64

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

IMPLANTACION DE HERRAMIENTAS PARA EL ANALISIS ESTRUCTURADO

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N :

LIZBETH MARIN CASTELAN

MAJAHAIRA RAVELO PICHARDO

DIRECTOR DE TESIS: CRISTOBAL JUAREZ CASTELLANOS

MEXICO, D. F. 1996

Agradecemos a Dios por todas las bendiciones recibidas y por habernos permitido concluir una de nuestras principales metas.

A nuestras madres por el empeño que dedicaron para que tuvieramos una buena educación y el apoyo que nos brindaron para seguir adelante y cumplir nuestros objetivos.

A todas aquellas personas que nos ofrecieron su comprensión y apoyo en el transcurso de nuestros estudios.

A nuestros profesores, que nos mostraron la luz del conocimiento.

A nuestra alma mater la UNAM, que fue el escenario de nuestros mejores tiempos de estudiantes.

Índice

Capítulo 1

Introducción

1.1 Ingeniería de Software

1.1.1 Ciclo de Vida de un Sistema de Información

Por ciclo de vida de un sistema de información se entiende el conjunto de fases por los que pasa a lo largo del tiempo, desde la fase de estudio y concepción hasta la realización, explotación y mantenimiento. Para el desarrollo de sistemas existen diferentes metodologías, sin embargo podemos distinguir las siguientes etapas:

- **Estudio de Factibilidad**

  Antes de comenzar el desarrollo de un proyecto se realiza una toma inicial de datos y se define el marco de aplicación del sistema. Para llevar a cabo estas tareas, es necesario realizar una serie de entrevistas con directivos y usuarios a los que afectará el proyecto. Una vez recabada esta información será de utilidad para estimar los alcances del proyecto, la posibilidad de desarrollarlo y fijar los objetivos muy generales a cubrir con el desarrollo del mismo.

- **Análisis de Requerimientos**

  El análisis de requerimientos tiene como objetivo modelar el sistema a realizar de acuerdo a los requerimientos del usuario. Para cumplir este objetivo se cuentan con diversas herramientas de modelado de sistemas, donde cada una nos ayudará a representar el comportamiento de un sistema, enfocandose en algunos aspectos importantes de este y a su vez minimizando otros. Es necesario realizar un análisis de requerimientos eficaz, para hacer un buen diseño del sistema

- **Diseño**

  En esta fase se especificarán las estructuras de datos necesarias para satisfacer los requerimientos del sistema.

- **Implementación**

  Comprende la generación de código y el ensamblaje e integración de todos los módulos.

- **Pruebas**

  Se realizan pruebas con la totalidad del sistema hasta llegar a la aceptación del mismo por parte del usuario.

- **Implantación**

  En esta etapa, el sistema debe estar terminado y probado, y se le debe proporcionar al usuario capacitación del uso del sistema.

- **Mantenimiento**

  Una vez cubierta la etapa de implantación se generaran peticiones de mejoras o ampliaciones al sistema.

El ciclo de vida del sistema terminará cuado se abandone o se sustituya por otro, debido a cambios de estrategias en la empresa o el sistema se vuelva obsoleto.

1.1.2 Problemas en el Desarrollo de Software

En el desarrollo de software existen diversos parámetros para medir la calidad del sistema, algunos de los cuales son: eficacia, eficiencia y flexibilidad. Decimos que un sistema es eficaz cuando hace lo que debe de hacer, se concidera eficiente cuando el sistema haga lo que debe de hacer. en el menor tiempo y con el mínimo de recursos, finalmente un sistema es flexible cuando este se puede ampliar y modificar con el menor esfuerzo posible conservando la eficacia y la eficiencia.

El uso de metodologías de trabajo para normalizar procesos y etapas en el desarrollo de software mejoran en gran medida la calidad y reducen el tiempo de mantenimiento. Sin embargo el uso de metodologías en el desarrollo puede resultar bastante laborioso, si se realizan manualmente ciertas tareas y técnicas. Se mejora la calidad a cambio de un costo más elevado en el desarrollo, a causa de una dedicación mayor a labores de documentación y al cuidado de las técnicas.

