informal, dentro de un enunciado que define la solución del problema, presentado en lenguaje natural y a un nivel consistente de detalle, o como un proceso intuitivo de identificación de entidades, teniendo en cuenta más que a la solución, a los elementos del problema.

2) Definir estructuras. Buscar relaciones entre clases y representarlas como estructuras generales-a-específicas (donde se reconoce a los objetos susceptibles de especificarse aún más o si un objeto identificado es una especificación de otro) o como estructuras completas-a-elementales (identificando si los objetos requieren de otros para su definición completa o si a su vez son necesarios para lograr una definición).

3) Definir áreas de sujetos. Examinar los objetos de nivel superior con jerarquías Completa-a-elemental y marcarlas como candidatas a ser área de sujetos, refinándose al minimizar interdependencia entre los sujetos.

4) Definir atributos. Identificar las características atómicas de los objetos como atributos. Buscar también relaciones asociativas entre objetos y determinar la cardinalidad de esas relaciones.

5) Definir servicios. Para cada clase y objeto, identificar todos los servicios que efectúa, dentro de su ámbito o en beneficio de otras clases y objetos.

Las herramientas primarias para el AOO de Coad-Yourdon son los diagramas de objeto y las cartas de servicio. Los diagramas de clase y objeto tienen cinco niveles que se construyen incrementalmente durante cada una de las cinco fases descritas. Las cartas de servicio son más parecidas a un diagrama de flujo tradicional, se usan durante la última fase, para representar la lógica interna de los servicios.

Este método soporta explícitamente cada uno de los principios esenciales de la OO. Los diagramas de clase y objeto proveen de la clasificación de objetos y de las posibles relaciones de herencia; el encapsulamiento es modelado por el concepto de servicio exclusivo y la conectividad de los mensajes. El método hace énfasis en el modelado de la información.

II.7.3 Análisis orientado al objeto de Shaler-Mellor

Shaler y Mellor desarrollaron su metodología de análisis en el curso de varios años de asesoría en modelado de la información. Proponen una descripción completa del dominio del problema mediante tres formas de ver al sistema: información interrelacionada, modelo del estado y modelo del proceso, aplicadas durante seis etapas:

1) Desarrollar un modelo de la información. Consistente de objetos, atributos, relaciones y construcciones de objetos múltiples (basados en relaciones asociativas, y en el análisis de los enunciados "es un" y "es parte de"). Para los autores, un objeto es un equivalente a la noción convencional de entidad, es decir, una persona, un lugar o un evento que tenga lugar en el mundo real.

2) Definir el ciclo de vida de los objetos. Efectuar el análisis del ciclo de vida de cada objeto (desde su creación hasta su destrucción) y formalizar el ciclo de vida en colecciones de

estados (algunas condiciones predefinidas de un objeto), eventos (señales que causan la transición de un estado a otro), reglas de transición (especifican transiciones disponibles entre estados) y acciones (actividades u operaciones que deben ser efectuadas por un objeto para llegar a un estado). También se definen los *timers*, mecanismos usados por las acciones para generar un evento futuro. La herramienta principal en esta etapa es el modelo de estado.

3) Definir la dinámica de las relaciones. Se desarrolla un modelo de estado de aquellas relaciones entre objetos que ocurren dinámicamente. Para cada relación dinámica se define un objeto asociado en el modelo de la información. Se definen modelos especiales de asignación para relaciones en que puede haber contención de instancias de objeto por los recursos de otra instancia de objeto.

4) Definir la dinámica del sistema. Aquí se produce un modelo del tiempo y control al nivel del sistema. Un modelo de comunicación de objeto (MCO) muestra el control síncrono, mientras que un modelo de acceso de objetos muestra el control asíncrono. Los autores describen un procedimiento para el trazado de flujos de control a alto y bajo nivel.

5) Desarrollo de modelos de proceso. Para cada acción, un diagrama de flujo de acciones se crea para mostrar todos los procesos para esa acción, y el flujo de datos entre el proceso y su almacenamiento. Se utiliza un diagrama de flujo de datos convencional (De Marco). Se definen cuatro tipos de proceso: accesantes, generadores de eventos, transformaciones y pruebas, y se proveen las guías para ubicar las acciones dentro de estos cuatro tipos.

6) Definición de dominios y subsistemas. Para problemáticas mayores, es útil descomponer al sujeto en dominios conceptualmente distintos, de los que se identifican cuatro: aplicación, servicio, arquitectura e implantación. Adicionalmente, algunas veces es útil descomponer el dominio de aplicación en múltiples subsistemas.

Shaler y Mellor otorgan un soporte más implícito que explícito para los tres principios esenciales de la OO. Los objetos y las relaciones contenidas en el diagrama de estructura de información, aunque no idénticas a los conceptos de clasificación y herencia, pueden fácilmente ser utilizadas durante el diseño.

II.8 Diseño orientado al objeto (DOO)

Diseño es el proceso de interpretar los requerimientos del sistema definidos durante la etapa del análisis en una representación abstracta de una conceptualización basada en el sistema, tomando en cuenta costos, calidad y eficiencia. El DOO transforma las clases del análisis en un modelo computarizado perteneciente al espacio de solución del problema que obedezca a una descripción de clase.

Las metodologías para el diseño de SOO presentan diferencias substanciales contra las metodologías tradicionales, siendo las principales:

- Diseño de datos que soporta la definición de clases, herencia y métodos.
- Utiliza descomposición modular OO; mientras que en el diseño tradicional, módulo puede ser un programa, subrutina o función que sólo contiene código procedimental, en el DOO, el objeto es la unidad primaria de modularidad.
- Se utiliza la descomposición OO; que resulta en una colección de métodos encapsulados dentro de los objetos, en lugar de la descomposición funcional que resulta en módulos de programa.

Dentro de la clasificación Incremental-Radical, el DOO tendría que colocarse como un cambio Radical para los paradigmas orientados al proceso y orientado al dato, al proveer una forma distinta de contemplar, dividir, estructurar y representar al problema. Cabe también anotar que el DOO exige el uso de herramientas adecuadas para la representación detallada de las definiciones de clases y herencias, de las relaciones entre clases y objetos, y de las conexiones entre mensajes y operaciones de objeto. Existen decisiones implícitas en el proceso del DOO que determinan el estilo de diseño y que pueden afectar grandemente al resultado, como es, por ejemplo, el balance correcto entre el máximo encapsulamiento (al enfatizar las responsabilidades de los objetos), la máxima herencia (enfatizando las similitudes entre clases), o la máxima reusabilidad (diseñando clases que sean independientes del contexto en el cual son usadas).

Actualmente la definición de métodos de DOO, en oposición a los del diseño estructurado, se considera un aspecto inmaduro de la TOO, encontrándose en una etapa de definición y refinamiento. A continuación se expondrán brevemente las características de tres de los métodos de diseño difundidos hasta 1994.

II.8.1 Diseño estructurado orientado al objeto (DEOO) de Wasserman

El DEOO fue desarrollado por Wasserman, Pircher y Muller, como una metodología que provee una notación detallada para describir el diseño estructural o de alto nivel que identifica los módulos individuales, pero sin representar su detalle interno. Para Wasserman, la meta principal del DEOO, es proporcionar una notación de diseño estándar, capaz de soportar a cualquier diseño de software, tanto OO como convencional; para lo que ofrece una notación híbrida que incorpora conceptos de trabajos previos de diseño como las cartas de estructura, la notación de Booch para paquetes y tareas, jerarquía y herencia, y el concepto de monitores para la programación concurrente.

Wasserman propone el uso de símbolos y notaciones típicos del diseño estructurado, como las cartas de estructura, incluyendo concepto de módulos, datos parámetro y parámetros de control y añade nuevas notaciones OO basadas en el esquema de Booch para los elementos de diseño en ADA equivalentes a objetos, clases, herencia, métodos, instanciamiento y concurrencia (Cartas de Estructura Orientada a Objetos).

II.8.2 Diseño orientado a objetos de Booch

En la década de los 80's, Booch incursionó en el campo del DOO, al proponer una metodología basada en el lenguaje ADA, que posteriormente fue significativamente expandida y generalizada. Booch plantea su metodología más como una alternativa al diseño estructurado que como una extensión. En su planteamiento, se describen un conjunto de técnicas y herramientas, desde una lista informal hasta diagramas y plantillas formales; sin prescribir un orden fijo en las fases de su propuesta, recomienda seguir un trabajo iterativo e incremental para la construcción de los formalismos con técnicas informales. Booch delinea los cuatro pasos que, según él deben seguirse para realizar un DOO.

1) Identificar Clases y Objetos. Identificar abstracciones clave en el espacio del problema y etiquetarlas como candidatas a Clase/Objeto.

2) Identificar la semántica de clases y objetos. Establecer el significado de las clases y objetos identificados en el paso anterior usando varias técnicas, que incluye la creación de "scripts" que definen el ciclo de vida de cada objeto, desde su creación hasta su destrucción.

3) Identificar relaciones entre clases y objetos. Establecer interacciones de clase y objetos, tales como patrones de herencia entre clases y patrones de cooperación entre objetos. Este paso también captura algunos detalles de visibilidad entre clases y objetos.

4) Crear Clases y Objetos. Construir vistas detalladas internas de clases y objetos, incluyendo definiciones de sus servicios. Alojar las entidades (objetos y clases) de acuerdo al lenguaje usado y alojar programas para los procesadores (en los ambientes que soporten multiproceso).

Las herramientas primarias usadas durante el DOO son:

- ✓ Diagramas y plantillas de clase (muestran definiciones de clase y relaciones de herencia).
- ✓ Diagramas de Objeto y de tiempos (que enfatizan definiciones de mensaje, visibilidad y flujos de control).
- ✓ Diagramas de transición de estado (para modelar la transición y el estado de los objetos).

- ✓ Plantillas de operación (capturan la definición de servicios).
- ✓ Diagramas y Plantillas de módulo (para capturar decisiones de diseño físico acerca de la asignación de los módulos en objetos y clases).
- ✓ Diagramas y Plantillas de proceso (para asignar módulos a procesos en situaciones donde se usa una configuración multiproceso).

El DOO de Booch ha provisto a la mayoría de las metodología modernas de las herramientas de modelado básicas de la OO.

II.8.3 Diseño de manejo de responsabilidades (DMR) de Wirfs-Brock

Wirfs-Brock, Wilkerson y Wiener desarrollaron el método DMR como producto de varios años de experiencia en el desarrollo interno de Software corporativo. DMR se basa en el modelo computacional de Cliente-Servidor, en el cual el sistema es visto como una colección de Servidores que acaparan responsabilidades privadas y proveen servicios a los Clientes, de acuerdo a contratos que definen la naturaleza y ámbito de las interacciones Cliente-Servidor válidas.

En el contexto de la terminología OO, clientes y servidores son diferentes tipos de objeto, mientras que los servicios y responsabilidades son los métodos. Los contratos y las colaboraciones son metáforas para la idea de que para conservar el encapsulamiento, algunos objetos tienen la capacidad de efectuar ciertas tareas (p. ej., modificar el valor de sus variables internas) para el beneficio de otros objetos y de que algunos objetos necesitan los servicios de otros para alcanzar un resultado deseado.

Esta metodología se dice de "Manejo de Responsabilidades" por que se avoca, durante el diseño, a los contratos entre los objetos clientes y los servidores. Dichos contratos indican expresamente de qué acciones es responsable el objeto y qué datos está obligado a compartir. Los autores contrastan su método contra lo que llamaron Diseño de Manejo de Datos (DMD), que enfatiza el enfoque en las clases y herencia por medio del diseño de las estructuras de datos internas para los objetos y la construcción de relaciones de herencia basados en sus atributos comunes. Por el contrario, el enfoque del DMR intenta maximizar el nivel de encapsulamiento en el diseño resultante, poniendo mayor atención a las interacciones de objeto y su encapsulamiento.

Como Booch, Wirfs-Brock recomienda un desarrollo incremental-iterativo de su método, sin orden fijo. El DMR presenta seis pasos que se aplican en dos etapas: exploración, donde se busca por clases candidatas, responsabilidades, colaboraciones y construcción de jerarquías, donde se definen subsistemas y protocolos.

1) Encontrar Clases. Extraer sustantivos de las frases que componen la especificación de requerimientos y construir una lista de clases candidatas al observar los sustantivos que se

refieran a objetos físicos, entidades conceptuales, categorías de objetos o interfaces externas. También se identifican los atributos de objetos y superclases candidatas.

2) Encontrar responsabilidades y asignarlas a las clases. Considerar el propósito de cada clase y examinar la especificación de las frases de acción para encontrar responsabilidades candidatas. Asignar responsabilidades a las clases de tal forma que la inteligencia del sistema se distribuya, las propiedades residirán en la información relacionada y las responsabilidades que se comparten entre clases afines.

3) Búsqueda de colaboraciones. Examinar responsabilidades asociadas con cada clase y considerar qué otras clases son necesarias y que su colaboración complemente la responsabilidad.

4) Definir jerarquías. Construir jerarquías de clase para las relaciones hereditarias de tipos comunes y para las clases que presentan responsabilidades parecidas.

5) Definir subsistemas. Dibujar un diagrama de colaboración para el sistema completo; observando aquellas colaboraciones frecuentes y complejas para considerarlas como candidatas a subsistemas. Las clases dentro de un subsistema deben soportar un conjunto de responsabilidades pequeño y fuertemente cohesionado, además de ser fuertemente interdependiente.

6) Definir protocolos. Desarrollar un diseño detallado, escribiendo las especificaciones para las clases, subsistemas y protocolos.

El DMR presenta un contraste significativo contra las metodologías anteriores, puesto que en su afán de enfocarse a las responsabilidades y características dinámicas de los objetos, intenta construir una simulación cercana al funcionamiento del sistema; en lugar de establecer de entrada una jerarquía de clase, haciendo énfasis en las relaciones estáticas de sus contrapartes del diseño.

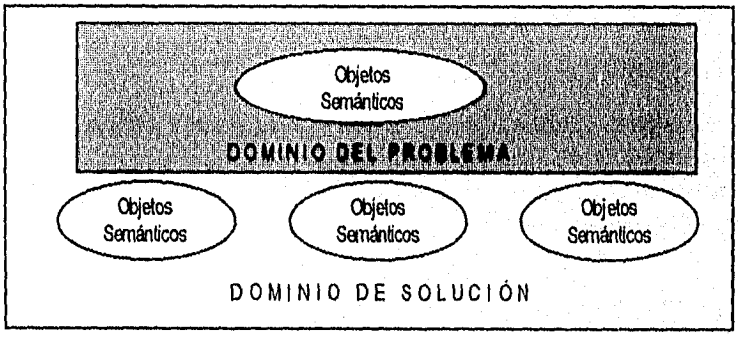
II.9 Comentarios a los métodos orientados a objetos

El DOO y el AOO conviven en la práctica de una forma en la que parece indistinguible una frontera real a partir de la que se pueda hacer una separación de los dos conceptos; y es que durante el proceso de desarrollo, se pasa constantemente de un lado a otro de la línea, como se aprecia al observar que la concepción de "objeto" es más bien dinámica a lo largo del ciclo de creación del sistema. Podemos notar cómo en el análisis se denomina objeto al elemento de modelado y que al pasar al diseño se le trata como un elemento de clasificación o clase, para luego, durante la programación y ejecución de los programas, regresar al concepto de objeto como una instancia.

En un esfuerzo por categorizar nuestra labor de desarrollo, podríamos distinguir nuestra estadía en un lado y en otro, al identificar actividades típicas, es decir, si queremos saber cómo se llama lo que estamos haciendo, debemos detenernos un poco y tratar de explicar qué es exactamente lo que hacemos y de qué manera. Para comprender este marco de referencia, observemos que:

AOO: Es la etapa donde se modela el dominio del problema, identificando y especificando un conjunto de objetos semánticos que interactúan de acuerdo a los requerimientos del sistema. Durante el análisis se especifican (abstraen) los objetos que forman parte de la descripción del problema y sólo tienen significado allí, perteneciendo al dominio del problema. La reespecificación de responsabilidades y los enunciados semánticos que detallan a un objeto, determinan la aplicación del análisis.

DOO: Modela el dominio de solución del problema, especificando la intercomunicación entre objetos, traduce los objetos semánticos que intervendrán con el usuario, en clases semánticas de interface; los que controlan directamente la aplicación o la secuencia de las funciones en clases de Aplicación y los objetos semánticos identificados como componentes independientes como clases de utilería. Durante esta etapa se organizan las clases desde el punto de vista dinámico y se estructura la complejidad.



CAPÍTULO III

CARTAS ASM

CAPÍTULO III

CARTAS ASM

En 1973 fue propuesto por Clare[1], un método auxiliar para el diseño de sistemas digitales. Llamó a los sistemas digitales como "Máquina de Estados Algorítmica" (ASM: Algorithm State Machine) y formalizó la relación entre los estados y el funcionamiento del sistema, en diagramas semejantes a los diagramas de flujo. A estos diagramas se les denomina "Cartas ASM".

Las Cartas ASM son entonces, representaciones de algoritmos, y las podemos relacionar con los diagramas de flujo que se elaboran antes de escribir un programa de computación. La Carta ASM representa al "programa" que va a gobernar el funcionamiento del sistema, siendo posible "escribir" este programa en memoria o de forma alambrada por hardware.

Una Carta ASM se define como una representación formal, -mediante un diagrama de flujo- del algoritmo que da solución a un problema de diseño de un sistema digital planteado.

Una descripción fundamental del comportamiento de cualquier sistema de procesamiento de información, ya sea de hardware o software, está provista de un algoritmo. Éste, es una secuencia de eventos cuidadosamente diseñada, la cual es necesaria para obtener un conjunto de resultados o acciones a partir de un conjunto de datos (así, un algoritmo está inherentemente ligado a los datos y al control del flujo). Un algoritmo tiene un punto de inicio y termina después de un número finito de pasos, los cuales están definidos con precisión y sin ambigüedades.

En el diseño de circuitos secuenciales, es imposible desarrollar un solo procedimiento formal que exprese correctamente cualquier tabla de transición; de cualquier forma, a partir de una descripción verbal del sistema a diseñar, se puede utilizar el Algoritmo de la Máquina de Estados para implementarlo, y esto es posible de lograr, gracias a la semejanza de una Carta ASM con un diagrama de flujo común y a la conversión directa de un diagrama de flujo a una Carta ASM.

La definición básica de un sistema que realiza un algoritmo está provista por la estructura de la MÁQUINA DE ESTADOS GENERAL. Esta estructura es mostrada en la Fig. III.1 y comprende 3 submódulos:

[1] Clare, Designing Logic System Using State Machines

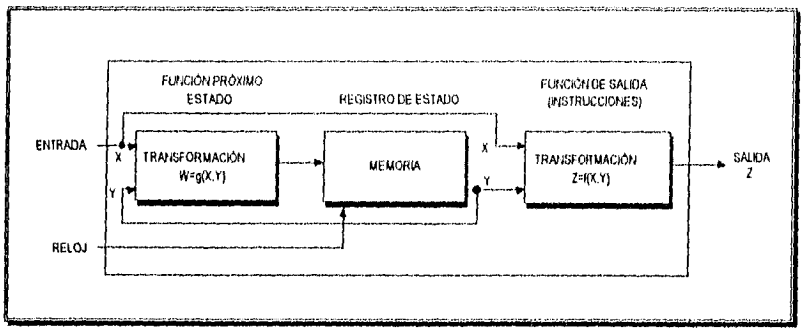


Fig.III.1.- Diagrama de la Máquina de Estados General.

1. Módulo de Función Próximo Estado. Es un módulo de transformación, implementado con un circuito combinacional, y que se encarga de definir a que estado debe pasar la máquina, después de permanecer en el actual, durante un tiempo determinado por una entrada periódica de reloj; el tiempo durante el cual la máquina permanece en un estado, se conoce como: "tiempo de estado".

2. Módulo Registro de Estado. Es un módulo que contiene elementos de memoria que almacenan el código del estado actual, desde un intervalo de tiempo a otro. Un estado está definido por las variables de estado, que son dígitos binarios, y que definen su código. Al estado definido por el código presente en un instante dado le llamaremos "Estado Activo". Este módulo puede ser implementado de varias formas, desde la utilización de elementos biestables (flip-flops) o registros para almacenar palabras (agrupación de bits) o bloques de palabras, hasta la utilización de circuitos integrados de memorias donde podemos depositar registros con información más completa.

3. Módulo de Función de Salida. Al igual que el módulo de Función Próximo Estado, es un módulo de transformación implementado con un circuito combinacional, pero éste se encarga de definir las salidas que están activas mientras la máquina permanece en el estado presente.

Estos tres módulos operan sobre un conjunto de entradas X para producir un conjunto de salidas Z. Las entradas, representadas por cualquier dato en la trayectoria de entrada, son usadas para obtener los datos de salida, la trayectoria de salida y las señales de control para el funcionamiento interno de la máquina. La trayectoria de salida, puede tomarse como trayectoria de control o puede ser usada para dar inicio a acciones en otros módulos de la máquina. De esta forma, pueden ser referenciadas como "instrucciones".

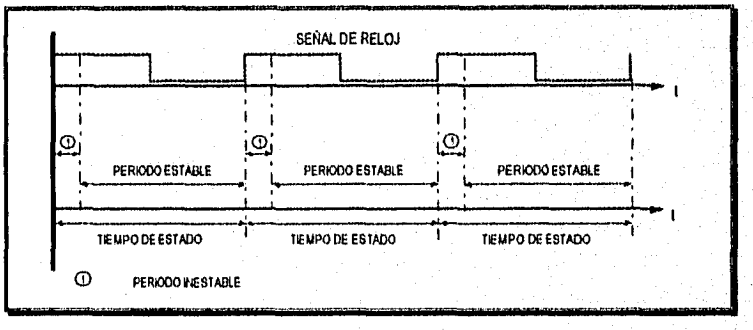
Esta estructura general es suficientemente flexible para cubrir todas la clases de circuitos combinacionales y secuenciales, así como algunos aspectos de unidades de procesamiento de

información más complejos. En cada caso los submódulos son interpretados de la manera más apropiada. En el contexto general podemos referirnos a un sistema de este tipo, como una "MÁQUINA DE ESTADOS ALGORÍTMICA" o "ASM".

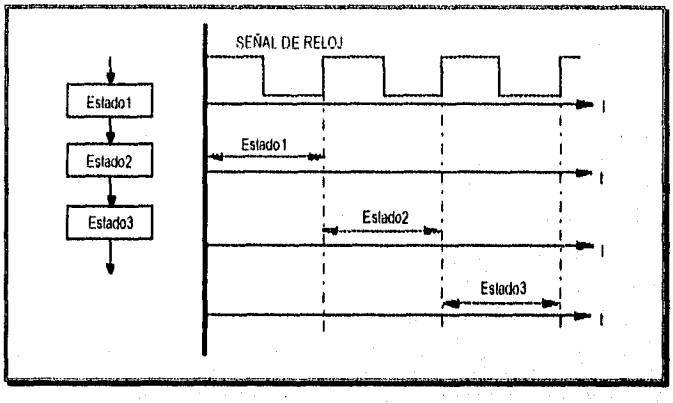
Dentro del modelo general ASM, los módulos de memoria almacenan información -de un intervalo de tiempo a otro-, y pueden ser implementados de varias formas, desde simples almacenamientos de bits en elementos biestables, hasta registros de bytes o palabras, o en bloques de varias palabras. Los módulos de memoria son obviamente dependientes del tiempo en cuanto a que los valores de salida de éstos, dependen del estado en que se encuentre la máquina en un instante dado.

Los módulos de transformación producen un conjunto de salidas por cada conjunto de entradas de acuerdo a relaciones lógicas bien definidas. En términos de hardware, esto puede ser representado por un circuito combinacional o algunos otros dispositivos para cambio de información de una forma a otra; en este contexto, esto es mejor conocido como INTERFACES. El módulo de transformación no es dependiente del tiempo, excepto por el retardo natural de la operación, presente en cualquier realización física. El módulo de transformación puede ser también un componente de software.

Cada estado de la máquina tiene un estado siguiente, determinado por la función próximo-estado; la máquina permanece en el estado actual durante un "tiempo de estado", que es determinado por una entrada periódica o "reloj" al registro de estados. Estando en el nuevo estado, la "Función Próximo Estado" calcula el siguiente nuevo estado, y la "Función de Salidas" calcula el nuevo conjunto de valores de salida, usando los valores de entrada actuales en el nuevo estado, este proceso tiene un retardo antes de ser completado, así, la primera parte del "tiempo de estado" es ocupado por este retardo, dando un período de inestabilidad, el resto del "tiempo de estado", es un período estable donde las salidas están disponibles y el valor del siguiente estado está listo para la siguiente transición.



Aunque el diseño con Cartas ASM tiene mucho parecido con la técnica de Diagramas de Flujo usada en el diseño de programas de computadora, existe una diferencia fundamental entre ambos, y es con respecto a la relación que existe entre los estados (o procesos) y el tiempo. Los Diagramas de Flujo, usualmente representan un flujo continuo en el tiempo desde el inicio hasta el final, reflejado en la secuencia de la realización de las operaciones del programa; éstas se ejecutan una inmediatamente después de terminada la anterior; las Cartas ASM son conceptualmente diferentes, ya que un estado dura al menos el tiempo determinado por la duración de un "ciclo de reloj", el próximo estado no puede iniciar si no hasta el siguiente "ciclo de reloj", por lo menos. En el siguiente esquema se muestra lo anterior.



En el esquema anterior se observa como el Estado1 permanece activo durante un tiempo mínimo (a ésta duración se le llama Tiempo de Estado) determinado por un ciclo de la señal de reloj; el Estado2 no se activará sino hasta el inicio del siguiente ciclo.

Otra diferencia, entre los diagramas de flujo y las Cartas ASM, se encuentra en la forma de desarrollar las "decisiones"; en los diagramas de flujo, éstas son instrucciones condicionales, mientras que en las Cartas ASM son funciones alambradas físicamente, en las que se evalúan las condiciones de una señal para determinar el flujo a seguir.

En el diseño tradicional de sistemas digitales, se divide al sistema en una parte de datos y otra de control de flujo. Afortunadamente el método ASM puede ser empleado para describir ambas partes, porque en él se pueden especificar los datos y el control del flujo, simultáneamente. Es posible configurar la estructura de las Cartas ASM para reflejar esta dicotomía. Esencialmente, esto requiere la partición de los datos del sistema en: "entradas de control" (variables de prueba), "salidas de estado" y "salidas de control" (instrucciones). Cada una de las partes del sistema (datos y control de

flujo) o su totalidad pueden ser descritas por un conjunto de funciones booleanas (llamadas así en honor a George Boole, 1815-1864) derivadas de la estructura ASM.

III.1 Descripción de los elementos de una Carta ASM

Como las Cartas ASM proveen una representación diagramática del comportamiento de cualquier sistema de procesamiento y forman parte de la documentación del diseño, se utilizan en su construcción tres símbolos básicos; estos son: la Caja de Estado, el Diamante de Decisión y la Caja de Salidas Condicionales.

III.1.1 Caja de estado

La "Caja de Estado" representa un estado o conjunto de condiciones de la Carta ASM; al tiempo que la máquina permanece con este estado como activo se le llama "tiempo de estado". El símbolo de la "Caja de Estado", se muestra en la Fig. III.2, que indica que cada estado tiene un "Nombre" (usualmente un mnemónico o un número) y un "Código de Estado"; este último, está representado por una combinación única de las variables de estado (que son las variables mediante las cuales se definen los estados), y que por lo regular son definidas durante el proceso de "asignación de estados". Así, cuando se dibuja por primera vez la carta, el código de estado es desconocido. Las salidas generadas durante el estado, igualmente representadas en forma de mnemónicos, están listadas dentro de la caja. Una "salida de estado" está activa, solamente cuando la máquina está en un estado que la incluye dentro de su lista de "salidas de estado". Cada Caja de Estado tiene una "Trayectoria de entrada" y una "Trayectoria de salida". La trayectoria de salida puede llevar a otra caja de estado, en este caso, se tiene una transición incondicional o directa; o puede llevar a un "Diamante de decisión", en cuyo caso se tendrá una transición condicional.

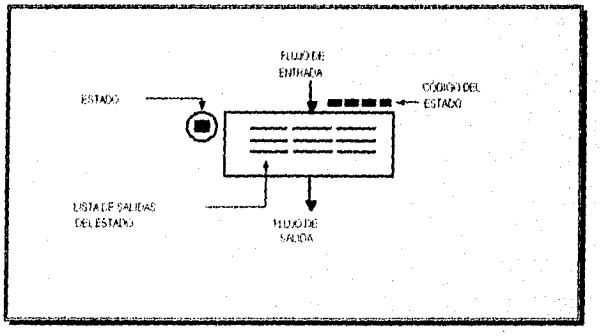


Fig.III.2.- Caja de Estado.

III.1.2 Diamante de decisión

El diamante de decisión se muestra en la Fig. III.3, e involucra las entradas al sistema. Representa un punto en el que el estado siguiente depende del valor que adquiera la variable de entrada o variable de prueba. La variable de prueba se anota dentro del diamante, y puede ser una expresión booleana. Las "Trayectorias de salida" del diamante son siempre dos y se señalan como 0 y 1 por fines de seguridad, evitando así la incertidumbre generada si se marcan como "verdadero" y "falso" cuando la presencia de una variable se detecta en valor negado. El flujo de la trayectoria es desviado de acuerdo al valor de la variable de prueba. Así, el diamante tendrá una sola entrada y como se dijo anteriormente, sólo dos salidas.

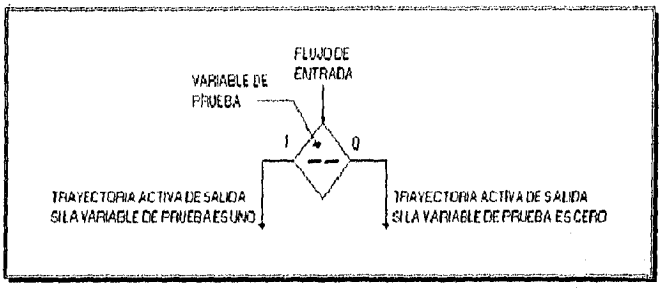


Fig.III.3.- Diamante de Decisión.

III.1.3 Caja de salidas condicionales

La Caja de salidas condicionales está siempre asociada a un diamante de decisión y a una caja de estado (formando un bloque ASM, que describiremos más adelante); describe aquellas salidas que están activas, sólo si ciertas condiciones -definidas en términos de las entradas al sistema- son verdaderas y, siempre y cuando la máquina se encuentre en el estado al que está asociada. Así, la caja de salidas condicionales, contiene una lista de salidas condicionadas y como se dijo antes, siempre está asociada a un diamante de decisión. La trayectoria de entrada siempre proviene de un diamante de decisión, pero la trayectoria de salida puede dirigirse a otro diamante de decisión o a una caja de estado. Su símbolo se muestra en la Fig. III.4.

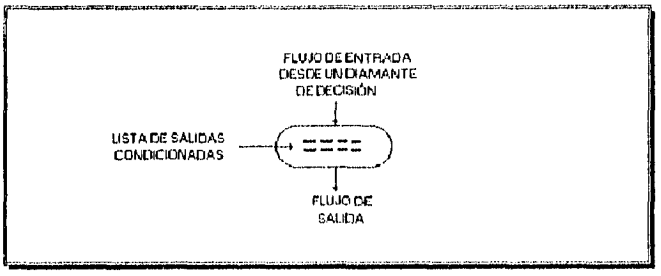


Fig.III.4.- Caja de Salida Condicional.

Nodo conector

Aunque no forma parte de los elementos básicos de una Carta ASM, se puede llegar a utilizar para hacer más claras las representaciones de entradas múltiples hacia un símbolo ASM. Como una convención particular podemos fijar en un nodo hasta 3 entradas múltiples pero sólo una salida.

III.1.4 Bloque ASM

Los elementos básicos (antes descritos), pueden ser combinados para formar un bloque ASM. Ésta estructura consiste de una caja de estado y la red de diamantes de decisión y cajas de salidas condicionales que puedan existir hasta antes del o de los estados siguientes. Según lo anterior, un bloque ASM puede estar formado con las siguientes combinaciones de elementos: un solo estado, un estado y una red de diamantes de decisión, o un estado y una red de diamantes de decisión combinada con salidas condicionales. Lo anterior es mostrado en la Fig. III.5.

Un bloque ASM tiene sólo una trayectoria de entrada y cualquier número de trayectorias de salida. Cada trayectoria de salida, debe dirigirse a otro estado, y así se convierte en la trayectoria de entrada de otro bloque o de sí mismo. Cada posible trayectoria de un estado a otro, determina una "Trayectoria de Enlace". Cada trayectoria de enlace corresponde a una parte de una expresión booleana, que en su totalidad define las salidas del bloque o la función de siguiente estado.

Dentro de un bloque ASM, la caja de estado es el único elemento que afecta al factor tiempo, todos los demás elementos son asumidos como actividades concurrentes. Así, todos los diamantes de decisión dentro del bloque, son evaluados simultáneamente sin importar su posición dentro de éste; las salidas listadas en las cajas de salidas condicionales que pertenecen al bloque, se activan simultáneamente, aunque no estén todas en la misma caja. Desde este punto de vista, las Cartas ASM difieren de un programa o de un diagrama de flujo. Una Carta ASM consiste de uno o más

bloques ASM interconectados de una manera consistente, que describe totalmente el comportamiento de la máquina de estados.

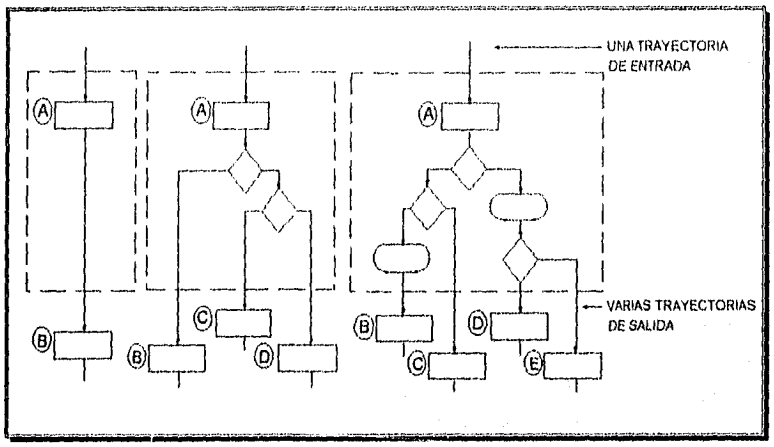


Fig. III.5.- Bloques ASM.

Las Cartas ASM contienen más información que los diagramas de estado de los tradicionales diseños de circuitos secuenciales, pues además de la secuencia del funcionamiento de la máquina de estados que ejecuta el algoritmo, muestra los componentes de las funciones requeridas en la síntesis de la carta. Para comparar lo anterior, se mostrará el diseño de un circuito para la detección de la siguiente secuencia -00 00 11 10- en dos variables de entrada. Se supondrá que, de alguna forma, se controla el número de datos introducidos, y que éstos serán sólo cuatro pares de valores (puesto que el diseño del circuito que controla el número de datos introducidos no es nuestro objetivo, éste no se mostrará). Cuando se detecte el final de esta secuencia, se tendrá un valor de "1" en la salida ISEQ del sistema, indicando la terminación de la secuencia válida.

La Fig. III.6 muestra los diagramas de estado de manera tradicional en el diseño de circuitos secuenciales.

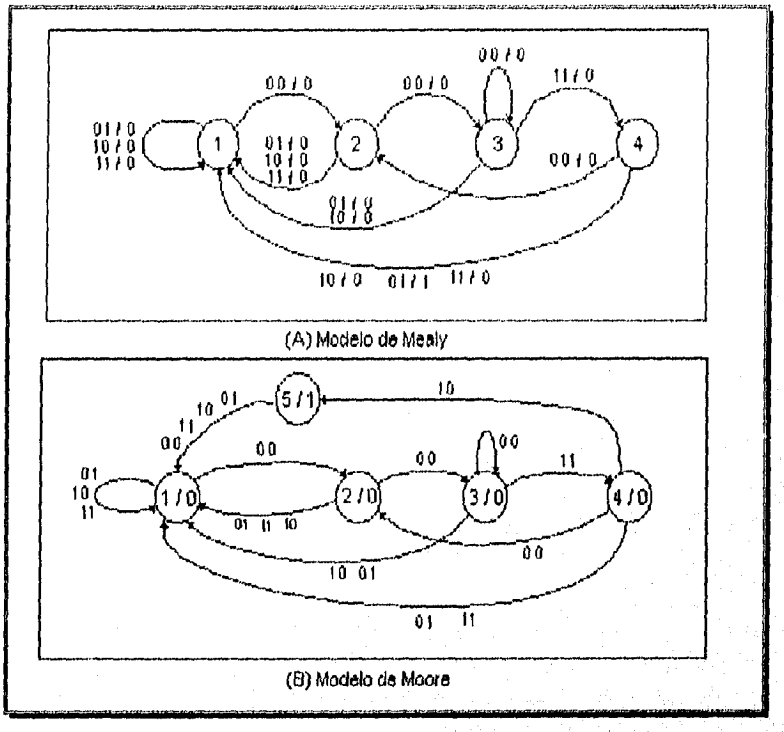


Fig. III.6. Diagramas de estado para un detector de secuencia.

La Fig. III.7, muestra el diseño con Cartas ASM para la detección de la misma secuencia, en ésta, se muestran los componentes X_1 y X_2 con los que se forman las funciones necesarias para la implementación. En la Fig. III.7a, las decisiones son hechas usando los elementos X_1 y X_2 como "entradas de control" del sistema. En la Fig. III.7b, se muestra una forma más compacta con menos diamantes de decisión pero con funciones de decisión más complejas. En cualquiera de los dos casos, se tiene una salida ISEQ (que indica que la secuencia correcta fue introducida) activa sólo en una de las trayectorias de enlace del estado D.

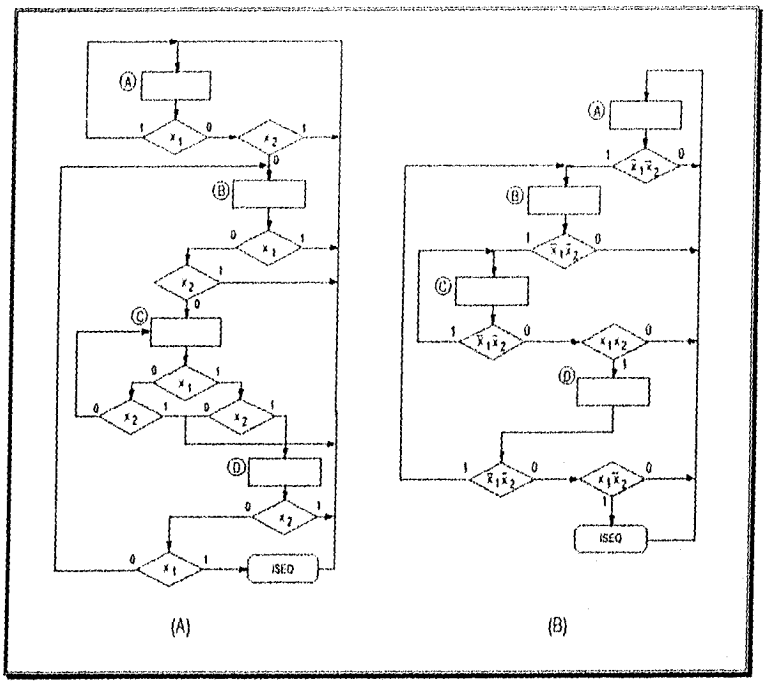


Fig. III.7. Cartas ASM para un detector de secuencia.

Como se ve en la Fig. III.6, existen dos modelos aceptados -con diferente estructura- en los diagramas de estado.

En la parte (A) de la figura, la representación del modelo de Mealy, las salidas son mostradas como funciones de las entradas y de las variables internas del estado. En la representación del modelo de Mealy, las trayectorias de transición están etiquetadas con la combinación de los valores de entrada que causan la transición, y la combinación de los valores de salida resultantes en la transición.

En la parte (B) de la figura, la representación del modelo de Moore, las salidas están asociadas con los estados particulares y así aparecen como etiquetas junto con el nombre del estado o número de nodo. La trayectoria de transición está etiquetada sólo con la combinación de entradas que da lugar a ésta. En este caso, se tiene un estado extra, porque se hace necesario pasar a él para activar la salida cuando la secuencia completa se ha detectado.

Como la descripción de estas representaciones nos desviaría del tema desarrollado, no se detallarán más.

III.2 Reglas de diseño

En el diseño de Cartas ASM se tiene una gran flexibilidad, que es lo que da la gran ventaja al método, pero para asegurar que el diseño sea correcto y por lo tanto se tenga un buen funcionamiento del sistema, debemos cuidar que el diseño de los bloques y de la carta en general, sea significativo y físicamente realizable; para esto debemos de cuidar algunos aspectos que podemos tomar como reglas para el diseño de tas cartas y que son los siguientes:

Debemos asegurarnos de que cada estado se dirige a un único próximo-estado, para cada conjunto de condiciones estables en la entrada.

Un error en este punto, comúnmente es debido a un mal arreglo de los diamantes de decisión, donde las condiciones para continuar son imposibles de cumplirse, la trayectoria de enlace es ficticia, o la transición de los estados es ambigua. En la Fig. III.8a y III.8b, la aparente trayectoria de enlace entre los estados A y B no es factible, ya que está requiere que la variable X tenga simultáneamente un valor de 0 y 1. De la misma forma, pero menos obvio, tenemos el arreglo de la Fig. III.8c que tiene una trayectoria de enlace desde el estado A al estado B que es imposible de realizarse porque la expresión que define la trayectoria de enlace siempre tendrá un resultado de "cero". Esto es: $X\bar{Y}(\bar{X}+Y) = X\bar{Y}\bar{X} + X\bar{Y}Y = 0$

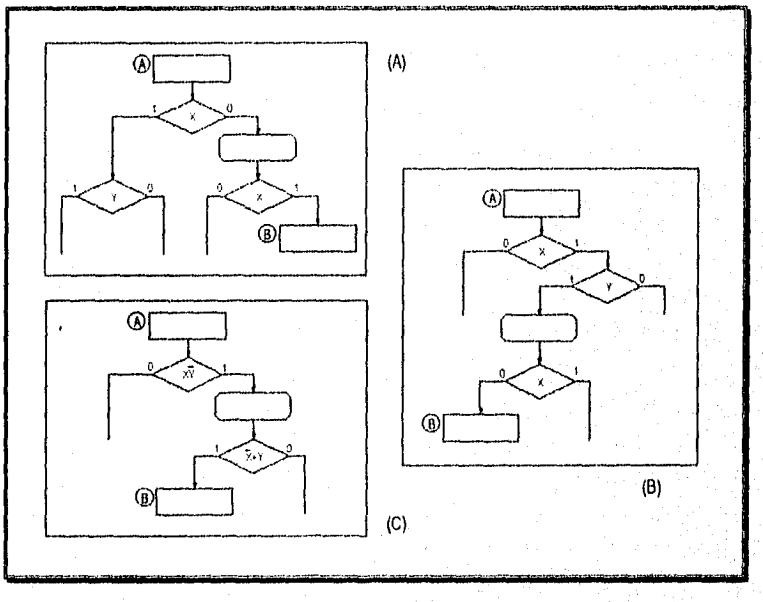


Fig. III.8. Estructuras de Cartas ASM con trayectorias de enlace imposibles.

La Fig. III.9 muestra algunos arreglos inválidos de elementos; la parte (A) de la figura, es claramente imposible de implementar ya que tiene, indicados con ambigüedad, dos próximos-estados simultáneos. La parte (B) de la figura es menos obvia; en ésta, si X y Y tienen un valor de 1, existe un próximo-estado único, sin embargo, si cualesquiera de X o Y tiene un valor de 0, se tendrán entonces indicados dos próximos-estados distintos, y no se sabrá a cuál de ellos se debe pasar.