Para solucionar este tipo de problemas en el desarrollo de software, se cuenta con herramientas computarizadas que soportan las diversas metodologías y sus técnicas, y facilitan la utilización de las mismas, de tal manera que disminuyen tanto el costo como el tiempo en el desarrollo.

1.1.3 Herramientas Computarizadas para el Desarrollo de Software

Existen herramientas para el desarrollo de software denominadas CASE (Computer-Aided Software Engineering), un CASE debe cubrir todas las etapas del desarrollo de sistemas, y debe llevar además, el control y la administración de proyectos. Algunas de las características de una herramienta CASE son las siguientes: manejo de gráficos para los diversos tipos de modelos que se aplican en las metodologías, utilizadas en cada una de las etapas para el desarrollo de software y verificación de consistencia internamente en cada modelo y a nivel general entre los diversos modelos.

Este tipo de herramientas, se consideran un soporte presente y futuro de las metodologías y técnicas de desarrollo y como base del ambiente de trabajo en el desarrollo de sistemas de información.

La utilización de herramientas CASE, trae como consecuencia que la automatización de la producción de software sea cada vez mayor. Por otro lado se tenderá a herramientas integradas, independientes del entorno de implementación y cuyo nivel de compatibilidad con diversas plataformas de hardware sea cada vez más alto.

El desarrollo de estas herramientas requiere una serie de recursos humanos y tecnológicos sumamente especializados, debido a que se trata de una tarea bastante compleja y de amplios alcances, implicando varios años/hombre de trabajo. Estas características hacen de este un producto de alto costo, el cual sólo puede ser adquirido por grandes compañías.

1.2 Panorama del Análisis Estructurado

Existen diversas metodologías para efectuar un análisis de sistemas, por ejemplo el análisis estructurado o el análisis orientado a objetos. A su vez en el análisis estructurado existen diversos enfoques, para este trabajo de tesis, utilizaremos el manejado por Edward Yourdon, en su libro Modern Structured Analysis (Análisis Estructurado Moderno).

El objetivo fundamental del análisis de requerimientos es transformar sus dos entradas principales, las políticas del usuario y el esquema del sistema a realizar, en una especificación estructurada. Para lograr dicho objetivo, en el análisis estructurado, se establece el desarrollo de un modelo del ambiente y de un modelo del comportamiento; los cuales constituyen el modelo de la esencia del sistema a desarrollar. El modelo de la esencia es independiente de la tecnología que se utilizará para la implantación.

El modelo del ambiente define la frontera entre el sistema y el resto del mundo. Este modelo consiste de: un diagrama de contexto, una lista de eventos y una descripción breve del propósito del sistema.

El modelo del comportamiento describe la forma en la que debe de funcionar el sistema para que interactue de manera exitosa con el ambiente. El modelo involucra la utilización de las siguientes herramientas de modelado:

- **Diagrama de Flujo de Datos**

  El diagrama de flujo de datos se utiliza para describir la transformación de entradas en salidas. Los diagramas de flujo de datos estan formados por cuatro componentes:

  - Proceso
  - Flujo
  - Almacén
  - Terminador

- **Diagrama Entidad Relación**

  El diagrama entidad relación presenta la relación que existe entre los datos, en un nivel alto de abstracción. El diagrama entidad relación consta de dos componentes principales:

  - Conjunto de Entidades
  - Interrelación

- **Diagrama de Transición de Estados**

  El diagrama de transición de estados se enfoca al comportamiento dependiente del tiempo del sistema. El diagrama de transición de estados representa los estados en los que se puede encontrar el sistema, los cambios de estado o transiciones de un estado a otro, las condiciones asociadas con cada cambio de estado y las acciones que se llevan a cabo como parte del cambio de estado.

- **Diccionario de Datos**

  El diccionario de datos es un listado organizado de todos los datos pertinentes al sistema, con definiciones precisas y rigurosas para que tanto el usuario como el

analista tengan un entendimiento común de todas las entradas, salidas, componentes de almacenes y cálculos intermedios.

- **Especificación de Procesos**

  La especificación de procesos es una descripción detallada de los procesos de más bajo nivel.

Las herramientas de modelado presentadas anteriormente se utilizan para:

- Concentrarse en ciertas propiedades y al mismo tiempo restar atención a otras.
- Discutir cambios y correcciones de los requerimientos del usuario a bajo costo y con el riesgo mínimo.
- Verificar que el analista comprenda correctamente el ambiente del usuario y que lo haya respaldado con información documental para que los diseñadores de sistemas y los programadores puedan construir el sistema.

1.3 Objetivo de la Tesis

El objetivo de este trabajo de tesis es desarrollar un sistema que maneje dos herramientas de modelado para el análisis estructurado:

- Diagrama de Contexto
- Diagrama de Flujo de Datos

Dado que la complejidad del desarrollo de estas herramientas, así como la cantidad de operaciones que se deban realizar rebasan el grado de complejidad esperado para un trabajo de tesis de licenciatura nuestro objetivo en particular es desarrollar un sistema que proporcione manejo gráfico y diferentes estrategias de desarrollo: ascendente, descendente y combinada que sirva como base para el desarrollo para un segundo trabajo de tesis que proporcione un sistema funcional para el usuario final.

1.4 Alcances de la Tesis

Proporcionar un sistema, que cuente con las operaciones necesarias para el desarrollo de un diagrama de flujo de datos y del diagrama de contexto. Además se realizará la verificación automática de consistencia, internamente en cada figura y a un nivel general entre las diversas figuras.

Para realizar un diagrama de flujo de datos existen tres estrategias de desarrollo, la primera consiste en iniciar el diagrama a partir del diagrama de contexto y posteriormente ir desarrollando niveles inferiores hasta llegar a los procesos más simples, la segunda es exactamente lo contrario, es decir iniciar desde los procesos más simples hasta llegar al

diagrama de contexto y la tercera consiste en combinar ambas estrategia. En este sistema se podrán utilizar cualquiera de estas estrategias.

Debido a que el análisis estructurado es un proceso iterativo, los diagramas se modifican constantemente, siendo incluso necesario retomar diagramas que habían sido desechados. Bajo estas consideraciones es necesario manejar diferentes versiones de los diagramas. En este trabajo se manejarán versiones para los diagramas de flujo de datos y para el diagrama de contexto.

En un proyecto además de estar incluidas todas las etapas del desarrollo de un sistema, se debe llevar el control y la administración del proyecto. En este sistema se manejarán proyectos pero de una manera muy sencilla, ya que sólo cubrirá dos herramientas del análisis de sistemas y no se desarrollorá ninguna herramienta para el control y la administración de proyectos.

Debido a que cada empresa tiene diferentes estándares de trabajo, para realizar la representación gráfica de los diagramas, el analista debe adecuarse a estos estándares y utilizar la simbología indicada. Aunque en este trabajo de tesis sólo se manejarán dos simbologías, el sistema se diseñará de tal manera, que se puedan agregar más simbologías en trabajos posteriores de manera sencilla..

1.5 Metodologías de Desarrollo del Sistema

El sistema constituye parte del desarrollo de un CASE. La metodología que se seguló para llevar a cabo este proyecto fué:

- Para el análisis de los requerimientos se utilizó el análisis estructurado.
- Para el diseño se utilizó el diseño orientado a objetos.
- Para la programación del sistema se utilizó la programación orientada a objetos siendo desarrollada en ambiente windows de PC's utilizando la herramienta de desarrollo de Visual C++.

1.6 Organización de la Tesis

La tesis se organizó en seis capítulos. En el capítulo 2 se exponen los conceptos fundamentales del análisis estructurado siguiendo la metodología de Edward Yourdon. En este capítulo se hace énfasis a las herramientas de modelado de diagrama de contexto y diagrama de flujo de datos, debido a que estas son las herramientas que se implementaron en el sistema.

El capítulo 3 describe las conceptos y las características principales del diseño orientado a objetos, así como las notaciones propuestas por Grady Booch para representar los modelos lógicos y físicos, que nos fueron de utilidad en el diseño del sistema .

La descripción general del sistema se presenta en el capitulo 4, planteando las características fundamentales de los tres módulos que conforman el sistema, siendo estos: el proyecto, el diagrama de flujo de datos y el diagrama de contexto. Además se proporciona la interface con el usuario, presentado las principales ventanas y explicando detalladamente como se pueden realizar ciertos procesos en el sistema.