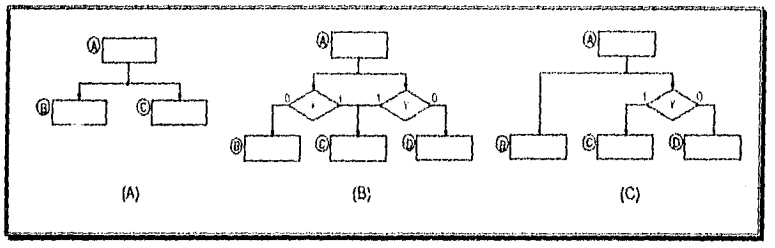


Fig. III.9. Algunas estructuras ASM inválidas.

Lo que probablemente se desea en la estructura (B), es mostrado en la Fig. III.10, donde la ambigüedad desaparece con un simple reacomodo de los diamantes de decisión.

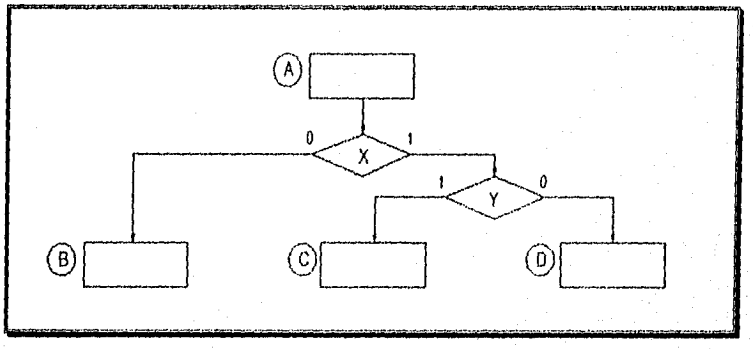


Fig. III.10. Estructura ASM correcta para la fig. III.9b.

Otro punto a cuidar, es que los ciclos que se forman deben tener siempre una última caja de estado, es decir, el bloque que forma el ciclo debe de tener al menos una trayectoria de salida para después reentrar a éste por la trayectoria normal de entrada, o dirigirse a otro bloque. El arreglo de la Fig. III.11a, muestra el caso inválido, y en la Fig. III.11b es redibujado en la forma correcta.

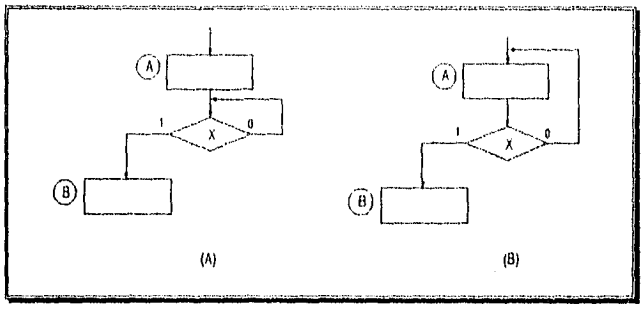


Fig. III.11. Representaciones (A) inválida y (B) válida de un ciclo.

Es posible tener la misma variable de prueba en distintos diamantes de decisión -dentro de un mismo bloque-, y tener una estructura o bloque ASM válido, siempre y cuando las trayectorias de enlace del estado actual estén definidas sin conflictos entre ellas; conflictos como dos posibles estados siguientes para una sola combinación de las variables de entrada, o que no exista un próximo estado definido para al menos una de las trayectorias de salida. En la Fig. III.12 se muestran dos ejemplos de estructuras válidas donde una variable aparece en más de un diamante de decisión.

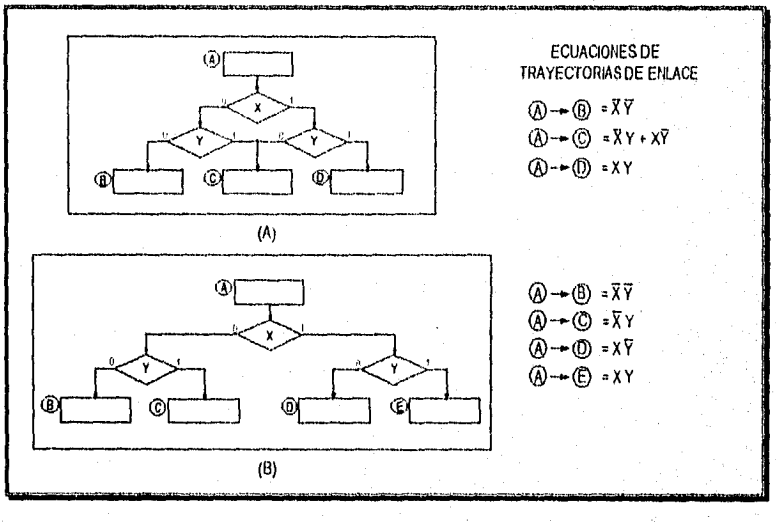


Fig. III.12. Algunas estructuras de Cartas ASM válidas.

Tenemos la libertad de utilizar en forma equivalente, series de diamantes de decisión interconectados en serie o en paralelo dentro de un bloque ASM. Como dentro de un bloque, todos sus elementos son evaluados simultáneamente, las decisiones y las salidas de las cajas de salida condicional, son evaluadas y aparecen -respectivamente- de manera simultánea, esto hace que las dos estructuras mostradas en la Fig. III.13 sean correctas y equivalentes.

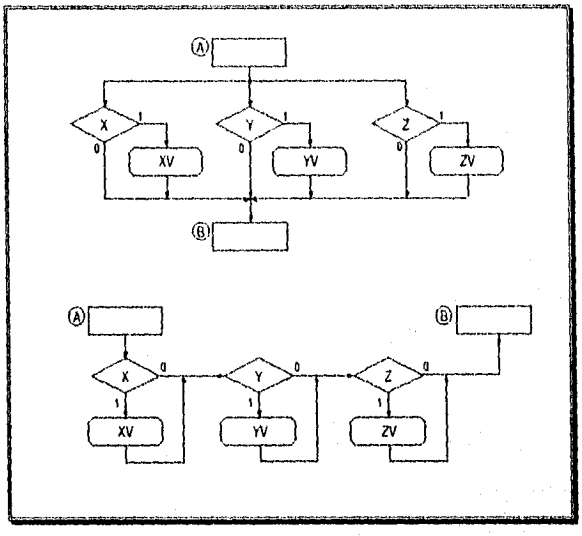


Fig. III.13. Estructuras de Cartas ASM válidas y equivalentes.

Con las estructuras en serie es menos probable que se presenten ambigüedades en las trayectorias de transición, pero con las estructuras en paralelo suelen construirse cartas más compactas.

En ocasiones es posible hacer Cartas ASM más compactas si los bloques comparten diamantes de decisión o cajas de salidas condicionales, evitando así duplicaciones. El hecho de que los bloques ASM puedan traslaparse no es problema, siempre y cuando sean identificables y sus trayectorias de enlace sean bien definidas. Como un ejemplo de esto, se puede considerar la Carta ASM de la Fig. III.14, en la cual, un diamante de decisión es compartido entre el bloque del estado B y el del estado C. El hecho de que esto pueda ocurrir, es gracias a que las condiciones de transición para cada caso, se presentan en instantes distintos.

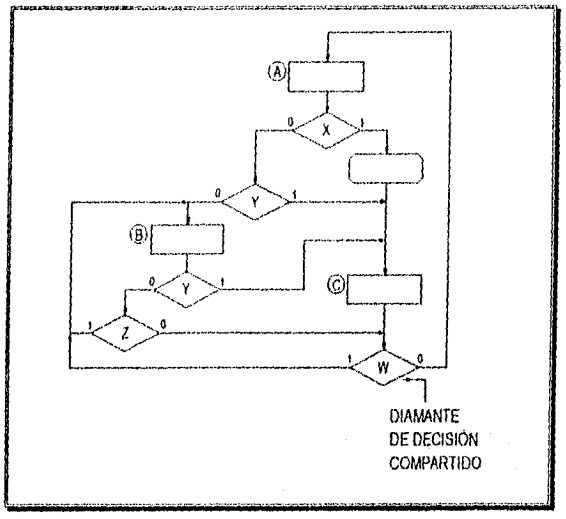


Fig. III.14. Ejemplo de Carta ASM.

En esta figura podemos fácilmente extraer los tres bloques ASM correspondientes a los tres estados y podemos también identificar sus trayectorias de enlace, como se muestra en la Fig. III.15. Con práctica esto puede lograrse sin redibujar los bloques.

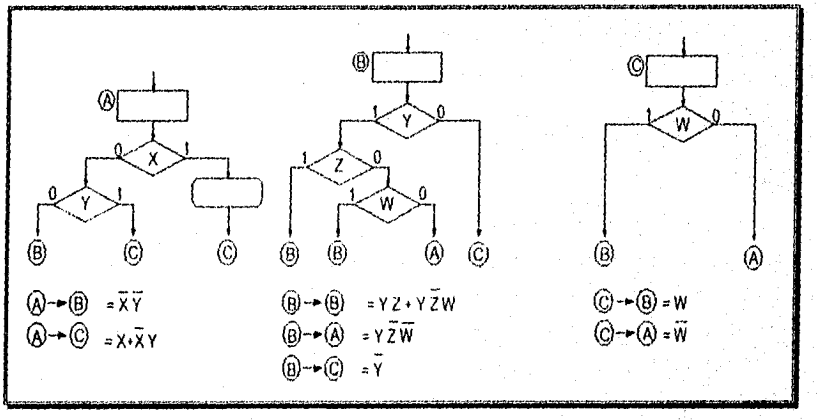


Fig. III.15. Bloques ASM y trayectorias de enlace de la carta de la Fig. III.14.

La conexión directa de un estado con una salida condicional no está permitida, puesto que las salidas condicionales están siempre asociadas a una de las trayectorías de salida de un diamante de decisión.

III.3 Ejemplo de aplicación

Para mostrar el método de diseño con Cartas ASM, se desarrollará un ejemplo paso a paso, siguiendo el algoritmo que da solución a un problema o requerimiento dado. En el siguiente procedimiento, se busca mostrar como se puede llevar a cabo la elaboración de una Carta ASM a partir de un problema dado.

Aunque el procedimiento de diseño puede variar dependiendo del diseñador, aquí se presentan los siguientes pasos a realizar para alcanzar nuestro objetivo:

1.- Planteamiento por escrito del problema (especificar requerimientos del problema).

2.- Planteamiento de un diagrama a bloques general del sistema (bloque de entradas, bloque controlador y bloque de salidas).

3.- Detallar en módulos el bloque controlador.

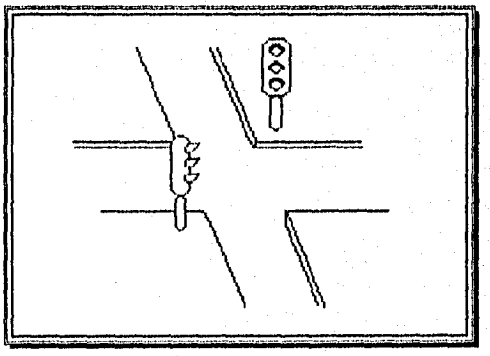
4.- Desarrollar un algoritmo de solución.

5.- Realizar la Carta ASM del controlador.

Solución:

1.- Planteamiento del problema.

El objetivo es crear la Carta ASM que describa el funcionamiento del circuito controlador de un semáforo ubicado en el cruce de dos avenidas, una con circulación de sur a norte (SN) y la otra con circulación de este a oeste (EO). El funcionamiento del semáforo tiene los siguientes requisitos:



En condiciones iniciales, tendremos: En la dirección sur-norte (SN) luz verde encendida, y en la dirección este-oeste (EO) luz roja.

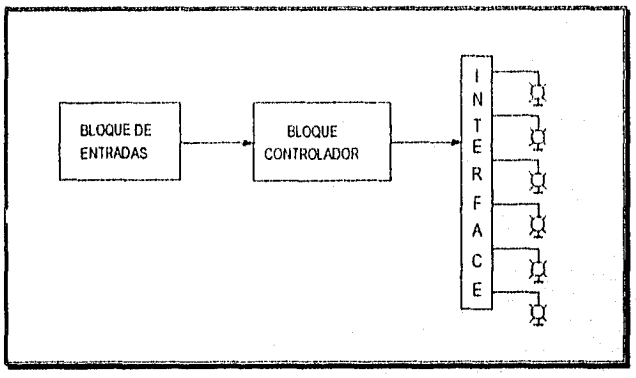
En la dirección SN las luces tendrán los siguientes tiempos de encendido: verde, 30 unidades de tiempo (UT); amarilla, 5 UT; roja, 30 UT.

Los tiempos de encendido para las luces de la dirección EO estarán de acuerdo a los de las luces de la dirección SN, considerando que la luz EO amarilla debe estar encendida por 5 UT mientras la dirección SN mantiene su luz roja.

Una vez cumplidos los tiempos anteriores, se volverá a condiciones iniciales para empezar de nuevo.

Las unidades de tiempo serán proporcionadas por un circuito temporizador.

2.- Diagrama a bloques general.



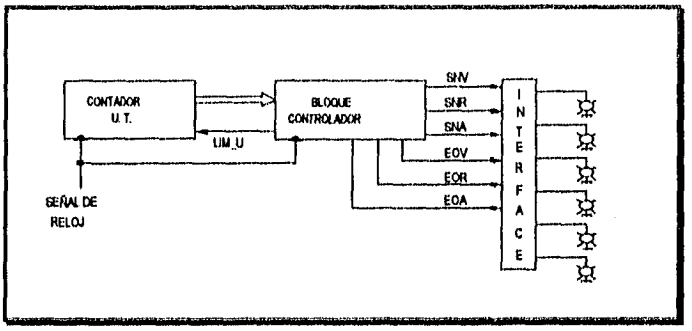
3.- Bloque controlador.

i. Del enunciado del problema, determinamos que el controlador sólo tendrá como entrada la señal proporcionada por el contador de unidades de tiempo, ya que deberá funcionar automáticamente una vez que se habilite, generando internamente las señales necesarias para su funcionamiento y para la inicialización del contador de unidades de tiempo.

ii. Se tendrá un contador de unidades de tiempo para determinar los periodos de tiempo requeridos en cada salida.

iii. Se contará con una interface para traducir las señales del controlador y activar las luces del semáforo.

Del análisis anterior, se obtiene el diagrama a bloques más detallado.



4.- Desarrollo del algoritmo de solución.

En un principio, se proporcionará el algoritmo en su totalidad para mostrar un panorama completo de los requerimientos del sistema a implementar, después se seguirá paso a paso para ir construyendo la Carta ASM y finalmente se mostrará ésta en forma completa.

⇒ El semáforo se encuentra en el cruce de dos avenidas, una con circulación de sur a norte (SN) y la otra con circulación de este a oeste (EO).

⇒ Inicialmente, la dirección SN tendrá luz verde y por lo tanto la dirección EO tendrá luz roja.

⇒ Estas condiciones permanecerán fijas durante 30 unidades de tiempo o ciclos de reloj.

⇒ Después de esto, la dirección SN pasará a luz amarilla durante 5 unidades de tiempo, y la dirección EO mantendrá su luz roja.

⇒ A continuación, la dirección SN mostrará luz roja y la dirección EO luz verde, esto durante 25 unidades de tiempo.

⇒ Una vez cumplido lo anterior, la dirección EO tendrá luz amarilla durante 5 unidades de tiempo, mientras que la dirección SN mantiene su luz roja.

⇒ Cuando los pasos anteriores se cumplan, el semáforo pasará a las condiciones iniciales, es decir, dirección SN en verde y dirección EO en rojo, para continuar con el ciclo descrito.

5.- Realización de la carta ASM.

Lo anterior describe el funcionamiento completo requerido para el semáforo, a continuación seguiremos el algoritmo paso a paso e iremos construyendo la Carta ASM que describe el circuito controlador.

⇒ El semáforo se encuentra en el cruce de dos avenidas, una con circulación de sur a norte (SN) y la otra con circulación de este a oeste (EO).

En base a este párrafo, se determina que deben existir al menos las siguientes salidas:

SNV.- Luz verde para la dirección SN.

SNA.- Luz amarilla para la dirección SN.

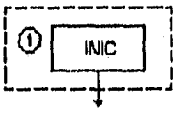
SNR.- Luz roja para la dirección SN.

EOV.- Luz verde para la dirección EO.

EOA.- Luz amarilla para la dirección EO.

EOR.- Luz roja para la dirección EO.

Una medida sana para asegurar que inicialmente todos los dispositivos (contadores, registros, elementos biestables, etc.) estén limpios de información que pudiera alterar el funcionamiento del circuito, es que la Carta ASM contenga como primer estado, uno donde exista una señal de salida (INIC) para inicializar o mandar a cero todo. Esto nos lleva a que generalmente, las Cartas ASM tengan como primer bloque el siguiente:



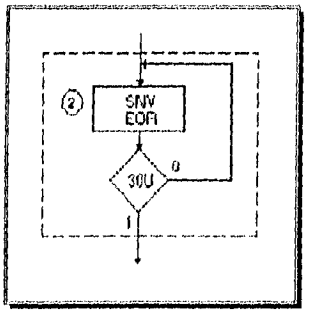
Ahora analicemos los dos siguientes pasos del algoritmo dado.

⇒ Inicialmente, la dirección SN tendrá luz verde y por lo tanto la dirección EO tendrá luz roja.

⇒ Estas condiciones permanecerán fijas durante 30 unidades de tiempo o ciclos de reloj.

Esto nos lleva a tener un estado donde existan las salidas SNV y EOR; como esto debe permanecer durante 30 unidades de tiempo, tenemos que verificar la entrada 30U que aparece cuando un contador de unidades de tiempo llegue a 30 (este contador se inicializa en cero con la señal de salida del primer estado), cuando esto ocurra seguiremos con el próximo paso del algoritmo.

Lo expresado en el párrafo anterior nos lleva a un bloque ASM como el siguiente:

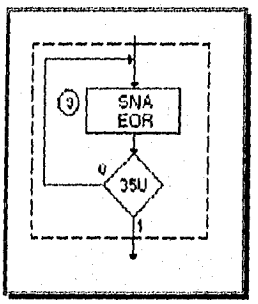


En este bloque, la trayectoria de entrada viene de la trayectoria de salida del estado anterior.

Continuando con el análisis del algoritmo tenemos:

⇒ Después de esto, la dirección SN pasará a luz amarilla durante 5 unidades de tiempo, y la dirección EO mantendrá su luz roja.

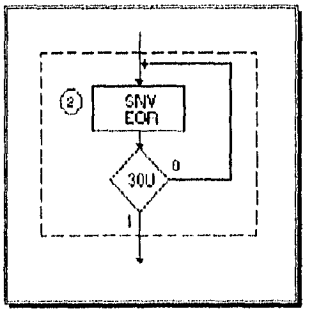
Esto lo realizaremos con un estado que contenga las salidas SNA y EOR; como se especifica que esto debe permanecer por 5 unidades de tiempo, la siguiente entrada a verificar debe activarse o presentarse cuando el contador de unidades de tiempo llegue a 35, esto nos dará las 5 unidades de tiempo requeridas, sin tener que inicializar de nueva cuenta el contador. El siguiente bloque ASM realiza lo anterior.



Continuando con el siguiente paso del algoritmo, tenemos:

⇒ A continuación, la dirección SN mostrará luz roja y la dirección EO luz verde, esto durante 25 unidades de tiempo.

Lo expresado en el párrafo anterior nos lleva a un bloque ASM como el siguiente:

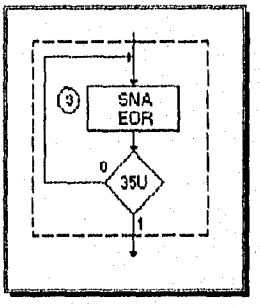


En este bloque, la trayectoria de entrada viene de la trayectoria de salida del estado anterior.

Continuando con el análisis del algoritmo tenemos:

⇒ Después de esto, la dirección SN pasará a luz amarilla durante 5 unidades de tiempo, y la dirección EO mantendrá su luz roja.

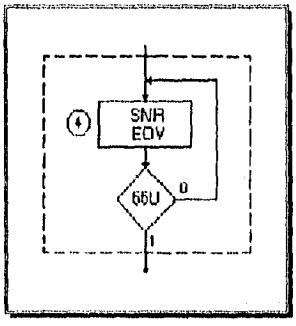
Esto lo realizaremos con un estado que contenga las salidas SNA y EOR; como se especifica que esto debe permanecer por 5 unidades de tiempo, la siguiente entrada a verificar debe activarse o presentarse cuando el contador de unidades de tiempo llegue a 35, esto nos dará las 5 unidades de tiempo requeridas, sin tener que inicializar de nueva cuenta el contador. El siguiente bloque ASM realiza lo anterior.



Continuando con el siguiente paso del algoritmo, tenemos:

⇒ A continuación, la dirección SN mostrará luz roja y la dirección EO luz verde, esto durante 25 unidades de tiempo.

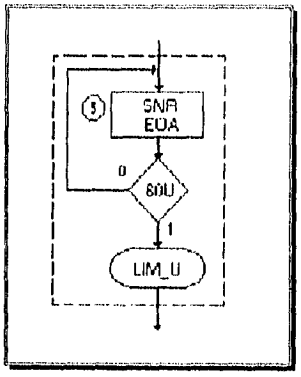
Como la dirección SN debe mostrar luz roja y la dirección EO luz verde, entonces, en el siguiente estado deben existir las salidas SNR y EOV y estar activas durante 25 unidades de tiempo; para conseguir esto, dejaremos que el contador de unidades de tiempo llegue a 55 unidades y no a los 60 que podría esperarse, esto previendo que la luz roja de la dirección SN debe permanecer encendida aún cuando cambie la luz -de verde a amarillo- en la dirección EO; las 5 unidades de tiempo faltantes para completar el tiempo de la salida SNR, se presentarán en el siguiente bloque. El bloque que se muestra a continuación representa las anteriores condiciones:



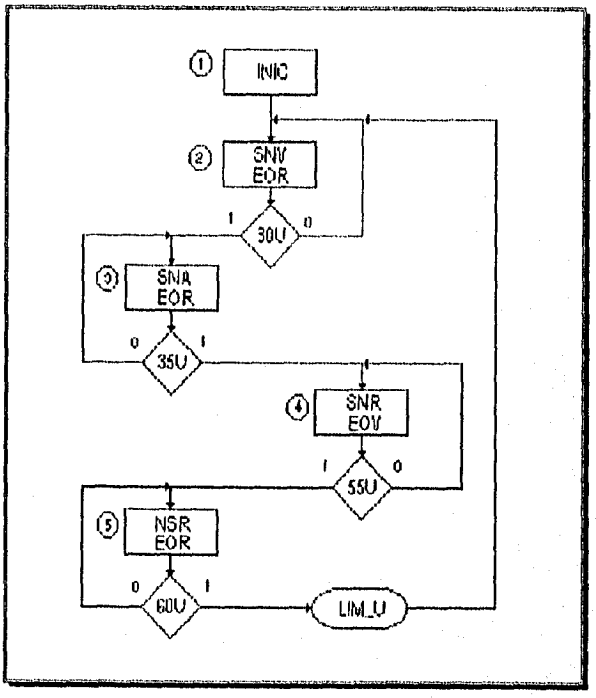
Por último analizaremos los dos últimos pasos del algoritmo:

- ⇒ Una vez cumplido lo anterior, la dirección EO tendrá luz amarilla durante 5 unidades de tiempo, mientras que la dirección SN mantiene su luz roja.
- ⇒ Cuando los pasos anteriores se cumplan, las condiciones del semáforo pasarán a las condiciones iniciales, es decir, dirección SN en verde y dirección EO en rojo, para volver a iniciar el ciclo descrito.

El bloque ASM que sirve para describir lo anterior tendrá un estado donde se activaran las salidas SNR y EOA durante 5 unidades de tiempo, con esto se cumplirá el requerimiento para la luz amarilla en la dirección EO y se completará la duración de la luz roja de la dirección SN. Dentro de este bloque incluiremos también una caja de salidas condicionales donde se activará la salida LIM_U -sólo cuando la cuenta llegue a 60- que servirá para volver el contador de unidades de tiempo a cero ya que con esto podemos reiniciar el algoritmo bajo las condiciones iniciales, por lo tanto, en este instante, la trayectoria de salida de este bloque será la trayectoria de entrada al bloque del estado 2. Hacemos que la trayectoria de salida corresponda a la trayectoria de entrada del estado 2 y no a la del estado 1, porque en éste se tiene una señal de salida para limpiar o inicializar todos los elementos del sistema y para lo que buscamos sólo es necesario inicializar el contador de unidades de tiempo.



Al unir los bloques anteriores, obtenemos la Carta ASM completa.



Lista de variables:

INIC.- Salida para inicializar o mandar a cero todos los dispositivos del sistema.

SNV.- Salida para habilitar la luz verde de la dirección sur-norte.

SNR.- Salida para habilitar la luz roja de la dirección sur-norte.

SNA.- Salida para habilitar la luz amarilla de la dirección sur-norte.

EOV.- Salida para habilitar la luz verde de la dirección este-oeste.

EOR.- Salida para habilitar la luz roja de la dirección este-oeste.

EOA.- Salida para habilitar la luz amarilla de la dirección este-oeste.

30U.- Señal interna que indica que el contador de unidades de tiempo está en 30.

35U.- Señal interna que indica que el contador de unidades de tiempo está en 35.

55U.- Señal interna que indica que el contador de unidades de tiempo está en 55.

60U.- Señal interna que indica que el contador de unidades de tiempo está en 60.

LIM_U.- Salida activa cuando el contador de unidades de tiempo llegue a 60, utilizada para regresar el contador a cero y volver a condiciones iniciales.

CAPÍTULO IV

DESARROLLO DE LA APLICACIÓN

CAPÍTULO IV

DESARROLLO DE LA APLICACIÓN

Dentro del panorama de la OO, al usar cualquiera de las metodologías antes mencionadas, podemos sentir cierto grado de libertad en varios de los puntos del análisis o del diseño que ponen de manifiesto una realidad dentro de las TOO: aún no existe un consenso generalizado sobre cuál es la técnica o combinación de ellas que represente significativamente al enfoque de objetos. Y es que ciertamente, la TOO es una disciplina nueva que demanda aún mucho tiempo de investigación y estandarización, sin embargo, existen autores que de manera entusiasta sugieren un camino a seguir, una lista de proposiciones que a juicio de ellos mismos es útil en ciertos tipos de problemas pero que de ninguna manera son definitivos.

Por todo lo anterior, el desarrollo de un sistema como el que nos ocupa enfrenta al equipo de desarrollo al trabajo de decidir qué métodos de análisis y diseño usar y cómo deben interactuar entre sí. Tomando en cuenta que cada una de las técnicas propuestas ha sido formulada de acuerdo a un contexto de problemas específicos a los que sus autores se han enfrentado, puede considerarse adecuada la construcción de un método a la medida de la complejidad de nuestro problema, tomando elementos técnicos que ya han sido probados.

En 1990 en su libro "Ingeniería del Software", Pressman propone un proceso de DOO que reúne en sí mismo varias técnicas en un afán por conducir al desarrollo de sistemas por un camino sencillo y práctico de lo que es la TOO. El método propuesto por Pressman como DOO, se presenta de forma concisa en los siguientes pasos:

1. **Definir el problema.**
2. **Desarrollar una estrategia informal para la realización del software del dominio del problema en el mundo real.**
3. **Formalizar la estrategia usando los siguientes subpasos:**

    A. **Identificar objetos.**
    B. **Identificar las operaciones y atributos que pueden aplicarse a los objetos.**
    C. **Establecer interfaces para mostrar las relaciones entre los objetos y las operaciones.**
    D. **Decidir los aspectos del diseño detallado que harán una descripción de los objetos.**

4. **Repetir los pasos 2, 3 y 4 recursivamente hasta que se cree un diseño completo.**

Si sólo obedeciéramos al conjunto de pasos que Pressman expone, pasaríamos por alto el hecho de que se han reunido ahí al análisis y al diseño en una forma que, como ya hemos mencionado en el Capítulo de "Tópicos de Orientación a Objetos", se vuelven indistinguibles. Sin embargo, para delimitar cada una de las etapas, observemos como los pasos 1 y 2 corresponden al método propuesto por Yourdon para el análisis; y notemos además cómo la técnica de Yourdon continúa su desarrollo y se sobrepone implícitamente a algunas de las etapas que componen al paso 3.

El DOO de Booch está claramente circunscrito en el propio paso 3, donde sus cuatro proposiciones son enunciadas exactamente como una serie de subpasos. En el paso 4, vale la pena hacer alto en la propuesta *recursiva* de los pasos precedentes. La recursividad en el método debe entenderse como la aplicación de las proposiciones 2 y 3 al producto de esos mismos pasos, de manera que, si partimos de un análisis inicial, acatando los pasos 1, 2 y 3 tendremos como resultado un conjunto de entidades identificables, que en la primera recursión serán una a una sometidas a los pasos 2 y 3, que a su vez nos proporcionarán más elementos para el análisis y así sucesivamente hasta agotar los elementos obtenidos o llegar al grado de detalle deseado.

El equipo de desarrollo consideró que la adopción de este método implica ventajas que harán decididamente de éste nuestro método de trabajo. Entre las ventajas podemos mencionar las siguientes:

- Fácil de interpretar.
- Sencillo en su uso.
- Emplea herramientas estándar a la mayoría de los métodos.
- Hace énfasis en la identificación intuitiva de objetos, sin descartar conceptos útiles de las metodologías estructuradas.
- Flexible en su implantación y conciso en su definición.

De hecho, un método basado en la metodología de Booch, necesariamente cumple con la mayoría de los preceptos de OO -recuerde que fue uno de los pioneros en el área- y si además se cuenta con el respaldo de un metodologista como lo es Yourdon, no hacemos más que obedecer a una elección obvia. Por otra parte, puede no parecer tan claro que este método es adecuado para sistemas de tamaño moderado, puesto que el control de las características pertenecientes a los elementos tiende a dificultarse con el crecimiento o la expansión, haciéndose necesario un método más riguroso.

IV.1 Definición del problema

Puesto que no puede existir análisis sin problema a analizar, es requisito indispensable la existencia de uno que debe estar, dicho sea de paso, bien definido. Al hacerlo, estamos asentando el **requerimiento** que describe a algunos de los elementos dentro del universo del problema, el conocimiento intuitivo del ambiente del caso planteado completará el cuadro de manera que sea comprensible y el entendimiento del problema sea mejor. La generación de una solución al problema depende en gran medida de la exactitud de la **definición del problema**, que nos permitirá identificar al **dominio de solución** en base al requerimiento y el reconocimiento de los elementos de este dominio de solución que se relacionarán entre sí en una **definición de la solución**. Para fines prácticos debemos tomar en cuenta que la definición de la solución en software debe establecerse de modo sencillo y gramaticalmente correcto, para unificar el problema a un alto nivel de abstracción.

La primera tarea para la definición del problema en DOO es escribir una declaración en una sentencia del problema. Esto, por ejemplo, lo podemos hacer con la ayuda de un Diagrama a Bloques del sistema o de un Diagrama de Flujo de Datos a un primer nivel (DFD).

La Fig. IV.1 nos ayuda a obtener el siguiente enunciado:

Paso 1. Definir el problema: "LA AUSENCIA DE UN SISTEMA DE SOFTWARE QUE AUXILIE A LOS ALUMNOS DE INGENIERÍA EN EL PROCESO DE COMPRENSIÓN-ENSEÑANZA DEL MÉTODO DE DISEÑO DE CONTROLADORES DIGITALES DENOMINADO CARTAS ASM".

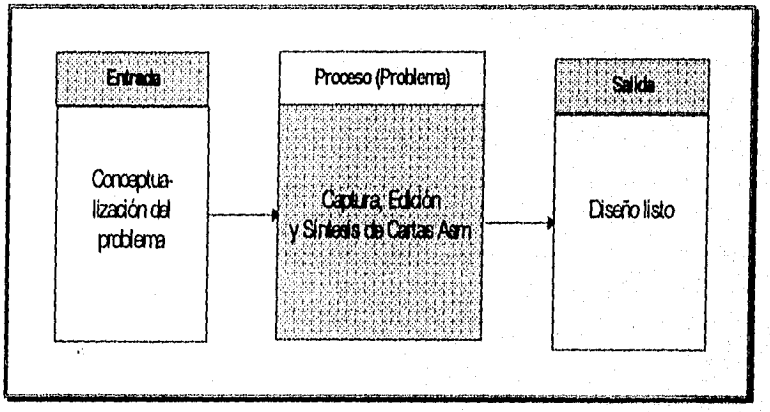


Fig. IV.1.- Diagrama a bloques del problema.

IV.2 Una estrategia informal

El siguiente paso es escribir una estrategia informal para la solución del problema establecido en la definición. La estrategia es una descripción en lenguaje natural de la solución del problema que hay que resolver mediante software, representado a un nivel consistente de detalle. Como regla general, la estrategia informal tiene las siguientes características:

1. Se escribe como un párrafo sencillo y claro.
2. Se escribe al mismo nivel de abstracción, esto es, el nivel de detalle debe ser consistente a lo largo de todo el párrafo.
3. Está enfocado sobre lo que debe hacerse para resolver el problema, en vez de en los detalles procedimentales de cómo se obtiene la solución, ni de cómo se descomponen los bloques o cómo se procesan y codifican los elementos de control.
4. El párrafo debe intentar repetir los mismos términos para describir la información y el procesamiento en vez de usar sinónimos.
5. Se describe la estrategia informal usando una terminología a nivel de sistema, en vez de un vocabulario típico de software.

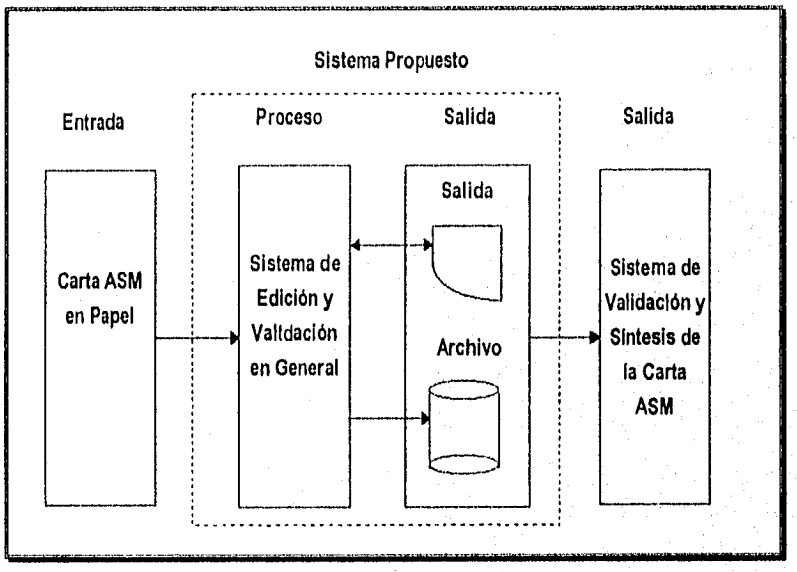


Fig. IV.2.- Detallado del diagrama a bloques del sistema.

De acuerdo a la estrategia informal y con la ayuda de la Fig. IV.2, la solución al enunciado del problema quedaría representada de la siguiente forma:

Paso 2: "DISEÑO Y CONSTRUCCIÓN DE UN ENTORNO GRÁFICO DE SOFTWARE QUE PERMITA LA EDICIÓN DE UNA CARTA ASM CONCEPTUALIZADA POR EL USUARIO".

La función primordial del sistema a implementar, consiste en el procesamiento de la información derivada de una Carta ASM diseñada previamente en papel (ENTRADA), a fin de generar unos archivos que describan correctamente la estructura de dicha carta (SISTEMA PROPUESTO: PROCESO Y SALIDA), para que otro sistema (SALIDA: el complemento del proyecto propuesto) obtenga la solución al problema por medio de una técnica aplicable a los controladores digitales.

Es importante mencionar que pueden existir dos clases de enunciados de acuerdo a su intención, por una parte los que explican al objeto mencionando todo aquello que hace, y por otra parte los que especifican un objeto al decir de qué se compone y por qué. No es extraño encontrarse con algún enunciado específico-explícito o también enunciados complejos compuestos por más de una frase.

IV. 3 Formalización de la estrategia

Es en este punto donde comienza realmente el DOO. El objetivo es aislar e identificar los objetos, operaciones y sus interrelaciones, de forma que pueda derivarse un diseño.

IV.3.1 Identificación de objetos

La identificación de los objetos es el corazón del DOO. Abbott ofrece una valiosa discusión al respecto:

> *La OO enfatiza la importancia de la identificación precisa de los objetos y sus propiedades. Sin esta identificación cuidadosa, es casi imposible ser precisos en las mismas operaciones que han de ejecutarse y en los efectos de las mismas.*
> *Los nombres y frases nominales[1] de la estrategia informal son buenos indicadores de los objetos y sus clasificaciones (es decir, tipos de datos) en nuestra solución del problema.*

Esta etapa consiste en reconocer los sustantivos que componen al enunciado de la etapa anterior, que son a su vez clasificados como:

[1] Aquellas frases compuestas por un sustantivo y un adjetivo que no pueden ser separadas sin ofrecer un significado ambiguo.

Sustantivo	Significado	
	Contexto de la solución	Contexto del Diseño
Nombre común.	Nombre de una clase de seres o cosas, por ejemplo "vehículo".	Una clase de objeto o abstracción de datos (clase "vehículo").
Nombre propio.	Nombre de seres o cosas específicas, por ejemplo bicicleta o auto.	Una instancia de clase (de la clase "vehículo").
Nombre masivo o abstracto.	Es el nombre de una cantidad o una medida, por ejemplo "tránsito".	Agrupación jerárquica de las clases de la solución.

Es importante tener en cuenta que contexto y semántica deben usarse para determinar las categorías nominales mostradas en el cuadro anterior. Una palabra puede ser un nombre común en un contexto, un nombre propio en otro y, en algunos casos, un nombre de masa o abstracto en un tercer contexto.

Después de que se han identificado todos los nombres y frases nominales, puede completarse una tabla de objetos. La tabla contiene cada nombre como un objeto; identifica si el objeto cae dentro del espacio del problema (los objetos que son parte del problema) o del espacio de solución (objetos que definen la solución del problema). Lo que puede ser mostrado en una tabla como la siguiente:

Objeto	Espacio	Comentario

Es importante observar que no todos los nombres y frases nominales serán de interés en la realización final de la solución. Algunos objetos, como ya hemos dicho, caerán fuera de los límites del espacio de solución. Otros objetos, aunque relevantes al problema, pueden ser redundantes o extraños cuando se refina la solución. Construyendo una tabla de objetos, se establecen los objetos potenciales y se determina el conjunto final de objetos.

Continuando con el desarrollo de la solución al problema, tenemos:

"DISEÑO Y CONSTRUCCIÓN DE UN ENTORNO GRÁFICO DE SOFTWARE (EGS) QUE PERMITA LA EDICIÓN DE UNA CARTA ASM CONCEPTUALIZADA POR EL USUARIO".

Tabla de objetos		
Objeto	Espacio	Comentario
EGS	Solución	
Carta ASM	Solución	
Usuario	Problema	

IV.3.2 Operaciones y atributos aplicados a los objetos

Una vez que se han identificado los objetos del espacio de solución, se selecciona el conjunto de operaciones que actúan sobre los mismos. Las operaciones se identifican examinando todos los verbos establecidos en la estrategia informal. Los verbos significan acciones u ocurrencias que en el contexto de la normalización del DOO, se consideran conjuntamente con las frases verbales[2] descriptivas y los predicados como operaciones potenciales. "Leer", "calcular", "es menor que", "determinar la diferencia entre", "es igual a", y "mantener", deberán ser todas operaciones candidatas.

Hay muchos casos en los que una operación se refiere a dos o más objetos. ¿A cuál de estos objetos debe asignarse la operación?. El Object Oriented Design Handbook presenta algunos criterios para completar este subpaso y estaolecer cada operación potencial:

- ✓ *Si sólo es necesario un objeto para que ocurra una operación, entonces ese es el objeto sobre el que tiene efecto la operación.*
- ✓ *Si una operación requiere el conocimiento de más de dos tipos (objetos), entonces la operación no es fundamentalmente coherente y debe rechazarse como parte de la estrategia informal.*
- ✓ *Si se requieren dos o más objetos para que ocurra una operación, entonces se debe determinar de que objeto es menester conocer la estructura interna (parte privada) por la operación.*

Aplicando estos criterios a las operaciones, derivaremos una relación entre los objetos ya identificados y sus operaciones, donde cada una de éstas se asigne a un único objeto y aquellos objetos y operaciones que existen en el espacio del problema sean desechados. Además, los objetos redundantes, extraños o abstractos (p. ej., un candidato a objeto, como el objeto "sistema" que abarca a todos los demás objetos y sus operaciones, y no añade nada a la solución del diseño) deben desecharse también; se combinarán las operaciones sinónimas y los nombres de las operaciones se extenderán para hacer más descriptivas las funciones de procesamiento.

---

[2] Frases verbales: Aquellas compuestas por un verbo y un calificador y que no pueden desprenderse de cualquiera de ellos sin perder su significado.

Todos los objetos deben tener al menos una operación que actúa sobre él. Sin embargo, frecuentemente existen casos en los que un objeto parece que está solo; esto es, aparentemente no existe operación alguna que requiera información sobre la construcción subyacente del objeto. Alternativamente, se puede encontrar una operación que parezca que no se aplica a ningún objeto de la tabla. Esta situación puede ocurrir por distintas razones:

1) La estrategia informal es incompleta y una operación u objeto importante ha sido omitido.
2) Un objeto u operación que pertenece al espacio del problema ha sido asignado al espacio de soluciones (o viceversa).
3) Una operación mostrada en la tabla de operaciones de los objetos, requiere conocimiento del objeto que "permanece solo", pero esta relación no ha sido reconocida.
4) La estrategia informal se ha escrito a diferentes niveles de abstracción; esto es, se especifican los objetos u operaciones de bajo nivel, pero sólo se tratan las operaciones u objetos de alto nivel.

En resumen, es crucialmente importante escribir, editar, revisar y reeditar la estrategia informal dada, así como el tratar a los objetos y operaciones a un nivel consistente de abstracción.

Frecuentemente es necesario asociar atributos a los objetos y operaciones. Un atributo nos ayuda a comprender las características del objeto u operación. Los **adjetivos** y **adverbios** se convierten en atributos de los objetos y operaciones, respectivamente. Cada uno de los atributos debe ser definido completamente. Sin embargo, la definición puede posponer su refinamiento hasta que se restablezca la estrategia a un nivel más bajo de abstracción.

En algunos casos, los atributos de los objetos pueden combinarse directamente en el nombre del objeto. Esto ocurre sóto cuando se establece la estructura informal a un nivel relativamente alto de abstracción. Conforme se refinan estos objetos y sus respectivas operaciones, puede que los atributos haya que desmontarlos del objeto y especificarlos en detalle. A continuación se presenta el formato para este refinamiento:

Objeto	Operación	Espacio	Atributo del Objeto	Atributo de la operación	Comentarios

Retomando el enunciado de la solución del problema, señalemos con cursivas la presencia de verbos, frases verbales, adjetivos y adverbios:

"*DISEÑO Y CONSTRUCCIÓN* DE UN ENTORNO GRÁFICO DE SOFTWARE (EGS) QUE *PERMITA LA EDICIÓN* DE UNA CARTAS ASM *CONCEPTUALIZADA* POR EL USUARIO".