En el capítulo 5 se muestran las estructuras de clases básicas utilizadas en cada uno de los módulos del sistema, las cuales fueron necesarias para satisfacer los requerimientos del problema y la estructura de clases proporcionada por el framework de Visual C++.

En el último capitulo, se realiza una evaluación de resultados, plasmando las estragias de diseño e implementación utilizadas, las ventajas y desventajas que se encontraron en el software de desarrollo utilizado, las ampliaciones que se le prodrian hacer al sistema y por último y más importe las conclusiones a las que se llegaron con la elaboración de este trabajo de tesis.

Capítulo 2

Análisis Estructurado

Introducción

Un modelo representa un sistema del mundo real. El análisis estructurado se basa en la creación de los modelos del sistema que se desea llevar acabo. Las razones por las que se construyen modelos son las siguientes:

- Para enfocar características importantes del sistema, a la vez que para minimizar las características menos importantes.
- Para discutir cambios y correcciones a los requerimientos del usuario, a bajo costo y con riesgo mínimo.
- Para verificar que se entiende el ambiente del usuario, y que se ha documentado de tal manera que los diseñadores y programadores puedan construir el sistema.

Existen muchos tipos diferentes de modelos que se pueden construir para el usuario como los modelos narrativos, modelos de prototipos, modelos gráficos diversos, etc.

2.1 Características de las Herramientas de Modelado

La mayoría de sistemas requieren de múltiples modelos: cada modelo se enfoca a un número limitado de aspectos del sistema, a la vez que minimiza (o ignora totalmente) otros de sus aspectos.

Una buena herramienta de modelado suele emplear una notación sencilla, con pocas reglas, símbolos y vocabulario nuevo que el usuario tenga que aprender.

Cualquier herramienta de modelado debe tener las siguientes características:

- Debe ser gráfica, con detalles textuales de apoyo.
- Debe permitir que el sistema sea visto en segmentos, en forma descendente.
- Debe tener redundancia mínima.
- Debe ayudar al lector a predecir el comportamiento del sistema.
- Debe ser transparente para el lector.

2.1.1 Modelos Gráficos

Una imagen bien realizada puede transmitir de manera concisa y compacta una gran cantidad de información. Lo que no significa necesariamente que una imagen pueda describir todo lo referente a un sistema; hacerlo causaría un desorden.

En general, se utilizan los gráficos para identificar los componentes de un sistema y su interfaz. Los demás detalles (que contestan a las preguntas ¿Cuántos?, ¿En qué orden?, etc.) se presentan en documentos textuales de apoyo.

Uno o más gráficos deben ser el documento primario al que se dirige el usuario para poder entender el sistema; los documentos textuales deben servir de material de referencia para consulta en caso necesario.

2.1.2 Modelos Segmentables en Forma Descendente

Un segundo aspecto importante de una herramienta de modelado es su capacidad de mostrar un sistema por partes en forma descendente. No es posible que una sola persona, sea usuario, analista o programador, se enfoque a todo el sistema al mismo tiempo. Tampoco es posible presentar un modelo gráfico de un sistema grande y complejo en una sola hoja de papel. Por lo que las herramientas deben permitirnos mostrar partes individuales del sistema de manera independiente, junto con una forma sencilla de moverse de una parte a otra del modelo del sistema.

Un modelo de un sistema complejo de información debe proceder de manera descendente. Una porción de la vista global de nivel alto del modelo debe dar al lector buena idea de los principales componentes de nivel alto y de las interfaces del sistema. Las siguientes porciones del modelo deben proporcionar información respecto a los componentes

detallados de nivel bajo. Y proporcionar un mecanismo conveniente para pasar sin problemas de un nivel alto a uno bajo.

2.1.3 Modelos Minimamente Redundantes

Los modelos representa sistemas reales, dichos sistemas pueden ser estáticos (no cambiantes) o dinámicos. Si el sistema cambia entonces debe cambiar el modelo, para mantenerse actualizado. Si sólo cambia un aspecto local del sistema. se desearía cambiar sólo un aspecto local correspondiente del modelo, sin tener por fuerza que cambiar algún otro y esto se lograría si la redundancia es mínima. De hecho, si se requieren múltiples cambios, existe una buena probabilidad de que no se hagan o de que se harán desordenadamente. Y esto significa que el modelo se volverá gradualmente menos preciso.

2.1.4 Modelos Transparentes

Un modelo debe ser tan fácil de leer que el lector no tenga que detenerse a pensar siquiera que se trata de la representación de un sistema y no del sistema mismo.

Los científicos han estudiado el comportamiento y la organización del cerebro humano y han encontrado que el hemisferio izquierdo realiza los procesos secuenciales. También se hace cargo de los textos. El hemisferio derecho trata con las imágenes y el procesamiento asincróno, donde todo sucede a la vez. Esto indica que si estamos tratando de modelar algo que es intrínsecamente lineal y secuencial, como el flujo de control en un programa de computadora, debemos usar una herramienta de modelado textual que quepa cómodamente en el hemisferio izquierdo. Y si estamos tratando de modelar algo que es intrínsecamente multidimensional, con muchas actividades que se dan a la vez, debemos usar una herramienta gráfica.

2.2. Modelo de la Esencia

El proceso de análisis de sistemas son los pasos que un analista lleva acabo cuando construye el modelo de un sistema. El proceso o metodología para construir un sistema involucra el desarrollo de diversos tipos de modelos, el último de los cuales será el producto del análisis de sistemas; es decir; el material para los responsables de diseñar la arquitectura general de las computadoras y su software, y finalmente, de escribir y probar los programas.

El enfoque clásico del análisis estructurado, argumenta que el analista debe desarrollar cuatro modelos distintos, el físico actual, el lógico actual, el lógico nuevo y el físico nuevo. Esta práctica involucra un gran desperdicio de tiempo debido a que el analista realiza un modelo demasiado detallado del sistema actual, pensando en él como un fin en si mismo. Por lo cual el análisis estructurado que Yourdon propone se enfoca en modelar el nuevo sistema lógico, denominado modelo de la esencia, evitando modelar el sistema actual de ser posible.

2.2.1 Definición del Modelo de la Esencia

El modelo de la esencia del sistema es un modelo de lo que el sistema debe hacer para satisfacer los requerimientos del usuario, diciendo lo mínimo posible (de preferencia nada) acerca de cómo se implantará.

2.2.2 Componentes del Modelo de la Esencia

El modelo de la esencia consiste en dos componentes principales:

- **Modelo del ambiente.** El cual define la frontera entre el sistema y el resto del mundo. (es decir, el ambiente en el cual existe el sistema).
- **Modelo del comportamiento.** Este modelo describe el comportamiento que del sistema se requiere para que interactúe de manera exitosa con el ambiente.

2.3 Modelo del Ambiente

El modelo del ambiente define las interfaces entre el sistema y el resto del Universo, determina las fronteras del sistema e identifica los acontecimientos que ocurren en el ambiente a los cuales debe responder.

2.3.1 Herramientas Usadas para Definir el Modelo del Ambiente

El modelo del ambiente consta de tres componentes:

1. Declaración de propósitos.
2. Diagrama de contexto.
3. Lista de acontecimientos.

2.3.2 La Declaración de Propósitos

Es una declaración textual breve concisa del propósito del sistema, dirigida al nivel administrativo superior, la administración de los usuarios y otras personas que no están directamente involucrados con el desarrollo del sistema.

La declaración de propósitos no debe llegar a ser mayor que un párrafo, ya que la intención no es proporcionar una descripción completa y detallada del sistema. El propósito del resto del modelo ambiente y del modelo de comportamiento es dar todos los detalles.

2.3.3 Diagrama de Contexto

El diagrama de contexto es un caso especial del diagrama de flujo de datos, en donde un sólo proceso representa todo el sistema.

El diagrama de contexto enfatiza varias características importantes del sistema:

- Los sistemas, organizaciones y personas con los que se comunica el sistema. Que se conocen como terminadores.
- Los datos que el sistema recibe del mundo exterior y que deben procesarse de alguna forma.
- Los datos que el sistema produce y que se envían al mundo exterior.
- Los almacenes de datos que el sistema comparte con los terminadores. Estos almacenes de datos se crean fuera del sistema para su uso, o bien son creados en él y usados fuera.
- La frontera entre el sistema y el resto del mundo.