Observemos que Cartas ASM no presenta una operación asignada a él, lo que se explica analizando los diagramas IV.1 y IV.2, así como su participación en la frase "permita la edición de Cartas ASM" y en "Carta ASM conceptualizada por el Usuario". En realidad este sustantivo forma parte del problema, lo que quedaría más claro si lo reemplazamos por "conceptualización de una Carta ASM".

A continuación presentamos la actualización de la tabla junto con las operaciones relacionadas con los objetos:

Objeto	Operación	Espacio	Atributo Objeto	Atributo Oper.	Comentarios
--------	Diseñar, Construir	Problema			
EGS	Permitir edición	Solución			
Carta ASM		Problema			
Usuario	Conceptualizar	Problema			

IV.4 Componentes e interfaces del programa

Un aspecto importante de la calidad del diseño de software es la modularidad, esto es, la especificación de las componentes (módulos) que se combinan para formar un sistema completo. El DOO define al objeto como una componente del sistema que se enlaza con otras componentes. Pero la definición de objetos y operaciones no es suficiente. Debemos también identificar las interfaces que existen entre los objetos y la estructura global de éstos (considerada en un sentido arquitectónico).

Aunque una componente de programa es una abstracción de diseño, debe ser representada en el contexto del lenguaje de programación con el que se implementará el diseño. Para acomodar el DOO, el Lenguaje de Diseño de Programa (LDP) que vaya a usarse para la construcción debe ser capaz de crear algo como lo siguiente:

```
PAQUETE nombre-componente-sistema ES
    TIPO especificación de objetos de datos
        ...
    PROC especificación de operaciones relacionadas
        ...
PRIVADO
    detalles de estructuras de datos para los objetos
CUERPO DE PAQUETE nombre-componente-sistema ES
    PROC operación.1 (descripción de interface) ES
        ...
    FIN
    PROC operación.n (descripción de interface) ES
        ...
    FIN
FIN nombre-componente-sistema
```

Refiriéndonos al LDP mostrado anteriormente, se especifica una componente del sistema indicando sus datos y operaciones. La parte de especificación de la componente indica todos los datos (declarados con la sentencia TIPO) y las operaciones (PROC para procedimiento) que actúan sobre ellos. La parte privada (PRIVADO) de la componente suministra otros detalles ocultos de las estructuras y procesamiento de los datos. En el contexto de la discusión anterior, el PAQUETE es conceptualmente similar a un objeto.

La primera componente del sistema que hay que identificar, debe ser el módulo de más alto nivel desde el que se originan todos los procesamientos y todas las estructuras de datos. Debe observarse que no se debe hacer ningún intento para especificar la información de la parte privada de los objetos de datos. Los detalles de construcción se considerarán más tarde.

Una vez que se han identificado las componentes del sistema, se está preparado para examinar el diseño subsecuente y establecer críticamente la necesidad de los cambios. Frecuentemente, una revisión de la definición de "primer corte" de los paquetes dará como resultado modificaciones, que añaden o devuelven nuevos objetos de datos a los primeros pasos del DOO (es decir, la estrategia informal), para establecer la entereza de las operaciones que se hayan especificado.

La interface de la componente deberá reconocerse como aquellas operaciones que estarán a disposición de otras componentes para accesar a su parte privada o pública, ya sea para consultar su estado actual o para requerir una operación.

IV.5 Representaciones gráficas para el DOO

Las componentes del sistema pueden representarse gráficamente para ayudar a establecer las conexiones de la interface y dar una forma sencilla de reconocer la representación del diseño. Booch propone una notación, llamada algunas veces diagramas de Booch (Fig. IV.3), para las componentes del sistema.

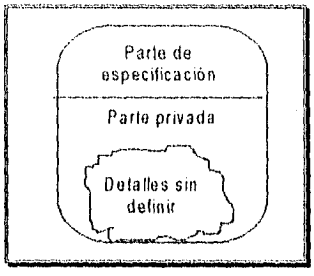


Fig. IV.3. Representación gráfica de una componente del sistema.

En la Fig. IV.3, una componente del sistema (objeto) se representa por una caja que puede ser dividida en una parte de especificación (visible al exterior) y una parte privada que permanece oculta al exterior. La "nube" sin forma representa los detalles de la parte privada o cuerpo del objeto sin especificar.

Una forma más específica la representamos como se muestra en la Fig. IV.4. Los objetos de datos se representan como rectángulos redondeados, mientras que las operaciones que actúan sobre los objetos se representan mediante rectángulos normales. De nuevo, la forma de "nube" se utiliza para representar los detalles de implantación actualmente indefinidos.

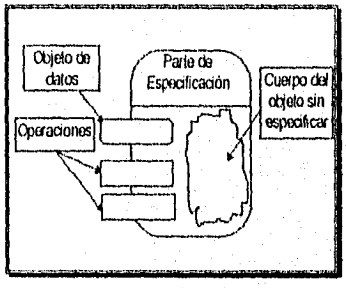


Fig. IV.4. Notación de paquete (objeto).

La Fig. IV.5 ilustra el uso del diagrama de Booch para representar las dependencias entre componentes del sistema. Las flechas de conexión indican dependencia, esto es, el paquete o componente en la punta de flecha, depende del paquete o componente de programa en el origen de la flecha. En la figura, la componente de sistema de más alto nivel, X, depende de los objetos y operaciones contenidas en el paquete 1 y paquete 2 para satisfacer su función. El paquete 2 depende de los objetos y operaciones contenidas en el paquete 3 y 4.

El uso de notación gráfica para el DOO no es esencial, pero da una indicación de la dependencia entre los paquetes (objetos y operaciones), que falta en una representación LDP. El diagrama de Booch se utiliza normalmente para representar las componentes del programa a un

nivel relativamente alto de abstracción. Cuando comienza el diseño de detalle de construcción, se abandona la notación gráfica y se utiliza el LDP como representación del diseño.

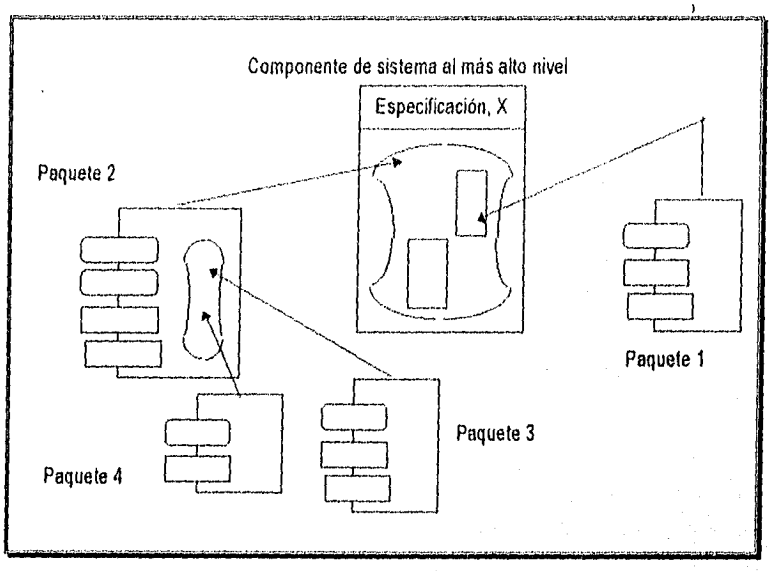


Fig.IV.5 Componentes e interfaces del sistema y diagrama de Booch.

IV.6 Detalle de implementación

El paso de diseño detallado del DOO es similar en muchos aspectos al de cualquier metodología de diseño del software. Se describen las interfaces en detalle; se refinan y especifican las estructuras de datos; los algoritmos se diseñan para cada unidad de programa, usando los conceptos fundamentales de diseño, tales como refinamiento sucesivo y programación estructurada. La diferencia clave para el DOO es que el proceso descrito anteriormente puede aplicarse recursivamente. De hecho, la definición recursiva de la estrategia de la solución es esencial para adquirir un nivel de diseño y de abstracción de datos, a partir del cual pueda derivarse el detalle de implementación.

IV.7 Desarrollo simplificado del método

Para proceder a aplicar el método repasemos las actividades efectuadas hasta este momento:

1. El problema fue definido en términos de una necesidad y,
2. se estableció una estrategia informal.

Nos queda entonces el proceso de formalización de la estrategia -que dicho sea de paso, es el objeto de la recursividad-, para lo que contamos con las siguientes directivas:

a) Generar un enunciado que dé solución al problema o que describa a un objeto para reducir su abstracción, e identificar objetos señalando los sustantivos que componen al enunciado.

b) Obtener las operaciones a partir de los verbos, así como los atributos al reconocer los adjetivos y adverbios del enunciado.

c) Identificar y establecer las interfaces entre objetos con la ayuda de los diagramas de Booch y generar el código necesario.

d) Decidir el diseño detallado para codificar la funcionalidad de las operaciones, atributos y objetos.

Para efectos prácticos podemos elaborar una herramienta que nos ayude a la aplicación del método y que ilustre a la vez su evolución. Esta herramienta concentrará los pasos recursivos de la metodología, de forma que se facilite su entendimiento, a la vista de sus cuatro elementos:

SEGMENTO DE NIVEL SUPERIOR: Ilustra un enunciado o descripción realizado a un nivel "inicial" de abstracción, así como las operaciones, objetos y atributos desprendidos de allí.

ENUNCIADO DEL PROBLEMA O DESCRIPCIÓN DE OBJETOS DE NIVEL SUPERIOR	
Tabla de Operaciones, OBjetos y Atributos de Nivel Superior (TOOBANS)	Consideraciones y comentarios a la identificación de los elementos de la tabla adjunta de nivel superior.

El enunciado del problema debe apegarse a las reglas que del método informal se disponen, cada objeto será subindexado con un número de referencia progresivo que permitirá, dado el caso, aclarar redundancias. La identificación de los objetos, operaciones y atributos para llenar la TOOBANS se efectuará como sigue:

- **Objetos**: Se identificarán en primer lugar -en apego a la directiva 'a'- y se indicarán en **negrita** los sustantivos candidatos a objeto.

- *Operaciones*: Se identifican en segundo lugar -directiva 'b'- y se indicarán con los verbos candidatos en *cursiva o itálica*.
- Atributos: Se identifican en último lugar como los adjetivos y adverbios del enunciado, permaneciendo con tipografía ordinaria.

SEGMENTOS DE NIVEL INMEDIATO INFERIOR: Básicamente es un bloque formado por réplicas del segmento de nivel superior desarrolladas sobre la TOOBANS, mostrando la recursividad del método al presentar para todos y cada uno de los candidatos a objeto enlistados lo siguiente:

- Un enunciado que lo describe y reduce su abstracción.
- Una Tabla de Operaciones, OBjetos y Atributos de Nivel Inmediato Inferior (TOOBANII).
- Los comentarios y observaciones a las directivas aplicadas al enunciado del segmento para generar la TOOBANII.

Lo anterior se repite hasta agotar los elementos de la TOOBANS.

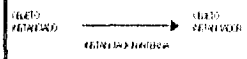
Enunciado/Descripción del candidato a objeto miembro de la TOOBANS.	Tabla de Operaciones, Objetos y Atributos de Nivel Inmediato Inferior (TOOBANII).	Comentarios y Observaciones a la generación de la TOOBANII.

DIAGRAMAS DE BOOCH: Presenta un estado actual del proceso, mostrando gráficamente los elementos del diseño y la relación existente entre los mismos, auxiliando así a la definición de las interfaces descrita con el LDP, cumpliendo con la directiva 'c'. Aunque este elemento se desarrolla con cada uno de los segmentos, se presentarán los diagramas hacia el final del ciclo (agotamiento de candidatos a objetos para todos los candidato a objeto primitivo). Se hará uso de dos tipos de diagramas, el de descripción de clase y el de definición de clase. En el primero se muestra la jerarquía de clase aunado a la relación existente entre las clases mismas, y en el segundo se mostrará el diagrama de objeto de Booch correspondiente a cada clase; este último puede también ser reemplazado por la definición completa de clase, desarrollada sobre el LDP en la última fase del método.

Se han tomado una serie de convenciones para mostrar con claridad las relaciones existentes entre las clases (para el diagrama de descripción de clase), que es conveniente explicar y que son presentadas en la siguiente tabla:

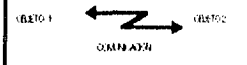
SIMBOLOGÍA DEL DIAGRAMA JERÁRQUICO-OPERACIONAL DE DESCRIPCIÓN DE CLASE

Este símbolo indica Jerarquía de Herencia, donde el estado Hijo hereda las características acumuladas por el padre.

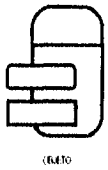


La instanciación interna muestra qué objetos hacen uso de otros para efectuar sus funciones.

La instanciación externa indica qué objetos hacen uso de un objeto ya creado por el sistema y de qué objeto proviene.



La comunicación relaciona las operaciones entre dos o más objetos.



El símbolo de objeto contiene su nombre y refleja su posición dentro del sistema.

Adicionalmente al diagrama de descripción de clase, se presenta una discusión de objetos, donde se determinan las operaciones que se derivan del análisis-diseño o se modifica la estructura propuesta por el diseño.

DETALLE DE IMPLEMENTACIÓN: Es una lista con las instrucciones necesarias para realizar los procedimientos que llevarán a efecto las operaciones, y con las declarativas necesarias para realizar los procedimientos. Este elemento se realiza cuando las interfaces están ya definidas y dada su magnitud no se incluirán como parte impresa del presente trabajo.

A partir de aquí, haremos uso de la metodología sin detallar los pasos de la misma.

Diseño y Construcción de un **Entorno Gráfico de Software (EGS)** que *permita la edición* de una **Carta ASM** *conceptualizada* por el **Usuario.**

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
--------	Problema	Diseñar, Construir	
EGS_1	Solución	Permitir edición	
Carta ASM_2	Problema		
$Usuario_3$	Problema	Conceptualizar	

Para el caso de --------, *Diseñar* y *Construir* se consideran operaciones miembro del problema puesto que es una declaración del problema mismo y no hacen referencia a un objeto del presente enunciado

EGS es una frase nominal que se considera indivisible y *Permitir edición* es un verbo que actúa directamente sobre **Cartas ASM**.

Cartas ASM aquí se refiere a la *conceptualización* de la **Carta ASM**, y se cataloga como parte del problema

El EGS_1 será una interface que *presente* al **Usuario** un **Área** sobre la que serán *visualizados* y *manejados* los **Elementos De Edición (EDE)** y donde estará disponible un **Conjunto de Elementos Gráficos (CEG)** que permitan *manipular, organizar e interpretar (PMOI)* la **Información**.

OBJETO	ESPACIO	OPERACION	ATRIBUTO
EGS_1	Problema	Presentar Área	interface
$Usuario_3$	Problema	Manejar Elem	
$Área_4$	Solución	Visualizar Elem	
EDE_5	Solución	Edición	
CEG_6	Solución	PMOI	Estará disponible
$Información_7$	Solución	--------	

El problema ahora es la definición del **EGS** y todo lo que hace referencia a él (**Usuario** y **EGS**) es parte de su dominio *Manejados* se asigna a **Usuario** por contestación a la pregunta ¿Quién maneja x sobre el **Área**?

PMOI, son operaciones del **CEG** y no de la **Información**, que es un sinónimo de lo que será *visualizado* y *manejado* dentro del **Área**

La interface, en términos de un enunciado como éste, se considera un sinónimo y es colocado como atributo en la TOOBANII

El **EGS_1** será una interface que *presente* al **Usuario** un **Área** sobre la que serán *visualizados y manejados* los **Elementos De Edición (EDE)** y donde estará disponible un **Conjunto de Elementos Gráficos (CEG)** que permitan *manipular, organizar e interpretar (PMOI)* la **Información**

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
EGS_1	Problema	Presentar Área	Interface
$Usuario_3$	Problema	Manejar Elem.	
$Área_4$	Solución	Visualizar Elem.	Información$_7$ a manejar, Estará disponible
EDE_5	Solución	Edición	
CEG_6	Solución	PMOI	Estará disponible

La forma en que la interface interactúa con el **usuario** supone la necesaria existencia de un dispositivo que permita la introducción de datos, teclado para el texto y un apuntador para las posiciones. Podemos intuir estos objetos y esperar a que sean definidos en otra fase de más detalle.

Área comparte el atributo dado por la frase estará disponible que en términos de una interface gráfica se traduce en *"Mostrar"* y la frase *"serán visualizados"* que reafirma este concepto

El **EGS** por ser el problema mismo queda descartado, así como **Información**, que es tanto una redundancia del **EDE** como un atributo del **Área** y como **Usuario** que es parte evidente de la definición del problema original

El **Área** deberá *mostrar* al **Usuario** una **Superficie** *delimitada* donde *construirá* el **Diagrama de la Carta ASM (D. Carta ASM)** *ensamblando* entre si **EDE** e indicando su secuencia. Deberá también *mostrar* una **Lista con las Operaciones (Lista Oper.)** que *administrarán* al **EGS**, al Contenido del **Espacio Disponible** para la **Edición** y facilitará el *guardar y recuperar* el **Trabajo Hecho**.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
$Área_4$	Problema	Mostrar	
$Usuario_3$	Problema	Construir, Ensamblar, Indicar Sec	
$Superficie_8$	Solución	Delimitar	Esp. Disp Trab. Hecho
D. Carta ASM_2	Solución	Ensamblar	
EDE_5	Problema		
$Lista\ Oper_9$	Solución	Administrar, grabar, recuperar	
EGS_1	Problema	Presentar	

De este enunciado se desprende que el **Área** consta de una **superficie**, una **lista de operaciones** y un **espacio disponible**, que contendrá al **D. Carta ASM** compuesto por los **EDE** de la carta seleccionados de algún lugar dentro del **Área**. Llamémosle a este ambiente **Área de Trabajo**

Área y **Usuario** quedan descartados de la tabla por pertenecer al problema, mientras que **Trabajo Hecho** también deberá descartarse por redundar con el contenido del **espacio disponible**. Por otra parte **Edición** que puede confundirse con otro objeto puede deslindarse al responder a ¿Qué es lo que se edita? (La **Carta ASM**) y ¿Quién la edita? (El **Usuario**), por lo tanto es una operación de un objeto del problema, específicamente del **Usuario**. El **Espacio Disponible** es un sinónimo de superficie que contiene al **Trabajo Hecho.**

Los **Elementos De Edición (EDE)** son *definidos* por el **Diagrama de Carta ASM (D. Carta ASM)** como **Figuras Geométricas (Figuras Geo.)** que representan **Estado, Salida Condicional y Decisión**, que son los **Elementos de Cartas ASM** y deberán estar disponibles para su *selección y uso* en la *construcción* del **Diagrama de la Carta ASM (D. Carta ASM)**

OBJETO	ESPACIO	OPERACION	ATRIBUTO
EDE_5	Problema	----------	
D. Carta ASM_2	Problema	Definir	
Elemento ASM_{10}	Solución	Seleccionar, Construir (usar)	Figura Geo., Edo., Salida C y Decisión

Los **EDE** y **D. Carta ASM** son parte de la descripción del problema, deben ser eliminados.

Estado, **Salida Cond.** y **Decisión** son los elementos definidos en la teoría de **Cartas ASM**

Figuras geométricas es un objeto vinculado estrechamente con los **Elementos ASM**, de hecho es una extensión que añade la característica geométrica a la definición, es por tanto un atributo. Lo mismo puede decirse de **Estado**, **Salida** y **Decisión**, que sólo extienden la definición de **Elemento ASM** a este nivel de abstracción

El **CEG** es un *grupo* compuesto de **Dispositivos** localizados en la **Pantalla** dedicados a *controlar la forma* en que *interactúa* con el **Área** al *modificar su escala*, *desplazándola* para ser observada por el **Usuario** y *mostrando los* ***Mensajes De las Operaciones (MDO)*** en una **Ventana** *desplegada* dentro del **Área**. Un dispositivo **Apuntador** y un **Cursor de Texto** ayudarán a *introducir los datos* y a *dirigir las acciones* del resto.

OBJETO	ESPACIO	OPERACION	ATRIBUTO
CEG_6	Problema	Agrupar	Grupo Comp.
$Dispositivos_{11}$	Solución	Controlar forma, Interactuar, Modif escala, Desplazar	Localizados en Pantalla
$Pantalla_{12}$	Solución	Mostrar	
$Área_4$	Problema	Mostrar	
$Usuario_3$	Problema	Observar	
$Ventana_{13}$	Solución	Mostrar MDO, Desplegar	
$Apuntador_{14}$	Solución	Seleccionar Dispositivos	
$Cursor\ Texto_{15}$	Solución	Introducir datos, Dirigir acciones	

El enunciado propone la existencia implícita de los objetos que interactúan con el **Área** y que se pueden identificar también como atributos del objeto **Dispositivos.**

La operación *Muestra* de **Pantalla** se deduce del atributo de **Dispositivos** localizados. Por otra parte, si **Dispositivos** designa a elementos que forman parte de la interface y **Pantalla** al elemento físico de video del sistema, podríamos catalogar **Dispositivos** físico al componente de hardware y como **Dispositivo** lógico al componente del programa que refleja un elemento de control

El **Área** deberá *mostrar* al **Usuario** una **Superficie** *delimitada* donde *construirá* el **Diagrama de la Carta ASM (D. Carta ASM)** *ensamblando* entre sí **EDE** e *indicando* su *secuencia*. Deberá también *mostrar* una **Lista con las Operaciones (Lista Oper.)** que *administrarán* al **EGS**, al Contenido del **Espacio Disponible** para la **Edición** y facilitará el *guardar* y *recuperar* el **Trabajo Hecho**.

OBJETO	ESPACIO	OPERACION	ATRIBUTO
Área$_4$	Problema	Mostrar	
Usuario$_3$	Problema	Construir, Ensamblar,Indicar Sec.	
Superficie$_8$	Solución	Delimitar	Esp. Disp. Trab. Hecho
D. Carta ASM$_2$	Solución	Ensamblar	
EDE$_5$	Problema		
Lista Oper$_9$	Solución	Administrar, grabar, recuperar	
EGS$_1$	Problema	Presentar	

De este enunciado se desprende que el **Área** consta de una **superficie**, una **lista de operaciones** y un **espacio disponible**, que contendrá al **D. Carta ASM** compuesto por los **EDE** de la carta seleccionados de algún lugar dentro del **Área**. Llamémosle a este ambiente **Área de Trabajo**.

Área y **Usuario** quedan descartados de la tabla por pertenecer al problema, mientras que **Trabajo Hecho** también deberá descartarse por redundar con el contenido del **espacio disponible**. Por otra parte **Edición**, que puede confundirse con otro objeto puede deslindarse al responder a ¿Qué es lo que se edita? (La **Carta ASM**) y ¿Quién la edita? (El **Usuario**), por lo tanto es una operación de un objeto del problema, específicamente del **Usuario**. El **Espacio Disponible** es un sinónimo de superficie que contiene al **Trabajo Hecho.**

La **Superficie$_8$** aparentará *extenderse* más allá de la **Pantalla** de forma que pueda obtenerse una mayor amplitud siempre que se requiera. Debe también *conservar* y *mostrar* consistentemente su **Contenido**.

OBJETO	ESPACIO	OPERACION	ATRIBUTO
Superficie$_8$	Problema	Extender, Contener, Mostrar, Conservar Contenido	Obtiene mayor Amplitud
Pantalla$_{12}$	Problema	Mostrar	

A primera vista en este enunciado no existen objetos de solución y esto nos indica que en realidad estamos describiendo un atributo de objeto o su sinónimo. Si **superficie** es un sinónimo de **Área**, entonces debemos suponer la existencia de las operaciones adecuadas para manejarla, como por ejemplo, *abrir*, *cerrar*, *ampliar* o *definir*, todas ellas serán atributos u operaciones de **Área de trabajo**.

Una **Carta ASM**$_2$ es un **Diagrama** que *organiza* la *ejecución* de un **Algoritmo** a lo largo del **Flujo** representado por los **Elementos de Cartas ASM (Elementos ASM)** de su Método. Es una **Entidad** que puede *imprimirse* en cualquier momento, *recuperarse* y *almacenarse* de y en disco.

OBJETO	ESPACIO	OPERACION	ATRIBUTO
Carta ASM$_2$	Problema	Organizar, ejecutar, imprimir, Recuperar, Almacenar	Diagrama Entidad
Algoritmo$_{16}$	Solución	Ejecutar Flujo	
Elementos ASM$_{18}$	Solución	Representar	

Ya que **Diagrama** profundiza en la descripción de lo que es una **Carta ASM** se considera un atributo, al igual los términos **algoritmo e entidad** hacen mención a **Cartas ASM** de una forma en que se consideran un sinónimo.

Aquí observamos como **Cartas ASM** y **elementos** de **Cartas ASM** mantienen una relación. Esto lo habíamos especificado en la definición a **EDE**.

La **Lista de Operaciones** podrá *mostrarse* por un **Menú Desplegable de Texto (MDT)** o por un **Menú De iconos (MDI)** que *agrupen* actividades acerca de la operación del **EGS** como son la *inicialización* del **Área de trabajo,** la **Función** que permite la vista *ampliada* y/o *reducida* del contenido del **Área de trabajo** y la capacidad de *cambiar* **Parámetros** del sistema.

OBJETO	ESPACIO	OPERACION	ATRIBUTO
Lista de Oper$_9$	Problema	Enlistar, Mostrar, Recup. y Almac. de disco, reducir y ampliar conten. área	Funciones
Área$_4$	Solución	Inicializar	Contenido
Parámetros$_{17}$	Solución	Cambiar	
MDT$_{12}$	Solución	Agrupar, Mostrar	Actividades
MDI$_{19}$	Solución	Agrupar, Mostrar	Actividades, Iconos

El **Menú** es una forma de *mostrar* las Operaciones de los diferentes objetos del sistema y de ponerlas a disposición del usuario, claro que no es la única forma pues las operaciones también pueden mostrarse como iconos específicos que no dejan por eso de clasificarse dentro de un **Menú (Menú de iconos)**.

En este enunciado, la descripción en términos de un objeto ya definido se convierte en un objeto de la solución del problema que nos ayuda a conceptualizar la naturaleza de las funciones que se incluyen en la lista. Es de esperarse que esta lista contenga una serie de conexiones con otros objetos actuando como una interface entre ellas. Los objetos de este tipo ceden sus operaciones a los objetos "primitivos" u originales, ayudando a definirlos.

Contenido es un atributo inherente a la definición de **Área de Trabajo**. Sin embargo, este atributo nos manifiesta la existencia de una función pública que permitirá reducir y ampliar la apariencia del **Área**. Recuperar y Almacenar en disco son frases verbales que forman dos operaciones de la **lista**. El sustantivo contenido del **Área** denota una relación entre la **lista** y el **Área de trabajo**.

La **Superficie**$_8$ aparentará *extenderse* más allá de la **Pantalla** de forma que pueda obtenerse una mayor amplitud siempre que se requiera. Debe también *conservar* y *mostrar* consistentemente su **Contenido**				
OBJETO	**ESPACIO**	**OPERACION**	**ATRIBUTO**	A primera vista en este enunciado no existen objetos de solución y esto nos indica que en realidad estamos describiendo un atributo de objeto o su sinónimo. Si **superficie** es un sinónimo de **Área**, entonces debemos suponer la existencia de las operaciones adecuadas para manejarla, como por ejemplo, *abrir*, *cerrar*, *ampliar* o *definir*, todas ellas serán atributos u operaciones de **Área de trabajo**
Superficie$_8$	Problema	Extender, Contener, Conservar, Mostrar Contenido	Obtiene Mayor Amplitud	
Pantalla$_{12}$	Problema	Mostrar		

Una **Carta ASM**$_2$ es un **Diagrama** que *organiza* la *ejecución* de un **Algoritmo** a lo largo del **Flujo** representado por los **Elementos de Cartas ASM (Elementos ASM)** de su Método. Es una **Entidad** que puede *imprimirse* en cualquier momento, recuperarse y *almacenarse* de y en disco.				
OBJETO	**ESPACIO**	**OPERACION**	**ATRIBUTO**	Ya que **Diagrama** profundiza en la descripción de lo que es una **Carta ASM** se considera un atributo, al igual los términos **algoritmo** e **entidad** hacen mención a **Cartas ASM** de una forma en que se consideran un sinónimo. Aquí observamos como **Cartas ASM** y **elementos** de **Cartas ASM** mantienen una relación. Esto lo habíamos especificado en la definición a **EDE**.
Carta ASM$_2$	Problema	Organizar ejecutar, Imprimir, Recuperar, Almacenar	Diagrama, Entidad	
Algoritmo$_{16}$	Problema	Ejecutar	Flujo	
Elementos ASM$_{18}$	Solución	Representar		

Estado, **Salida Condicional (Sal Cond)** y **Decisión** son todos **Elementos ASM** que conforman el **Flujo Estructural** que describe a una **Carta ASM.** Puede ser: *presentado*, *seleccionado* para *editarse*, *movido* a través del **Área** y *modificado* su **Contenido**.	**OBJETO**	**ESPACIO**	**OPERACIÓN**	**ATRIBUTO**	En este enunciado nos auxiliamos de una descripción que pertenece a un nivel anterior. Las operaciones Presentar, Seleccionar, Mover y Modificar y el atributo de contenido debieron obtenerse en aquel nivel y ahora debe manejarse como operaciones comunes a todos los elementos. Existe relación con **Área** y **EDE** **Flujo Estructural** describe a la **Carta ASM** por lo que es un atributo.
	Estado$_{20}$	Solución	Presentar, Seleccionar, Mover, Modificar	Contenido	
	Sal Cond$_{21}$	Solución	Presentar, Seleccionar, Mover, Modificar	Contenido	
	Decisión$_{22}$	Solución	Presentar, Seleccionar, Mover, Modificar	Contenido	
	Elem ASM$_{10}$	Problema			
	Carta ASM$_{2}$	Problema	Describir	Flujo Estructural	
	Área$_{4}$	Problema			

Estado, Salida Condicional (Sal Condicional) y **Decisión** son todos **Elementos ASM** que conforman el **Flujo Estructural** que describe a una **Carta ASM.** Puede ser: *presentado, seleccionado* para *editarse, movido* a través del **Área** y *modificado* su **Contenido**.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Estado$_{20}$	Solución	Presentar, Seleccionar Mover, Modificar,	Contenido
Sal Condicional$_{21}$	Solución	Presentar, Seleccionar, Mover, Modificar	Contenido
Decisión$_{22}$	Solución	Presentar, Seleccionar, Mover, Modificar	Contenido
Elementos ASM$_{10}$	Problema		
Carta ASM$_{2}$	Problema	Describir	Flujo Estructural
Área$_{4}$	Problema		

En este enunciado nos auxiliamos de una descripción que pertenece aun nivel anterior. Las operaciones Presentar, Seleccionar, Mover y Modificar y el atributo de **contenido** debieron obtenerse en aquel nivel y ahora debe manejarse como operaciones comunes a todos los elementos.

Existe relación con **Área** y **EDE**.

Flujo Estructural describe a la **Carta ASM** por lo que es un atributo.

Estado$_{20}$ es una entidad compuesta por un rectángulo, una etiqueta de nombre, una etiqueta de código y una lista de variables de salida, todos ellos *editables*. Es el destino de un flujo y la fuente de otro.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Estado$_{20}$	Problema	Editar	Rectángulo, Código, Etiq Nombre, Etiq Código, Lista de Var , Entidad

Se detiene la abstracción para los atributos de **estado**. Haciendo de este objeto terminal, sin embargo, existe la posibilidad de definir un nuevo objeto "Objeto".

Entidad podemos considerarlo como un sinónimo de Estado.

Salida Condicional$_{21}$ es una entidad compuesta por un Rectángulo de esquinas redondeadas y una lista de variables de salida también *editables*. Es el destino de un flujo y la fuente de otro.

OBJETO	ESPACIO	OPERACION	ATRIBUTO
Sal Condicional$_{21}$	Problema	Editar	Rectángulo Redondeado, Lista de Var., Entidad

De acuerdo a la teoría de Cartas ASM, el flujo fuente debe provenir necesariamente de una **decisión**. Este es un objeto del desarrollo del método.

Decisión$_{22}$ es una entidad compuesta por una figura rómbica y una variable de prueba *editable*. Tiene dos flujos de salida, uno etiquetado con un "1" y el otro con un "0" dirigidos ambos hacia otros elementos.	**OBJETO**	**ESPACIO**	**OPERACIÓN**	**ATRIBUTO**	La teoría de Cartas ASM indica a éste como el único con más de un flujo de salida. Es también un objeto terminal del desarrollo de análisis y diseño
	Decisión$_{22}$	Problema	Editar	Fig. Rómbica, Var. Prueba, Entidad	

Estado$_{26}$ es una entidad compuesta por un rectángulo, una etiqueta de nombre, una etiqueta de código y una lista de variables de salida, todos ellos *editables* es el destino de un flujo y la fuente de otro.				
OBJETO	**ESPACIO**	**OPERACIÓN**	**ATRIBUTO**	Se detiene la abstracción para los atributos de **estado**. Haciendo de este objeto terminal, sin embargo, existe la posibilidad de definir un nuevo objeto "Objeto".
Estado$_{26}$	Problema	Editar	rectángulo, Código, Etiq. Nombre, Etiq. Código, Lista de Var., Entidad	**Entidad** podemos considerarlo como un sinónimo de Estado.

Salida Condicional$_{21}$ es una entidad compuesta por un Rectángulo de esquinas redondeadas y una lista de variables de salida también *editables*. Es el destino de un flujo y la fuente de otro.				
OBJETO	**ESPACIO**	**OPERACIÓN**	**ATRIBUTO**	De acuerdo a la teoría de Cartas ASM, el flujo fuente debe provenir necesariamente de una **decisión**. Éste es un objeto del desarrollo del método.
Sal Condicional$_{21}$	Problema	Editar	Rectángulo redondeado, Lista de Var., Entidad	

Decisión$_{22}$ es una entidad compuesta por una figura rómbica y una variable de prueba *editable*. Tiene dos flujos de salida, uno etiquetado con un "1" y el otro con un "0" dirigidos ambos hacia otros elementos.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Decisión$_{22}$	Problema	Editar	Fig. Rómbica,Var. Prueba, Entidad

La teoría de Cartas ASM indica a éste como el único con más de un flujo de salida. Es también un objeto terminal del desarrollo de análisis y diseño.

La **Lista de Operaciones (Lista Oper)** podrá *mostrarse* por un **Menú Desplegable de Texto (MDT)** o por un **Menú De iconos (MDI)** que *agrupen* actividades acerca de la operación del **EGS** como son la *inicialización* del **Área de trabajo,** la **Función** que permite la vista *ampliada* y/o *reducida* del contenido del **Área de trabajo** y la capacidad de *cambiar* **Parámetros** del sistema.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Lista Oper$_9$	Problema	Enlistar, Mostrar, Recup. y Almac. de disco, reducir y ampliar conten. área	Funciones
Área$_4$	Solución	Inicializar	Contenido
Parámetros$_{17}$	Solución	Cambiar	
MDT$_{13}$	Solución	Agrupar, Mostrar	Actividades
MDI$_{13}$	Solución	Agrupar, Mostrar	Actividades, Iconos

El **Menú** es una forma de *mostrar* las Operaciones de los diferentes objetos del sistema y de ponerlas a disposición del usuario, claro que no es la única forma pues las operaciones también pueden mostrarse como iconos específicos que no dejan por eso de clasificarse dentro de un **Menú (Menú de iconos)**.

En este enunciado, la descripción en términos de un objeto ya definido se convierte en una objeto de la solución del problema que nos ayuda a conceptualizar la naturaleza de las funciones que se incluyen en la lista. Es de esperarse que esta lista contenga una serie de conexiones con otros objetos, actuando como una interface entre ellas. Los objetos de este tipo ceden sus operaciones a los objetos "primitivos" u originales, ayudando a definirlos.

Contenido es un atributo inherente a la definición de **Área de Trabajo**. Sin embargo, este atributo nos manifiesta la existencia de una función pública que permitirá reducir y ampliar la apariencia del **Área**. Recuperar y Almacenar en disco son frases verbales que forman dos operaciones de la **lista**. El sustantivo contenido del **Área** denota una relación entre la lista y el **Área de trabajo**.

Los **Parámetros**$_{17}$ son un grupo de **Valores** modificables que *determinan* las *características* del **Software**

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Parámetros$_{17}$	Problema		
Valores$_{23}$	Solución	Determinar Características	Modificables
Software$_{24}$	Problema		

Al indicar que los **valores** hacen a un grupo de **Parámetros**, estamos mencionando un sinónimo, con la salvedad de que son objetos de un distinto nivel **Parámetros** agrupa a los **valores** y cada valor por si mismo puede ser un tipo de atributo

El objeto **Parámetros** lo podemos considerar opcional para el desarrollo de la aplicación debido que ofrece características deseables pero no definitivas para la solución propuesta

Aquí aparece un posible objeto llamado **Software**, pero como este es en realidad el problema que se desea resolver, por lo que es 100 % parte del problema

El **Menú Desplegable de Texto (MDT)** será una barra horizontal que *enlistará* nombres de **Submenús,** que al ser *Seleccionados* por **Teclado** o **Apuntador**, *mostrarán* **Ventanas** *enlistando* ***Opciones*** *seleccionables* compuestas por el **Nombre de la operación** y su **Tecla rápida** para accesar mas rápidamente, o **Ventanas de Diálogo** para *capturar información*

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
MDT$_{18}$	Problema	Enlistar Submenús	Barra Horiz
Submenús$_{25}$	Solución	Mostrar, Seleccionar por teclado o apuntador	Ventanas Elegibles
Ventana$_{13}$	Solución	Enlistar opciones	
Ventana Dial$_{26}$	Solución	Capturar Info	
Opciones$_{27}$	Solución	Seleccionar Operación,	Nombre de la Tecla rápida

Un **menú desplegable** es un objeto familiar en las interfaces de usuario que se maneja tanto con el **teclado** como con el **Ratón** La lista horizontal inicial contiene nombres de **Submenús** o en algunos casos, nombres de acciones concretas que se efectúan en cuanto se seleccionan

Este es un ejemplo de combinación de niveles de abstracción, pues se define **Submenús**, que pertenece a un nivel inferior de abstracción o de mayor detalle

El **Menú De Iconos$_{19}$ (MDI)** *mostrará* un **Área** con **Botones** *seleccionables* identificados por **Iconos** representando la **Operación** que *ejecutan*.

OBJETO	ESPACIO	OPERACION	ATRIBUTO
MDI$_{19}$	Problema	Mostrar Área	
Botones$_{28}$	Solución	Seleccionar	Icono
Operación$_{29}$	Solución	Ejecutar	

El **Menú de iconos** agrupa **botones** con un atributo especial, sobre un área que puede llamarse área de menú de iconos definida sobre el **Área de trabajo** Cada **botón** tiene un pequeño dibujo distintivo de la acción que efectúa, (Icono). La pertenencia del Icono es un atributo de **Botones** distinguido por la palabra identificados

Los **Parámetros$_{17}$** son un grupo de **Valores** modificables que *determinan* las *características* del **Software**.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Parámetros$_{17}$	Problema		
Valores$_{23}$	Solución	Determinar Características	Modificables
Software$_{24}$	Problema		

Al indicar que los **valores** hacen a un grupo de **Parámetros**, estamos mencionando un sinónimo, con la salvedad de que son objetos de un distinto nivel. **Parámetros** agrupa a los **valores** y cada valor por si mismo puede ser un tipo de atributo.

El objeto **Parámetros** lo podemos considerar opcional para el desarrollo de la aplicación, debido que ofrece características deseables pero no definitivas para la solución propuesta.

Aqui aparece un posible objeto llamado **Software**, pero como éste es en realidad el problema que se desea resolver, por lo que es 100 % parte del problema.

El **Menú Desplegable de Texto (MDT)** será una barra horizontal que *enlistará* nombres de **Submenús,** que al ser *Seleccionados* por **Teclado** o **Apuntador**, *mostrarán* **Ventanas** *enlistando* **Opciones** *seleccionables* compuestas por el **Nombre de la operación** y su **Tecla rápida** para accesar mas rápidamente, o **Ventanas de Diálogo** para *capturar* información.

OBJETO	ESPACIO	OPERACION	ATRIBUTO
MDT$_{18}$	Problema	Enlistar Submenús	Barra Horiz.
Submenús$_{25}$	Solución	Mostrar, Seleccionar por teclado o apuntador	Ventanas Elegibles
Ventana$_{13}$	Solución	Enlistar opciones	
Ventana Dial$_{26}$	Solución	Capturar Info.	
Opciones$_{27}$	Solución	Seleccionar Operación	Nombre de la Tecla rápida

Un **menú desplegable** es un objeto familiar en las interfaces de usuario que se maneja tanto con el teclado como con el **Ratón**. La lista horizontal inicial contiene nombres de **Submenús** o en algunos casos, nombres de acciones concretas que se efectúan en cuanto se seleccionan.

Este es un ejemplo de combinación de niveles de abstracción, pues se define **Submenús** que pertenece a un nivel inferior de abstracción o de mayor detalle

Los **Submenús$_{25}$** son **Ventanas** que *Emergen* verticalmente sobre el **MDT** y *Muestran* su *contenido* seleccionable, que pueden ser: **Opciones**, Ventanas de Información o **Sub-menús.**

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Submenús$_{25}$	Problema	Emerger verticalmente, Mostrar Contenido información	Ventanas, Opción, Ventanas Submenús

El **SubMenú** es una característica del **MDT** que puede ser tratado como un objeto o como parte de las funciones propias del **MDT**, razón por la que hemos trasladado los elementos del contenido como atributos que más tarde serán construidos con el LDP.

La **Ventana$_{13}$** será un área rectangular que intermediará el diálogo sistema-usuario, *mostrando mensajes* y *requiriendo acciones* **Específicas.**

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Ventana$_{13}$	Problema	Mostrar mensajes, Requerir acciones	Área rectangular

Una **ventana** asoma al usuario a un **mensaje** del sistema y es el medio en que en general ocurre la comunicación de la interface. Rigurosamente hablando podríamos decir que el **Área de Trabajo** es una **ventana**, sin embargo la definición de ellas que se da aquí es que sólo sirven para la presentación de texto o para el requerimiento de acciones. El **Área de trabajo** no requiere nunca de acciones por sí misma y es sólo una vista de la superficie editable

Las **Ventanas Diálogo$_{26}$ (Ventana Diál)** son **Ventanas** *abiertas* para *mostrar* su **Contenido** al **Usuario** y/o *recibir* información, esperando la confirmación de una **Acción** efectuada sobre ella para cerrarse.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Ventana Diál$_{26}$	Problema	Abrir, Mostrar, Accionar Dentro, Cerrar, Recibir	Ventana
Usuario$_3$	Problema		

Este enunciado nos aclara que una vez abierta, **la Ventana Diálogo** es cerrada con una acción sobre ella. Éste podría ser el nivel máximo de detalle para esta parte del análisis.

Las **Opciones$_{27}$** son **Alternativas de Acción** *enlistadas* en los submenús utilizadas para *ejecutar funciones*.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Opciones$_{27}$	Problema	Enlistar, Ejecutar función	Alternativa de acción

Las **opciones** pueden ser tratadas como funciones o como objetos, dependiendo del diseño. Si como funciones, entonces pertenecerán al espacio del problema.

Los **Submenús$_{25}$** son **Ventanas** que *Emergen verticalmente* sobre el **MDT** y *Muestran su contenido* seleccionable, que pueden ser: **Opciones, Ventanas de Información o Sub-menús.**

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Submenús$_{25}$	Problema	Emerger verticalmente, Mostrar Contenido información	Ventanas, Ventanas Opción, SubMenús

El **SubMenú** es una característica del **MDT** que puede ser tratado como un objeto o como parte de las funciones propias del **MDT**, razón por la que hemos trasladado los elementos del contenido como atributos que más tarde serán construidos con el LDP.