2.3.4 Lista de Eventos

La lista de eventos es una lista narrativa de los "estímulos" que ocurren en el mundo exterior a los cuales el sistema debe responder.

Cada evento se etiqueta como F, T o C. Con ellos se muestra si es de tipo de flujo, temporal, o de control.

Un evento de flujos es el que se asocia con un flujo de datos; es decir, el sistema da cuenta de que ha ocurrido el acontecimiento cuando llega algún dato (o posiblemente varios).

Los eventos temporales arrancan con la llegada de un momento dado en el tiempo. No se inician con la entrada de flujos de datos; podría imaginarse que el sistema tiene un reloj interno con el cual puede determinar el paso del tiempo. Estos eventos podrían requerir que el sistema solicite entradas de uno o más terminadores. Por ello podrían asociarse uno o más flujos de datos con un evento temporal, aunque los flujos de datos, en sí, no representan el evento mismo.

Los eventos de control deben considerarse un caso especial del evento temporal: un estímulo externo que ocurre en algún momento impredecible. El evento de control no se asocia con el paso regular del tiempo, por lo que el sistema no puede anticiparlo utilizando un reloj interno. El evento de control se asocia con un flujo de control en el diagrama de contexto. El flujo de control puede considerarse como un flujo de datos binario: está encendido o apagado, y puede cambiar de un estado al otro en cualquier momento, señalando así al sistema que se necesita tomar alguna acción inmediata.

2.4 Modelo del Comportamiento

El modelo del comportamiento tiene como fin modelar el comportamiento interior del sistema, esto es como el sistema actuará ante los estímulos del exterior.

2.4.1 Herramientas para Definir el Modelo del Comportamiento

El modelo consta de las siguientes herramientas:

- Diagramas de Flujo de Datos.
- Diagramas Entidad-Relación.
- Diagramas de Transición de Estados.
- Diccionario de Datos.
- Especificación de Procesos.

2.4.2 Diagramas de Flujo de Datos

Un diagrama de flujo de datos es la herramienta de modelado que se utiliza para describir la transformación de entradas a salidas.

Los diagramas de flujo de datos se componen de procesos, almacenes de datos, flujos y terminadores.

- Los *procesos* representan las diversas funciones individuales que el sistema lleva a cabo, las cuales transforman entradas en salidas.

- Los *flujos* son las conexiones entre los procesos (funciones del sistema) y representan la información que dichos procesos requieren como entrada o la información que generan como salida.

- Los *almacenes de datos* muestran colecciones de datos que el sistema debe recordar por un periodo de tiempo.

- Los *terminadores* muestran las entidades externas con las que el sistema se comunica. Generalmente se trata de individuos o grupos de personas, sistemas de cómputo externos u organizaciones externas.

El diagrama de flujo de datos proporciona una visión global de los componentes funcionales del sistema, pero no da detalles de éstos. Para mostrar detalles acerca de qué información se transforma y de cómo se transforma, se ocupan dos herramientas textuales de modelado: el diccionario de datos y la especificación de procesos.

Por lo general los diagramas de flujo de datos son demasiado grandes por lo que es necesario realizarlos por niveles.

2.4.3 Diagramas Entidad-Relación

El diagrama entidad-relación es un modelo que describe con un nivel de abstracción alto la distribución de datos almacenados en un sistema. Esta herramienta es valiosa para un sistema con múltiples almacenes y relaciones complejas entre datos, ya que se enfoca totalmente a las relaciones entre datos, sin dar información acerca de las funciones que los crean o usan.

Los componentes de un Diagrama Entidad-Relación son:

- **Tipos de objetos**, que representan una colección o conjunto de objetos del mundo real cuyos miembros individuales (o instancias) tienen las siguientes características:
    - Cada una puede identificarse de manera única por algún medio.
    - Cada uno juega un papel necesario en el sistema que se construye, se puede decir que el sistema no puede operar sin tener acceso a esos miembros.
    - Cada uno puede describirse por uno o más datos. Los atributos deben aplicarse a cada instancia del tipo de objeto.

- **Relaciones**. Los objetos se conectan entre sí mediante relaciones. Una relación representa un conjunto de conexiones entre objetos, y se representa por medio de un rombo. Cada instancia de la relación representa una asociación entre cero o más ocurrencias de un objeto y cero o más ocurrencias del otro.

- **Indicadores asociativos** de tipo de objeto. Representa algo que funciona como objeto y como relación. Esto es una relación de la cual se desea mantener alguna información.

- **Indicadores de generalización/especialización**. Consisten en tipos de objetos de una o más subcategorías, conectados por una relación. La generalización se describe por datos que se aplican a todos las especializaciones. Cada especialización se describe por medio de datos diferentes.

2.4.4 Diagramas de Transición de Estados

Los diagramas de transición de estados son una herramienta poderosa de modelado para describir el comportamiento requerido de los sistemas de tiempo real.

Los principales componentes del diagrama de transición de estados son:

- **Estados**
  Los estados representan algún comportamiento del sistema que es observable y que perdura durante algún período finito.

- **Flechas**
  Las flecha las cuales indican la transición de estados.

- **Condiciones**
  Las condiciones son acontecimientos en el ambiente externo que el sistema es capaz de detectar y que causan el cambio de estado.

- **Acciones**
  Las acciones son respuestas del sistema al ambiente externo o bien cálculos cuyos resultados el sistema recuerda, para poder responder a algún acontecimiento futuro.

2.4.5 Diccionario de Datos

El diccionario de datos es una listado organizado de todos los datos pertinentes al sistema, con definiciones precisas y rigurosas para que, tanto el usuario como el analista tengan un entendimiento común de todas las entradas, salidas, componentes de almacenes y cálculos intermedios.

El diccionario de datos define los datos haciendo lo siguiente:

- Describe el significado de los flujos y almacenes que se muestran en los diagramas de flujo de datos.

- Describe la composición de paquetes de datos que se mueven a lo largo de flujo, paquetes complejos, que pueden descomponerse en unidades más elementales.

- Describen la composición de los paquetes de datos en los almacenes.

- Especifica los valores y unidades relevantes de piezas elementales de información en los flujos de datos y en los almacenes de datos.

- Describe los detalles de las relaciones entre almacenes que se enfatizan en un diagrama entidad-relación.

Construir un diccionario de datos es una de las labores más tediosas, y largas, del análisis de sistemas. Pero también es una de las más importantes: sin el diccionario formal que defina el significado de los términos, no se puede esperar precisión.

2.4.6 Especificación de Procesos

La especificación de procesos es la descripción de qué es lo que sucede en cada proceso de nivel más bajo en un diagrama de flujo de datos. Más no la descripción de como se lleva acabo dicho proceso.

Existen una variedad de herramientas que se pueden utilizar para producir especificaciones de procesos tales como tablas de decisiones, lenguaje estructurado, condiciones pre/post, diagramas de flujo, diagramas de Nassi/Shneiderman, etc.
La especificación de procesos debe satisfacer lo siguiente:

- Debe expresarse de una manera tal que lo puedan verificar tanto el usuario como el analista.
- El proceso debe especificarse en una forma que pueda ser comunicada a cualquier persona que esté involucrada con el sistema.
- No debe imponer o implicar decisiones de diseño e implementación arbitrarias.

Puede utilizarse una combinación de herramientas para realizar la especificación de procesos, según las preferencias del usuario, las preferencias del analista o la naturaleza propia de los diversos procesos.

2.5 Diagrama de Flujo de Datos

Es una de las herramientas más comúnmente usadas, sobre todo en sistemas en donde las funciones son de gran importancia y son más complejas que los datos que éste maneja.

Los diagramas de flujo de datos, se utilizaron por primera vez en la ingeniería de software como notación para el estudio del diseño de sistemas. A su vez, la notación se tomó prestada de artículos anteriores sobre teoría de gráficas, y continúa siendo utilizada por los ingenieros de software que trabajan en la implantación directa de modelos de los requerimientos del usuario.

Los diagramas de flujo de datos no sólo se pueden utilizar para modelar sistemas de cómputo de procesos de información, sino también para modelar organizaciones enteras, es decir, como una herramienta para la planeación estratégica y de negocios.