La **Ventana$_{13}$** será un área rectangular que intermediará el diálogo sistema-usuario, *mostrando mensajes* y *requiriendo acciones* **Específicas.**

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Ventana$_{13}$	Problema	Mostrar Mensajes, Requerir Acciones	Área Rectangular

Una **ventana** asoma al usuario a un **mensaje** del sistema y es el medio en que en general ocurre la comunicación de la interface. Rigurosamente hablando podríamos decir que el **Área de Trabajo** es una **ventana**, sin embargo la definición de ellas que se da aquí es que sólo sirven para la presentación de texto o para el requerimiento de acciones. El **Área de trabajo** no requiere nunca de acciones por sí misma y es sólo una vista de la superficie editable.

Las **Ventanas Diálogo$_{26}$ (Ventana Diál)** son **Ventanas** abiertas para *mostrar* su **Contenido** al **Usuario** y/o *recibir* información, esperando la confirmación de una **Acción** efectuada sobre ella para cerrarse.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Ventana Diál$_{26}$	Problema	Abrir, Mostrar, Accionar Dentro, Cerrar, Recibir	Ventana
Usuario$_3$	Problema		

Este enunciado nos aclara que una vez abierta, la **Ventana Diálogo** es cerrada con una acción sobre ella. Éste podría ser el nivel máximo de detalle para esta parte del análisis.

Las **Opciones$_{27}$** son **Alternativas de Acción** *enlistadas* en los submenús utilizadas para *ejecutar funciones*.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Opciones$_{27}$	Problema	Enlistar, Ejecutar función	Alternativa de acción

Las **opciones** pueden ser tratadas como funciones o como objetos, dependiendo del diseño. Si como funciones, entonces pertenecerán al espacio del problema.

El **Menú De Iconos$_{19}$ (MDI)** *mostrará* un **Área** con **Botones** *seleccionables* identificados por **Iconos** representando la **Operación** que *ejecutan*.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
MDI$_{19}$	Problema	Mostrar Área	
Botones$_{28}$	Solución	Seleccionar	Icono
Operación$_{29}$	Solución	Ejecutar	

El **Menú de iconos** agrupa **botones** con un atributo especial, sobre un área que puede llamarse área de menú de iconos definida sobre el **Área de trabajo**. Cada **Botón** tiene un pequeño dibujo distintivo de la acción que efectúa, (icono). La pertenencia del Icono es un atributo de **Botones** distinguido por la palabra identificados.

Los **Botones$_{28}$** son **Áreas Dibujadas** a semejanza de un botón físico que *reaccionan* análogamente tras su *selección*, cada botón estará *ligado a la ejecución* de una *operación* específica.	**OBJETO**	**ESPACIO**	**OPERACIÓN**	**ATRIBUTO**	Este objeto puede guardar relación directa **con Ventana Diálogo** o con cualquier **EGS** que necesite una acción de parte del usuario.
	Botones$_{28}$	Problema	Reaccionar selección, Ejecutar operación	Área dibujada a semejanza de botón Fís	
Las **Operaciones$_{29}$** son el **Conjunto de Acciones** que pueden *ejecutarse* con y entre los **CEG** y los **EDE**.	**OBJETO**	**ESPACIO**	**OPERACIÓN**	**ATRIBUTO**	Las **operaciones** no son un objeto en sí, si no que son las funciones propias de los **CEG** y **EDE**.
	Operación$_{29}$	Problema	Ejecutar	Conjunto Acciones	

Los **Botones$_{28}$** son **Áreas Dibujadas** a semejanza de un botón físico que *reaccionan* análogamente tras su *selección*, cada botón estará ligado a la *ejecución* de una *operación* específica.

OBJETO	**ESPACIO**	**OPERACIÓN**	**ATRIBUTO**	Este objeto puede guardar relación directa con **Ventana Diálogo** o con cualquier **EGS** que necesite una acción de parte del usuario.
Botones$_{28}$	Problema	Reaccionar Selección, Ejecutar operación	Área dibujada a semejanza de botón Fís.	

Las **Operaciones$_{29}$** son el **Conjunto de Acciones** que pueden *ejecutarse* con y entre los **CEG** y los **EDE**.

OBJETO	**ESPACIO**	**OPERACIÓN**	**ATRIBUTO**	Las **operaciones** no son un objeto en sí, si no que son las funciones propias de los **CEG** y **EDE**.
Operación$_{29}$	Problema	Ejecutar	Conjunto Acciones	

Los **Elementos De Edición (EDE)** son *definidos* por el **Diagrama de Carta ASM (D. Carta ASM)** como **Figuras Geométricas (Figuras Geo)** que representan **Estado, Salida Condicional y Decisión**, que son los **Elementos de Cartas ASM** y deberán estar disponibles para su *selección* y *uso* en la *construcción* del **Diagrama de la Carta ASM**.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
EDE_s	Problema	———	
D. Carta ASM_2	Problema	Definir	
Elemento ASM_{10}	Solución	Seleccionar, Construir (usar)	Figura Geo., Edo., Salida C. y Decisión

Los **EDE** y **D. Carta ASM** son parte de la descripción del problema, deben ser eliminados.

Estado, **Salida Cond** y **Decisión** son los elementos definidos en la teoría de **Cartas ASM**.

Figuras geométricas es un objeto vinculado estrechamente con los **Elementos ASM**, de hecho es una extensión que añade la característica geométrica a la definición, es por tanto, un atributo. Lo mismo puede decirse de **Estado**, **Salida** y **Decisión**, que sólo extienden la definición de **Elemento ASM** a este nivel de abstracción.

El **CEG** es un grupo compuesto de **Dispositivos** localizados en la **Pantalla** dedicados a *controlar* la *forma* en que *interactúa* con el **Área** al *modificar* su escala, *desplazándola* para ser observada por el **Usuario** y *mostrando* los **Mensajes De las Operaciones (MDO)** en una **Ventana** *desplegada* dentro del **Área**. Un dispositivo **Apuntador** y un **Cursor de Texto** ayudarán a *introducir* los *datos* y a *dirigir* las *acciones* del resto.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
CEG$_6$	Problema	Agrupar	Grupo Comp.
Dispositivos$_{11}$	Solución	Controlar forma, Interactuar, Modif. escala, Desplazar	Localizados en Pantalla
Pantalla$_{12}$	Solución	Mostrar	
Área$_4$	Problema	Mostrar	
Usuario$_3$	Problema	Observar	
Ventana$_{13}$	Solución	Mostrar MDO, Desplegar	
Apuntador$_{14}$	Solución	Seleccionar Dispositivos	
Cursor Texto$_{15}$	Solución	Introducir datos, Dirigir acciones	

El enunciado propone la existencia implícita de los objetos que interactúan con el **Área**, y que se pueden identificar también como atributos del objeto **Dispositivos.**

La operación *Muestra* de **Pantalla** se deduce del atributo de **Dispositivos**: Localizados. Por otra parte, si **Dispositivos** designa a elementos que forman parte de la interface y **Pantalla** al elemento físico de video del sistema, podríamos catalogar **Dispositivos** físico al componente de hardware y como **dispositivo** lógico al componente del programa que refleja un elemento de control.

Los **Dispositivos$_{11}$** son **Elementos de Software** que *complementan* el *manejo* del **Área**, dando *acceso* a las partes que componen el **EGS**	**OBJETO**	**ESPACIO**	**OPERACIÓN**	**ATRIBUTO**	**Dispositivos** es una generalización de todos los elementos del **EGS** proyectado. Los agrupa y define su interacción con el usuario. Este tipo de enunciados denota la existencia de una generalización de la cual se desprenderán los objetos **EDE** y **CEG.** Es importante mencionar que los **Dispositivos** Lógicos comparten características como se denota si observamos que todos los **dispositivos** que se presentan en pantalla tienen que ser dibujados sobre ella, a partir de un solo punto. Todos tienen que estar autodocumentados para ser usados adecuadamente. Una forma de documentación es el empleo de iconos que expliquen la acción a seguir después de seleccionar un **dispositivo** o mediante el despliegado de un mensaje en una **ventana**. Otro objeto como por ejemplo, una **barra de desplazamiento**, necesita otro tipo de elementos, como lo es mostrar un porcentaje de avance con un cursor y unos **botones** con flechas indicadoras de la dirección.
	Dispositivos$_{11}$	Problema	Complementar manejo	Elementos de Software	
	Área$_4$	Problema	Accesar a los EGS		
La **Pantalla$_{12}$** es el **Dispositivo Físico** de Despliegue Gráfico que *Mostrará* al **Entorno Gráfico de Software (EGS).**	**OBJETO**	**ESPACIO**	**OPERACIÓN**	**ATRIBUTO**	La **pantalla** es un elemento físico que posee ciertas características propias del hardware que afectan directamente al software, proporcionando una plataforma de ejecución que debe configurarse y prepararse para la modalidad gráfica.
	Pantalla$_{12}$	Problema	Mostrar EGS	D. Físico de despliegue gráfico	
	EGS$_1$	Problema			

El **Apuntador**$_{14}$ es un **Dispositivo** que permite el *señalamiento* y *selección* de cualquier **Punto** en la **Pantalla** que coincida con un **EDE** o **CEG**, su *ejecución* o -si es posible- su *transporte*	**OBJETO**	**ESPACIO**	**OPERACIÓN**	**ATRIBUTO**	El **Apuntador** es otro elemento de software que modela a un **dispositivo** físico reconocido directamente por la computadora. Una pluma electrónica, una palanca de juego (JoyStick) o un **Ratón** (Mouse) son los más conocidos, sin embargo el sistema sólo se diseñará para funcionar con un Mouse Microsoft (Tres botones), que es el más popular actualmente. Este aparato requiere -para su reconocimiento- de programación especializada que maneje las interrupciones definidas para su manejador (Driver)
	Apuntador$_{14}$	Problema	Señalar, Seleccionar puntos, Transportar, Ejecutar	Dispositivo	
	Punto$_{38}$	Solución	Seleccionar		
	Pantalla$_{12}$	Problema			
	EDE$_5$	Problema			
	CEG$_6$	Problema			
El **Cursor de Texto**$_{15}$ permitirá *reconocer* el **Lugar Actual** en que será *introducido* el texto para cualquier dispositivo que lo permita, *señalando* una posición mediante un guión de las dimensiones de un carácter.	**OBJETO**	**ESPACIO**	**OPERACIÓN**	**ATRIBUTO**	El **Cursor de Texto** es un auxiliar del usuario para la *introducción* y *selección* del texto que compone ciertas partes de los **EDE**. A pesar de ser considerado como un objeto su instrumentación puede reemplazarse en la forma de una función de captura de texto.
	Cursor de Txto$_{15}$	Problema	Reconocer, Señalar, Seleccionar puntos, Introducir texto	Dispositivo, Lugar actual	

Los **Dispositivos$_{11}$** son **Elementos de Software** que *complementan* el *manejo* del **Área**, dando *acceso* a las partes que componen el **EGS**				
OBJETO	**ESPACIO**	**OPERACION**	**ATRIBUTO**	**Dispositivos** es una generalización de todos los elementos del **EGS** proyectado. Los agrupa y define su interacción con el usuario. Este tipo de enunciados denota la existencia de una generalización de la cual se desprenderán los Objetos **EDE** y **CEG.** Es importante mencionar que los **Dispositivos** Lógicos comparten características como se denota si observamos que todos los **dispositivos** que se presentan en pantalla tienen que ser dibujados sobre ella, a partir de un solo punto. Todos tienen que estar autodocumentados para ser usados adecuadamente. Una forma de documentación es el empleo de iconos que expliquen la acción a seguir después de seleccionar un **dispositivo** o mediante el desplegado de un mensaje en una **ventana**. Otro objeto como por ejemplo, una **barra de desplazamiento**, necesita otro tipo de elementos, como lo es mostrar un porcentaje de avance con un cursor y unos **botones** con flechas indicadoras de la dirección.
Dispositivos$_{11}$	Problema	Complementar el manejo	Elementos de Software	
Área$_4$	Problema	Accesar a los EGS		

La **Pantalla$_{12}$** es el **Dispositivo Físico** de Despliegue Gráfico que *Mostrará* al **Entorno Gráfico de Software (EGS).**				
OBJETO	**ESPACIO**	**OPERACIÓN**	**ATRIBUTO**	La **pantalla** es un elemento físico que posee ciertas características propias del hardware que afectan directamente al software, proporcionando una plataforma de ejecución que debe configurarse y prepararse para la modalidad gráfica.
Pantalla$_{12}$	Problema	Mostrar EGS	D. Físico de despliegue gráfico	
EGS$_1$	Problema			

El **Apuntador**$_{14}$ es un **Dispositivo** que permite el *señalamiento* y *selección* de cualquier **Punto** en la **Pantalla** que coincida con un **EDE** o **CEG**, su *ejecución* o -si es posible- su *transporte*.

OBJETO	ESPACIO	OPERACION	ATRIBUTO
Apuntador$_{14}$	Problema	Señalar, Seleccionar puntos, Transportar, Ejecutar	Dispositivo
Punto$_{36}$	Solución	Seleccionar	
Pantalla$_{12}$	Problema		
EDE$_5$	Problema		
CEG$_6$	Problema		

El **Apuntador** es otro elemento de software que modela a un **dispositivo** físico reconocido directamente por la computadora. Una pluma electrónica, una palanca de juegos (JoyStick) o un **Ratón** (Mouse) son los más conocidos, sin embargo el sistema sólo se diseñará para funcionar con un Mouse Microsoft (Tres botones), que es el más popular actualmente. Este aparato requiere -para su reconocimiento- de programación especializada que maneje las interrupciones definidas para su manejador (Driver).

Un **Punto**$_{36}$ es una coordenada en pantalla que puede ser *establecido*, *dibujado*, *borrado* y *movido*.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Punto$_{36}$	Problema	Establecer, Dibujar, Borrar, Mover	Coordenada

Como se mencionó para **dispositivo**, esta es la base para todos los elementos que forman el **EGS**

Un **Punto**$_{36}$ es una coordenada en pantalla que puede ser *establecido*, *dibujado*, *borrado* y *movido*.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Punto$_{36}$	Problema	Establecer, Dibujar, Borrar, Mover	Coordenada

Como se mencionó para **dispositivo**, esta es la base para todos los elementos que forman el **EGS**

El **Cursor de Texto$_{1S}$** permitirá *reconocer* el **Lugar Actual** en que será *introducido* el texto para cualquier dispositivo que lo permita, *señalando* una posición mediante un guión de las dimensiones de un carácter.

OBJETO	ESPACIO	OPERACIÓN	ATRIBUTO
Cursor de Texto$_{1S}$	Problema	Reconocer, Señalar, Seleccionar puntos, Introducir texto	Dispositivo, Lugar Actual

El **Cursor de Texto** es un auxiliar del usuario para la introducción y selección del texto que compone ciertas partes de los **EDE**.

A pesar de ser considerado como un objeto su instrumentación puede reemplazarse en la forma de una función de captura de texto.

IV.8 Comentarios al desarrollo del LDP

OBJETO - OPERACIÓN	COMENTARIO
Objeto: EGS_1	**** Considerado como sinónimo del sistema solución, relaciona al CEG con el Hardware. Se construirá un objeto denominado **Unidad Gráfica** encargado de inicializar el Ambiente Gráfico en la computadora y de conmutar al hardware a ese modo.
Operación: Permite Edición	**** Operación trasladada a **Área de Trabajo** y a **Carta ASM**.
Objeto: **Carta ASM**$_2$	**** Construido como un objeto independiente de **Dispositivo**, pues no pertenece al EGS.
Operación: Permite Edición	**** Operación trasladada a **Área de trabajo**.
Operación: Ensambla	**** Construida en forma de funciones para el manejo de una lista.
Objeto: $Usuario_3$	**** Abstracción que no necesita ser construida, pero podría ser representada por el Apuntador y el teclado alfanumérico.
Operación: Maneja Elemento	**** Operación trasladada a **Área de Trabajo**.
Operación: Construir	**** Operación trasladada a **Área de Trabajo**.
Operación: Ensamblar	**** Operación trasladada a **Cartas ASM**.
Operación: Indicar Secuencia	**** Operación trasladada a **Área de Trabajo**.
Objeto: **Área de Trabajo**$_4$	**** Construido como objeto dependiente de **Dispositivo**.
Operación: Visualizar **Elementos**	**** Construido como una operación inherente de **Área de Trabajo**.
Operación: Mostrar	**** Sinónimo de la operación anterior.

Objeto: EDE$_5$	**** Conceptualiza un grupo de elementos exclusivos para la edición, sin construir.
Operación: Edición	**** Operación identificada dentro de **Área de Trabajo**
Objeto: CEG$_6$	**** Conceptualiza al total de los elementos exclusivos del Entorno Gráfico (**Dispositivo**)
Operación: Permite Manipular Organizar	**** Construida independientemente por cada Objeto derivado de **Dispositivo** e Interpretar
Objeto: Superficie$_8$	**** Se considera un atributo de **Área de Trabajo**.
Operación: Delimitar	**** Pasa a ser un límite intrínseco a una función de **Área de Trabajo**
Objeto: Lista Oper$_9$	**** Agrupa objetos que administran al sistema (**Área de Trabajo**, MDT, etc), Construido como funciones independientes en distintos objetos
Operación: Administrar, Grabar, Recupera	**** Operaciones Trasladadas a sus respectivos objetos.
Objeto: **Elemento** ASM$_{10}$	**** Construido como objeto derivado de dispositivo que concentra las características comunes a los distintos elementos.
Operación: Seleccionar, Construir	**** Cada **Elemento** ASM se representa a sí mismo, la selección la opera el objeto **Carta ASM** y la Construcción es dirigida **por Área de Trabajo** y almacenada por **Carta ASM**.
Objeto: **Dispositivos$_{11}$**	**** Objeto fundamental para concentrar características de los objetos que componen al EGS
Operación: Controla Forma	**** Atributo de cada uno de los dispositivos, no construida
Operación: Interactuar	**** Atributo de cada uno de los dispositivos, no construida
Operación: Modificar Escala	**** Construida como una función ligada al botón de Zoom.
Operación: Desplazar	**** Construida como un nuevo objeto de **Barra de Desplazamiento**.

Objeto: Pantalla$_{12}$	**** Se considera sinónimo de **Área de Trabajo** y su operación se transfiere a éste.
Operación: Mostrar	**** Atributo asignado a **Área de Trabajo**.
Objeto: **Ventana$_{13}$**	**** Objeto que agrupa a otros tipos de ventana hereda de **Dispositivo**
Operación: Mostrar MDO	**** Construida.
Operación: Desplegar	**** Construida.
Objeto: Apuntador$_{14}$	**** Objeto complementario de EGS que se construye con nombre "**Ratón**" tanto para el modo texto como para el modo gráfico.
Operación: Selecciona **Dispositivos**	**** Función construida como parte del flujo principal del programa (main) basado en una operación de **Ratón** que obtiene la posición actual.
Objeto: Cursor Texto$_{15}$	**** En el modo gráfico el cursor texto no existe, pero puede ser manejado por funciones.
Operación: Introducir Datos	**** Construida como una función que coloca un cursor y espera información por teclado.
Operación: Dirigir Acciones	**** Considerada atributo.
Objeto: Algoritmo$_{16}$	**** Una abstracción de una función o de un procedimiento sin necesidad de implantarse.
Operación: Ejecutar	**** Atributo de cualquier función, eliminada.
Objeto: Parámetros$_{17}$	**** Pueden incorporarse al sistema como parte del objeto de menú o como objeto en sí.
Operación: Cambiar	**** Construida como proceso implícito del menú.
Objeto: Menú Desplegable de Texto$_{18}$	**** No necesariamente un objeto, si se organiza de manera adecuada puede ser una extensión funcional del **Área de Trabajo**.
Operación: Agrupar	**** Considerado atributo, es una consecuencia de utilizar un menú.

Operación: Mostrar	**** Construida como una función que hace uso de **Ventana**
Objeto: Menú de Iconos (MDI)$_{19}$	**** Dado que los iconos pueden crearse con **Botones**, el menú es la designación a la función que hace aparecer a dichos objetos
Operación: Agrupar	**** Considerada atributo
Operación: Mostrar	**** Operación trasladada al flujo principal del programa por su escaso uso posterior
Objeto: **Estado**$_{20}$	**** Objeto derivado de **Elemento** ASM.
Operación: Presentar	**** Construida localmente como "Dibuja".
Operación: Seleccionar	**** Construida en **Elemento** ASM.
Operación: Mover	**** Construida originalmente en **Elemento** ASM, fue trasladada a **Dispositivo**.
Operación: Modificar	**** Construida.
Objeto: **Salida Condicional** $_{21}$	**** Objeto derivado de **Elemento** ASM.
Operación: Presentar	**** Construida localmente como "Dibuja".
Operación: Seleccionar	**** Construida en **Elemento** ASM en Coordinación de **Área de Trabajo**.
Operación: Mover	**** Construida originalmente en **Elemento** ASM pero movida a **Dispositivo** como algo más general aplicable a otros objetos.
Operación: Modificar	**** Construida.

Objeto: **Decisión**$_{22}$	**** Objeto derivado de **Elemento** ASM.
Operación: Presentar	**** Construida.
Operación: Seleccionar	**** Construida en **Elemento** ASM en coordinación de **Área de Trabajo**.
Operación: Mover	**** Construida en **Dispositivo**.
Operación: Modificar	**** Construida.
Objeto: Valores$_{23}$	**** Objeto omitido por hacer referencia a valores que por su cantidad reducida serán mantenidos implícitamente en el sistema.
Operación: Determina Características	**** Considerada atributo de los valores mantenidos dentro del sistema.
Objeto: Submenús$_{25}$	**** Es un **Elemento** constituyente del MDT que puede desarrollarse como funciones.
Operación: Mostrar **Ventana**	**** Construida.
Operación: Seleccionar	**** Construida.
Operación: Emerger Verticalmente	**** Operación distinta a las de **Ventana**, puesto que sobre ella se elegirá un Texto.
Operación: Mostrar Contenido	**** Operación común a todas las **Ventanas** "Implantada".
Objeto: **Ventana Dialogo** $_{26}$	**** Como **Ventana**, pero con **Botones** de interacción (OK, Cancelar, etc.).
Operación: Capturar Información	**** Operación que podría hacer uso del Cursor de Texto.
Objeto: Opciones$_{27}$	**** Elementos de los Submenús, pueden ser conjunto de apuntadores a funciones.
Operación: Seleccionar	**** Operación instrumentada como procedimiento en el MDT.

Objeto: **Botones**[28] **** Objeto dependiente de **Punto** para dibujarse que "responde" a su selección.

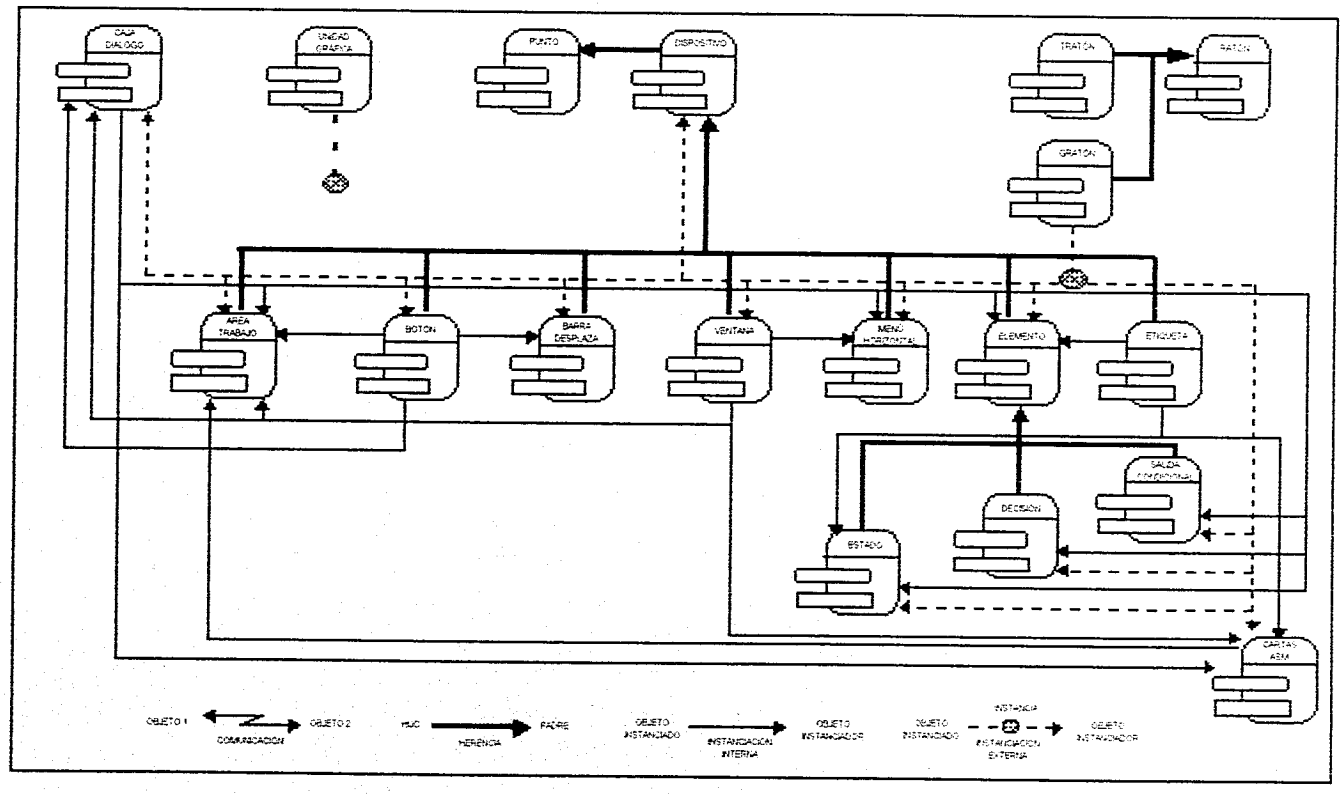
Operación: Seleccionar **** Operación que aparentará el presionado visual del **Botón**.

Objeto: Operación[29] **** Otro sinónimo de una función o procedimiento, esta vez asignado a un **Botón**.

Operación: Ejecutar **** Atributo del objeto **Botón**.

Objeto: Punto[30] **** La Base de cualquier elemento gráfico, puesto que todas las imágenes están formadas por puntos, es la clase inicial para el dibujado en pantalla.

Operación: Seleccionar **** Construida


CAJA DIALOGO
UNIDAD GRAFICA
PUNTO
DISPOSITIVO
TRATOR
RATON
GRATOR
AREA TRABAJO
BOTON
BARRA DESPLAZ
VENTANA
MENU HORIZONTAL
ELEMENTO
ETIQUETA
SALIDA CONVENCIONAL
DECISION
ESTADO
CARTAS ASM
OBJETO 1
COMUNICACION
OBJETO 2
HIJO
HERENCIA
PADRE
OBJETO INSTANCIADO
INSTANCIACION INTERNA
OBJETO INSTANCIADOR
OBJETO INSTANCIADO
INSTANCIA
INSTANCIACION EXTERNA
OBJETO INSTANCIADOR

CONCLUSIONES

Conclusiones

Adentrarse en un proyecto como éste -el diseño e implementación de una interface gráfica con OO para la captura de Cartas ASM-, permite al participante encarar situaciones que sin duda difícilmente enfrentaría en el mundo laboral, por ejemplo, en la actualidad se ha extendido mucho el uso de lenguajes "visuales" que ponen en manos de los programadores las herramientas necesarias para el manejo gráfico de interfaces predefinidas como el Microsoft Windows. Sin embargo, los sistemas basados en GUI's cuando se generalizan tienden a presentar un crecimiento considerable que debe ser soportado por un hardware cada vez más capaz.

Los ambientes gráficos son pues, voraces consumidores de tiempo y esfuerzo de parte de los programadores, así como de recursos de cómputo; pero si se diseñan adecuadamente otorgan una poderosa herramienta de comunicación con los usuarios de los sistemas y, si a eso le aunamos las ventajas de la OO, tendremos un sistema que además de versátil puede ser reutilizado total o parcialmente. El producto del proyecto, un editor gráfico totalmente funcional, responde a las expectativas del subobjetivo uno, como herramienta de software ha sido probado por el equipo de desarrollo, auxiliando el diseño de controladores digitales de pruebas.

Por otra parte el entorno gráfico creado, que opera sobre DOS y funciona con monitores de vídeo a color o monocromáticos, cumple con el aprovechamiento de hardware existente. Aún más, siendo un sistema que opera sobre máquinas AT 286 ó superiores, no exige software ni hardware adicional. además, la OO permite a sus componentes sacar ventaja de ello y ser reutilizados y fácilmente modificables.

El conjunto de reglas que gobiernan al diseño con Cartas ASM, al ser trasladadas al dominio del software -subobjetivo 2- permitió hacer el modelado de un procedimiento-método muy semejante al de los diagramas de flujo y así poder generalizar su uso a toda aquella tarea capaz de ser descrita por un fluxograma.

En cuanto a los conceptos de la OO utilizados (subobjetivo 4), podemos indicar que la gran mayoría de ellos fueron usados en el sistema, ejemplificándose casi en su totalidad en forma de elementos gráficos, fácilmente reconocibles por el usuario interesado en el tema. Si bien es cierto que la TOO puede facilitar enormemente el mantenimiento y la extensión de sistemas ya existentes, demanda una gran cantidad de tiempo inicial y un alto volumen de código. La información resultante de la captura (subojetivo 3), se pone a disposición de quien pretenda utilizarla, estructurada en dos archivos, cuyos formatos se describen en el Apéndice F.

Las perspectivas de un sistema con las características de éste, son en verdad prometedoras, como pueden serlo por ejemplo:

El aprovechamiento de la interface para la síntesis de controladores utilizando otras arquitecturas microprogramadas a parte de la implementada, como podrían ser el diseño con mica II, con bit slice o con dispositivos PLD's.

Considerando que dentro del sistema se tendrían todas las características tanto físicas como de funcionamiento del controlador, se pueden construir módulos que aprovechen esta información y realicen la simulación del funcionamiento del circuito.

La resolución de circuitos eléctricos por medio de los métodos de Kirchhoff o Maxwell, la documentación de procedimientos por fluxogramas, la programación visual, las aplicaciones administrativas, etc., etc.

El sistema esta desarrollado bajo el popular ambiente DOS usando el compilador de C++, pero como una adaptación seria deseable hacer un desarrollo para su manejo en otras plataformas tanto de software como de hardware, como pueden ser:

Hardware:

- HP-9000
- Silicon Graphics
- Pentium
- Next
- etc.

Software:

- Windows
- Unix
- NextStep
- Mac
- Vms
- etc.

Como ya hemos dicho con anterioridad, más que un trabajo documental, tiene por intención la de ser más bien un ejercicio de conjugación tecnológica que ayude al interesado a asimilar más

fácilmente nuestra experiencia, por tal motivo hemos anexado un disquete con el código fuente generado y que esperamos sea de utilidad.

El equipo de desarrollo.

APÉNDICE A

CONTROL DE IMPRESIÓN GRÁFICA CON EL ESTÁNDAR EPSON

APÉNDICE A

CONTROL DE IMPRESIÓN GRÁFICA CON EL ESTÁNDAR EPSON

Para las aplicaciones gráficas la presentación de resultados debe ser acompañada de una transferencia en papel, y cada usuario espera el reflejo fiel del trabajo que ha realizado a través de la imagen del monitor, este concepto ha sido tomado por aplicaciones como el Microsoft Windows, que anuncia la posibilidad de WYSIWYG (What You See is What You Get, "lo que ve, es lo que obtiene") como característica principal de su interface en oposición a la impresión por caracteres que dominaba hasta entonces la escena. Sin embargo, la impresión gráfica aunque es de gran utilidad se enfrenta al problema de los estándares de impresión electrónica basados mayormente en códigos de caracteres, lo que hace al desarrollo de rutinas de impresión gráfica una actividad altamente compleja.

Una impresora de puntos moderna cuenta con varios elementos para realizar su trabajo:

1.- Una cabeza de impresión para impactar a la cinta sobre el papel.

2.- Un puerto de entrada para recepción paralela de bits bajo el estándar Centrónics de asignación de patillas, y alternativamente un puerto serial para el mismo fin.

3.- Un servomecanismo que posiciona la cabeza a lo largo de una línea guía.

4.- Una memoria que almacena los mapas de caracteres y su código correspondiente, así como los códigos asignados para el control directo del servomecanismo de la impresora.

5.- Un controlador electrónico que interpreta la información recibida por el puerto Centronics, mediante los códigos almacenados para hacer funcional el servomecanismo.

En la impresión de gráficos, podemos manejar dos técnicas para su realización. En la primera, por GRÁFICOS DE BLOQUES, se utiliza el conjunto de caracteres semi-gráficos (definidos en los códigos ASCII del 169 al 223) para formar recuadros y líneas rectas, utilizando la impresora en modo texto; es decir, para imprimir líneas o rectángulos (o ambos) usando esta técnica, no es necesario conmutar la impresora al modo gráfico, puesto que las líneas que utilizamos en la construcción de los gráficos, se generan del mismo modo que se genera cualquier carácter alfanumérico. La segunda técnica, por GRÁFICOS DE PUNTOS, permite (por medio de programación) especificar exactamente donde será impreso cada punto. Para utilizar esta técnica, debemos conmutar la impresora al modo gráfico, puesto que aquí tendremos una resolución de manejo a nivel de puntos y no a nivel de caracteres, como en el modo texto.

Como el sistema está manejado en el modo gráfico, la impresión de los resultados se realizará también de esta forma, por lo que profundizaremos en el manejo de la técnica de gráficos de puntos.

En este apéndice, se explica, en forma general, las bases del funcionamiento de un controlador de gráficos para realizar la impresión monocromática de pantallas gráficas sobre impresoras de matriz de puntos. Las operaciones gráficas con matrices de puntos que aquí se describen, han sido diseñadas para cabezas de impresión de 9 agujas, por lo que no son directamente compatibles con las impresoras matriciales que utilizan cabezas de impresión de 24 agujas. Las operaciones, sin embargo, son análogas.

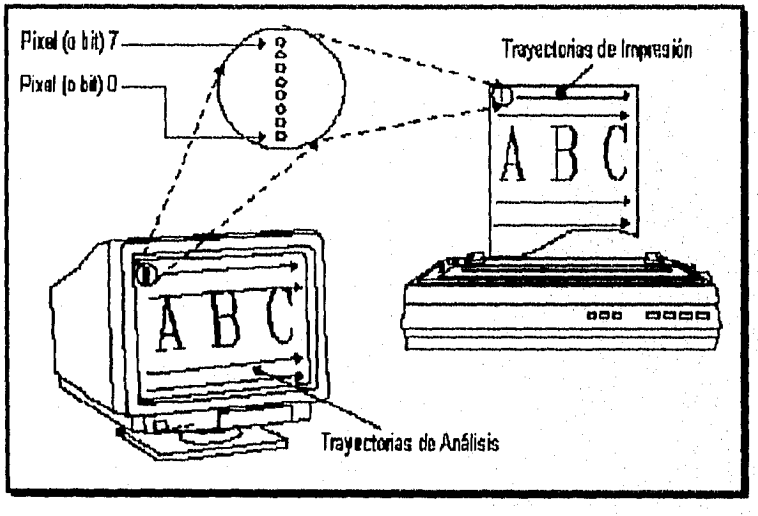
La impresora Epson FX-85 es considerada como el dispositivo estándar de las impresoras matriciales. Esto nos asegura la compatibilidad con las series MX, RX, y con la mayor parte de las impresoras de la serie LX (inyección o chorro de tinta) y las impresoras matriciales de otros fabricantes. Aunque casi todas ofrecen modos gráficos compatibles con Epson, se aconseja consultar el manual correspondiente con objeto de conocer el grado de compatibilidad y poder adaptar las funciones gráficas de la manera más conveniente. La impresora FX-85 ofrece ocho modos de operación gráfica según se muestra en la siguiente tabla.

0	Sencilla	60	dpi[4]	16	pulg/seg
1	Doble, velocidad baja	120	dpi	8	pulg/seg
2	Doble, velocidad alta	120	dpi	16	pulg/seg[1]
3	Cuádruple	240	dpi	8	pulg/seg[1]
4	CRT I	80	dpi	6	pulg/seg[2]
5	Plotter uno a uno	72	dpi	12	pulg/seg[2]
6	CRT II	90	dpi	8	pulg/seg
7	Plotter doble densidad	144	dpi	3	pulg/seg[3]

1. No imprime puntos consecutivos en ninguna fila.
2. Coincide con la densidad de pantalla Epson QX-10.
3. Modos del trazador gráfico; proporciona una densidad horizontal de puntos de uno a uno.
4. dpi (del ingles; Dot Per Inches) puntos por pulgada.

La representación de imágenes gráficas en papel se puede hacer en modo HORIZONTAL (apaisado[1]) y en modo VERTICAL. La orientación vertical representa el eje X de la pantalla a lo ancho de un papel, y el eje Y a lo largo del mismo, dando origen a una imagen de media hoja. La orientación horizontal relaciona el eje largo del papel con el eje X de la pantalla y el eje Y de la pantalla con el eje ancho del papel, produciendo una única imagen de pantalla por página. Esta última suele ser la orientación de salida preferida.

En el modo vertical, los pixeles de la pantalla son mapeados (o analizados) en grupos verticales de 8 (más adelante se verá por que en grupos de 8), empezando por la parte superior izquierda de la pantalla, con líneas de análisis que se desplazan hacia la derecha. En la impresora, el movimiento de la cabeza de impresión es el normal, es decir, con un desplazamiento (al imprimir) de izquierda a derecha, cuando se termina con una línea o renglón de impresión, se provoca un avance de papel y al mismo tiempo la cabeza de impresión regresa al extremo izquierdo.

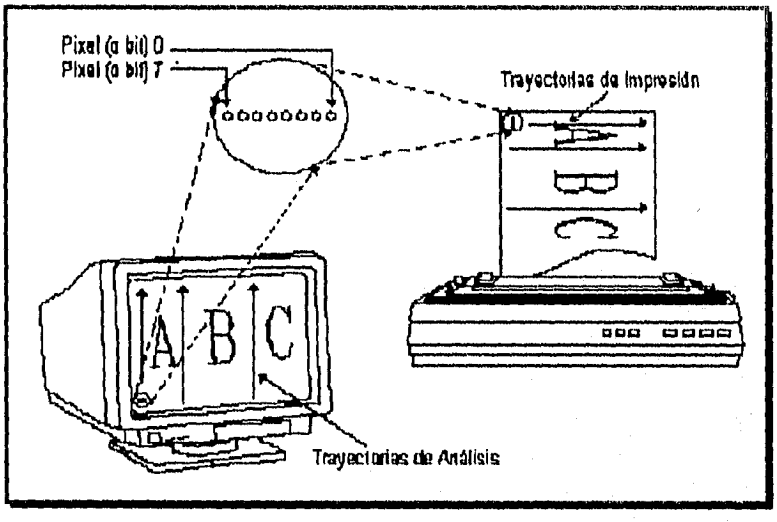


Representación Vertical (Portrait)

En modo horizontal, la dirección de impresión y el avance del papel permanecen constantes, pero los pixeles de la pantalla se examinan en grupos horizontales de ocho, con una línea de análisis que comienza en la parte inferior izquierda de la pantalla y se desplaza hacia arriba. Las siguientes líneas de

[1] Apaisado. Que es más ancho que largo.

análisis comienzan examinando en la parte inferior de la pantalla los siguientes 8 pixeles horizontales a la derecha de los de la anterior línea, la última línea de impresión examina los pixeles de más a la derecha de la pantalla (como se muestra en la siguiente figura).

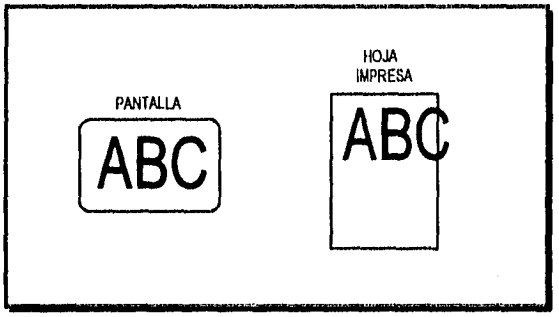


Representación Horizontal (Apaisada o Landscape)

La forma de indicar a la impresora como colocar los grupos de 8 puntos en el papel, será explicada más adelante.

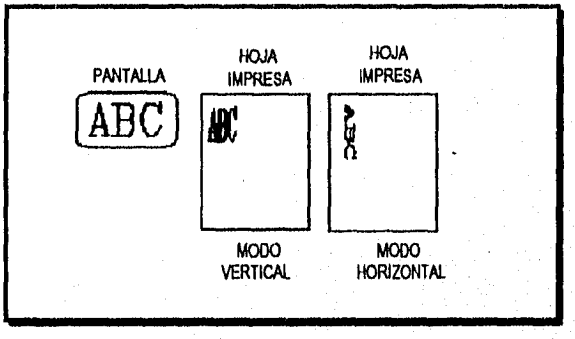
Para imprimir una pantalla gráfica hay que considerar tres criterios antes de seleccionar el modo de impresión:

Primero, el número de puntos por pulgada (ppp) debe ser lo suficientemente elevado para proyectar los pixeles de la pantalla dentro de los límites de la página física. Con una resolución de 60 ppp (modo 0) en la orientación vertical podría necesitarse una anchura de papel superior a las ocho pulgadas. Un valor de 60 ppp podría permitir que una impresora de 13 pulgadas de ancho imprimiera 780 pixeles en horizontal, pero con este valor, una impresora de ocho pulgadas sólo podría imprimir 480 pixeles.



Segundo, los puntos por pulgada en horizontal y las líneas por pulgada en vertical deben estar equilibrados para que la imagen de salida se ajuste lo máximo posible a la imagen de la pantalla. Los ppp horizontales pueden modificarse, no así los verticales (o no tan fácilmente).

Tercero, puesto que la imagen se imprime en formato uno a uno (un pixel/un punto), cuanto más alta sea la densidad de impresión (puntos por pulgada), menor será la imagen resultante. Por ejemplo, el modo 3 imprime una imagen de pantalla completa en modo vertical con una anchura de sólo 2.7 pulgadas, aproximadamente, (en proporciones muy distorsionadas).



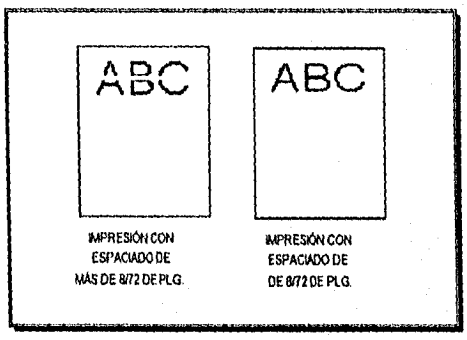
Para comprender las gráficas de puntos, usted necesitará conocer un poco acerca de cómo trabaja la cabeza de impresión.

La cabeza de impresión tiene 9 agujas y se mueve cruzando la página; impulsos eléctricos activan las agujas. Cada vez que una aguja se activa, golpea la cinta entintada y hace presión sobre el papel para producir un pequeño punto. Como la cabeza se mueve cruzando el papel, las agujas se activan cada vez en diferentes ubicaciones, para producir letras, números o símbolos.

En el modo gráfico, la epson imprime una columna de ocho puntos por cada código recibido y, utiliza solamente ocho de las nueve agujas, por lo que su programa gráfico deberá enviar códigos para modelos de ocho puntos; uno para cada columna en una línea impresa. Por cada una de estas columnas, la cabeza de impresión imprime el modelo de puntos que se haya especificado.

Para imprimir figuras mayores de ocho puntos, la cabeza de impresión realiza más de una pasada. Imprime una línea, avanza el papel y después imprime otra, y así sucesivamente.

Para evitar que la cabeza de impresión deje espacios entre líneas gráficas, tal como lo hace con líneas de texto, la línea de espaciado deberá ser modificada. Con un cambio en la línea de espaciado, la epson puede imprimir imágenes gráficas finamente detalladas, formando líneas adyacentes de no más de 8/72 de pulgada de alto.



Con cada pasada de la cabeza de impresión, se imprime una pieza del modelo total, el cual puede ser tan alto o tan bajo, tan amplio o tan angosto como usted lo desee y la impresora lo permita.

El modo gráfico requiere un método para decirle a la impresora qué agujas debe activar en cada columna. Existen 256 posibles combinaciones con las ocho agujas, usted necesitará emplear un sistema de numeración que le permita usar un solo número para especificar cual de los 256 posibles modelos desea. Etiquetando cada aguja con su propio número, usted puede usar un sistema numérico que le permita especificar exactamente que agujas deben ser activadas.

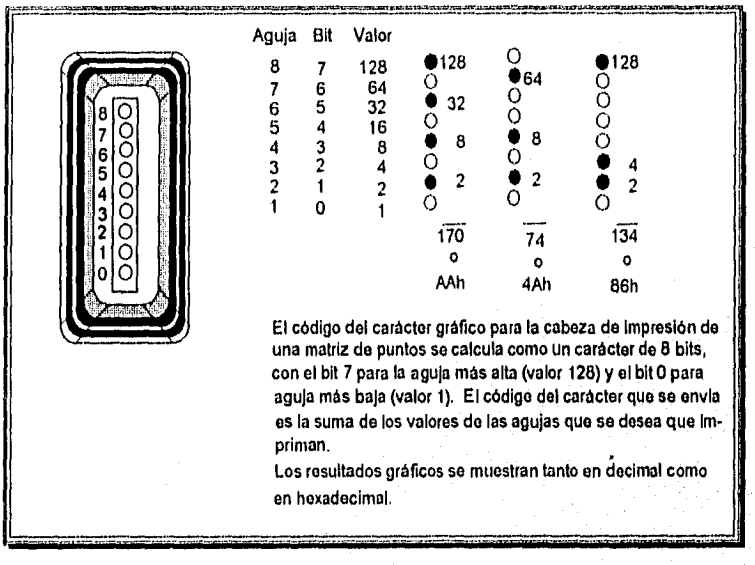
Para activar cualquier aguja, usted envía un número de acuerdo al sistema numérico explicado a continuación.

Cuando se utiliza una impresora de matriz de puntos para salidas normales (alfanuméricas), existe un código de caracteres de 8 bits que selecciona un patrón de 9x9 agujas entre los juegos de caracteres de la ROM de la impresora. En los modos gráficos, sin embargo, sólo se utilizan 8 de estas agujas y se

envía un carácter de 8 bits por cada posición horizontal de la cabeza de impresión, donde cada bit controla una de las agujas (como ya se mencionó antes).

El código del carácter gráfico se calculará de forma que la aguja más alta se controle con el bit de peso 7 del carácter, y así sucesivamente hasta llegar a la aguja más baja, que se controlará con el bit de peso 0. Para activar más de una aguja a la vez, sume el número de agujas a activar y envíe la suma a la impresora.

En la siguiente figura se muestran tres ejemplos de cálculo de caracteres gráficos.



Así, de acuerdo a las etiquetas de agujas, para activar la aguja superior usted deberá enviar el número 128. Para activar la aguja inferior usted deberá enviar el número 1. Si usted desea activar ambas agujas simplemente sume 128 y 1 y envíe 129.

Hasta este momento, se ha explicado como se generan los códigos que definen los caracteres gráficos, pero no se ha dicho como se le comunican a la impresora, esto se hará a continuación.

Comunicación Computadora-Impresora

Su computadora se comunica con su impresora usando un conjunto estandarizado de códigos numerados llamados códigos ASCII. Cuando usted oprime la letra "A" en su teclado, ésta es trasladada

al código ASCII para "A", transmitida a un recurso periférico como la pantalla de su computadora o su impresora, y después convertida nuevamente en la letra "A".

Existen códigos ASCII para todas las letras del alfabeto, tanto mayúsculas como minúsculas, y para números del 0 al 9. El conjunto de códigos ASCII también incluye la mayoría de los signos de puntuación y algunos códigos que controlan funciones de impresión.

Todas las letras, números y, signos de puntuación están asignados con números decimales de 32 a 255. Los códigos ASCII con valor decimal menor que 32 son llamados "códigos de control", debido a que ellos controlan la operación de su impresora y otros periféricos. Estos caracteres ASCII no corresponden por lo general a las teclas del teclado y no pueden ser impresos por su impresora.

Secuencias de Escape

Existen más de 30 códigos de control disponibles para la operación de su impresora, sin embargo, muchos más códigos son requeridos en la operación diaria de ésta. Los códigos ASCII son agrupados en secuencias para indicar a la impresora ciertas funciones u operaciones, estas secuencias utilizan los códigos ASCII con valores decimales del 32 al 255 (reservados normalmente para caracteres y puntuación) para controlar las funciones de impresión; esto se hace enviando, al inicio de la secuencia, un código estándar para decirle a la impresora que los códigos que le siguen serán usados como códigos de control, no como caracteres o signos de puntuación.

El código estándar que es enviado al comienzo de estas secuencias es el código escape (ESC), valor decimal 27. Cualquier secuencia de códigos que comienza con el código ESC es llamada **Secuencia de Escape**.

Comandos de Impresión

Para que la impresora reconozca las instrucciones recibidas, el código ASCII deberá ser enviado en un formato especial, llamado comando. Una secuencia de escape es un comando, tal como cualquier código ASCII o secuencia de códigos que instruyen a la impresora para la realización de una función específica. Los programas de aplicación envían continuamente comandos a la pantalla de su computadora y a su impresora. Estos comandos instruyen a la impresora para la realización de diversas funciones como la impresión de estilos de tipos, alimentar una cantidad de papel después de imprimir cada línea y comenzar a imprimir sobre un lugar específico de la página.

Los comandos listados en el "Sumario de Comandos" dentro de este apéndice, constan de varias combinaciones de códigos ASCII y son agrupados por el tipo de función que realizan.

Algunos de los comandos incluyen alguna variable como la letra *n*. Por ejemplo, el comando para seleccionar o cancelar el modo doble alto es *ESC w n*. Cuando *n* = 1, el modo doble alto está activado y cuando *n* = 0 el modo doble alto es desactivado.

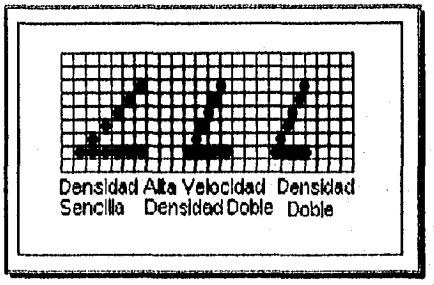
El formato del comando dependerá de su programa de aplicación. Algunos programas de aplicación aceptan solamente el formato decimal mientras que otros requieren cierta puntuación. Algunos programas no le permiten insertar comandos de impresión.

Caracteres Gráficos

El modo de gráficas de puntos le permite a su impresora producir caracteres personalizados dando las instrucciones adecuadas.

A manera de ejemplo se mostrará un carácter definido dentro de un arreglo de puntos de 8 x 8.

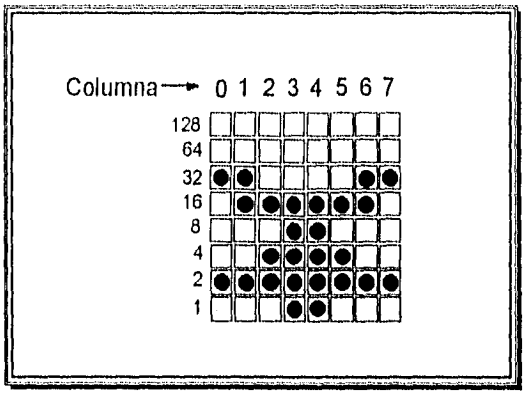
Primero se debe planear la forma del carácter en un área cuadriculada; pero antes de comenzar a colocar los puntos, deberá decidir que densidad de puntos desea. La siguiente figura muestra los tres modos más comunes.



En esta figura se pueden observar las principales reglas para el diseño de caracteres (y de gráficas en general) en las tres densidades:

- En densidad sencilla, los puntos no pueden ser colocados en líneas verticales.
- En alta velocidad doble densidad, los puntos pueden ser colocados en líneas verticales pero, no pueden encimarse.
- En doble densidad, los puntos pueden colocarse en líneas verticales y encimarse.

Ahora se muestra el diseño del carácter a definir en el modo de densidad sencilla.



Después de dibujar los puntos en el cuadriculado, calcule el código para cada modelo de aguja examinando columna por columna; en el ejemplo tendríamos:

Columna:	0	1	2	3	4	5	6	7
Código:	34	50	22	31	31	22	50	34

Los datos obtenidos se mandan a la impresora después de haber conmutado ésta al modo gráfico. A continuación se muestra como podrían quedar las instrucciones para imprimir el carácter definido, utilizando el lenguaje C.

Para empezar colocaremos en un arreglo los valores obtenidos.

```
char codigos[8] = [34, 50, 22, 31, 31, 22, 50, 34 };
```

A continuación le indicamos a la impresora que los siguientes ocho datos a recibir debe manejarlos como códigos gráficos de densidad sencilla (consultar el comando ESC K, en el "Sumario de comandos").

```
fprintf (strdprn, "\x1BK%c%c",8,0);
```

Lo único que resta es mandar los 8 datos, esto lo hacemos con las siguientes instrucciones:

```
for (int x = 0; x < 8; x++)
        fprintf (stdprn, "%c", codigos[x]);
```

Lo anterior se aplica a un carácter de 8 x 8 puntos, pero podemos hacer lo mismo para un carácter mayor o una gráfica o dibujo completo. Las instrucciones son dadas en lenguaje C pero puede usarse otro lenguaje con características adecuadas.

La manera más rápida para la realización de gráficas en su impresora es utilizar un programa comercial de gráficas. Por lo general, estos programas crean una imagen en su monitor y después proporcionan un comando para enviar la imagen a la impresora.

Si su programa de aplicación produce gráficas, todo lo que necesita para imprimir gráficas de puntos es aprender a manejar su programa de aplicación.

Comandos gráficos

Los comandos gráficos son diferentes de la mayoría de los otros comandos. Para la mayoría de los modos de impresión, tales como enfatizado y expandido (ver "Sumario de Comandos"), un comando lo activa y otro lo desactiva. Para gráficas, éste es más complicado debido a que además de activar un modo gráfico, especifica también la cantidad de columnas de gráficas que serán impresas. Después de que la impresora reciba este comando, interpretará los siguientes datos como códigos de modelos de agujas y los imprimirá sobre el papel.

Formato de Comandos Gráficos

Existen distintos comandos gráficos proporcionando diferentes densidades horizontales de puntos y velocidades de impresión. A continuación se muestra un ejemplo, usando el comando gráfico en densidad sencilla ESC K. En las gráficas de densidad sencilla existen horizontalmente 60 puntos por pulgada horizontal.

El comando para dar entrada al modo gráfico de densidad sencilla es ESC K n1 n2. En lenguaje C el comando es dado en el siguiente formato:

```
fprintf (stdprn,"\x1B \x4B %c%c",n1,n2);
```

o

```
fprintf (stdprn,"\x1B K %c%c",n1,n2);
```

En este comando, ESC K (en hexadecimal: \x1B \x4B) selecciona gráficas de densidad sencilla y, n1 y n2 especifican el número de columnas a reservar para gráficas.

Número de columnas reservadas

Debido a que una línea puede usar cientos de columnas, y el valor mayor de un código ASCII es 255 (decimal), los comandos gráficos requieren de dos números (n1 y n2) para especificar cuantas columnas deberán ser reservadas; de tal forma que el número total de columnas sea igual a n1+(n2 x 256); por lo tanto,

para calcular n1 y n2, divida el número total de columnas entre 256; el resultado es n1 y el restante es n2. Debido a que el comando es especificado para dos números, usted deberá proporcionar dos números aún si necesita uno sólo. Cuando requiera menos de 256 columnas, solamente sustituya n1 por el número de columnas que usted está reservando y n2 por un cero.

Por ejemplo, si usted desea enviar 1632 columnas de datos gráficos, n1 deberá ser 6 que es el resultado de dividir 1632 entre 256, y n2 deberá ser 96, con esto tenemos: 1632 = 96+(6 x 256) = n1 + (n2 x 256).

Después de recibir un comando gráfico como ESC K, la impresora imprime el número de códigos especificados por n1 y n2 como datos gráficos no importando de que códigos se trate. Esto significa que usted deberá asegurarse de proporcionar exactamente la cantidad adecuada de datos gráficos. Si proporciona pocos datos, la impresora se detendrá y esperará más datos, el siguiente dato enviado será impreso después como gráfica, aun si éste es en realidad texto. Por otro lado, si usted proporciona demasiados datos gráficos, el exceso será impreso como texto regular.

Programando gráficas

A continuación un ejemplo muestra como un comando gráfico, números de columnas reservadas y datos pueden ser usados para imprimir una simple línea de gráficas. El ejemplo se expresa en lenguaje C, usted por supuesto, puede utilizar otro lenguaje, los principios son los mismos.

La primera línea del ejemplo especifica gráficas de densidad sencilla para 40 columnas:

```
fprintf (stdprn, "\1B \x4B %c%c",40,0);
```

La segunda línea es el dato (el número 74) que es impreso como modelo de puntos. La instrucción for envía 40 columnas de datos:

```
for (int x=1, x==40, x++) fprintf (stdprn, "%c", 74);
```

Algunos programas de aplicación insertan automáticamente códigos de retorno de carro y avance de línea después de cada 80 ó 130 caracteres; por lo general, esto no resulta un problema con texto pero puede dañar sus gráficas. Dos columnas extra de gráficas son impresas a la mitad de la que usted envió y, dos datos son impresos a la izquierda como texto. Usted puede prevenir códigos de control no deseados en gráficas, colocando instrucciones que indiquen el ancho del renglón de impresión.

A continuación se listan y describen algunos (los de interés para la aplicación, para mayor información acerca de los comandos existentes, consulte el manual de su impresora) de los comandos del estándar Epson, divididos en los siguientes temas:

- Operación de la impresora.
- Control de datos.
- Movimiento vertical.
- Movimiento horizontal.
- Estilos de impresión avanzada.
- Mejoras de impresión.
- Procesando palabras.
- Conjunto de caracteres.
- Gráficas.
- Tamaño de impresión.

Cada comando tiene una sección de formato y una de comentarios. La sección de formatos, proporciona los valores ASCII, decimal y hexadecimal para el comando; la sección de comentarios, describe los efectos de los comandos y proporciona información adicional para el uso de éstos.

Los tres formatos son equivalentes, esto facilitará la elección de uno, de acuerdo a los propósitos para los que se necesite.

El tipo de comando más simple consta de un solo carácter que es enviado a la impresora. Por ejemplo, para imprimir en modo condensado, el formato del código es:

Modo ASCII:	SI
Decimal:	15
Hexadecimal:	0F

Este código puede ser enviado desde el programa, mediante el envío del código 15 directamente.

Los comandos más complejos constan de dos o más códigos de carácter. Por ejemplo, para imprimir en el modo expandido, el formato de código es el siguiente:

Modo ASCII:	ESC	W	n
Decimal:	27	87	n
Hexadecimal:	1B	57	n

En este caso **n** puede ser 1 ó 0, para activar o desactivar el modo expandido. Usted puede usar el siguiente comando, para activar la impresión de modo expandido desde C:

```
fprintf (stdprn, "\x1B \x57 %c", 1);
```

Para los siguientes comandos que utilizan solamente 0 ó 1 para la variable, pueden ser usados los códigos ASCII 1 y 0 ó los caracteres ASCII 1 y 0.

ESC s, ESC U, ESC x, ESC p, ESC W, ESC S, ESC -, ESC w, y ESC #.

Por ejemplo, en el lenguaje C, usted puede modificar el modo doble alto con la siguiente instrucción:

```
fprintf (stdprn, "\x1Bw%c", 1);
```

Tabla de tecla control

Algunos programas de aplicación pueden utilizar códigos de tecla **control** para valores decimales de 0 a 27. La tabla siguiente le muestra los valores apropiados. La columna de **tecla control** indica que usted deberá oprimir la tecla **control** al mismo tiempo que oprime la tecla para la letra o símbolo en la columna. Por ejemplo, usted oprime la tecla **control** y la letra "A" al mismo tiempo para enviar el valor 1.

Nota: Algunos programas utilizan la tecla **control** para otros propósitos. Además, algunos programas no utilizan todas estas teclas.

DEC	HEX	Tecla Control
0	00	@
1	01	A
2	02	B
3	03	C
4	04	D
5	05	E
6	06	F
7	07	G
8	08	H
9	09	I
10	0A	J
11	0B	K
12	0C	L
13	0D	M

DEC	HEX	Tecla Control
14	0E	N
15	0F	O
16	10	P
17	11	Q
18	12	R
19	23	S
20	14	T
21	15	U
22	16	V
23	17	W
24	18	X
25	19	Y
26	1A	Z
27	1B	[

Sumario de comandos

A continuación se muestra una lista con algunos de los comandos Epson (clasificados de acuerdo a la función que realizan), con sus valores decimales y hexadecimales.

OPERACIÓN DE LA IMPRESORA			
ASCII	DEC.	HEX.	DESCRIPCIÓN
ESC BEL	7	07	Alarma.
ESC DC1	17	11	Selecciona impresora.
ESC DC3	19	13	Deshabilita impresora.
ESC 8	56	38	Deshabilita el sensor de fin de papel.
ESC 9	57	39	Habilita el sensor de fin de papel.
ESC @	64	40	Inicializa impresora.
ESC s	115	73	Activa / desactiva modo de alta velocidad.

CONTROL DE DATOS			
ASCII	DEC.	HEX.	DESCRIPCIÓN
CR	13	0D	Retorno de carro.
CAN	24	18	Cancela línea.
DEL	127	7F	Borra carácter.

MOVIMIENTO VERTICAL			
ASCII	DEC.	HEX.	DESCRIPCIÓN
LF	10	0A	Avance de línea.
FF	12	0C	Avance de página.
ESC 0	48	30	Selecciona 1/8 de pulgada de línea de espaciado.
ESC 1	49	31	Selecciona 7/72 de pulgada de línea de espaciado.
ESC 2	50	32	Selecciona 1/6 de pulgada de línea de espaciado.
ESC 3	51	33	Selecciona n/216 de pulgada de línea de espaciado.
ESC A	65	41	Selecciona n/72 de pulgada de línea de espaciado.
ESC C	67	43	Especifica longitud de página en líneas.
ESC J	74	4A	Ejecuta n/216 de pulgada de avance de línea.
ESC N	78	4E	Especifica el salto sobre perforaciones.
ESC O	79	4F	Cancela el salto sobre perforaciones.

MOVIMIENTO HORIZONTAL			
ASCII	DEC.	HEX.	DESCRIPCIÓN
HT	9	09	Tabulador horizontal.
ESC Q	81	51	Especifica margen derecho.
ESC l	108	6C	Especifica margen izquierdo.

TAMAÑO DE [illegible] Y ANCHO DE CARACTERES			
ASCII	DEC.	HEX.	DESCRIPCIÓN
SO	14	0E	Selecciona modo expandido (una línea).
SI	15	0F	Selecciona modo condensado.
DC2	18	12	Cancela modo condensado.
DC4	20	14	Cancela modo expandido (una línea).
ESC M	77	4D	Selecciona 12 cpp.
ESC P	80	50	Selecciona 10 cpp.
ESC W	87	57	Activa / desactiva modo condensado.
ESC p	112	70	Activa / desactiva el modo proporcional.
ESC w	119	77	Activa / desactiva modo doble alto.

[illegible]			
ASCII	DEC.	HEX.	DESCRIPCIÓN
ESC !	33	21	Selector maestro.
ESC -	45	2D	Activa / desactiva modo subrayado.
ESC E	69	45	Selecciona modo enfatizado.
ESC F	70	46	Cancela modo enfatizado.
ESC G	71	47	Selecciona el modo doble golpe.
ESC H	72	48	Cancela modo doble golpe.
ESC k	107	6B	Selecciona tipo de letra en modo Calidad de Letra.
ESC x	120	1B	Selecciona modo Calidad de Letra o modo normal (draft).

PROCESAMIENTO PALABRAS			
ASCII	DEC.	HEX.	DESCRIPCIÓN
ESC SP	32	20	Especifica espacio entre caracteres.
ESC 4	52	34	Selecciona modo itálico.
ESC 5	53	35	Cancela el modo itálico.

[illegible]			
ASCII	DEC.	HEX.	DESCRIPCIÓN
ESC *	42	2A	Selecciona modo gráfico.
ESC ?	63	3F	Reasigna modo de graficación.
ESC K	75	4B	Selecciona modo gráfico de densidad sencilla.
ESC L	76	4C	Selecciona modo gráfico en doble densidad.
ESC Y	89	59	Selecciona modo gráfico doble densidad alta velocidad.
ESC Z	90	5A	Selecciona modo gráfico en cuádruple densidad.
ESC ^	94	5E	Selecciona modo gráfico de 9 agujas.

ESC BEL — **Alarma.**

Formato:	Código ASCII	ESC	BEL
	Decimal	27	7
	Hexadecimal	1B	07
Comentarios:	Hace que se active la alarma de la impresora.		

ESC DC1 — **Selecciona impresora.**

Formato:	Código ASCII	DC1	
	Decimal	17	
	Hexadecimal	11	
Comentarios:	Regresa la impresora al estado de selección si ésta ha sido deshabilitada mediante el código de deshabilitación de impresora (DC3). No selecciona la impresora si ésta ha sido colocada fuera de línea a través de oprimir el botón ON LINE (en línea). DC1 y DC3 no trabajan si la aguja 36 de la interface paralela ésta baja (por ejemplo, en IBM y algunas computadoras compatibles).		

ESC DC3 — **Deshabilita impresora.**

Formato:	Código ASCII	DC3	
	Decimal	19	
	Hexadecimal	13	
Comentarios:	Coloca la impresora en el estado de deshabilitación hasta que el código de selección de impresora (DC1) es recibido. La impresora no puede ser vuelta a seleccionar con el botón ON LINE (en línea).		

ESC 8 — **Deshabilita el sensor de fin de papel.**

Formato:	Código ASCII	ESC	8
	Decimal	27	56
	Hexadecimal	1B	38
Comentarios:	Desactiva el sensor de fin de papel, de tal manera que usted puede imprimir al final de una hoja suelta.		

ESC 9 — **Habilita el sensor de fin de papel.**

Formato:	Código ASCII	ESC	9
	Decimal	27	57
	Hexadecimal	1B	39
Comentarios:	Cancela ESC 8. De tal manera, la alarma de la impresora suena y deja de imprimir cuando la impresora rebasa un punto localizado aproximadamente a 1/2 pulgada del final del papel.		

ESC @ — **Inicializa impresora.**

Formato:	Código ASCII	ESC	@
	Decimal	27	64
	Hexadecimal	1B	40
Comentarios:	Inicializa la impresora y limpia el buffer de datos imprimibles precedidos del comando en la línea de impresión.		

ESC s — **Activa/desactiva modo de alta velocidad.**

Formato:	Código ASCII	ESC	s	n	
	Decimal	27	115	n	
	Hexadecimal	1B	73	n	
Comentarios:	Los siguientes valores pueden se usados para n: 1: El modo es activado (on). 2: El modo es desactivado (off). (Pueden ser usados los códigos ASCII 0 y 1 ó los caracteres ASCII 0 y 1).				

CR — **Retorno de carro.**

Formato:	Código ASCII	CR	
	Decimal	13	
	Hexadecimal	0D	
Comentarios:	Imprime los datos del buffer y regresa la posición de la impresora al margen izquierdo.		

CAN — **Cancela línea.**

Formato:	Código ASCII	CAN	
	Decimal	24	
	Hexadecimal	18	
Comentarios:	Retira el texto de la línea impresa pero no afecta los códigos de control.		

DEL — **Borra carácter.**

Formato:	Código ASCII	DEL	
	Decimal	127	
	Hexadecimal	7F	
Comentarios:	Retira el último carácter de la línea impresa pero no afecta los códigos de control.		

LF — Avance de línea.

Formato:	Código ASCII	LF
	Decimal	10
	Hexadecimal	0A
Comentarios:	Cuando este comando es recibido, son impresos los datos existentes dentro del buffer y el papel avanza una línea en la actual línea de espaciado.	

FF — Avance de página.

Formato:	Código ASCII	FF
	Decimal	12
	Hexadecimal	0C
Comentarios:	Imprime datos desde el buffer de la impresora y avanza el papel al inicio de la siguiente hoja, de acuerdo a la longitud de página actual.	

ESC 0 — Selecciona 1/8 de pulgada de línea de espaciado.

Formato:	Código ASCII	ESC 0
	Decimal	27 48
	Hexadecimal	1B 30
Comentarios:	Especifica el espacio entre líneas en 1/8 de pulgada para subsecuentes comandos de avance de línea. El 0 es el carácter cero y no el código ASCII 0.	

ESC 1 — Selecciona 7/72 de pulgada de línea de espaciado.

Formato:	Código ASCII	ESC 1
	Decimal	27 49
	Hexadecimal	1B 31
Comentarios:	Especifica el espacio entre líneas en 7/72 de pulgada para subsecuentes comandos de avance de línea. El 1 es el carácter uno y no la letra minúscula l o el código ASCII 1.	

ESC 2 — Selecciona 1/6 de pulgada de línea de espaciado.

Formato:	Código ASCII	ESC	2	
	Decimal	27	50	
	Hexadecimal	1B	32	
Comentarios:	Especifica el espacio entre líneas en 1/6 de pulgada para subsecuentes comandos de avance de línea. El 2 es el carácter dos y no el código ASCII 2. Este es el valor por omisión al encender la impresora.			

ESC 3 — Selecciona n/216 de pulgada de línea de espaciado.

Formato:	Código ASCII	ESC	3	n
	Decimal	27	51	n
	Hexadecimal	1B	33	n
Comentarios:	Especifica el espacio entre líneas en n/216 de pulgada para subsecuentes comandos de avance de línea. El 3 es el carácter tres y no el código ASCII 3. El valor de n deberá estar entre 0 y 255.			

ESC A — Selecciona n/72 de pulgada de línea de espaciado.

Formato:	Código ASCII	ESC	A	n
	Decimal	27	65	n
	Hexadecimal	1B	41	n
Comentarios:	Especifica el espacio entre líneas en n/72 de pulgada para subsecuentes comandos de avance de línea. El valor de n deberá estar entre 0 y 85.			

ESC C — Especifica longitud de página en líneas.

Formato:	Código ASCII	ESC	C	n
	Decimal	27	67	n
	Hexadecimal	1B	43	n
Comentarios:	Especifica longitud de página para n líneas en la actual línea de espaciado. El valor de n deberá se entre 1 y 127. La posición de inicio de página es especificada de acuerdo a la línea actual.			

ESC J — Ejecuta n/216 de pulgada de avance de línea.

Formato:	Código ASCII	ESC	J	n
	Decimal	27	74	n
	Hexadecimal	1B	4A	n
Comentarios:	Avanza el papel n/216 de pulgada. El valor de n deberá estar entre 0 y 255. Este comando produce un avance de línea inmediato, pero no afecta subsecuentes espacios entre líneas y no provoca retornos de carro.			

ESC N — Especifica el salto sobre perforaciones.

Formato:	Código ASCII	ESC	N	n
	Decimal	27	78	n
	Hexadecimal	1B	4E	n
Comentarios:	La variable n es el número de líneas saltadas entre la última línea impresa de una página y la primera línea de la siguiente. Por ejemplo, con las especificaciones estándar para el espaciado entre líneas (1/6 de pulgada), y para la longitud de página de 66 líneas, ESC N 6 hace que la impresora imprima 60 líneas y después se salte 6. Esta especificación es cancelada a través de ESC O y también a través de ESC C. El valor de n deberá estar entre 1 y 127.			

ESC O — Cancela el salto sobre perforaciones.

Formato:	Código ASCII	ESC	O
	Decimal	27	79
	Hexadecimal	1B	4F
Comentarios:	Cancela la especificación de salto sobre perforaciones dada a través de ESC N.		

HT — Tabulador horizontal.

Formato:	Código ASCII	HT
	Decimal	9
	Hexadecimal	09
Comentarios:	Avanza la posición de impresión al siguiente tabulador horizontal especificado. Las especificaciones por omisión están en intervalos de ocho caracteres dentro del tamaño de carácter por omisión, y las posiciones de los tabuladores no son afectadas por subsecuentes cambios en el tamaño de carácter.	

ESC Q — Especifica margen derecho.

Formato:	Código ASCII	ESC Q n
	Decimal	27 81 n
	Hexadecimal	1B 51 n
Comentarios:	Especifica el margen derecho para n columnas en el tamaño de carácter actual. Las especificaciones hechas en el modo proporcional son tratadas como 10 cpp. Este comando borra las especificaciones previas de comandos y todos los caracteres previos en la línea de impresión.	

ESC l — Especifica margen izquierdo.

Formato:	Código ASCII	ESC l n
	Decimal	27 108 n
	Hexadecimal	1B 6C n
Comentarios:	Especifica el margen izquierdo para n columnas en el tamaño de carácter actual. Las especificaciones hechas en el modo proporcional son tratadas como 10 cpp. Este comando borra las especificaciones previas de comandos y todos los caracteres previos en la línea de impresión. utilice la letra minúscula l y no el número uno. El mínimo espacio entre márgenes es del ancho de un carácter en 10 cpp expandido.	

SO — Selecciona modo expandido (una línea).

Formato:	Código ASCII	SO
	Decimal	14
	Hexadecimal	0E
Comentarios:	El modo expandido duplica en ancho de los caracteres. Este modo es cancelado a través de un retorno de carro o DC4. Este comando está disponible tanto en el modo normal como en el modo de Letra de Calidad.	

SI — Selecciona modo condensado.

Formato:	Código ASCII	SI
	Decimal	15
	Hexadecimal	0F
Comentarios:	Imprime caracteres en aproximadamente 60% de su tamaño normal. Por ejemplo, el modo condensado en 10 cpp tiene 17 caracteres por pulgada. El modo proporcional no puede ser condensado y anula el modo condensado. Este comando está disponible tanto en el modo normal como en el modo de Letra de Calidad.	

DC2 — Cancela modo condensado.

Formato:	Código ASCII	DC2
	Decimal	18
	Hexadecimal	12
Comentarios:	Cancela la impresión condensada especificada a través de SI, ESC SI o a través del Selector de Tipos.	

DC4 — Cancela modo expandido (una línea).

Formato:	Código ASCII	DC4
	Decimal	20
	Hexadecimal	14
Comentarios:	Cancela una línea de impresión expandida seleccionada a través de SO o ESC O, pero no la impresión expandida seleccionada a través de ESC W o ESC I.	

ESC M — Selecciona 12 cpp.

Formato:	Código ASCII	ESC M
	Decimal	27 77
	Hexadecimal	1B 4D
Comentarios:	Selecciona la impresión en 12 caracteres por pulgada. Este comando está disponible tanto en el modo normal como en el modo de Letra de Calidad.	

ESC P — Selecciona 10 cpp.

Formato:	Código ASCII	ESC	P	
	Decimal	27	80	
	Hexadecimal	1B	50	
Comentarios:	Selecciona la impresión en 10 caracteres por pulgada. Este comando es usado normalmente para cancelar el ancho de 12 cpp.			

ESC W — Activa / Desactiva modo condensado.

Formato:	Código ASCII	ESC	W	n
	Decimal	27	87	n
	Hexadecimal	1B	57	n
Comentarios:	Los siguientes valores pueden ser usados para n: 1: El modo es activado (on). 2: El modo es desactivado (off). Pueden ser usados los códigos ASCII 0 y 1 ó los caracteres 0 y 1.			

ESC p — Activa / Desactiva el modo proporcional.

Formato:	Código ASCII	ESC	p	n
	Decimal	27	112	n
	Hexadecimal	1B	70	n
Comentarios:	Los siguientes valores pueden ser usados para n: 1: El modo es activado (on). 2: El modo es desactivado (off). Pueden ser usados los códigos ASCII 0 y 1 ó los caracteres 0 y 1.			

ESC w — Activa / Desactiva modo doble alto.

Formato:	Código ASCII	ESC	w	n
	Decimal	27	119	n
	Hexadecimal	1B	77	n
Comentarios:	Los siguientes valores pueden ser usados para n: 1: El modo es activado (on). 2: El modo es desactivado (off). Pueden ser usados los códigos ASCII 0 y 1 ó los caracteres 0 y 1. El modo doble alto duplica la altura de los caracteres. Este comando está disponible tanto en el modo normal como en el modo de Letra de Calidad.			

ESC ! Selector maestro.

Formato:	Código ASCII	ESC	!	n
	Decimal	27	33	n
	Hexadecimal	1B	21	n

Comentarios:

Selecciona cualquier combinación válida de modos, de acuerdo a la siguiente tabla. La variable n es determinada a través de agregar al mismo tiempo los valores de los modos deseados desde la tabla.

Modo	Decimal	Hexadecimal
10 cpp	0	00
12 cpp	1	01
Proporcional	2	02
Condensado	4	04
Enfatizado	8	08
Doble golpe	16	10
Expandido	32	20
Itálica	64	40
Subrayado	128	80

Este comando es aplicable tanto en el modo normal como en el modo de Letra de calidad. 10 cpp no puede ser combinado con 12 cpp y, proporcional no puede ser condensado. Si son seleccionados tanto el proporcional como el condensado, el proporcional anula el condensado. El doble golpe es ignorado en el modo de Letra de Calidad.

ESC - Activa/Desactiva modo subrayado.

Formato:	Código ASCII	ESC	-	n
	Decimal	27	45	n
	Hexadecimal	1B	2D	n
Comentarios:	Los siguientes valores pueden ser usados para n: 1: El modo es activado (on). 2: El modo es desactivado (off). Pueden ser usados los códigos ASCII 0 y 1 ó los caracteres 0 y 1. Este modo proporciona subrayado continuo, incluyendo espacios. Este comando está disponible tanto en el modo normal como en el modo de Letra de Calidad.			

ESC E Selecciona modo enfatizado.

Formato:	Código ASCII	ESC	E
	Decimal	27	69
	Hexadecimal	1B	45
Comentarios:	Torna el texto más oscuro a través de imprimir cada punto dos veces, el segundo punto ligeramente a la derecha del primero. Este comando está disponible tanto en el modo normal como en el modo de Letra de Calidad.		

ESC F Cancela modo enfatizado.

Formato:	Código ASCII	ESC	F
	Decimal	27	70
	Hexadecimal	1B	46
Comentarios:	Cancela el modo enfatizado seleccionado a través de ESC E. Este comando está disponible tanto en el modo normal como en el modo de Letra de Calidad.		

ESC G — Selecciona el modo doble golpe.

Formato:	Código ASCII	ESC	G	
	Decimal	27	71	
	Hexadecimal	1B	47	
Comentarios:	Torna el texto más oscuro a través de imprimir cada línea dos veces, con la segunda impresión ligeramente abajo de la primera. Doble golpe no está disponible en el modo de Letra de Calidad.			

ESC H — Cancela modo doble golpe.

Formato:	Código ASCII	ESC	H	
	Decimal	27	72	
	Hexadecimal	1B	48	
Comentarios:	Desactiva el modo doble golpe seleccionado a través de ESC G.			

ESC k — Selecciona fuente en modo Calidad de Letra.

Formato:	Código ASCII	ESC	k	n
	Decimal	27	107	n
	Hexadecimal	1B	6B	n
Comentarios:	Este comando afecta solamente el estilo de tipo en modo de Letra de Calidad, no así el modo normal. Los siguientes valores pueden ser usados para n: 0 = Romano. 1 = Sans Serif. Borra las especificaciones del Selector de Tipos.			

ESC x — **Selecciona modo Calidad de Letra o modo normal.**

Formato:	Código ASCII	ESC	x	n	
	Decimal	27	120	n	
	Hexadecimal	1B	78	n	
Comentarios:	Los siguientes valores pueden ser usados para n: 1: El modo es activado (on). 2: El modo es desactivado (off). Pueden ser usados los códigos ASCII 0 y 1 ó los caracteres ASCII 0 y 1. Borra las especificaciones del Selector de Tipos. Cuando el modo Calidad de Letra es seleccionado, la fuente utilizada puede ser Romano, Sans Serif o definido por el usuario, dependiendo cual haya sido seleccionado.				

ESC SP — **Especifica espacio entre caracteres.**

Formato:	Código ASCII	ESC	SP	n	
	Decimal	27	32	n	
	Hexadecimal	1B	20	n	
Comentarios:	Especifica la cantidad de espacio agregado a la derecha de cada carácter, en adición al espacio ya permitido en el diseño del carácter. El número de unidades de espacio es igual a n, el cual deberá encontrarse entre 0 y 127. Cada unidad de espacio es de 1/120 de pulgada. Este comando está disponible tanto en el modo normal como en el modo de Letra de Calidad.				

ESC 4 — **Selecciona modo Itálico.**

Formato:	Código ASCII	ESC	4	
	Decimal	27	52	
	Hexadecimal	1B	34	
Comentarios:	Llama caracteres desde el conjunto de caracteres itálicos para ser impresos. Este comando está disponible tanto en el modo normal como en el modo de Letra de Calidad.			

ESC 5 — **Cancela el modo Itálico.**

Formato:	Código ASCII	ESC	5	
	Decimal	27	53	
	Hexadecimal	1B	35	
Comentarios:	Cancela el modo seleccionado a través de ESC 4. Este comando está disponible tanto en el modo normal como en el modo de Letra de Calidad.			

ESC * — **Selecciona modo gráfico.**

Formato:	Código ASCII	ESC	*	m	n1	n2
	Decimal	27	42	m	n1	n2
	Hexadecimal	1B	2A	m	n1	n2

Comentarios: Activa el modo gráfico m. Consulte la siguiente tabla para detalles sobre modos disponibles. El número total de columnas está dado por n1 + (n2 x 256).

Opción	Código Alterno	m	Densidad Horizontal (puntos/pulgada)
Densidad Sencilla	ASC K	0	60
Doble Densidad	ESC L	1	120
Alta velocidad doble densidad*	ESC Y	2	120
Cuádruple densidad*	ESC Z	3	240
CRT I	Ninguno	4	80
Plotter (1:1)	Ninguno	5	72
CRT II	Ninguno	6	90
Plotter doble densidad	Ninguno	7	144

* Puntos adyacentes no pueden ser impresos en este modo.

ESC ? — **Reasigna modo de graficación.**

Formato:	Código ASCII	ESC	?	s	n	
	Decimal	27	63	s	n	
	Hexadecimal	1B	3F	s	n	
Comentarios:	Cambia de un modo gráfico a otro. La variable s es un carácter (K, L, Y o Z), la cual es reasignada a un modo n (0-7).					

ESC K — Selecciona modo gráfico de densidad sencilla.

Formato:	Código ASCII	ESC	K	n1	n2
	Decimal	27	75	n1	n2
	Hexadecimal	1B	4B	n1	n2
Comentarios:	Activa modo gráfico de densidad sencilla de ocho agujas (60 puntos por pulgada). El número total de columnas está dado por n1 + (n2 x 256).				

ESC L — Selecciona modo gráfico en doble densidad.

Formato:	Código ASCII	ESC	L	n1	n2
	Decimal	27	76	n1	n2
	Hexadecimal	1B	4C	n1	n2
Comentarios:	Activa el modo gráfico en doble densidad, baja velocidad de ocho agujas (120 puntos por pulgada). El número total de columnas está dado por n1 + (n2 x 256).				

ESC Y — Selecciona modo gráfico doble densidad alta velocidad.

Formato:	Código ASCII	ESC	Y	n1	n2
	Decimal	27	89	n1	n2
	Hexadecimal	1B	59	n1	n2
Comentarios:	Activa el modo gráfico en doble densidad, alta velocidad de ocho agujas (120 ppp). El número total de columnas está dado por n1 + (n2 x 256).				

ESC Z — Selecciona modo gráfico en cuádruple densidad.

Formato:	Código ASCII	ESC	Z	n1	n2
	Decimal	27	90	n1	n2
	Hexadecimal	1B	5A	n1	n2
Comentarios:	Activa el modo gráfico en cuádruple densidad, de ocho agujas (120 ppp). El número total de columnas está dado por n1 + (n2 x 256).				

ESC ^ Selecciona modo gráfico de 9 agujas.

Formato:	Código ASCII	ESC	^	m	n1	n2
	Decimal	27	94	m	n1	n2
	Hexadecimal	1B	5E	m	n1	n2
Comentarios:	Activa el modo gráfico de nueve agujas. Para este comando la variable m define la densidad de impresión: 0 para densidad sencilla (60 cpp) y 1 para doble densidad (120 cpp). El número total de columnas está dado por n1 + (n2 x 256). Este modo requiere dos datos para cada columna de impresión.					

APÉNDICE B

EL ENFOQUE PARADIGMÁTICO

APÉNDICE B

EL ENFOQUE PARADIGMÁTICO

Paradigma: Un modelo generalizado a gran escala que proporciona un punto de vista desde donde el mundo real puede ser dilucidado.

Los paradigmas se establecen en la sociedad de manera casi inconsciente, y por regla general, dictan la conducta y costumbres de una comunidad, la forma de observar distintos fenómenos y de interpretarlos. Así por ejemplo, el paradigma del geocentrismo, durante la época medieval se veía como indiscutible, la tierra se convirtió en el centro de todas las cosas y pensar lo contrario, como en el caso de Copérnico se tomaba por irreverente y blasfemo. Sin embargo, la visión cosmológica de la época tuvo que cambiar; poco a poco las bases que sustentaban a las antiguas teorías fueron dejando camino hacia una nueva forma de contemplar la realidad y con ello, a nuevas teorías para esta vez dar forma al nuevo paradigma. Este cambio, de un paradigma a otro, con todas las ventajas y dificultades inherentes, se ha dado en llamar "cambio paradigmático".

En un cambio paradigmático, un punto de crisis del paradigma actual ha sido alcanzado y es necesaria una revolución, es decir, reemplazarlo por uno nuevo más amplio y que admita nuevas soluciones a los mismos problemas. Es una nueva forma de ver las cosas que se apoya generalmente en nuevos desarrollos de la ciencia, tecnología, arte o cualquiera otra disciplina; tales cambios son necesarios porque cambios importantes en la realidad demandan un nuevo enfoque en la conceptualización.

El concepto fue introducido originalmente por el filósofo e historiador Thomas Khun en su libro "La estructura de las revoluciones científicas", de 1962. Marilyn Ferguson en 1976, popularizó esta noción cuando escribió que el "cambio paradigmático comprendía la dislocación, conflicto, confusión e incertidumbre entre los partidarios del paradigma original, causando que los nuevos paradigmas sean recibidos con frialdad, incredulidad y hasta hostilmente". Esta idea fue retomada en 1987 para describir la idea de que la era de la informática estaba llegando a su primer cambio de fondo.

Los paradigmas no pueden compararse entre sí para determinar cuál de ellos es el mejor, pues pertenecen a ambientes distintos, utilizan distinta terminología y en general obedecen a distintas reglas. Una evaluación entre ellos solamente daría por resultado una indicación de cuál de ellos es mejor para un contexto de solución específico.

APÉNDICE C

MANUAL DE USUARIO

APÉNDICE C

MANUAL DE USUARIO

Bienvenida

Bienvenido al Diseñador Gráfico de Microcontroladores AuXiliar o Sistema de Apoyo al Diseño Digital con Arquitectura Microprogramada (DGMAX/SADDAM), el primer sistema de procesamiento de Cartas ASM por computadora realizado en México. El DGMAX/SADDAM incluye muchas características del estándar para interfaces gráficas de usuario (GUI's por sus siglas en inglés), que transformará el trabajo de construir el diagrama de la Carta ASM junto con su síntesis en papel, a archivos de computadora para su posterior procesamiento. Desarrollado con la tecnología de la Orientación a Objetos, el DGMAX/SADDAM liberará al diseñador de controladores digitales -que usa la metodología de Cartas ASM-, de la laboriosa tarea de sintetizar el problema en una serie de tablas, microcódigo y diagrama electrónico del circuito -posterior a alambrarse-, a una pequeña serie de "tikis" (palabra que el equipo de desarrollo emplea en sustitución del tecnicismo inglés "click") del dispositivo apuntador denominado ratón, con la finalidad de obtener los mismos resultados aunque en forma más eficiente.

¿A quiénes está dirigido DGMAX/SADDAM?. DGMAX/SADDAM está dirigido a todas aquellas personas que alguna vez diseñen controladores digitales empleando la metodología de Cartas ASM y síntesis de la misma con la arquitectura Mica I, además a todas aquellas interesadas en el desarrollo de sistemas de computadoras en ambiente gráfico y con la metodología de la Orientación a Objetos.

Para ambos perfiles de usuarios está disponible el sistema operando en forma correcta, para el segundo perfil están disponibles además, los fuentes de la aplicación, así como el diseño del sistema en el capítulo correspondiente a su desarrollo. Se hace del conocimiento del usuario que se cuenta con la libertad de desarrollar otras aplicaciones usando la mayoría de las componentes del sistema, como es la interface gráfica de usuario, teniendo la libertad inclusive de modificar alguna de dichas componentes. Bueno, como lo que realmente importa es saber como podemos trabajar con DGMAX/SADDAM, como entrar y como salir de la aplicación, las tres secciones siguientes están dedicadas a ello, así que ¡manos a la obra!.

Instalación del sistema

En primer lugar, Usted debe contar con el DGMAX/SADDAM, proporcionado en un disquete en formato de 3½" de alta densidad (1.44 MB) como material del presente trabajo. El cual contiene los siguientes archivos:

- **Manual.txt**. Contiene -en código ASCII- la información del presente documento relativo a la instalación y manejo de la aplicación.
- **Fuentes.exe**. Desempaqueta los archivos fuente de la aplicación.
- **Sistema.exe**. Desempaquete los archivos del sistema de la aplicación.
- **Instalar.bat**. Permite instalar los archivos de sistema y/o los archivos fuentes -los archivos fuentes no son necesarios para el proceso de sesión de trabajo con el sistema- de la aplicación en el disco duro.

Continuando, se debe determinar la unidad de disco en el cual se instalará el sistema. Una vez elegida la unidad de instalación, se está en posibilidad de ejecutar el archivo de instalación **instalar.bat**. En el proceso de instalación sólo se desempaquetan los archivos del sistema, los correspondientes a los archivos fuentes de la aplicación se dejan empaquetado (**Fuentes.exe**), esto con la finalidad de ahorrar espacio en disco, aunque se cuenta con la opción de desempaquetarlo también. En principio de cuenta veremos como instalar los archivos de sistema:

Pasos de la instalación:

1. Ejecutar el archivo **instalar.bat** con la siguiente sintaxis:

instalar UDisco [Enter]

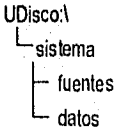
donde:

UDisco. Es la unidad de disco seleccionada (c, d, ...).

Comentario: La unidad de disco sólo se representa con la letra indicada para ella y seguida de : (dos puntos). Lo que está encerrado en corchetes indica que se debe presionar la tecla etiquetada con la palabra "Enter".

El programa instalar.bat le preguntará si desea desempaquetar el código del sistema, si rechaza esta opción, el código fuente del sistema se grabará en UDisco, en el directorio "\sistema\fuentes", empaquetado como fuentes.exe.

Al terminar de ejecutarse el programa de instalación, Ud. tendrá grabado en el disco de la unidad seleccionada, la siguiente estructura de directorios con sus respectivos archivos:



Si ningún error ocurrió, estará en posibilidad de manejar la aplicación, por lo que puede ir directamente a las secciones de "¿Cómo entrar al sistema?" y "¿Cómo salir del sistema?". En caso de error, realice de nuevo el proceso de instalación como se indicó.

En el directorio sistema, estarán disponibles los siguiente archivos: asm_mica.exe (archivo ejecutable del sistema), archivos con extensión chr y bgi. Inicialmente el subdirectorio **datos** se encontrará vacío, este subdirectorio está reservado para almacenar los diferentes archivos generados en el proceso de edición de una Carta ASM, es decir, cada vez que usted cree una nueva Carta ASM o modifique una existente, al salvarla, estos archivos se almacenarán ahí. En el subdirectorio fuentes estará un archivo llamado Fuentes.exe, el cual puede ser ejecutado para desempaquetar los fuentes de la aplicación, esto es útil, siempre y cuando se desee conocer el código de alguno o de todos los archivos fuentes desarrollados -recomendado para los programadores-.

2. El procedimiento para desempaquetar los archivos fuentes es el siguiente (opcional):

- ✓ Ubicarse en el subdirectorio UDisco:>\Sistema\Fuentes y teclear
- ✓ fuentes [Enter]

Al final de este procedimiento veremos en el directorio "Fuentes", archivos con extensión i, h y cpp, con lo que termina el procedimiento adicional de instalación.

¿Cómo entrar al sistema?

Una vez realizada la fase de instalación, Ud. está listo para trabajar con el sistema, la forma de hacerlo es la siguiente:

Si está ubicado en un directorio diferente al directorio llamado **sistema**, debe teclear lo siguiente:

cd\ [Enter]
cd sistema [Enter]
asm_mica [Enter]

En caso de que halla estado en el directorio sistema, simplemente teclee la última instrucción.

Una vez hecho lo anterior, si no hubo error en el proceso, estará listo para trabajar con el sistema, esto lo confirmamos porque se presentará una interface como la que se muestra en la Fig. C.1. En caso de error simplemente corregirlo volviendo a teclear todo a la parte donde se cometió el error.

Sugerencia: Puede crear un archivo .bat ("punto bat") incluyendo las instrucciones anteriores para poder ahorrarse el escribirlas cada vez que desee ejecutar el sistema, además de poder iniciar desde

cualquier directorio. Para mayor explicación de los archivos .bat refiérase a su manual de usuario de DOS.

¿Cómo salir del sistema?

A reserva de explicar con mayor profundidad los elementos de la interface gráfica del editor, mencionaremos que para salir del sistema, simplemente se debe apuntar con el ratón el botón  (botón con icono de una puerta abierta, una flecha y una etiqueta de salir) y presionar el botón izquierdo del ratón.

Convenciones:

- Todas las operaciones realizadas con el ratón serán por medio del botón izquierdo de éste, a menos que se especifique otro.
- Cuando se indique, "seleccionar un botón" o una "orden de la interface", significa apuntar con el ratón y proporcionar un "tiki" con el botón izquierdo de éste.

Descripción de la interface gráfica del sistema

Las Figs. C.1 a, b y c muestran los elementos más relevantes de la interface gráfica del sistema.

Elementos permanentes de la interface gráfica del sistema

A continuación presentaremos los elementos de la interface gráfica del sistema, los cuales se encuentran siempre disponibles. La Fig. C.1.a es la pantalla principal de la aplicación, estos elementos son permanentes, es decir, se conservan desde el inicio hasta el final de una sesión de trabajo.

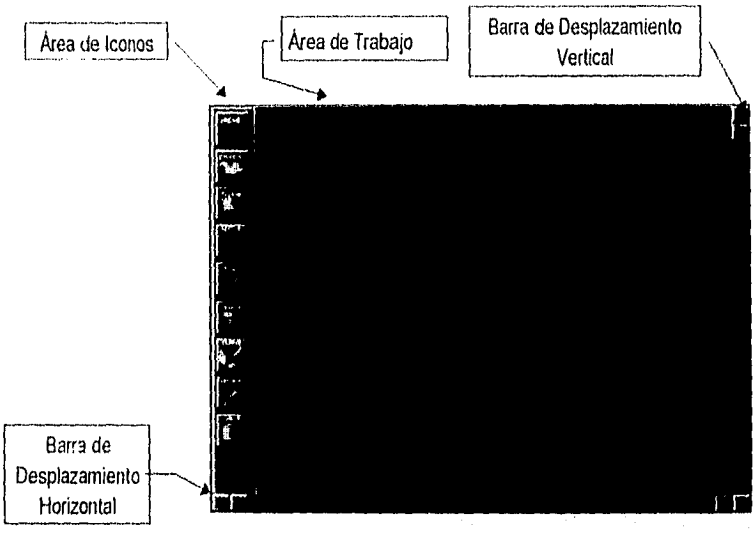


Fig. C.1.a. Descripción de los elementos permanentes de la Interface Gráfica del Sistema.

En la Fig. C.1.a observamos que la interface está dividida en varias secciones:

Área de Iconos: La cual está reservada para establecer los botones con un icono, representando las operaciones más comunes del editor. Las operaciones disponibles en esta área son accesadas mediante la sencilla operación de apuntar al botón correspondiente y proporcionar un "tiki" con el botón izquierdo del ratón. Posteriormente ampliaremos la explicación de esta área.

Área de trabajo: Ésta está destinada para contener los elementos de la Carta ASM que se desee editar. Como observamos en dicha área, ésta presenta una cuadrícula para la ubicación de los diversos elementos que contendrá la Carta ASM, además de presentar en ella los resultados de la síntesis de la carta en edición.

Adicional al Área de Trabajo se cuenta con un Área Virtual. El área virtual adicional es proporcionada al hacer uso de las Barra de Desplazamiento ya sea vertical u horizontal.

Dos Barras de Desplazamiento (Vertical y Horizontal): Las funciones que realizarán dichas barras, como ya se mencionó anteriormente, es la de navegar en el área virtual de trabajo, es decir, si la Carta ASM en edición se hiciese demasiado extensa para ubicarla en la porción del Área de Trabajo disponible, ésta se desplazará hacia arriba, hacia abajo, a la izquierda o la derecha dependiendo del desplazamiento deseado.

Menú de elementos de Cartas ASM

En la Fig. C.1.b se presenta otro elemento de la interface el cual es el **Menú de elementos** de Cartas ASM.

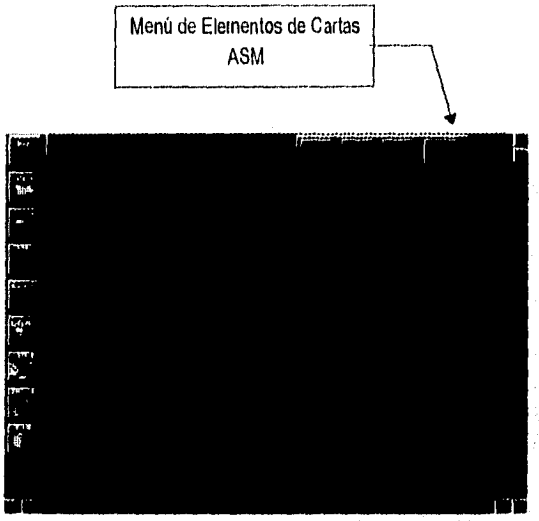


Fig. C.1.b. Descripción del Menú de Elementos de la Carta ASM.

Menú de elementos de Cartas ASM: Este menú contiene botones con iconos que representan los elementos (estado, decisión, salida condicional y unión de elementos) para construir una Carta ASM. Es invocado con un "tiki" del botón derecho del ratón, independiente de la ubicación de éste. El elemento deseado de la Carta ASM es seleccionado por su botón correspondiente (recuerde: apuntándolo y proporcionando un "tiki" al botón izquierdo del ratón). Posteriormente ampliaremos la explicación de este menú.

Descripción de los iconos del área de iconos

Icono Barra de Menú. Al seleccionar este botón, aparece en la parte superior de la pantalla la Barra de Menú para poder realizar operaciones de edición por medio de teclado o ratón. Reservado para ampliación del sistema.

Icono Impresión. Al presionar este botón, aparece una caja de diálogo para configurar las opciones disponibles de impresión de resultados: tablas generadas de la metodología, microprograma, diagrama del circuito electrónico, etc. La caja de diálogo con sus respectivas opciones se presentarán más adelante.

Icono Salvar Carta ASM. Al presionar este botón, aparece una caja de diálogo preguntado si se desea salvar la carta con el mismo nombre o con uno diferente o cancelar la operación Salvar carta. La carta salvada es almacenada en el directorio **datos**. Si es una carta inicial, el sistema preguntará por el nombre a asignarle.

Icono Edición. Este botón proporciona el mecanismo para cambiar los nombres, los códigos de los estados y los nombres de las variables de entrada de la Carta ASM.

Icono Salidas. El presionar este botón, nos posibilita el introducir variables de salidas de la Carta ASM mediante una ventana de edición y su posterior ubicación en los diversos estados en los que aparezca.

Icono Cargar Carta ASM. Al presionar este botón aparece, una caja de diálogo solicitando el nombre de la Carta ASM deseada para su posterior manejo.

Icono Borrar. El presionar este botón, nos posibilita la forma de borrar un elemento de la Carta ASM, borrar la unión entre dos elementos, borrar una salida de un estado o de todos los estados o borrar la carta en su totalidad. Cuando solicitamos esta operación, el sistema presenta una serie de ventanas indicando las operaciones disponibles.

Icono Síntesis de Carta ASM. El presionar este botón, presenta los resultados siguientes en pantalla (si la Carta ASM es válida): Tabla del Conjunto de Instrucciones, Tabla del Contador Programable, Tabla de Estados, Tabla de Salidas, Microprograma en formato binario y en hexadecimal, y el diagrama electrónico representado la solución del problema.

Icono Salida. El presionar este botón nos permite salir del sistema de edición de Cartas ASM. Al hacerlo restablece las condiciones iniciales de la computadora. No verifica si las modificaciones de la carta han sido salvadas.

Nota: La operación presionar botón la realizamos haciendo un "tiki" sobre el botón izquierdo del ratón.

Descripción de las Barras de Desplazamiento

Como se observa en la Fig. C.2, cada barra de desplazamiento consta de una barra que representa la longitud disponible de edición, en los extremos de cada barra se dispone de dos botones con un icono de flecha, los cuales nos sirven para desplazar la información ubicada en el área de trabajo, ya sea hacia arriba, hacia abajo, a la izquierda o a la derecha.

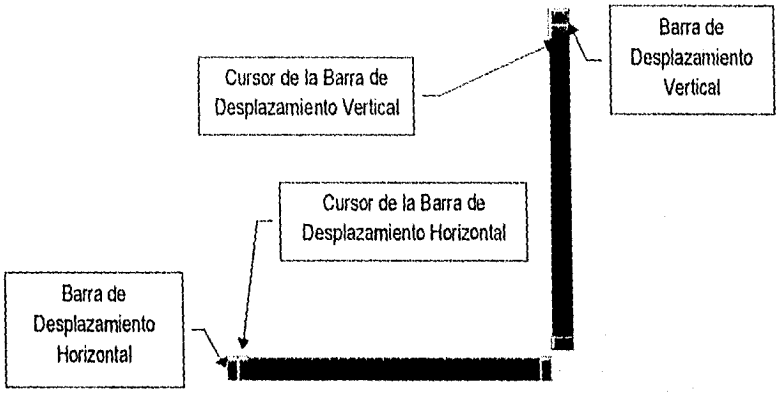


Fig. C.2. Barras de Desplazamiento: Vertical y Horizontal.

Además, observamos que consta de otro botón, el cual es el cursor de la barra y se desplazará según la indicación que le demos por medio de los botones extremos.

Descripción de los botones del menú de elementos

El Menú de elementos [redacted] está conformado por cuatro elementos con los que se representan las Cartas ASM, y que son: estado, representado por un botón con la figura de un rectángulo; salida condicional, representado por un botón con la figura de un rectángulo con esquinas redondeadas; diamante de decisión, representado por un botón con la figura de un rombo estilizado y por último la unión, representada por un botón con la figura de una flecha para la unión de los elementos. Para mayor información acerca de las características de cada elemento refiérase al Capítulo de Cartas ASM.

[redacted] **Icono Elemento Estado.** El presionar este botón crea un nuevo Estado, con un nombre y un código por omisión. La reubicación al lugar deseado, cambio de nombre y/o código de estado y unirlo con otro elemento de la Carta ASM se realiza posteriormente.

[redacted] **Icono Elemento Salida Condicional.** El presionar este botón crea un nuevo elemento Salida Condicional, al igual que al Estado sólo nos restaría reubicarlo en la posición adecuada, establecerle la(s) salida(s) y unirlo a los elementos deseados.

[redacted] **Icono Elemento Decisión.** El presionar este botón crea una nueva Decisión para la Carta ASM, le asigna una variable de prueba por omisión, a la que le podemos cambiar el nombre. Al igual que a los elementos anteriores, deberemos reubicarlo y unirlo a los elementos correspondientes.

Icono Unión de Elementos. Técnicamente hablando éste no es un elemento de una Carta ASM, sin embargo aquí lo consideraremos así. El presionar este botón nos permite unir dos elemento de la Carta ASM. Más tarde ampliaremos el procedimiento que tenemos que seguir para realizar la unión.

Operación arrastrar elementos

Para arrastrar un elemento, ya sea un estado, una decisión, una salida condicional o alguna Variable de salida del sistema, se procede de la siguiente manera:

- ✓ Se apunta al elemento deseado.
- ✓ Se presiona el botón izquierdo del ratón, sin soltarlo.
- ✓ Se arrastra el elemento seleccionado hasta el lugar deseado.
- ✓ Se libera el botón del ratón.

Con esto queda completo el proceso de arrastre y el elemento arrastrado se ubica en su nueva posición.

Nota.- Si el elemento arrastrado fue una variable, ésta es ubicada en una posición, siempre y cuando sea dentro de un estado.

Operación imprimir

A continuación detallaremos las opciones de la operación de impresión. Supongamos que ya hemos presionado el botón  (botón con el icono de impresora), a lo que el sistema respondió con la siguiente caja de dialogo:

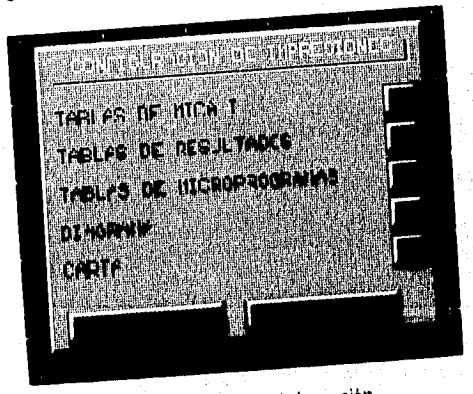


Fig. C.3. Opciones de Impresión.

Por omisión, todas las opciones presentes en la anterior caja de diálogo están activas, es decir, todas ellas se imprimirán si seleccionamos la opción de imprimir. Lo anterior lo sabemos porque se presentan los botones a la derecha de la opción con el símbolo "paloma" ( ). En la caja de diálogo podemos seleccionar/deseleccionar las opciones que nos interesa imprimir, esto lo realizamos simplemente apuntando y proporcionando un "tiki" al botón correspondiente. Las opciones son las siguientes:

Tabla de Mica I. Con esta opción activa se imprime el conjunto de instrucciones correspondiente a la arquitectura Mica I.

Tablas de Resultados. Si esta opción está activa se imprimen las Tablas correspondiente al funcionamiento del contador programable, la combinación de las Tablas del Conjunto de Instrucciones con la del contador, la Tabla de estados de la Carta ASM, la Tabla de salidas, la Tabla de Entradas.

Tabla de Microprograma. Con esta opción activa se imprime el microprograma resultante en formato binario y hexadecimal.

Diagrama. Cuando esta opción está activa, se incluye en la impresión, el diagrama del circuito controlador.

Carta. Con esta opción activa se imprime el diagrama de la Carta ASM.

Nota: Antes de proceder a imprimir, asegúrese que su impresora esté configurada como Epson.

Operación edición del nombre y/o código de estado y/o variables de prueba de los elementos de decisión'

Como se mencionó anteriormente, este botón  (botón con icono de cursor de ratón) proporciona el mecanismo para cambiar los nombres, los códigos de los estados y los nombres de las variables de entrada de la Carta ASM.

Esto se realiza de la siguiente manera:

I. Inmediatamente después de presionar este botón, apuntamos al nombre, al código del estado o a la variable de prueba dando un "tiki" con el botón izquierdo del ratón.
II. A continuación se presenta una pequeña ventana de edición, en la cual podemos teclear el nuevo nombre, al terminar de teclearlo se actualiza la etiqueta y eso es todo. El término de la edición puede ser mediante un [Enter] o porque el nombre de la variable excede de cuatro caracteres.

Operación edición de variables de salidas y ubicación de ellas en los estados correspondientes

Para proporcionar las salidas de cada uno de los estados de la Carta ASM, presionamos el botón  (botón con icono de lápiz), a lo que el sistema responde con la siguiente ventana de edición:

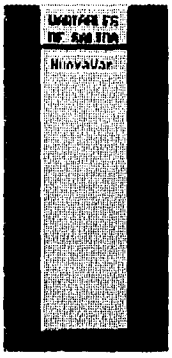


Fig. C.4. Ventana de edición de Variables.

En un inicio, esta ventana aparece vacía de variables, con una etiqueta de "NuevaVar", lo que indica que el programa está listo para recibir las salidas del sistema. En caso de que previamente se hallan proporcionado variables de salida, ésta o éstas aparecerán en la ventana de edición anterior. Los pasos para proporcionar salidas al sistema son los siguientes:

I. Con la ventana de edición abierta y posicionado en la etiqueta "NuevaVar", presionar la tecla [Enter], a continuación se presenta un pequeño cursor dentro de la ventana, en la posición de la etiqueta "NuevaVar", con lo que nos indica que podemos empezar a teclear el nombre de una salida, el término de ésta es mediante otro [Enter] o porque el nombre de la variable excede de cuatro caracteres, y así sucesivamente.
II. Una vez tecleada(s) la(s) salida(s) y dada una carta en pantalla, simplemente seleccionamos la variable correspondiente (apuntarla, presionar el botón izquierdo sin liberarlo) y la arrastramos dentro de la pantalla hasta ubicarla en el estado deseado.
III. Una vez ahí liberamos el botón del ratón y eso es todo, la variable queda relacionada con el estado elegido. Para salir de este modo apuntamos y damos un "tiki" del ratón fuera de la ventana.

Observaciones:

- ✗ Si por alguna razón seleccionamos una salida y no deseamos ubicarla en algún estado, simplemente la colocamos en cualquier parte diferente de un estado con lo que cancelamos esta operación.
- ✗ Si deseáramos eliminar una variable de salida de algún estado, ver **"Operación borrado de elementos del sistema"**.

Operación cargar carta de disco

Esta operación y la de "Salvar carta a Disco" son similares pero con sus características propias. Después de presionar el botón  (botón con el icono de un disquete y flecha apuntado hacia afuera) el sistema responde con la siguiente caja de dialogo:

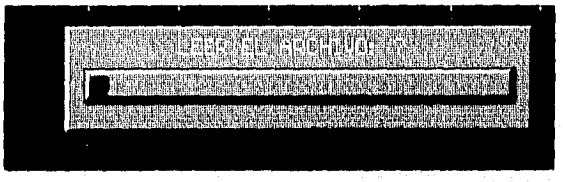


Fig. C.5 Ventana de lectura de archivo.

Aquí tenemos que proporcionar el nombre de una Carta ASM previamente grabada, dicho nombre obedece a las reglas de todo archivo de DOS, es decir, el primer carácter debe ser una letra, se permiten dígitos después del primer carácter, el nombre completo no debe ser mayor de ocho caracteres, etc.

En el caso de que la carta esté bien grabada y no exista ningún error, se cargará en el Área de Trabajo para su posterior manejo, en caso contrario, nos proporcionará un mensaje de error indicando que el archivo no existe o que no se puede cargar.

Operación borrado de elementos del sistema

Después de presionar el botón  (botón con icono de goma de borrar), el sistema responde con las siguientes ventanas:

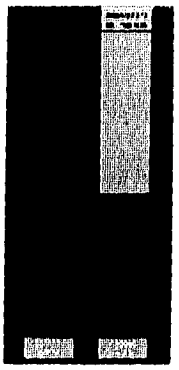


Fig. C.6. Ventanas de Borrado de Elementos.

Estas ventanas nos indican los diversos elementos que podemos borrar, de las cuales podemos deducir que podemos eliminar:

- Uniones entre elementos. Para esto el sistema presenta una barra en la parte inferior de la pantalla solicitando el origen de la unión, esto se lo proporcionamos con un "tiki" al elemento deseado, Si el elemento donde inicia la unión es un diamante de decisión la unión a borrar dependerá de si el elemento fue seleccionado en su extremo derecho o izquierdo. Esto es, si se seleccionó el extremo derecho se borrará la unión a la derecha del diamante, en caso contrario la unión izquierda será la eliminada. Para el estado o la salida condicional no importa el extremo seleccionado
- La carta completa. Simplemente presionando el botón que indica "carta".
- Una Variable de un estado. Esto lo realizamos apuntando y proporcionando un "tiki" a la variable en la ventana de variables de salida, a lo que el sistema responderá con otra ventana donde se listarán los estados en donde aparece: si queremos borrarla de alguno, seleccionamos de cual, y si queremos borrarla de todos, elegimos la opción "todos".

Adicionalmente podemos eliminar un elemento de la carta, simplemente selecionandolo.

Operación unión de dos elementos de la Carta ASM

Después de presionar el botón  (botón con el icono de una flecha), el sistema responde con lo siguiente:

Se presenta una barra en la parte inferior de la pantalla solicitando el elemento fuente de la unión, a lo cual le respondemos seleccionando alguno.

A continuación se presenta la misma barra solicitando el final del primer segmento de la unión -que puede ser el final de la unión si es que seleccionamos otro elemento-. Este procedimiento continua hasta que se selecciona el elemento destino.

En caso de que el elemento inicial haya sido una decisión:

a) Para seleccionar el elemento que va unido a la rama etiquetada con "0" (cero), establecemos una linea vertical imaginaria dividiendo el Diamante de decisión en dos. seleccionamos la parte izquierda de este elemento (recuerde, apuntar y presionar el botón izquierdo del ratón).
b) Para seleccionar el elemento que va unido a la rama etiquetada con "1" (uno), realizamos lo mismo que el paso a) excepto que seleccionamos la parte derecha del diamante de decisión.
c) Completamos la operación siguiendo los pasos descritos para los otros elementos.

Ejemplo de edición de una Carta ASM

A continuación desarrollaremos un pequeño ejemplo demostrativo del editor, se contempla introducir estados, Decisiones, Variables de salida, cambiar nombre a los estados, a las Variables de prueba, etc. La carta que obtendremos será como la siguiente:

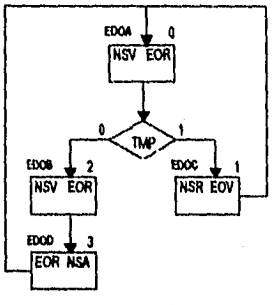


Fig. C.7.a. Carta ASM a editar.

Los pasos que realizaremos son:

1) Generar los estados, esto lo realizamos de la siguiente manera:

a) Presionar el botón derecho del ratón, a continuación aparece el menú de elementos. De este menú presionar el botón de estado. Una vez concluido lo anterior el sistema nos proporciona un estado ubicado en un recuadro vacío de la primera columna del Área de

Trabajo. Observe que este estado tiene un nombre etiquetado con la letra "a", su código es el "0" (cero).

b) Hacer lo anterior para los estados restantes. Observe que cada vez que generamos un nuevo estado, éste se ubica en el cuadro inferior del anterior, además, observemos que los siguientes estados tiene como nombre la siguiente letra del alfabeto y su código es el número consecutivo del anterior.

Una vez realizado lo anterior, deben estar presentes los estados: "a", "b", "c" y "d".

2) Generar la decisión. Esto lo realizamos presionando el botón derecho del ratón, una vez que aparece el menú de elementos, presionar el botón de decisión. La decisión que se genera aparece en el recuadro inferior del último elemento creado.

Nota: El elemento que se genera, aparece en el primer recuadro libre de la primera columna, si ésta está llena, entonces aparece en el primer recuadro libre de la siguiente columna y así sucesivamente.

3) A continuación reubicaremos los elementos de tal manera que queden de la siguiente forma:

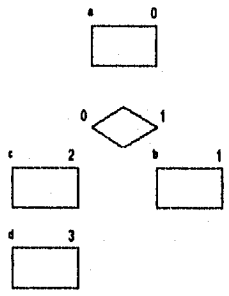


Fig. C.7.b. Carta ASM a editar.

Para lo anterior, realizaremos los siguientes pasos:

a) Apuntar al elemento que deseamos mover.
b) Presionar el botón izquierdo del ratón y sin liberarlo, ubicarse en un recuadro apropiado (sugerencia: ubíquelo en el centro del Área de Trabajo); ahora sí, libere el botón del ratón.
c) Haga los dos pasos anteriores para los elementos restantes, de tal manera que queden como en la figura anterior.

4) Ahora procederemos a unir cada uno de los elementos, los pasos necesarios son:

Pasos para uniones en donde el primer elemento no es una decisión.

a) Presione el botón derecho del ratón.
b) Del menú de elementos, presione el botón que representa una flecha; en la parte inferior de la pantalla verá una barra solicitando que seleccione el elemento fuente -seleccione el estado etiquetado con la letra "a" como elemento fuente-.
c) A continuación verá la misma barra solicitándole el final del primer segmento de la unión, seleccione la decisión como elemento destino -final del segmento-, si no existió error verá una línea uniendo estos dos elementos.

Pasos para unir el elemento diamante de decisión.

a) Presione el botón derecho del ratón.
b) Del menú de elementos, presione el botón que representa una flecha, en la parte inferior de la pantalla verá una barra solicitando que seleccione el elemento inicial. Si desea unir la rama etiquetada con "0" (cero) seleccione el diamante por su extremo izquierdo, de lo contrario hágalo del extremo derecho.
c) A continuación verá la barra inferior solicitándole el final del primer segmento de la unión, seleccione la casilla ubicada a la izquierda de la decisión para definir el primer segmento de la unión izquierda -verá una línea como primer segmento-. La barra inferior solicitará el final del nuevo segmento, seleccione el estado con el código 2. Realizado lo anterior queda definida la unión de la rama izquierda. Para definir la unión derecha el procedimiento es similar. Nota. Los segmento deben ser siempre ortogonales.

Ahora proceda para cada uno de los elemento presentes uniendo:

a) El estado "a" con la decisión,
b) la decisión unida con el estado "c" por su rama "0" o izquierda,
c) la misma decisión con el estado "b" por su rama "1" o derecha,
d) el estado "b" con el estado "a",
e) el estado "c" con el estado "d" y
f) el estado "d" con el estado "a".

La carta debe quedar conformada de la siguiente manera:

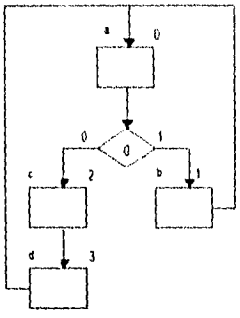


Fig. C.7.c. Carta ASM a editar.

5) Bueno, ahora es momento de establecer las variables de salida de la Carta ASM, para esto presionamos el botòn (botón con icono de lápiz), aparece una pequeña ventana de edición en la parte superior derecha del Área de Trabajo, como no se ha definido ninguna variable, ésta sólo se presenta con un mensaje que indica "NuevaVar" encerrada en un rectángulo.

   Para proceder a teclear una variable presionamos la tecla [Enter] y empezamos a introducir el nombre de la variable deseada, en este caso NSV; para finalizar presionamos otra vez la tecla [Enter].

   Lo anterior hay que realizarlo para cada una de las variables deseadas, a continuación presentamos la lista completa de variables que deben ser introducidas:

   - ✓ NSV
   - ✓ EOR
   - ✓ NSA
   - ✓ NSR
   - ✓ EOV
   - ✓ EOA

   Observe que cada vez que realizamos lo anterior, el mensaje de "NuevaVar" se desplaza hacia abajo y las variables quedan en la parte superior.

6) Continuando con el proceso, relacionaremos las variables de salida con los estados correspondientes, es decir, ubicaremos cada variable de salida con el estado que le corresponda. Realicémoslo así:

De la ventana de edición de variables de salida seleccionamos por ejemplo a NSV (NSV aparece en los estados "a" y "c"), es decir, apuntamos a esta variable y presionamos el botón izquierdo del ratón y sin liberarlo, la arrastramos hacia uno de estos estados, digamos el "a", si ya estamos ubicados en este estado liberamos el botón del ratón.

Observe que inmediatamente se estable la variable en ese estado, de manera similar se procede para establecer la variable en el estado "c" y así para cada una de las variables. Para finalizar con el proceso de introducir variables, realícelo en el siguiente orden:

- ✓ NSV debe establecerse en los estados "a" y "c",
- ✓ EOR debe establecerse en los estados "a", "c" y "d",
- ✓ NSA debe establecerse en el estado "d",
- ✓ NSR debe establecerse en el estado "b" y
- ✓ EOV también debe establecerse en el estado "b".

La carta debe quedar conformada de la siguiente manera:

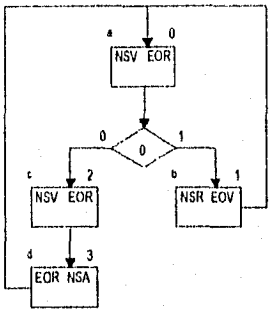


Fig. C.7.d. Carta ASM a editar.

7) Finalizaremos este ejemplo de edición con el cambio de nombre de los estados y de la Variable de prueba "0" del elemento decisión (no confundir con la rama "0" de dicha decisión). Para realizar el cambio de nombre presionamos el botón  (botón con icono de flecha de apuntador de ratón), a continuación seleccionamos el nombre del estado, en este caso "a", con lo que aparece una ventana solicitando el nuevo nombre -no debe ser mayor de cuatro caracteres-, en este punto teclee ..OOA.

Proceda de manera similar para los siguientes estados:

- ✓ al estado "b" renombrelo con la etiqueta EDOC (note que no lo estamos renombrando como EDOB),
- ✓ al estado "c" renombrelo con la etiqueta EDOB (también observe que no lo renombramos como EDOC),
- ✓ al estado "d" renombrelo con la etiqueta EDOD y,
- ✓ de manera similar a la variable de prueba "0" renombrela con la etiqueta TMP.

Con lo que la Carta ASM queda como la de la Fig. C.7.a.

8) A continuación, obtendremos los resultados de la Carta ASM anterior, para realizar esto simplemente presionamos el botón  (botón con icono de diagrama de un circuito electrónico), para sintetizar la carta, al realizar esto observaremos las primeras tablas que genera el sistema, observará una barra de mensajes en la parte inferior de la pantalla, indicando las teclas que debe presionar para continuar observando los resultados, para regresarse a ver los resultados anteriores o para salir de este modo. Las teclas que se proporcionan para realizar lo anterior son:

- ✓ [Pg Up] para observar resultados siguientes,
- ✓ [Pg Dn] para regresarse a ver resultados anteriores,
- ✓ [Esc] para salir del modo síntesis.

9) Si desea imprimir los resultados anteriores sólo tiene que presionar el botón de impresión  (botón con icono de impresora) y configurarlo para obtener los resultados deseados, recuerde que debe configurar la impresora en modo Epson.

Con los pasos anteriores se cuentan con los elementos necesarios para hacer uso del sistema DGMAX/SADDAM.

Esperamos que la información contenida en este manual sea de utilidad para simplificar su proceso de diseño.

El equipo de desarrollo

APÉNDICE D

BIBLIOGRAFÍA

APÉNDICE D

BIBLIOGRAFÍA

NO.	AUTOR	TITULO	PAÍS	AÑO
1		Microsoft MS-DOS Guía del Usuario V. 3.3	Microsoft Corporation, México	1987
2		Funciones de DOS y BIOS	Editorial Addison-Wesley, México	1990
3		Manual del Usuario Impresora FX-850/1050	Epson	1990
4		User's Manual, Citizen 200GX, Dot Matrix Printer	Citizen America Corporation, Santa Monica, California	1990
5		User's Manual NX-1001 multi-font dot matrix printer	Star Micronics Co., Ltd., USA	1990
6		Easy guide to graphics file and video standards	Compute's getting started with graphics standars, Compute publications International ltd., pp. 18	1991
7		Laser Printer HL-6/6T User's Guide	Brother Industries, Ltd, Japan	1993
8		Turbo C++, User's Guide	Borland International, Inc., USA	1992
9	A. Wiatrowski, Claude & H. House, Charles	Circuitos Lógicos y Sistemas de Microcomputadoras	Limusa, 1a. Edición, México	1987
10	Abrash, Michael	Graphics Programming	Dr. Dobb's Journal, Vol. 16, Núm. 8, pp. 165	1991
11	Abrash, Michael	Graphics Programming	Dr. Dobb's Journal, Vol. 16, Núm. 10, pp. 155	1991
12	Abrash, Michael	Graphics Programming	Dr. Dobb's Journal, Vol. 16, Núm. 11, pp. 123	1991
13	Betsira, Mark	VGA Palette Mapping Using BSP Trees	Dr. Dobb's Journal, Vol. 18, Núm. 7, pp. 28	1993

NO.	AUTOR	TÍTULO	PAÍS	AÑO
14	Betz, Mark	Interoperable objects	Dr. Dobb's Journal, Vol. 19, Núm. 11, pp. 18	1994
15	C. Lee, Samuel	Digital circuits & logical design	Prentice Hall, EUA	1976
16	Campbell, Tom	How to get started with graphics cards and monitors	Compute's getting started with graphics standars, Compute publications international ltd., pp. 4	1991
17	Cannavino, James A.	Object Insider: The future of object technology	Object Magazine, Vol. 4, Núm. 7, pp. 12	1994
18	Ceballos, Francisco Javier	Curso de Programación con "C"	Macrobit Editores, México	1990
19	English, David	How to with graphics standards	Compute's getting started with graphics standars, Compute publications international ltd., pp. 2.	1991
20	Ezzell, Ben	Graphics Programming in Turbo C++	Addison-Wesley Publishing Company, Inc., Massachusetts, EUA	1990
21	Faison, Ted	Borland C++ 3.1 Programación Orientada a Objetos	Prentice Hall, 2a. Edición, México	1993
22	Fernández Villarroel, David	Diccionario de dudas e irregularidades de la lengua española	Editorial Teide, S.A., Barcelona, España	1991
23	Floyd, Michael	The object d'art	Dr. Dobb's, Journal, Vol. 16, Núm. 10, pp. 52	1991
24	Floyd, Michael	Comparing Object Oriented languaje	Dr. Dobb's Journal, Vol. 18, Núm. 11, pp. 104	1993
25	Götze, Ralf & Starrermayr, Gerhard	El gran libro de las HP-DeskJet	Marcombo, Barcelona, España	1994
26	G. Fichman, Robert & F. Kemerer, Chris	Object-Oriented and Conventional Analysis and Design Methodologies	Computer, Vol. 25, Núm. 10, pp. 22, IEEE Computer Society, N.J.	1992

NO.	AUTOR	TÍTULO	PAÍS	AÑO
27	Gaona Castillo, Areli	Diseño de interfaces ¿El secreto del éxito?	PC/Tips-Byte, Año 5, Núm. 51, pp. 10	1992
28	Goodwin, Mark	Graphical User Interfaces in C++ & Object-Oriented Programming	MIS: PRESS, Portland, Oregon	1990
29	Gossain, Sanjiv	Modelling Rules in Object-Oriented Analysis	Object Magazine, Vol. 4, Núm. 7, pp. 28	1994
30	Graham, Ian	Migration Strategies: You was my object ... but I done you wrong	Object Magazine, Vol. 4, Núm. 7, pp. 68	1994
31	Green, David	Modern Logic Design	Addison-Wesley, EEUU	1986
32	Harter, Richard	Object-Oriented Software Configuration Management	Dr. Dobb's, Journal, Vol. 16, Núm. 10, pp. 36	1991
33	Hekmatpour, Sharam	C++ guía para programadores en "C"	Prentice Hall Hispanoamericana, S. A. México, D. F.	1992
34	Henderson-Sellers, B.	A Book of Object-Oriented Knowledge	Prentice Hall, Australia	1991
35	I. Fletcher, William	An Engineering Approach to Digital Design	Prentice All, EEUU	1980
36	IBM	Object Technology, The Developer's & manager's Guide to the Future	IBM, International Business Machines Corporation	1994
37	IBM	On course for the future, How you can plan now to exploit the benefits of Object Technology	IBM, International Business Machines Corporation	1994
38	J. Hill, Frederick & R. Peterson, Gerald	Teoría de conmutación y diseño lógico	Editorial Limusa, México, D. F.	1990
39	J. Rochkind, Marc	Advanced C Programming for display	Prentice Hall, EEUU	1988
40	J. Tocci, Ronald	Sistemas digitales, principios y aplicaciones	Prentice Hall, 5a. Edición, México	1993

NO.	AUTOR	TITULO	PAIS	AÑO
41	Johnson, Nelson	Advanced Graphics in C: Programming and Techniques	Osborne McGraw-Hill, Berkeley, California	1990
42	Kay Russell	Objects in use	Byte, Vol. 19, Núm. 4, pp. 99	1994
43	Ladd, Jim	Developing a medical diagnostic system wit OOA/RD	Object Magazine, Vol. 4, Núm. 7, pp. 48	1994
44	Leblanc, G.	Turbo C para IBM-PC y Compatibles	Editorial Gustavo Gili, S. A., Barcelona	1988
45	Lin Wang, Barbara & Wang Jin	Is a deep class hierarchy considered harmful?	Object Magazine, Vol. 4, Núm. 7, pp. 35	1994
46	M. Schevin-Tejera, Geneviève & G. Tejera, Héctor	Diccionario Moderno de Informática, Inglés-Español	Grupo Editorial Iberoamérica, México, D. F.	1989
47	Martin, James & J. Odell, James	Object-Oriented Analysis and Design	Prentice Hall, Englewood, New Jersey	1992
48	Meyer, Bertrand	Applying "Design by Contract"	Computer, Vol. 25, Núm. 10, pp. 40, IEEE Computer Society, N.J.	1992
49	Minasi, Mark	How to upgrade your graphics card and monitor	Compute's getting started with graphics standars, Compute publications international ltd., pp. 10	1991
50	Minassi, Mark	How to choose a VGA System.	Compute, Vol. 13, Núm. 11, pp. 52	1991
51	Mondragón Ballesteros, Jorge	Programe su propia interface gráfica.	PC/Tips - Byte, Año 5, Núm. 51, pp. 77	1992
52	Morris Mano, M.	Lógica digital	Prentice Hall, Colombia	1982
53	O'Brien Stephen	Turbo Pascal 6, manual de referencia	McGraw-Hill/Interamericana, España	1991
54	Oros, Juan Carlos & Montes, Antonio	Impresoras Matriciales, Chorro de Tinta y Laser	Editorial Paraninfo, Madrid, España	1991
55	P. Deschamps, Jean & Angulo José Maria	Diseño de Sistemas Digitales	Editorial Paraninfo, 2a. Edición, España	1992

NO.	AUTOR	TÍTULO	PAÍS	AÑO
56	Peñaloza Romero, Ernesto	Fundamentos de Programación	U.N.A.M., México	1994
57	Pfaffenberger, Bryan	Diccionario para usuarios de computadoras	Prentice Hall Hispanoamericana, Naocalpan de Juárez, Edo. de México	1993
58	Poor, Alfred	Monitores de 16 pulgadas	PC Magazine, Vol. 2, Núm. 8, pp. 70	1991
59	Porras, Alejandro y Plácido, Antonio	Autómatas programables	McGraw-Hill, México	1990
60	Rieble, Richard	Generalized Methods in Object-based Languages	Computer Language, Vol. 8, Núm. 10, pp. 79	1991
61	S. Pressman, Roger	Ingeniería del Software, un enfoque práctico	Borland-Osborne/McGraw-Hills, Madrid España	1990
62	S. Weiner, Richard	Applications of object-oriented programming	Addison Wesley, Massachusetts, EUA	1991
63	Schildt, Herbert	Lenguaje C, programación avanzada	McGraw-Hill/Interamericana, México, D. F.	1990
64	Schildt, Herbert	Programación en Turbo C	Borland-Osborne/McGraw-Hill, Madrid España	1990
65	Schwork, Paul	About the paradigm	Object Magazine, Vol. 4, Núm. 7, pp. 60	1994
66	Shelton, Ted	Objects in Business: The challenge of change	Object Magazine, Vol. 4, Núm. 7, pp. 20	1994
67	Soley, Richard	Standards: Defining interfaces	Object Magazine, Vol. 4, Núm. 7, pp. 71	1994
68	Stevens, Al	C Programming	Dr. Dobb's Journal, Vol. 16, Núm. 8, pp. 149	1991
69	Stevens, Al	C Programming	Dr. Dobb's Journal, Vol. 18, Núm. 11, pp. 133	1993
70	Stevens, Al	C Programming	Dr. Dobb's Journal, Vol. 18, Núm. 7, pp. 115	1993

NO.	AUTOR	TITULO	PAIS	AÑO
71	Stroustrup, Bjorne	The C++ Programming Languaje	AT&T Bell Laboratories, Murray Hill, New Jersey, Addison-Wesley Publishing Company	1992
72	T. Demel John & J. Miller Michael	Gráficas por computadoras	McGraw-Hill/Interamericana, México, D. F.	1990
73	Ungar, David; B. Smith, Randall; Chambers, Craig & Holzle Urs	Object, Message, and Performance: How They Coexist in Self	Computer, Vol. 25, Núm. 10, pp. 53, IEEE Computer Society, N.J.	1992
74	Voss, Greg & Chui Paul	Turbo C++ DiskTutor	Osborne McGraw-Hill, 2a. edición, United States of America	1991
75	W. Koontz, Richard	Lessons from the experts	Object Magazine, Vol. 4, Núm. 7, pp. 64	1994
76	Wayt Gibbs, W.	Software's Chronic Crisis	Scientific American, Vol. 271, Núm. 3, pp. 72	1994
77	Wegner, Peter	Dimensions of Object-Oriented Modeling	Computer, Vol. 25, Núm. 10, pp. 12, IEEE Computer Society, N.J.	1992

APÉNDICE E

GLOSARIO

APÉNDICE E

GLOSARIO

Abstracción	Modo de pensar mediante el cual consideramos algo aisladamente o no tenemos en cuenta algunas de sus cualidades o características, esto es, la separación de detalles innecesarios de los requerimientos o especificaciones del sistema, para así reducir la complejidad o comprensión de los requerimientos o especificaciones.
Abstracción por clasificación	En la OO, administración de colecciones de objetos por medio de sus características representativas, separándolas en clases.
ADA	Lenguaje de alto nivel desarrollado por el Departamento de Defensa de los Estados Unidos y requerido en todas las aplicaciones de programación militar. Llamado así en honor de Lady Augusta Ada Byron, el lenguaje Ada cubre las necesidades militares de un lenguaje estándar capaz de controlar procesos en tiempo real (por ejemplo, la operación de un dispositivo sumamente complejo como un misil). Basado en Pascal y Modula-2, Ada usa los principios de la programación estructurada, como los módulos de programa que pueden compilarse de forma independiente (al igual que los de Modula-2). Ada es un lenguaje altamente estructurado para programación de propósito general y un lenguaje especializado para el control de proceso en tiempo real.
Adaptador de Video	Dispositivo físico que genera la salida necesaria para presentar texto y gráficos de computadora a través de un monitor.

Algoritmo	Conjunto específico de procedimientos matemáticos y lógicos, simples y bien definidos, que pueden seguirse para resolver un problema en un número determinado de pasos. Sin embargo, no toda lista de instrucciones es un algoritmo. Para serlo, debe cumplir tres criterios básicos: ✓ La lista de instrucciones debe ser finita y suficientemente corta para que pueda ser completada. ✓ Cada instrucción debe ser ejecutable; esto es, debe ser capaz de realizar las acciones u operaciones designadas. ✓ El algoritmo debe permitir que la ejecución termine en algún momento. El lógico británico Alan Turing, probó que cualquier problema lógico o matemático capaz de resolverse, y para el que hay una solución conocida, se puede resolver mediante el enfoque algorítmico. Todo problema resoluble conocido se puede manejar con una computadora; la solución depende de encontrar el algoritmo correcto.
Ambiente/Entorno/ Sistema Gráfico	Hardware y/o Sistema Operativo con elementos provistos de capacidad gráfica para programas de aplicación. También se conoce como entorno/sistema gráfico.
Análisis	Método de investigación que comienza por separar una situación o problema en las partes que lo componen y luego trata de entender cómo se afectan las partes entre sí. En computación, etapa de desarrollo de sistemas en el cual los analistas y los usuarios establecen las especificaciones del sistema que se desea implantar. Se clarifican los objetivos dentro del dominio del lenguaje del problema.
Animación	Creación de una ilusión de movimiento en un programa de computadora, para el cual se registra una serie de imágenes que representan ligeros cambios incrementales en uno de los objetos mostrados y se reproducen con la suficiente velocidad para que el ojo perciba un movimiento suave.
AOO	(Léase "a doble o"). Análisis Orientado a Objetos. Método de análisis en el cual los requerimientos son examinados desde la perspectiva de las clases y los objetos encontrados en el vocabulario que es dominio del problema.

Apuntador	En las interfaces de usuario de computadora, símbolo en pantalla -por lo general una flecha- que presenta la ubicación de un dispositivo de entrada, por lo regular un ratón.
Arquitectura Cliente-Servidor	Modelo de diseño para aplicaciones que corren en redes de área local, en la que la mayor parte del proceso final se lleva a cabo en el servidor. El proceso de la primera fase, que implica comunicación con el usuario, lo manejan varios programas pequeños, distribuidos en las estaciones de trabajo destino (clientes). En la OO, esta arquitectura se refleja en los servicios que ofrece un objeto (Servidor) a otros objetos (Clientes) que hacen uso de ellos.
Arquitectura de Computadora	Diseño total mediante el cual se interrelacionan los componentes individuales del hardware de una computadora. Este término se emplea a menudo para describir la capacidad de manipulación de datos de una computadora. Por ejemplo, la arquitectura de 8 bits del microprocesador Intel 8088 está determinado por el bus de datos de 8 bits, el que transmite un byte de datos a la vez.
ASM	Del inglés, Algorithm State Machine, es decir, la Máquina de Estados Algorítmica, la cual es la formalización de la relación de los Estados y el funcionamiento del sistema digital. También conocido como Carta ASM.
ATT400	Norma de visualización de gráficos en mapa de bits. Los adaptadores ATT400 muestran al mismo tiempo 2 colores de variación continua, con resoluciones de 320 x 200 pixeles, 640 x 200 pixeles ó 640 x 400 pixeles. Nota. En baja resolución, los colores dependen de la paleta gráfica definida.
Autocontención	Característica que presenta una entidad u objeto, al consistir tanto de código como de datos sobre los que se opera.
Barra de Mensajes	Barra que está a lo largo de la parte inferior de la pantalla (o de la ventana) que contiene mensajes de información de la operación realizada o una ayuda en particular.
Barra de Menú	De acuerdo con las normas industriales y las interfaces gráficas para usuarios, barra que está a lo largo de la parte superior de la pantalla (o de la ventana) que contiene los nombres de menús descendentes.
Biestable	Circuito digital que constituye una unidad de memoria de un bit.

BIOS	Del inglés Basic Input/Output System, sistema básico de entrada/salida. Conjunto de programas codificados en la ROM de las computadoras personales IBM y compatibles.
Bit	Unidad básica de información en un sistema de numeración binario (BInary digiT), representando "0" o "1" lógico (Falso/Verdadero, Cerrado/Abierto, etc.).
Bloque ASM	Combinación de Caja de Estado, Diamante de Decisión y Caja de Salidas Condicionales. Ésta estructura consiste de una Caja de Estado y/o una red de Diamantes de Decisión y/o Cajas de Salidas Condicionales
Bus	Trayectoria interna a lo largo de la cual se envían señales de una parte de la computadora a otra parte de la misma. Las computadoras personales tienen un bus diseñado con tres trayectorias: ✓ El bus de datos envía información de ida y regreso entre la memoria y el microprocesador. ✓ El bus de direcciones identifica qué localidad de memoria se usará. ✓ El bus de control conduce las señales de la unidad de control.
Byte de Atributos	Byte en la memoria de vídeo que contiene la información de despliegado del carácter que le precede, esta información son: los colores de forma y fondo, brillo, parpadeo, vídeo inverso, etc.
Byte, KiloByte (KB), MegaByte (MB), GigaByte (GB)	Unidades de medida de almacenamiento en computadoras, sus equivalencias son: 1 Byte = 8 bits, 1 KB = 2^{10} bits = 1024 bits, 1 MB = 2^{20} bits = 1 048 576 bits, 1 GB = 2^{40} bits = 1,099,511,627,776 bits
C++	Lenguaje de programación de alto nivel desarrollado por Bjarne Stroustrup en los Laboratorios Bell de AT&T. A C++, que combina las ventajas del lenguaje C con las de la programación orientada a objetos, se le denomina lenguaje híbrido.
CAD/CAM	Computer Aided Design/Manufacture (Diseño/Manufactura Asistido por Computadora). Uso de la computadora y de un programa de diseño asistido por computadora como ambiente para el diseño y manufactura de una variedad de artefactos industriales, que van desde componentes de máquinas hasta casas modernas.

Caja de Diálogo	De acuerdo con las normas industriales y las interfaces gráficas para usuarios, ésta es un recuadro en pantalla con un mensaje que transmite o solicita información al usuario.
Caja de Estado	Representación diagramática del Elemento Estado de la ASM. Está compuesta de un "Nombre", un "Código de estado", una forma rectangular donde por lo regular serán listadas las salidas generadas durante un tiempo de Estado.
Caja de Salidas Condicionales	Representación diagramática del Elemento Salidas Condicional de la ASM. Está compuesta de una forma rectangular redondeada, donde serán listadas las salidas activas, sólo si ciertas condiciones son verdaderas y, siempre y cuando la ASM se encuentre en el estado al que está asociada.
Carácter	Cualquier letra, número, signo de puntuación o símbolo que puede generarse en pantalla al presionar una tecla. Nota: Es un error pronunciar y escribir *caracter*, es decir, sin acento prosódico.
Caracteres ASCII	Patrones de caracteres en un arreglo matricial de 8 x 9 bits almacenados en la ROM de la computadora.
Cartas ASM	Ver ASM.
Cartas de Estructura	Muestra la arquitectura de un sistema como una jerarquía de funciones (cajas) arregladas en una estructura semejante a un árbol. Identifica interconexiones entre funciones y parámetros de entrada y salida. No muestra estructuras de control como condición, secuencia, iteración o selección.
CASE	Ingeniería de Software Asistido por Computadora, herramienta para el desarrollo de aplicaciones basadas en computadoras.
CGA	Color Graphics Adapter, Adaptador de Gráficos en Colores. Adaptador para presentar gráficos en mapa de bits para Computadoras Personales. Este adaptador despliega cuatro colores de forma simultánea con una resolución de 200 pixeles en posición horizontal y 320 líneas en posición vertical, o un color con una resolución de 640 pixeles en posición horizontal por 200 líneas en posición vertical.
Ciclo de la Señal de Reloj	Tiempo determinado por una entrada periódica de reloj a un dispositivo electrónico.

Caja de Diálogo	De acuerdo con las normas industriales y las interfaces gráficas para usuarios, ésta es un recuadro en pantalla con un mensaje que transmite o solicita información al usuario.
Caja de Estado	Representación diagramática del Elemento Estado de la ASM. Está compuesta de un "Nombre", un "Código de estado", una forma rectangular donde por lo regular serán listadas las salidas generadas durante un tiempo de Estado.
Caja de Salidas Condicionales	Representación diagramática del Elemento Salidas Condicional de la ASM. Está compuesta de una forma rectangular redondeada, donde serán listadas las salidas activas, sólo si ciertas condiciones son verdaderas y, siempre y cuando la ASM se encuentre en el estado al que está asociada.
Carácter	Cualquier letra, número, signo de puntuación o símbolo que puede generarse en pantalla al presionar una tecla. Nota: Es un error pronunciar y escribir *caracter*, es decir, sin acento prosódico.
Caracteres ASCII	Patrones de caracteres en un arreglo matricial de 8 x 9 bits almacenados en la ROM de la computadora.
Cartas ASM	Ver ASM.
Cartas de Estructura	Muestra la arquitectura de un sistema como una jerarquía de funciones (cajas) arregladas en una estructura semejante a un árbol. Identifica interconexiones entre funciones y parámetros de entrada y salida. No muestra estructuras de control como condición, secuencia, iteración o selección.
CASE	Ingeniería de Software Asistido por Computadora, herramienta para el desarrollo de aplicaciones basadas en computadoras.
CGA	Color Graphics Adapter, Adaptador de Gráficos en Colores. Adaptador para presentar gráficos en mapa de bits para Computadoras Personales. Este adaptador despliega cuatro colores de forma simultánea con una resolución de 200 pixeles en posición horizontal y 320 líneas en posición vertical, o un color con una resolución de 640 pixeles en posición horizontal por 200 líneas en posición vertical.
Ciclo de la Señal de Reloj	Tiempo determinado por una entrada periódica de reloj a un dispositivo electrónico.

Circuitos Integrados	Circuito semiconductor que contiene más de un transistor y otros componentes electrónicos.
Clase	Un conjunto o colección de objetos con características comunes. La clase es la "plantilla" codificada y por lo tanto incluye detalles de implementación completa mientras permanece en conformidad con la especificación del Tipo Abstracto de Dato (TAD). Una clase "diferida" o "abstracta" es una en la cual algunos servicios no son implementados, esto es diferida a una clase derivada. Tales clase diferidas no pueden por lo tanto ser instanciadas por si mismas.
Clase Base	En el argot de la OO, la clase base es aquella que contiene la mayor generalización en una estructura de clases. De las clases bases se crean nuevas, que contienen las características de dicha clase y las propias.
Clase Derivada	Clase derivada es aquella que se crea de una clase previamente definida.
Clasificación	Noción abstracta de agrupamiento de objetos similares en clases.
Código de Estado	En la teoría de Cartas ASM, combinación única de las variables de estado.
Concurrencia	Es descrita en términos de múltiples secuencias que pueden ser ejecutadas en paralelo por múltiples procesadores o simuladas secuencialmente por un solo procesador.
Configuración	Selecciones realizadas en la inicialización de un sistema de cómputo o en un programa de aplicación, para que cumpla con las necesidades del usuario.
Coordenadas Físicas	Son las coordenadas que establecen los dispositivos físicos.
Coordenadas Lógicas	Son las coordenadas que se establecen al mover el origen del sistema coordenado en pantalla a un nuevo origen determinado.
CPU	Del inglés, Central processing unit, Unidad Central de Procesamiento. Circuitería de control, procesamiento y almacenamiento interno de la computadora, incluyendo la Unidad Aritmética Lógica (ALU, por sus siglas en inglés), la Unidad de Control y el almacenamiento principal.
CUI	Interface de usuario basada en caracteres. Aquí no se usa la capacidad gráfica de la computadora, es decir, se usa el modo texto de la computadora.
Cursor	Carácter parpadeante en pantalla que le muestra dónde aparecerá el siguiente carácter. Ver apuntador.

DEFD	Diagrama de Entidad-Flujo de Datos (Entity-DataFlow Diagram). Una variante en el diagrama flujo de dato en el cual cada nodo de proceso contiene ya sea una entidad activa o alguna función relacionada a una entidad activa, mejor que procesos desincorporados. Las entidades y funciones activas son encerradas en una burbuja, Las burbujas son conectadas a cada una de las otras y los datos almacenados por arcos etiquetados conteniendo flujos de datos. Los flujos de datos y los datos son entidades pasivas.
Definición de Clase	Especificaciones distintivas de su clase.
Definición/ Construcción Dinámica	Instanciación de un identificador o variable, definido por uso de un lenguaje orientado a objetos, con un objeto durante la ejecución de un sistema.
Densidad de Impresión	Calidad de reproducción de imágenes sobre papel.
DEOO	Diseño estructurado orientado al objeto. Metodología desarrollada por Wasserman, Pirchar y Muller, que provee una notación detallada para describir el diseño estructural o de alto nivel, que identifica los módulos individuales pero sin representar su detalle interno.
Desarrollo Orientado a Objetos	Construcción de un sistema orientado a objetos de sus requerimientos por análisis, diseño y programación orientada a objetos.
Descomposición Funcional	Técnica mediante la cual se diseña un sistema de computadora, considerando las funciones que interaccionarán unas con otras, además del manejo de los elementos de datos del propio sistema. De lo anterior podemos decir que es la representación del sistema mediante un conjunto de funciones.
DFD	Diagrama de Flujo de Datos, herramienta de la metodología "Análisis y Diseño Estructurado" de Gane & Sarson, que permite modelar de forma lógica un problema del mundo real para su posterior implantación por computadora.
Diagrama de Estado	Representación diagramática en la que los estado son representados mediante círculos, sin nombre, ni código, y las transiciones entre estados, mediante flechas. No existe una forma de representar una condición ni salidas condicionales.

Diagramas de Flujo	Diagrama que contiene símbolos referentes a operaciones de cómputo que describen cómo se realiza un programa, o dicho de otra forma, es un método gráfico para describir algoritmos.
Diamante de Decisión	Representación diagramática del elemento Decisión de la ASM. Está compuesta de una "Variable de prueba", una forma rómbica donde es listada la variable de prueba y dos ramas de salida, etiquetadas como "0" y "1" lógicos. Aquí se involucran las entradas al sistema.
Digitalización	Transformar una imagen continua en información legible para la computadora mediante un dispositivo conocido como digitalizador.
Diseño	Método de desarrollo de sistemas en el cual se establece la arquitectura del sistema. Primero se establece la forma lógica en el cual se interrelacionan los elementos, la segunda parte consiste del desarrollo de la forma física en la cual se implantará el sistema.
Diseño de Sistemas Digitales	Método de desarrollo de sistemas electrónicos basado en dispositivos digitales y álgebra de Boole.
Dispositivo Apuntador	Dispositivo de entrada -un ratón, una pluma óptica o una tablilla de gráficos con pluma, por ejemplo- empleado para mostrar un apuntador en pantalla.
Disquete	Disco flexible tanto en 3½ como en 5¼ de pulgada, empleado como medio de almacenamiento secundario en computadoras personales. En el formato de 3½ el disco magnético va dentro de una funda de plástico rígido.
DMR	Método para el desarrollo de aplicaciones, denominado "Diseño de manejo de responsabilidades" de Wirfs-Brock, Wilkerson y Wiener. La metodología se aboca, durante el diseño, a los contratos entre los objetos clientes y los servicios. Dichos contratos indican expresamente de qué acciones es responsable el objeto y qué datos está obligado a compartir.
DOO	(Léase "d doble o"). Método de diseño que redunda en el proceso de la descomposición orientada a objetos y a una notación lógica y física, así como modelos estáticos y dinámicos del sistema bajo diseño. Específicamente, esta notación incluye diagramas de clase, diagramas de objetos, etc.

EGA	Del inglés, Enhanced Graphics Adapter, Adaptador Gráfico Mejorado a color en *mapa de bits para computadoras personales*, introducido en 1984. Los adaptadores EGA exhiben de manera simultánea hasta 16 colores con una resolución de 640 x 350, 640 x 200 y 320 x 200 pixeles. Compatible con los *modos* CGA.
Eiffel	Medio ambiente de programación basado en el concepto de objeto, desarrollado por Bertrand Meyer.
Encapsulamiento	El Encapsulamiento es el empaquetamiento de un objeto de la información de lo que es (estructura) y de cómo es (funcionalidad).
Entidad Autocontenida	Ver autocontención.
Entidades Activas/Pasivas	En la especificación de requerimientos orientado a objetos de Bailin, las entidades activas son las que efectúan operaciones (sobre ellas mismas o en otras entidades) lo suficientemente importante para ser consideradas en detalle durante la fase de análisis. Las pasivas son de menor importancia y pueden ser tratadas como cajas negras hasta la fase de diseño.
Entrada de Control	En la teoría de Cartas ASM, variable que sirven para tomar una decisión. También llamada variable de prueba, estas son variables físicas (pertenecientes al problema mismo, como por ejemplo, alguna señal para controlar una luz de un semáforo) y variables lógicas (de control interno de la ASM).
Espacio de Solución	En el contexto de la OO, son aquellos objetos que definen la solución del problema.
Espacio del Problema	En el contexto de la OO, son aquellos objetos que forman parte del problema, es decir, no aportan elementos de solución al problema.
Estado	En el contexto de sistemas digitales, es la condición de las Variables de Estado en un instante dado.
Estado activo	Es el estado definido por el valor de la variables de estado en un instante dado. Sinónimo de Estado Presente.
Estado Local	En la orientación a objetos, estado que los elementos internos de un objeto mantienen, compuesto por variables de uso permitido solamente a él mismo.

Estado Siguiente	Es el estado determinado por la Función Próximo Estado de la Máquina de Estados General. Concretando, podemos decir que es el estado al que pasará la Máquina después de estar en el estado presente durante un tiempo de estado.
Estrategia Informal	Descripción en lenguaje natural de la solución del problema que hay que resolver mediante software, representado a un nivel consistente de detalle.
Estructura de Datos/Función Privada	En la OO, declaración que forma parte de una clase, objeto o módulo y que es no visible para otra clase, objeto o módulo. De lo anterior se deduce que sólo la clase, objeto o módulo la puede modificar.
Estructuras de Control	Organización lógica para un algoritmo que gobierna la secuencia en que se ejecutan las instrucciones de un programa. Las estructuras de control gobiernan el flujo de control en un programa. El uso de estructuras secuenciales, de ramificación y de ciclos para expresar un algoritmo es mucho más que una buena práctica; esta técnica es válida por razones de importancia científica. Una brillante prueba, matemática demostró que estas tres estructuras son apropiadas para la expresión de procedimientos de cualquier algoritmo conocido.
Estructuras Orientadas a Objetos	Estructuras que han ido apareciendo en la ciencia, así como el pensamiento orientado a objetos, para mejor comprensión de la realidad.
Fondo del Carácter (Background)	Características de un carácter, como son: el color de fondo y parpadeo.
Forma del Carácter (Foreground)	Características de un carácter, como son: el color de la forma, brillo y parpadeo.
Fuente de Letra	Del inglés Fonts, colección completa de letras, signos de puntuación, números y caracteres especiales con una tipografía, peso (Roman o negrita), postura (vertical o itálica) y tamaño de fuente consistente e identificables. Hay dos clases de fuentes de tipos: las fuentes en mapas de bits y las fuentes por algoritmo.
Función de Programación	En lenguaje de programación, procedimiento designado y guardado que regresa un valor.

Gráficos de Bloques	Al trabajar con computadoras personales, gráficas que se forman en la pantalla o se imprimen en papel mediante caracteres semigráficos del conjunto extendido de caracteres. Los caracteres semigráficos del conjunto extendido de caracteres gráficos de IBM son adecuados para crear rectángulos en pantalla, pero no para detalles finos. Como los caracteres gráficos de bloques se manejan de la misma manera que los caracteres ordinarios, la computadora puede mostrar gráficas de bloques con más rapidez que una gráfica en mapa de bits.
Gráficos de Punto	Gráficos que se forman al especificar exactamente donde será impreso cada punto en pantalla.
Hardware	Componentes electrónicos y/o electromecánicos, como tarjetas, periféricos y equipo que conforman un sistema de cómputo.
Hércules	Adaptador gráfico monocromático de video para computadoras personales, este adaptador exhibe texto y gráficas en un monitor con una resolución de 720 pixeles en posición horizontal por 320 líneas en posición vertical.
Herencia	Equivalente a una relación taxonómica entre "padres" e "hijos", posiblemente sobre muchas "generaciones". En la OO, una relación entre dos clases de objetos tales que una de las clases, el hijo, toma todas las características relevantes de la otra clase, el padre. Esta relación puede ser de 1 a 1, 1 a muchos, muchos a 1 ó muchos a muchos.
IBM8514/A	Adaptador de video para las computadoras de la serie PS/2 (sistema personal) de IBM que, con una tarjeta VGA (matriz de gráficos de video), produce una resolución de 1,024 pixeles en posición horizontal y 768 líneas en posición vertical. El adaptador también contiene sus propios circuitos de proceso, lo que reduce la demanda de la CPU de la computadora.
Icono	Signo que mantiene una relación de semejanza con el objeto que representa.
Impresión Gráfica	Reprocesamiento de la información contenida en archivo o pantalla de computadora para imprimirla en papel.
Informática	Procesamiento automático de información mediante la computadora, tal nombre tuvo su origen en el término francés informatique. Corresponde a las designaciones inglesas de computer science y electronic data processing.

Ingeniería de Software	Aplicación de un aprovechamiento sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software.
Instancia	En la OO, creación de un elemento individual (un "objeto" en tiempo de ejecución) de la plantilla de la clase.
Inteligencia Artificial (IA)	Campo de la ciencia de la computación cuyo propósito es mejorar las computadoras para tratar de dotarlas con algunas características asociadas con la inteligencia humana, como la capacidad de entender el lenguaje natural y razonar bajo condiciones de incertidumbre.
Interface	Es un término inglés peculiar, injertado en la nomenclatura de la computación y la informática y que proviene del campo de la física, de interface (la superficie intermedia, o interficie, que separa dos regiones) que se aplica a un dispositivo adaptador o medio de enlace entre equipos electrónicos de computación o de otra índole. Dado que se trasladó homográficamente al francés como interface, se ha tratado de adoptarlo en español con la misma grafía: interface. Sin embargo, ello ha hacho que se le confunda con otro término que en español es su homófono: interfase, y que tiene un significado originalmente distinto. Para evitar tal confusión es preferible el vocablo interfaz, que sirve para diferenciar los conceptos. Tal término es de género femenino y debe interpretarse como "entrecara".
Interface del Objeto	En el contexto de la OO, servicios proporcionados por un objeto, es decir, las funciones, métodos o procedimientos, que son conocidos por los elementos externos al objeto.
Interface Gráfica de Usuario (GUI)	Diseño para la parte de un programa que interactúa con el usuario y aprovecha en su totalidad la ventaja de las pantallas de gráficas en mapas de bits de las computadoras personales. Al igual que la interface estándar de la industria, cada vez más común en aplicaciones DOS, una GUI emplea Barra de Menú, Menús Desplegables, Cajas de Diálogo, Ventanas, Botones, etc.
Iteración	Repetición, ciclo de un conjunto de instrucciones.
Justificación	En tipografía, alineación de numerosas líneas de texto a lo largo del margen izquierdo, del margen derecho, o de ambos.

LAN	Local area network. Red de área local. Enlace de computadoras personales y de otro tipo dentro de una área limitada por medio de cables de alto desempeño, con el fin de que los usuarios puedan intercambiar información, compartir periféricos costosos y recurrir a los recursos de la unidad de almacenamiento masivo secundario, llamado servidores de archivos. Un conjunto de normas (protocolo de red) se encarga de controlar el flujo de información dentro de la red. Estas normas determinan el momento y la forma en que un nodo puede iniciar un mensaje. Los protocolos también se encargan de manejar los conflictos que se suscitan cuando dos nodos comienzan a transmitir al mismo tiempo. Los componentes básicos de una LAN son cables, una tarjeta de interface de red, un servidor de archivos (que incluye un almacenamiento central masivo), un sistema operativo para red (NOS) y computadoras personales o estaciones de trabajo enlazadas por el sistema. Existen tres tipos de topología para red (métodos para interconectar las estaciones de trabajo de la red): redes de bus, redes de anillo y redes en estrella. Asimismo, existen dos métodos para transmitir información a través de los cables de red: el de banda de base y el de banda ancha.
Lenguaje Basado en Objetos	Aquellos que utilizan objetos pero sin posibilidades de modificación o de creación de ellos.
Lenguaje Declarativo	Lenguaje de programación que libera al programador de especificar el procedimiento exacto que la computadora necesita llevar a efecto para completar una tarea. Los programadores usan el lenguaje para describir un conjunto de hechos y relaciones para que el usuario pueda consultar el sistema a fin de obtener un resultado específico. Por ejemplo, el lenguaje de consulta estructurado (SQL), le permite realizar una búsqueda preguntando por una lista de registros que muestre información específica, en vez de indicarle a la computadora que busque en todos los registros los que tienen las entradas apropiadas en los campos especificados.
Lenguaje Orientado a Objetos	Una notación bien definida que soporta propiedades y especificaciones orientadas a objetos de un sistema orientado a objetos. Lenguajes que utilizan objetos y además permiten la definición de clases y de herencia.

Lenguaje de Programación	Lenguaje de programación: Compuesto por un vocabulario fijo y un conjunto de reglas (llamado sintaxis), que se usa para crear instrucciones que debe seguir la computadora. De manera convencional, los lenguajes de programación se dividen en dos: alto nivel y los de bajo nivel. Otra forma de diferenciar los lenguajes de programación consiste en hacer una distinción entre los lenguajes de procedimientos y los declarativos. En un lenguaje de procedimientos, el programador debe especificar el procedimiento que la computadora seguirá para llevar a cabo un objetivo determinado. En un lenguaje declarativo (conocido también como lenguaje de no procedimiento), el lenguaje define un conjunto de hechos y relaciones y permite consultar resultados específicos. Entre los ejemplos de lenguajes declarativos están PROLOG y el lenguaje de preguntas estructurado (SQL, por sus siglas en inglés
Listas	En programación, estructura de datos que organiza y vincula cada elemento de datos con un apuntador que muestra la ubicación física del elemento en una base de datos. Mediante una lista, un programador puede organizar la información de varias maneras sin alterar la ubicación física de los datos.
Manejo de Responsabilidades	Ver DMR.
Mantenimiento	Etapa del ciclo de vida de los sistemas en que son sometidos a un proceso de actualización conforme a los nuevos requerimientos de sus usuarios.
Mapa de Bits	Representación de una imagen de vídeo almacenada en la memoria de la computadora. Cada elemento gráfico (pixel), correspondiente a un pequeño punto en la pantalla, es controlado por un código de encendido o apagado (on/off) guardado como un bit en la memoria de la computadora.
Máquina de Estados Algorítmica	Ver ASM.

Máquina de Estados General	Estructura compuesta de 3 submódulos: Módulo de Función Próximo Estado, Módulo de Registro de Estado y Módulo de Función de Salida. Estos tres módulos operan sobre un conjunto de entradas X para producir un conjunto de salidas Z. Ver Fig. III.1 Diagrama de la Máquina de Estados General.
Máscara	Patrón de símbolos o caracteres que, al imponerlo sobre un campo de datos, limita el tipo de caracteres que pueden escribirse en el campo.
Máximo Encapsulamiento	Énfasis en las responsabilidades que deben tener los objetos.
MCGA	Adaptador de gráficos en multicolores. Estándar de visualización de pantalla del Sistema Personal/2 de IBM. El MCGA agrega 64 tonos de gris a la norma CGA y suministra la resolución estándar EGA de 640 pixeles por 350 líneas con 16 colores posibles.
Memoria de Vídeo	Memoria RAM dedicada al vídeo, separada de la tarjeta principal de la computadora. La computadora almacena forma y fondo en ella para caracteres en modo texto o imágenes en modo gráfico
Mensaje	El requerimiento de un objeto por los servicios/asistencia de un segundo objeto -similar a una llamada a subrutina en un lenguaje procedural-. También se le conoce como protocolo, y está compuesto de un método o rutina y una referencia o dirección de objeto.
Menú	En programación, despliegue en pantalla que lista las opciones de comandos disponibles.
Método	En la OO, un procedimiento o función especificada en la clase objeto. También conocido como servicio, operación o rutina.
Microcomputadora	Cualquier computadora con su unidad lógica-aritmética (ALU) y unidad de control incluidas en un circuito integrado, llamado microprocesador.
Modalidad de Vídeo	En el modo gráfico, alguna configuración en particular de la tarjeta gráfica.
Modelado	Representación matemática o pictórica de un objeto o sistema que existe en el mundo. La efectividad de un modelo guarda relación directa con las hipótesis que lo fundamentan. Si las hipótesis son incorrectas, o si no se incluye información importante en el modelo, éste no reflejará con precisión el comportamiento del prototipo.

Modelo de Mealy/Moore	Concepto matemático correspondiente a una máquina de estados finito -una máquina hipotética que puede existir en sólo uno de un número finito de estados en un momento dado- que responde a estímulos externos, cambiando su estado y produciendo salidas.
Modo Gráfico	Modo operativo de las tarjetas de vídeo para computadoras personales, en el que la computadora puede exhibir imágenes en mapas de bits, caracteres con amplia gama de características. En este modo de despliegue en el que todo lo que aparece en pantalla -incluyendo texto y gráficas- está creado por la iluminación selectiva del adaptador sobre los pequeños puntos de la pantalla llamados pixeles.
Modo Texto	Modo operativo de las tarjetas de vídeo para computadoras personales en el que la computadora presenta únicamente las imágenes que pueden construirse con el conjunto estándar de 254 caracteres de IBM. Aunque este conjunto de caracteres incluyen un número limitado de caracteres gráficos, el modo de texto puede exhibir imágenes gráficas simples como cajas o líneas. Además, el texto puede aparecer en negritas y en vídeo inverso. En modo de texto no se pueden exhibir gráficas en mapa de bits, caracteres en cursivas, tipos de letras distintos a los de pantalla, tamaños de tipos, ni caracteres colocados arriba o debajo de la línea base. El modo texto corre con más rapidez porque dibuja sobre los caracteres integrados y preparados de la computadora en vez de formarlos de manera individual.
Modos Texto de Despliegue	En el Modo Texto de la computadora, son los modos particulares de trabajo de desplegados, los cuales controlan el tamaño de los caracteres y el color en el desplegado.
Modularidad	La propiedad de un sistema que tiene de ser descompuesto dentro de un conjunto de módulos. La modularidad es uno de los elementos funcionales del modelo de objeto.
Módulo	En un programa de computación, unidad o sección capaz de funcionar por sí misma.
Módulo de Función de Salida	En la teoría de Cartas ASM, módulo de transformación encargado de definir las salidas que estarán activas mientras la Máquina de Estados General permanece en el estado presente.

Módulo de Función Próximo Estado	En teoría de Cartas ASM, módulo de transformación encargado de definir a que estado pasará la Máquina de Estados General, después de permanecer en el estado actual, durante un tiempo de estado.
Módulo de Registro de Estado	En teoría de Cartas ASM, módulo que contiene elementos de memoria que almacenan el código del estado actual de la Máquina de Estados General.
Monitor	Dispositivo completo que produce una visualización en pantalla, incluyendo toda la circuitería de soporte interno necesaria. Al monitor también se le conoce como unidad de despliegue de vídeo o tubo de rayos catódicos (CRT). En programación, es el equivalente a un objeto utilizado para representar una entidad concurrente en un sistema de tiempo real.
Multimedia	Método basado en computadora que sirve para presentar información mediante el empleo de diversos medios de comunicación y en el que se destaca la interactividad. En general, la multimedia combina texto, gráficas y sonido. Como las gráficas y el sonido requieren bastante espacio de almacenamiento, la configuración mínima para un sistema de multimedia incluye una unidad CD-ROM.
Nivel de Abstracción	Jerarquía del modo de pensar mediante el cual consideramos nuevos elementos, cualidades y detalles que antes eran innecesarios pero que ahora nos proporcionan información para dilucidar más claramente el problema inicial.
Nodo Conector	Representación diagramática para entradas múltiples hacia un símbolo ASM. Este símbolo es opcional para la representación de la Carta ASM.
Objetive C	Lenguaje de programación basado en C con los conceptos de la orientación a objetos, desarrollado por Brad Cod de StepStone Corp. Este lenguaje corre para el sistema operativo NextStep, el cual a su vez es un sistema operativo orientado a objetos. A este lenguaje también se le conoce como C objetivo.
Objeto	En la OO, una abstracción de una entidad del mundo real. En tiempo de ejecución, una instanciación única de la plantilla clase. En otro contexto, una colección de cosas similares. Usado en análisis y diseño. también puede ser referido como objeto o entidad.

MCO	En la OO, modelo de comunicación de objeto, el cual muestra el control síncrono entre los objetos.
Ocultamiento de Información	Separación de la representación de detalles de un objeto, clase o sistema, de los detalles del dominio de aplicación.
OO	Léase "doble o". Es la aplicación sistemática de la forma de pensamiento -modelo orientado a objetos- relacionados a la computación, en el cual el sistema se descompone en entidades (objetos) jerarquizadas y clasificadas que corresponden a elementos específicos. Se hace énfasis en los conceptos de clase, herencia, encapsulamiento, etc. También lo podemos denominar paradigma de la orientación a objetos.
Operación	Ver método.
Operaciones Locales	En la OO, aquellas operaciones de un objeto que sólo pueden ser llamadas por otra operación local o por una de interface, más no por las llamadas directas de otros objetos.
Operaciones no Locales	En la OO, aquellas operaciones de un objeto que sólo pueden ser llamadas por medio de un mensaje originado por un objeto externo. También se le conoce como operaciones de interface.
Orientación a Objetos	Ver OO.
Origen Físico	En un sistema coordenado en la pantalla, es el origen establecido por los dispositivos físicos: monitor y adaptador de vídeo.
Origen Lógico	Es el origen establecido al moverlo a una posición determinada; a partir de ese instante, el origen (0,0) estará en la posición indicada.
Páginas de Vídeo	Bloque de tamaño fijo de memoria de acceso directo (RAM).
Palabra	Agrupación de bits en un dispositivo electrónico.
Paleta	En monitores de computadora, repertorio de colores que puede mostrar el sistema. Los monitores a color VGA ofrecen una paleta de 262,144 colores; no obstante, cada pantalla puede exhibir un máximo de 256 colores al mismo tiempo.
Paradigma	Ejemplo, modelo, prototipo. Un modelo generalizado a gran escala que proporciona un punto de vista desde donde el mundo real puede ser dilucidado.

Paradigma de la Orientación a Objetos	Ver OO.
Parte Pública	En la OO, parte que define la interface de una clase, objeto o módulo, y que es visible para todas las otras clases, objetos o módulos.
PC3270	Norma de visualización de gráficos a color en mapa de bits. Los adaptadores PC3270 muestran 2 colores, con resolución de 720 pixeles en posición horizontal y 350 líneas en posición vertical.
Persistencia	En la OO, característica que hace que la información que portan los objetos permanezca a lo largo de la operación de un sistema, haciendo accesible la "historia" del sistema a través de la información de sus objetos componentes
PGA	Del Inglés Professional Graphics Adapter, adaptador de gráficos profesional. Norma de visualización de gráficos a color en mapa de bits introducida en 1984. Los adaptadores PGA muestran hasta 256 colores, con resolución de 640 pixeles en posición horizontal y 480 líneas en posición vertical. Compatible con CGA.
Pixel	De la contracción de Picture Element, elemento de imagen más pequeño que se exhibe en pantalla y a partir del cual se construye una imagen desplegada.
Plataforma	Hardware o software estándar de computadoras. En lo relativo a software se refiere a los diferentes sistemas operativos existentes en el mercado de computadoras.
Polimorfismo	En el ámbito de los SOO, puede definirse como la capacidad que tiene un mensaje de ser adoptado por la interface de más de un objeto y ser interpretado distintamente, según sea el tipo de objeto receptor.
POO	(Léase "p doble o"). Método de programación en el cual los programas son organizados como colecciones cooperativas de objetos, cada uno de los cuales representa una instancia de algún tipo, y cuyos tipos son todos miembros de una jerarquía de tipos unidos vía otras relaciones heredables.
Portabilidad	Propiedad de los programas o sistemas que representa su fácil movimiento entre distintos plataformas hardware/software de trabajo.
Procedimiento	Los pasos que definen el uso específico de cada elemento del sistema o el contexto procedimental en que reside el sistema.

Procedural/ Procedimental	Que se enfoca a los procedimientos, mecanismo mediante el cual se le debe indicar a la computadora qué y cómo hacer las actividades deseadas.
Proceso Distribuido	El proceso se distribuye a lo largo de las distintas entidades informáticas de la empresa, esto es opuesto a los procesos centralizados en el cual el proceso productivo se centraliza junto con la administración en una computadora central.
Programación	Ciencia de planear la solución de los problemas al reducir el plan a un conjunto de instrucciones razonables para la máquina que dirige las acciones de un sistema de computación o procesamiento de datos. La computadora realiza operaciones aritméticas y lógicas que dan solución a los problemas.
Programación Declarativa	Ver lenguaje de programación.
Programación Estructurada	Conjunto de normas de calidad que hacen que los programas tengan una verbosidad excesiva pero que sean más legibles, confiables y fáciles de mantener. Se anima a los programadores para que usen ciclos y estructuras de control de bifurcación y a que designen los procedimientos en vez de los enunciados GOTO.
Protocolo de Comunicación	Lista de parámetros de comunicación (configuraciones) y estándares que controlan la trasferencia de información entre computadoras a través de telecomunicaciones. Ambas computadoras deben tener las mismas configuraciones y seguir los mismos estándares para evitar errores. Los programas de comunicaciones permiten elegir los parámetros necesarios, incluyendo la tasa de baudios, los bits de datos, el dúplex, la paridad y los bits de alto. Muchos servicios de comunicaciones usan ocho bits de datos y un bit de alto; también es usual el dúplex total. Quizá se tenga que especificar un parámetro adicional llamado protocolo de diálogo. Este parámetro establece la forma en que una computadora le indica a otro dispositivo cuándo esperar. Casi todas las computadoras y muchos dispositivos periféricos usan un protocolo de diálogo XON/XOFF, que es el parámetro por omisión para muchos programas de comunicaciones.
Prototipo	Ejemplo funcional de un elemento del modelado o del modelado en si mismo.

RAM	Del inglés, Random Acces Memory. Memoria principal de trabajo de una computadora en la que se guardan instrucciones de programa e información para que tengan acceso directo a la CPU. A la RAM se le conoce como memoria de lectura/escritura para diferenciarla de la memoria de sólo lectura (ROM), el otro componente de almacenamiento primario de una computadora personal. En la RAM, la CPU puede escribir y leer.
Ratón (Mouse)	Dispositivo de entrada que cuenta con uno o más botones de control. A medida que se mueve el ratón, sus circuitos transmiten señales que mueven en la misma correspondencia a un apuntador en la pantalla. Se puede usar el ratón para elegir comandos de los menús, seleccionar texto para edición, mover objetos y dibujar ilustraciones en la pantalla. A los ratones se les distingue por el mecanismo interno que emplean para generar sus señales y por los medios con que se conectan a la computadora. Los dos tipos de mecanismos internos más populares son: mecánico y óptico. Los ratones se conectan a la computadora en una de las formas siguientes: por bus, por puerto serial.
Recursión	En programación, es la invocación por parte de una función a si misma.
Redes	Sistema de intercambio de comunicaciones e información basado en computadora, creado mediante la conexión física de dos o más computadoras. Las redes para computadoras personales difieren por su topología, esto es, la geometría de sus conexiones. Entre las topologías comunes de red para computadoras personales están la de estrella, en la que las máquinas son enlazadas a un servidor de archivos, y la de bus, en la que las máquinas se enlazan a un simple cable general.
Refinamiento Sucesivo	Metodología en la cual se realiza una especificación de requerimientos iniciales y superficial. En base a ésta lista se diseña una primera versión del sistema. Tan pronto como se tiene este primer diseño, se realiza una lista más detallada de requerimientos y se procede a redefinir el diseño previo para que cumpla con las nuevas especificaciones. Este proceso se repite tantas veces como sea necesario.

Reloj	Circuito electrónico que genera pulsos uniformemente espaciados a velocidades -generalmente- de millones de ciclos por segundo; los pulsos se usan para sincronizar el flujo de información a través de los canales de comunicación interna de la computadora o un módulo electrónico.
Requerimientos	Necesidades y funciones que deberá cubrir un sistema. Se suele cubrir además las especificaciones sobre el tiempo de desarrollo y el equipo que se utilizará.
Resolución	Medida de la nitidez de una imagen generada por un dispositivo de salida, como un monitor o una impresora; por lo general se expresa en puntos por pulgada lineal, tanto horizontal como verticalmente. ☞ En los monitores, la resolución se expresa como el número de pixeles que aparecen en pantalla en posición horizontal y el de líneas que aparecen en posición vertical. ☞ En las impresoras, la resolución se mide por lo general por el número de puntos por pulgada (PPP) que la impresora es capaz de imprimir: cuanto más alto es el número, mayor es la resolución.
Responsabilidades	Ver DMR.
Reusabilidad	En OO, es la habilidad que tiene un elemento del sistema para ser aprovechado en la construcción de nuevos sistemas, entendiéndose como elemento del sistema a cualquier definición de objeto, jerarquía de clase, diseño o análisis propio de un problema.
ROM	Del inglés, Read Only Memory, Memoria de Sólo Lectura. Porción del almacenamiento principal de una computadora que no pierde su contenido cuando se interrumpe el flujo de energía eléctrica y que contiene programas de sistemas esenciales que no se pueden borrar.
Salidas Condicionales	En la teoría de Cartas ASM, salidas que están activas, sólo si ciertas condiciones son válidas, siempre y cuando la ASM se encuentra en el estado al que está asociada.
Salidas de Control	En teoría de Cartas ASM, son todas aquellas que proporcionarán una forma de comportarse a la ASM, también conocidas como instrucciones.
Salidas de Estados	En teoría de Cartas ASM, aquellas salidas que están presente en un estado particular de la ASM.

Reloj	Circuito electrónico que genera pulsos uniformemente espaciados a velocidades -generalmente- de millones de ciclos por segundo; los pulsos se usan para sincronizar el flujo de información a través de los canales de comunicación interna de la computadora o un módulo electrónico.
Requerimientos	Necesidades y funciones que deberá cubrir un sistema. Se suele cubrir además las especificaciones sobre el tiempo de desarrollo y el equipo que se utilizará.
Resolución	Medida de la nitidez de una imagen generada por un dispositivo de salida, como un monitor o una impresora; por lo general se expresa en puntos por pulgada lineal, tanto horizontal como verticalmente. ☛ En los monitores, la resolución se expresa como el número de pixeles que aparecen en pantalla en posición horizontal y el de líneas que aparecen en posición vertical. ☛ En las impresoras, la resolución se mide por lo general por el número de puntos por pulgada (PPP) que la impresora es capaz de imprimir: cuanto más alto es el número, mayor es la resolución.
Responsabilidades	Ver DMR.
Reusabilidad	En OO, es la habilidad que tiene un elemento del sistema para ser aprovechado en la construcción de nuevos sistemas, entendiéndose como elemento del sistema a cualquier definición de objeto, jerarquía de clase, diseño o análisis propio de un problema.
ROM	Del inglés, Read Only Memory, Memoria de Sólo Lectura. Porción del almacenamiento principal de una computadora que no pierde su contenido cuando se interrumpe el flujo de energía eléctrica y que contiene programas de sistemas esenciales que no se pueden borrar.
Salidas Condicionales	En la teoría de Cartas ASM, salidas que están activas, sólo si ciertas condiciones son válidas, siempre y cuando la ASM se encuentra en el estado al que está asociada.
Salidas de Control	En teoría de Cartas ASM, son todas aquellas que proporcionarán una forma de comportarse a la ASM, también conocidas como instrucciones.
Salidas de Estados	En teoría de Cartas ASM, aquellas salidas que están presente en un estado particular de la ASM.

SDLC	Ciclo de vida del desarrollo de sistemas. Metodología desarrollada a finales de los 60's. La cual consiste en la descomposición del proceso en fases discretas de proyecto que entregan documentos formales a la siguiente fase.
Segmento de Nivel Inmediato Inferior	Elemento de la herramienta elaborada para el desarrollo de la presente aplicación. Es un bloque formado por réplicas del segmento de nivel superior desarrolladas sobre la TOOBANS, que muestra la recursividad de la metodología, usada para presentar todos y cada uno de los candidatos a objetos enlistados. Se compone de: un enunciado que describe y reduce la abstracción de los objetos, una Tabla de Operaciones, OBjetos y Atributos de Nivel Inmediato Inferior (TOOBANII), comentarios y observaciones.
Segmento de Nivel Superior	Elemento de la herramienta elaborada para el desarrollo de la presente aplicación. Éste ilustra un enunciado o descripción realizado a un nivel "inicial" de abstracción, así como las operaciones, objetos y atributos desprendidos de allí. Se compone además de la TOOBANS y una sección de consideraciones y comentarios.
Selección	En programación, bifurcación o estructura de control condicional.
Self	Medio ambiente de programación basado en el concepto de objeto desarrollado por Ungar-Smith.
Servicio	Ver método.
Simula	Lenguaje de programación de propósito general desarrollado por Ole'Johan Dahl y Kristen Nygaard en Noruega en el año de 1967, que incluyó la clasificación y objetos aunque nunca se hizo popular. Su uso se suscribió al modelado de problemas y simulación, y sus características de objetos eran conocidas sólo por unos cuantos investigadores.
Síntesis de la carta	En teoría de Cartas ASM, procedimiento mediante el cual se obtienen los resultados en forma tabular de la representación de la Carta ASM, su programa que la gobierna (denominado también microprograma), y su representación electrónica (alambrado físico de la Carta ASM).
Sistema Coordenado	Método para localizar un punto en un espacio de dos dimensiones mediante la definición de un eje vertical y uno horizontal; lo creó el matemático René Descartes en el siglo XVII.
Sistema de Video	Sinónimo de adaptador de video.

Sistema Distribuido	Sistema de procesos distribuidos. Sistema de computadora diseñado para múltiples usuarios que proporciona a cada uno una computadora funcionalmente completa. Sin embargo, a diferencia de los sistemas individuales, un sistema distribuido está diseñado para facilitar la comunicación entre las computadoras enlazadas y ofrecer acceso compartido a archivos centrales.
Sistemas Digitales	Es un dispositivo físico, normalmente un circuito electrónico caracterizado por el hecho de tener un número finito de estados posibles.
SmallTalk	Lenguaje de programación de alto nivel y ambiente de programación que conceptúa los cómputos como objetos que se envían mensajes entre sí. Desarrollado por Alan Kay y otros de PARC (Palo Alto Research) de Xerox Corporation, SmallTalk es distinto a otros lenguajes de programación porque las funciones de programación se expresan en términos de la metáfora dominante de objetos que se envían mensajes entre sí. Más que un lenguaje de programación, SmallTalk es un ambiente de programación completo que cuenta con una interface gráfica de usuario, menús de despliegue descendente y soporte de ratón.
Software	Programa de sistema, utilerías o aplicaciones expresados en un lenguaje legible para las computadoras.
SOO	(Léase "s doble o"). Sistema orientado a objetos, es aquel que modela al sistema como una colección de "prototipos" de los objetos que componen el espacio de solución del sistema.
SOP	En el desarrollo de sistemas, sistema orientado al proceso
SVGA	Del Inglés, Super Video Graphics Array, Arreglo de Gráficos de Vídeo. Norma de visualización de gráficos a color en mapa de bits introducida en 1989. Los adaptadores SVGA muestran al mismo tiempo hasta 16 colores, con una resolución de 800 pixeles en posición horizontal y 600 líneas en posición vertical. Compatible con los modos: CGA, EGA y VGA
Tabla de Transición	En los sistemas digitales, es la tabla que describe el flujo del sistema, es decir, es la representación en forma de tabla de la relación Estado Presente-Estado Futuro.
Tarjeta Gráfica	Ver adaptador de vídeo.

Taxón	Conjunto de especificaciones que configuran un sistema jerárquico y que encuadran la condición que un ser mantiene dentro de su clasificación. Es la conjunción en las características de un objeto de aquellas propias de su herencia y las de sí mismo.
Tecnología de la Información	Conjunto de procesos y técnicas que, conjuntamente con la tecnología electrónica especializada han permitido la incorporación creciente del manejo electrónico de datos a los procesos productivos, incrementando la eficiencia, velocidad y productividad. En la práctica, la tecnología de la información se traduce en el análisis de requerimientos, selección del equipo adecuado y todo el proceso de desarrollo de sistemas que traen por resultado una herramienta software-hardware altamente especializada y que el usuario final del sistema podría usar.
Tiempo de Estado	En teoría de Cartas ASM, tiempo durante el cual la Máquina de Estados General permanece en un estado.
Tipo Abstracto de Datos (TAD)	Una descripción de una clase pero sin implementar detalle, así se provee la especificación. Es similar al concepto de tipos de datos definidos por el usuario.
Tipo de Dato Definido por el Usuario	Ver Tipo Abstracto de Datos.
TOO	(Léase "t doble o"). Tecnología de Orientación a Objetos. Conjunto de técnicas, métodos y conceptos que aplicados al desarrollo de sistemas da por resultado una aplicación orientada al objeto.
TOOBANII	Parte del Segmento de Nivel Superior de la herramienta elaborada para el desarrollo de la aplicación. Esta es una Tabla de Operaciones, OBjetos y Atributos de Nivel Superior.
TOOBANS	Parte del Segmento de Nivel Inmediato Inferior de la herramienta elabora para el desarrollo de la aplicación. Esta es una Tabla de Operaciones, OBjetos y Atributos de Nivel Inferior.
Trayectoria de Enlace	En teoría de Cartas ASM, es cada posible trayectoria de enlace de un estado a otro.

Valor Negado	En electrónica digital, se utiliza la convención de un valor ausente como un "0" lógico y un valor presente como un "1" lógico, a esto se le denomina lógica positiva, pero sin embargo, si cambiamos la asignación, es decir, ausencia como "1" y presencia como "0" se le denomina lógica negativa, es cuando decimos que una variable está en valor negado.
Variables de Estado	Son variables internas de control de la ASM, variables cuyos valores definen el código del estado de una Carta ASM.
Ventana	Marco rectangular en pantalla mediante el que se puede ver un documento, hoja de trabajo, base de datos u otra aplicación.
VGA	Del Inglés Video Graphics Array, Arreglo de Gráficos de Vídeo. Norma de visualización de gráficos a color en mapa de bits introducida por IBM en 1987. Los adaptadores VGA y los monitores analógicos muestran al mismo tiempo hasta 256 colores de variación continua, con una resolución de 640 pixeles en posición horizontal y 480 líneas en posición vertical. Es compatible con los modos: CGA, EGA en sus tres resoluciones
VGAMONO	Del Inglés Video Graphics Array Monocromo, Arreglo de Gráficos de Vídeo Monocromático. Norma de visualización de gráficos en mapa de bits. Los adaptadores VGAMONO muestran una resolución de 640 pixeles en posición horizontal y 350 líneas en posición vertical.
Vídeo Inverso	En monitores monocromáticos, medio de resaltar texto en pantalla para que los caracteres oscuros normales aparezcan brillantes sobre un fondo oscuro o para que los caracteres brillantes normales aparezcan oscuros sobre un fondo brillante.
WAN	Red de cómputo de cobertura amplia que usa redes de comunicación de larga distancia, alta velocidad o telecomunicaciones para conectar computadoras a distancias mayores -1.6 ó 3.2 Kms.- que las recorridas por las redes de área local.
XGA	Extended Graphics Array. Despliegue de vídeo estándar de IBM que intentó reemplazar su viejo estándar 8514/A para alcanzar una resolución de 1,024 por 768 para vídeos IBM y compatibles con IBM. Las tarjetas XGA equipadas con suficiente memoria (1 MB) pueden exhibir 65,536 colores en el modo de baja resolución (640 por 480) y 256 colores en el de alta (1,024 por 768).

APÉNDICE F

ARCHIVO DE ESTRUCTURA DE LA CARTA Y ARCHIVO DE SALIDAS

Definición de la estructura del archivo que describe a la Carta Asm.

NE	Valor entero que indica el número de elemento, éste es asignado de acuerdo a como se van creando en el sistema.
TE	Valor entero que indica el tipo de elemento, asignados como sigue: 0.- Para los estados. 1.- Para las salidas condicionales. 2.- Para los diamantes de decisión.
PX PY	Coordenadas (x,y) dentro del área de trabajo donde se ubica la esquina superior izquierda del elemento.
NOMBRE	Cadena de 4 cararacteres. Si el elemento es un estado indica el nombre del estado. Si el elemento es una salida condicional no se utiliza. Si el elemento es una decisión indica la variable de entrada asociada.
CÓDIGO	Cadena de cuatro caracteres. Se utiliza sólo si el elemento es un estado e indica el código asignado a éste.
SD	Es un apuntador que indica: Si el elemento es un estado o una salida condicional, el elemento al que está conectado el actual. Si el elemento es un diamante de decisión, el elemento al que está conectada la salida tomada en caso de que la variable de prueba sea cero.
SI	Es un apuntador utilizado solamente en los elementos diamante de decisión, indica el elemento al que está conectada la salida tomada en caso de que la variable de prueba sea uno.

Definición de la estructura del archivo que describe a las Salidas de la Carta Asm.

NombreVar	Nombre de la variable.
NoEdo	Ocho dígitos enteros que indican en que estados aparece o está asignada la variable. Una variable puede estar asignada hasta en ocho estados distintos.

APÉNDICE G

DEFINICIÓN DE CLASES

```
/*****************************************************************************
PROGRAMA:    DEFINE.I
COMENTARIO:          Codigo fuente Turbo C++ para la definicion de las clasesempleadas
                     en el sistema.
FECHA REVISION:  Enero/96
*****************************************************************************/

#ifndef DEFINE.I
#define DEFINE.I

#include <c:\tesis\programa\cabecera.h>
#include <c:\tesis\programa\grafinic.i>
/*****************************************************************************
Definicion de objetos y funciones
*****************************************************************************/
/*****************************************************************************
Definicion del objeto clase base Raton general
*****************************************************************************/
class Raton
{
    int Rver;           // Bandera de Visualizacion del cursor del raton
protected:
    Raton();            // constructor
    ~Raton();           // destructor
public:
    static raton_evento far* Reventos;  // Apuntador far global para registro
                                        // eventos del raton
    Rresultado *RRestaura();
    void RMuestra(int);
    Restado RPos();
    void RMueveA(int,int);
    Restado RPresionado(int);
    Restado RLiberado(int);
    void RxLimite(int,int);
    void RyLimite(int,int);
    Rmovimiento *RMueve();      // Movimiento neto del cursor
    void RMueve_Razon(int,int);
    void RAreaOculta(int,int,int,int);
    void RRapidez(int);
    void RLibera();         // Restaura el raton dejandolo en la posicion actual.
    void Pone_Cursor(g_cursor );
    void pone_cursor(int, int, unsigned, unsigned);
}; // class Raton

/*****************************************************************************
Definicion del objeto GRaton, hereda de Rston (para modo gráfico)
*****************************************************************************/
class GRaton : public Raton
{
private:        // figura de cursor grafico
    void pone_cursor(int,int,unsigned,unsigned);
public :
    void Pone_Cursor(g_cursor);
    void RPlumaOptica(int);
}; // class GRaton

/*****************************************************************************
Definicion del objeto TRaton, hereda de Raton
*****************************************************************************/
class TRaton : public Raton
{
public:             // figura de cursor texto
    void Pone_Cursor(int,unsigned,unsigned);
    void RPlumaOptica(int);
}; // class TRaton

/*****************************************************************************
Definicion de la Clase Unidad Grafica
*****************************************************************************/
class UnidadGrafica  // Esta clase se usa para inicializar la unidad grafica
{                    // como un objeto manejador del dispositivo grafico
int  TarjetaVideo,       // Tipo de la tarjeta de Video
     ModoGrafico,        // Valor del modo grafico
```

```
    MaxColores, // Maximos de colores disponibles
    CodigoError; // Reportes de cualquier error grafico
public:
    UnidadGrafica();            // Realiza la inicializacion de la unidad
    ~UnidadGrafica();           // Cierra la unidad grafica
}; // class UnidadGrafica

/*****************************************************************************
Definicion de la Clase Punto
*****************************************************************************/
class Punto
{
protected:
    int x,y,                    // (x,y) coordenadas del punto a dibujarse
    Color;                      // Color de visualizacion del punto a dibujarse
    viewporttype Pref;          // Salva las caracteristicas iniciales del puerto activo
public:
    Punto();
    virtual void Mueve(int,int);
    virtual void Dibuja();
    virtual void Crea(int,int,int);
    void RestauraPuerto();
    virtual void FijaColor(int);
    virtual void FijaLoc(int,int);
    virtual void Borra();
    int ObtenColor();
    int ObtenX();
    int ObtenY();
}; // class Punto

/*****************************************************************************
Definicion de la Clase Dispositivo (logicos)
*****************************************************************************/
class Dispositivo : public virtual Punto
{
protected:
```

```
/*****************************************************************************
**NOTA: Variables Heredadas de Punto.                                        **
**    (x,y): Vertice inicial del dispositivo.                                **
**    Color: Color del dispositivo.                                          **
**    PRef : Caracteristicas del puerto activo en el instante de creacion    **
*****************************************************************************/
    int  Edo,           // Estado de visualizacion del dispositivo Presionado/NoPresionado.
         Abierta,       // Indica si el dispositivo este o no dibujado en pantalla.
         TxtTam,        // Indica tamano de la fuente a usar.
         Ancho,         // Dimension en x del dispositivo.
         Alto,          // Dimension en y del dispositivo.
         Rotar;         // Indica si el dispositivo esta en posicion horizontal o vertical.
    void *AreaOrigen;   // Guarda el area original en el cual se dibujara el dispositivo.
public:
    Dispositivo();
    Dispositivo(int PtX,int PtY,int Anch,int Alt,int C);
    ~Dispositivo();
    virtual void Crea(int PtX,int PtY,int Anch,int Alt,int C);
    vertice Vertices();
    int Dentro(int PtX,int PtY);
    void *SalvaImagen();
    void DibujArea();
    int ObtenAncho();
    int ObtenAltura();
    int ObtenEstado();
    int ObtenAbierta();
    int ObtenTxtTam();
    int DispElegido();
    virtual void Borra();
    virtual void Mueve(int PtX,int PtY);
    virtual void Arrastra();
    virtual void FijaColor(int C);
    void AlturaTexto(int TxtTamano);
    void LiberAreaOrig();
    void Puerto();
}; // class Dispositivo
```

```
/*****************************************************************************
Definicion de la Clase Etiqueta
*****************************************************************************/
class Etiqueta : public Dispositivo
{
/*****************************************************************************
**NOTA:                                                                     **
**    (x,y): Posicion en la cual se escribira el texto.                     **
**    Color: Color de la fuente a escribir.                                 **
**    PRef : Caracteristicas del puerto activo en el instante de creacion.  **
**    Edo : Define el tipo de la fuente a usar.                             **
**    Abierta: Sin aplicacion.                                              **
**    TxtTam: Tamano de la fuente a usar.                                   **
**    Ancho : Longitud del texto.                                           **
**    Alto : Altura de la fuente usada.                                     **
**    Rotar : Sin aplicacion.                                               **
**    AreaOrigen: Guarda el area original en el cual se escribira el texto. **
*****************************************************************************/
    char texto[80];    // Almacena el texto de la etiqueta deseada.
public:
    Etiqueta();
    Etiqueta(int,int,char*);
    Etiqueta(int,int,int,int,int,char*);
    virtual void Cambia(int,int,char*);
    virtual void Crea(int,int,int,int,int,char *);
    virtual void FormaColor(int);
    void FuenteTamano(int);
    void Texto(char *);
    void EscribeTexto();
    void EscTexto();
    void EditaTexto(int NoCar = 59);
    char *ObtenTxt();
    void FijaColor(int);
    void FijaTamFte(int T); //Fija el Tamaño de la letra.
}; // class Etiqueta
```

```
/*****************************************************************************
Definicion de la Clase Boton (hereda de la clase Dispositivo)
*****************************************************************************/
class Boton : public Dispositivo
{
/*****************************************************************************
**NOTA:                                                                     **
**    (x,y): Posicion del vertice inicial del Boton.                        **
**    Color: Color del Boton a dibujarse                                    **
**    PRef : Caracteristicas del puerto activo en el instante de creacion   **
**    Edo : Estado de visualizacion del Boton Presionado/NoPresionado.      **
**    Abierta: Indica el Boton esta o no dibujado en pantalla.              **
**    TxtTam: Indica tamano de la fuente a usar.                            **
**    Ancho : Dimension en x del Boton                                      **
**    Alto : Dimension en y del Boton.                                      **
**    Rotar : Indica si el Boton es dibujado horizontal o verticalmente.    **
**    AreaOrigen: Almacena el area original en el cual se dibujara el Boton.**
*****************************************************************************/
    int TipoFte;          // Almacena el tipo de la fuente deseada.
    char BtnTxt[40];      // Almacena el mensaje del boton.
public:
    Boton();
    Boton(int PtX,int PtY,int Anch,int Alt,int C,char* Texto,int dib = 0,int borr = 0);
    ~Boton();
    virtual void Crea(int PtX,int PtY,int Anch,int Alt,int C,char* Texto,int dib = 0,int borr = 0);
    virtual void Mueve(int PtX,int PtY,int borr = 1);
    void Etiqueta(char* Texto,int borr = 1);
    virtual void FijaColor(int C,int borr = 1);
    void Estado(int BEdo,int borr = 1);
    void TipoFuente(int TxtFuente);
    void Invierte(int borr = 1);
    virtual void Dibuja(int borr = 1);
    int BotonElegido(int borr = 1);
}; // class Boton
```

```
/*****************************************************************
Definicion de la Clase BarrDesp (hereda de la clase Dispositivo)
*****************************************************************/
class BarrDesp : public Dispositivo
{
/*****************************************************************
**NOTA:                                                         **
**    (x,y): Vertice inicial de la Barra de Desplazamiento (BD). **
**    Color: Color de la Barra de Desplazamiento.                **
**    PRef : Caracteristicas del puerto activo en el instante de creacion. **
**    Edo : Color del contorno de la Barra de Desplazamiento (LineaColor). **
**    Abierta: Posicion del cursor de la Barra de Desplazamiento (BDPos). **
**    TxtTam : Tamano del paso de desplazamiento de la BD (Pasos **
**    Ancho: Dimension en x de la Barra de Desplazamiento.       **
**    Alto : Dimension en y de la Barra de Desplazamiento.       **
**    Rotar: Indica si la Barra de Desplazamiento es horizontal o vertical. **
**    AreaOrigen: Almacena la imagen del area donde se dibujara el cursor. **
*****************************************************************/
    int AntPos;         // Posicion anterior del cursor de la Barra de Desplazamiento.
    Boton BtnExt1, BtnCursor, BtnExt2; // Cursores extremos y central de la BD.
public:
    BarrDesp(int PtX,int PtY,int Tamano,int CFr,int CFn,int Dir);
    ~BarrDesp();
    virtual void Crea(int PtX,int Pty,int Tamano,int CFr,int CFn,int Dir);
    virtual void FijaLoc(int PtX,int PtY);
    PresTipo BDPresionada();
    int ObtenPosicion();
    int ObtenDireccion();
    int ObtenPorcentaje();
    void Restaura();
private:
    virtual void Dibuja();
    void Traza();
    void PonFlechas();
    void PonCursor();
}; // class BarrDesp
```

```
/*****************************************************************
Definicion de la Clase Ventana (hereda de la clase Dispositivo)
*****************************************************************/
class Ventana : public Dispositivo
{
/*****************************************************************
**NOTA:                                                         **
**    (x,y): Vertice inicial de la Ventana.                      **
**    Color: Color de la Ventana.                                **
**    PRef : Caracteristicas del puerto activo en el instante de creacion. **
**    Edo : Sin Aplicacion.                                      **
**    Abierta: Indica la Ventana esta o no dibujado en pantalla. **
**    TxtTam: Sin Aplicacion.                                    **
**    Ancho: Dimension en x de la Ventana.                       **
**    Alto : Dimension en y de la Ventana                        **
**    Rotar: Indica si la Ventana es horizontal o vertical       **
**    AreaOrigen: Guarda el area original en el cual se dibujara la Ventana. **
*****************************************************************/
public:
    Ventana();
    Ventana(int r1, int c1, int anch, int alt, int vcolor);
    ~Ventana();
    virtual void Dibuja(int borr = 0);
    void crea(int r1,int c1,int r2,int c2,int w,int b,int s);
    void Limpia(int borr = 0);
    void Cierra();
    void Informa(char *Msg,int espera,int borra,int color);
}; // class Ventana
```

```
/*****************************************************************
Definicion de la Clase MenuHorizontal (hereda de la clase Dispositivo)
*****************************************************************/
class MenuHorizontal : public Dispositivo
{
/*****************************************************************
**NOTA:                                                          **
**    (x,y): Vertice inicial del Menu Horizontal.               **
**    Color: Color del Menu Horizontal.                         **
**    Edo : Sin aplicacion.                                     **
**    Abierta: Indica si el Menu Horizontal esta o no dibujado en pantalla.  **
**    TxtTam:Color del texto del Menu                           **
**    Ancho: Dimension en x del Menu Horizontal                 **
**    Alto : Dimension en y del Menu Horizontal                 **
**    Rotar: Color de la Tecla Aceleradora                      **
**    AreaOrigen: Sin aplicacion.                               **
*****************************************************************/
    int NOpciones,              // # de opciones de la barra horizontal.
     Tabs,                      // Tabuladores de las opciones de la barra horizontal.
     OpcActiva,                 // Opcion latente sin desplegar su submenu.
     OpcElegida,                // Opcion activa con submenu desplegado.
     ColorLetra,                // Color de la letra aceleradora.
     OpcSubMenu;                // # de opcion del submenu.
    char *TeclasRapidas;        // Cadena de caracteres de las teclas aceleradoras.
    MENU_ENCABEZADO *OpcionesH;    // Contiene las opciones completas de la barra.
    Ventana VMenu,VSubMenu,        // Barra del Menu Horizontal y Ventana del SubMenu.
            VAyudaTxt,VAyudaMsg;   // Barra de mensages de ayuda.
    void (*Menu_Ayuda)(void);
    void EscribeOpcion(int PtX,int PtY,char* Opc,int Pos,int Clr);
    void TeclaRapida(int PtX,int PtY,char* Opc,int Clr);
    void CursorActivo(int,int,int,int,char*,int);
    void OpcSeleccionada(int,int);
    void DeseleccionaOpcion(int);
    void DesmarcaOpcion(int);
    void DespSubMenu();
    void NuevaOpcion(int,int);
public:
    MenuHorizontal(int,int,int,int,int,int,int,int,MENU_ENCABEZADO *,void(*ayuda)(void) = NULL);
    ~MenuHorizontal();
    void Despliega();
    virtual void Borra();
    void Selecciona();
}; // class MenuHorizontal

/*****************************************************************
Definicion de una Clase para manejar Cajas de Dialogo
*****************************************************************/
class Dialogo
{
public:
    Dialogo() { };
    void DialogoConfirmacion(int,int,int,int,int,char*);
    char *DialogoLee(int,int,int,int,int,char*,int LongCadena = 55);
    int DialogoOpciones(int Cx,int Cy ,int Ax,int Ay,int Color,char *Msg,int GpoOpc = 0);
    void DialogoImpresion(int *TabImp);
}; //class Dialogo

/*****************************************************************
Definicion de la Clase Elemento (hereda de la clase Dispositivo)
*****************************************************************/
class Elemento : public virtual Dispositivo
{
/*****************************************************************
**NOTA:                                                          **
**    (x,y): Vertice inicial del Elemento.                      **
**    Color: Color de contorno del Elemento.                    **
**    PRef : Caracteristicas del puerto activo en el instante de creacion.  **
**    Edo : Probablemente sea color de fondo!!!.                **
**    Abierta: Sin Aplicacion.                                  **
**    TxtTam: Tamano de la fuente a usar para calcular tamano de Elemento.  **
**    Ancho : Dimension en x del Elemento.                      **
**    Alto : Dimension en y del Elemento.                       **
```

```
**    Rotar: Sin Aplicacion.                                                    **
**    AreaOrigen: Guarda el area original en el cual se dibujara el Elemento.   **
*****************************************************************************/
protected:
    Etiqueta Nombre;// Nombre del Elemento en cuestion.
    UneManual *UnionDer,*UnionIzq;
public:
    Elemento();
    ~Elemento();
    void LlenaColor();
    virtual void FijaColor(int C) {Color = C;};
    void Modifica(int Tam = 1);
    int EditaNombre();
    char *ObtenNombre();
    UneManual *GeneraSegmento(Coord,int IzqDer = 0, int Inicio_Fin_Union=1); // IzqDer = 1 para la lista Izq.
    int BorraSegmento(int IzqDer = 0);      // Inicio_Fin_Union=1 -> primer segmento
    int BorraUnion(int IzqDer = 0);
    Coord DibujeSeg(Coord, Coord, int Dib = 1,int Inicio_Fin_Union=1);   // Dib = 0 para borrar el Seg.
                                                                         //Inicio_Fin_Union=1 -> primer
                                                                         //segmento
    void DibujeUnion(int IzqDer = 0, int Dib = 1);
    Coord InicioSeg(UneManual*);
    void DspUnVis(int Direccion, int Desplazamiento);
}; // class Elemento

/*****************************************************************************
Definicion de la Clase Estado (hereda de la clase Elemento)
*****************************************************************************/
class Estado : public virtual Elemento
{
/*****************************************************************************
**NOTA:                                                                      **
**    (x,y): Vertice inicial del Estado.                                     **
**    Color: Color de contorno del Estado.                                   **
**    PRef : Caracteristicas del puerto activo en el instante de creacion.   **
**    Edo : Probablemente sea color de fondo del Estado!!!.                  **
**    Abierta: Sin Aplicacion.                                               **
**    TxtTam: Tamano de la fuente a usar para calcular tamano del Estado.    **
**    Ancho : Dimension en x del Estado.                                     **
**    Alto : Dimension en y del Estado.                                      **
**    Rotar: Sin Aplicacion.                                                 **
**    AreaOrigen: Guarda el area original en la cual se dibujara al Estado.  **
**    Nombre : Nombre del Estado en cuestion.                                **
*****************************************************************************/
protected:
    Etiqueta Codigo;        // Codigo del Estado en cuestion.
    int Desp;               // Ubicacion del rectanguio del Estado
public:
    Estado();
    virtual void Cree(int Tam,char *Nombr,char *Codig,int PtX = 0,int PtY = 0,int CForme = WHITE,
            int CFondo = getbkcolor());
    virtual void Dibuje();
    int CodigoEntero();
    void EditaCodigo();
    char *ObtenCodigo();
    void NuevaPosNomCod();
    void CambiePos(Coord);
}; // class Estado

/*****************************************************************************
Definicion de la Clase Decision (hereda de la clase Elemento)
*****************************************************************************/
class Decision : public virtual Elemento
{
/*****************************************************************************
**NOTA:                                                                      **
**    (x,y): Vertice inicial de la Decision.                                 **
**    Color: Color de contorno de la Decision.                               **
**    PRef : Caracteristicas del puerto activo en el instante de creacion.   **
**    Edo : Probablemente sea color de fondo de la Desicion!!!.              **
**    Abierta: Sin Aplicacion.                                               **
**    TxtTam: Tamano de la fuente usada para calcular tamano de la Decision. **
```

```
**    Ancho : Dimension en x de la Decision.                                **
**    Alto : Dimension en y de la Decision.                                 **
**    Rotar : Sin Aplicacion.                                               **
**    AreaOrigen: Guarda area original en el cual se dibujara la Decision.  **
**    Nombre : Condicion Booleana de la Decision.                           **
*************************************************************************/
public:
    Decision();
    virtual void Dibuja();
    virtual void Crea(int Tam,char *Nombr,int PtX = 0,int PtY = 0,int CForma = WHITE,
                      int CFondo = getbkcolor());
    void EditaCondicion();
    char* ObtenCondBool();
    void NuevaPosCondBool();
    void CambiaPos(Coord);
    virtual void Arrastra();
}; // class Decision

/*************************************************************************
Definicion de la Clase SalCondicional (hereda de la clase Elemento)
*************************************************************************/
class SalCondicional : public virtual Elemento
{
/*************************************************************************
**NOTA:                                                                  **
**    (x,y): Vertice inicial de la Salida Condicional.                  **
**    Color: Color de contorno de la Salida Condicional.                **
**    PRef : Caracteristicas del puerto activo en el instante de creacion. **
**    Edo : Probablemente sea color de fondo de la Salida Condicional!!!. **
**    Abierta: Sin Aplicacion.                                          **
**    TxtTam: Tamano de la fuente a usar para calcular tamano de la SalCond. **
**    Ancho : Dimension en x de la Salida Condicional.                  **
**    Alto : Dimension en y de la Salida Condicional.                   **
**    Rotar : Bordeado de la Salida Condicional.                        **
**    AreaOrigen: Guarda area original en el cual se dibujara la Sal_Cond. **
**    Nombre : Sin Aplicacion.                                          **
*************************************************************************/
public:
    SalCondicional();
    virtual void Dibuja();
    virtual void Crea(int Tam,int PtX = 0,int PtY = 0,int CForma = WHITE,int CFondo = getbkcolor());
    void CambiaPos(Coord);
}; // class SalCondicional

/*************************************************************************
Definicion de la estructura para generar una lista de Estados, Decisiones y Salidas
Condicionales en CartasAsm
*************************************************************************/
struct Elem
{
    int Tipo;              // Tipo del Elemento: Estado/Salida Condicional/Decision.
    Estado *Est;           // Apuntador a un Estado en particular.
    SalCondicional *Sal;   // Apuntador a una Salida Condicional en particular.
    Decision * Dec;        // Apuntador a una Decicion en particular.
    Elem *ElemAnt,         // Apuntador a Elem anterior en la lista doblemente ligada.
         *ElemSig,         // Apuntador a Elem siguiente en la lista doblemente ligada.
         *ElemIzq,         // Apuntador a Elem izquierdo en la Carta ASM para la Decision
         *ElemDer;         // Apuntador a Elem derecho en la Carta ASM para la Decision.
}; // struct Elem

/*************************************************************************
Definicion de la Clase CartasAsm.
*************************************************************************/
class CartasAsm
{
    int  NoEdos,     // # de Estados totales de la Carta ASM.
         NoSal,      // # de Salidas Condicionales totales de la Carta ASM.
         NoDec,      // # de Decisiones totales de la Carta ASM.
         NoElem,     // # de Elementos totales de la Carta ASM. NOTA: Determinar si es necesaria.
         NoVarSal,   // # de Variables de Salidas de la Carta ASM. NOTA: Determinar si es necesaria.
         TamElem;    // Variable para dimensionar a los Elementos de la Carta ASM.
    Elem *ElemInicial,     // Apuntador al Elemento inicial de la Carta ASM.
```

```
        *ElemAsm,          // Apuntador al ultimo Elemento de la lista doblemente ligada.
        *ElemActual;       // Apuntador al ultimo Elemento activo de la Carta ASM.
    VarsEntSal Salidas[16];  // Tabla de Variables de Salida con sus Estados asociados.
    char NombreBase[9];
    void RestauraSalidas();
    void RestauraVariables();
    Estado *GeneraEdos(int PtX,int PtY,char *Nom = NULL,char *Cod = NULL);
    Decision *GeneraDecision(int PtX,int PtY,char *Nom = NULL);
    SalCondicional *GeneraSal(int PtX,int PtY);
    void EscribeSalidas(int,int,int);
    void ColocaSal(Elem*);
    void BorraVariable(int);
    void BorraSalEdo(int,int);
    void BorraElemento(Elem*);
    int VerificaUnion(Elem*,Elem*);
    void BorraUnion();
public:
    void EditaVar();
    void BorraE_S();
    void ColocaSalGral();
    CartasAsm();
    ~CartasAsm();
    void LibMemElem(Elem* Lib);
    void BorraAsm();
    virtual void Dibuja(int);
    Coord DaDimension();
    void PoneTamano(int Tam) {TamElem = Tam;};
    vertice EnlistaElementos(int Tipo,int CX = 0,int CY = 0,char *NOMBRE = NULL,
                             char *CODIGO = NULL);
    void NuevaPos(Elem*,Coord);
    Elem *ObtenElemActual() {return ElemActual;};
    Elem *ChecaElemActual();
    Elem *ChecaSeleccion(Coord Pos);
    Elem *ChecaSeleccion();
    vertice PosElem(Elem*);
    vertice ArrastraElemento(Elem*);
    void LiberaImagen(Elem*);
    void CapturaSalidas();
    void BorraSalida(int);
    int UnElementos(Elem*,Elem*);
    void CambiaPos(int,PresTipo);
    void LiberaImagenTotal();
    void GrabaArchivos(Coord OrigenAT);
    void LeeElementos();
    void GeneraArchASCII(Coord OAreaT,Coord Paso);
    void G_EDO_ASCII(FILE *Archivo,Coord CCasilla,char NombreSal[8][6]);
    Elem *ElementoInicial() {return ElemInicial;};
    int NoSalidas() {return NoVarSal;};
    VarsEntSal *TablaSal() {return Salidas;};
    void FijaNombre(char *NuevoNombre) {strcpy(NombreBase,NuevoNombre);};
    void LeeSecuencia();
    void ImprimeArch(Coord OAreaT,Coord Paso);
    void G_DEC_ASCII(FILE *Archivo,Coord PosEnArch)
    void CambiaCaracteristicas(Coord NuePos,int Tam);
    void CambiaDimPos(int,Coord);
    void CambiaElemActual(Elem* NuevoActual) {ElemActual=NuevoActual;};
    void LeeSalidas();
    void LeeSal();
    Elem *ElementoAnt(Elem*);
    Coord AsmXY(Elem*);
    int CartasAsm::Vacia() {if(!ElemInicial) return 1; return 0; };
    void FijaTamElem(int TamNuevo){ TamanoElem = TamNuevo; };
    int ObtenTamElem() { return(TamanoElem); };
    void ImprimeCarta(Coord OAreaT, Coord Paso);
    void G_SAL_ASCII(FILE *Archivo,Coord CCasillas); //Genera una salida condicional
                                                     //dentro de un archivo ascii
}; // class CartasAsm
```

```
/*************************************************************************
Definicion de la Clase Sintesis.
*************************************************************************/
class Sintesis
{
    int NoVarEnt;     // Numero de variables de Entrada (Condiciones Booleanas)
    char TablaProg[16][56];   // Table de almacenamiento del Micro Programa.
    VarsEntSal Entradas[16];  // Table de almacenamiento de las entradas de la Carta ASM.
public:
    Sintesis();
    int MicaValida(Elem *);
    void DefineOperaciones(Elem*, int, VarsEntSal*);
    void InstruccionesMical(int,int);
    void AccionModoCta(int,int);
    void Table_Ep_Ld(int, int);
    void MuestraEstados(int, int);
    void MuestraSalidas(int, int, int, VarsEntSal*);
    void MuestraEntradas(int, int);
    void MuestraOperaciones(int, int, int);
    void MuestraTabla(int, int, int);
    void MuestraHexa(int,int,int);
    void Diagrama(int,int,VarsEntSal*);
}; // class Sintesis

/*************************************************************************
Definicion de la Clase Area_trabajo (hereda de la clase Dispositivo)
*************************************************************************/
class Area_trabajo : public Dispositivo
{
/*************************************************************************
**NOTA:                                                                **
**    (x,y): Vertice inicial del Area de trabajo en pantalla.         **
**    Color: Color de la cuadricula del Area de Trabajo en pantalla.  **
**    PRef : Caracteristicas del puerto activo en el instante de creacion. **
**    Abierta: Sin Aplicacion.                                        **
**    TxtTam: Tamano de los elementos de la carta Asm, asi como de la cuadricula **
**          (Antes TamElementos).                                     **
**    Ancho : Dimension en x del Area de Trabajo.                     **
**    Alto : Dimension en y del Area de Trabajo.                      **
**    Rotar: Sin Aplicacion.                                          **
**    AreaOrigen: Sin aplicacion.                                     **
*************************************************************************/
// Estas constan de dos componentes: Long. de dx,dy de cada elemento de la cuadricula
    Coord   Origen,     // Coordenada de la casilla inicial de Area de Trabajo en pantalla.
            Margen,     // Sangria lateral del Elemento y su casilla de AT.
            OrigAT,     // Origen del Area de trabajo no con respecto a la pantalla.
            Paso,       // Dimension en x e y de las casillas de Area de Trabajo.
            PasElem;    // No se para que se usa. Tal vez este de mas.
    char NomCarta[9];   // Nombre del archivo o carta en edicion.
    CartasAsm Carta;    // instancia de CartasAsm

/*************************************************************************
*** Estructura dedicada al manejo de las posiciones de Area de trabajo
*** ocupadas por algun tipo de entidad
*************************************************************************/
    Lista *Ultimo;
    int AdicionALista(int x, int y, char Marca, char Salida);
    int BorraDeLista(int x, int y);
    char ObtenMarca(Coord *);
    char CambiaMarca(int x, int y, char Marca, char Salida);
    void CambiaListaXY(int x, int y, int x1, int y1);
    void CambiaTodaListaXY(int x1, int y1);
    void IniciaLista();
    void ConstruyeLista();

    void Dimensiona(int Tam);
    void PoneOrigen(int a,int b);
    Coord Coordxy(int i,int j);
    Coord Coordij(int x,int y);
    Coord LugarVacio();
    void IncOrigATX(int IncX) {OrigAT.x += IncX;};
    void IncOrigATY(int IncY) {OrigAT.y += IncY;};
```

```
    int RutaParcialOK(Coord Fte,Coord Dtn,Coord Fin,int Dir);
    int Direccion(Coord Fte, Coord Dtn, int Dir);
    Coord IncrementaPaso(Coord Fte, int Dir);
    Coord RutaParcial(Coord Fte, Coord Dtn, int Dir);
    char ObtenDir(char Direccion, char Suma);
    void FlechitasUnion(char Dir,Coord Pt,int Dib);
    int InvierteDireccion(int Dir);
public:
    Coord Ajusta(Coord puntoen,char Opcion);
    Coord Ubica(Coord puntoen);
    Coord AlVertice(Coord puntoen);
    Coord AlCentro(Coord puntoen);
    void Cuadricula(int bandera);
    void Zoom();
    Area_trabajo();
    void Agrega(int tipo);
    void Manipula();
    void CapturaVarSal();
    void RestauraCoord();
    void IncOrigenAT(PresTipo);
//========= Funciones miembro para el manejo de las Cartas ASM
    void Editar();
    void Borrar();
    void CartaCambiaPos(PresTipo Dir);
    Elem *VerificaSel();
    int PideArch();
    void LeeCarta();
    void EscribeCarta();
    void MuestraSintesis(int ConjOpc); // Manipula la presentacion de las tablas
    void GeneraElementos();
//========== funciones miembro para la union visual ================
    Coord Une(Coord * , char , char, Coord *, char Dib);
    Coord CreaUnion(Coord *, Coord *, char Dib);
    void DibujaUniones(char Dib);
    char FijaMarca(Coord *, char , char);
    char FijaMarca(Coord , char);
    char FijaSalida(Coord *, char Dir);
    void Imprimir();
    void Escala(int);
}; // class Area_trabajo

#endif
```

*** NOTA:
Para mayor detalle de la interface de las funciones consulta el código correspondiente a cada una de las funciones incluidos en el disco del sistema.