El diagrama de flujo de datos es tan sólo una de las herramientas de modelado disponibles y que únicamente proporciona un punto de vista de un sistema, el orientado a las funciones. Si se está desarrollando un sistema en donde las relaciones entre datos son más importantes que las funciones, tal vez se dé menos importancia al diagrama de flujo de datos (o incluso ni nos molestemos en elaborarlo), para concentrarse más bien en desarrollar un conjunto de diagramas entidad-relación. De otra manera si el comportamiento dependiente del tiempo de un sistema domina sobre cualquier otro factor, tal vez nos concentremos más en el diagrama de transición de estados.

Los diagrama de flujo de datos prácticamente no requieren explicación y su notación es sencilla y clara.

2.5.1 Componentes de un Diagrama de Flujo de Datos

Los elementos que constituyen un diagrama de flujo de datos son los siguientes:

- Procesos.
- Flujos.
- Almacenes.
- Terminadores.

2.5.1.1 Procesos

El primer componente del diagrama de flujo de datos se conoce como proceso. Los sinónimos comunes son burbuja, función, o transformación. El proceso muestra una parte del sistema que transforma entradas en salidas; es decir, muestra cómo es que una o más entradas se transforman en salidas. El proceso se representa gráficamente como circulo, como se muestra en la figura 2.1 (a) , aunque algunos analistas prefieren usar un rectángulo (figura 2.1 (b)) o un rectángulo con esquinas redondeadas, como se muestra en la figura 2.1 (c)

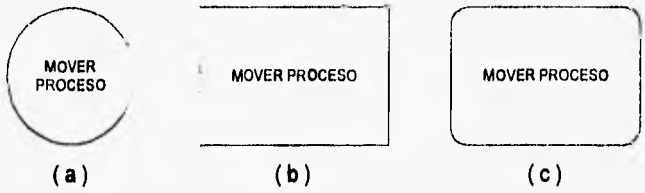


Figura 2.1 Representaciones de Procesos

Las diferencias entre estas tres formas son puramente estéticas, aunque obviamente es importante usar la misma forma de manera consistente para representar todas las funciones de un sistema.

2.5.1.2 Flujos

El flujo se representa gráficamente por medio de una flecha que entra o sale de un proceso. Se usa para describir el movimiento de bloques o paquetes de información de una parte del sistema a otra. Por ello, los flujos representan datos en movimiento.

Los flujos tienen nombre que representa el significado del paquete que se mueve a lo largo del flujo. El flujo sólo lleva un tipo de paquete de datos, como lo indica su nombre.

Los flujos muestran la dirección: una cabeza de flecha en cualquier extremo (o posiblemente ambos) del flujo indica si los datos se están moviendo hacia adentro o hacia afuera de un proceso (o ambas cosas).

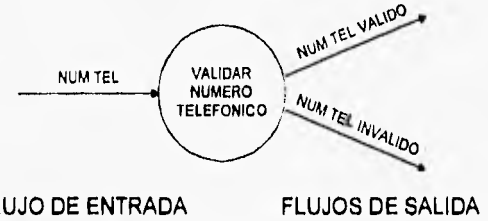


Fig. 2.2 Ejemplo de Flujos de Entrada y de Salida

Los flujos con dos cabezas de flecha se denominan diálogos, es decir, una pregunta y una respuesta en el mismo flujo. En el caso de un diálogo, los paquetes en cada extremo de la flecha deben nombrarse, ver el ejemplo en la figura 2.3

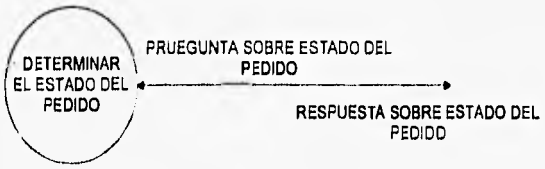


Fig. 2.3 Ejemplo de un Diálogo

Los flujos de datos pueden divergir o converger en un diagrama de flujo de datos. En el caso de los flujos divergentes, estos pueden representar que copias de los mismos datos son enviados a diferentes partes del sistema, ver figura 2.4, o que un conjunto complejo de datos se está dividiendo en varios paquetes individuales, cada uno de los cuales se está mandando a diferentes partes del sistema, como se muestra en la figura 2.5.

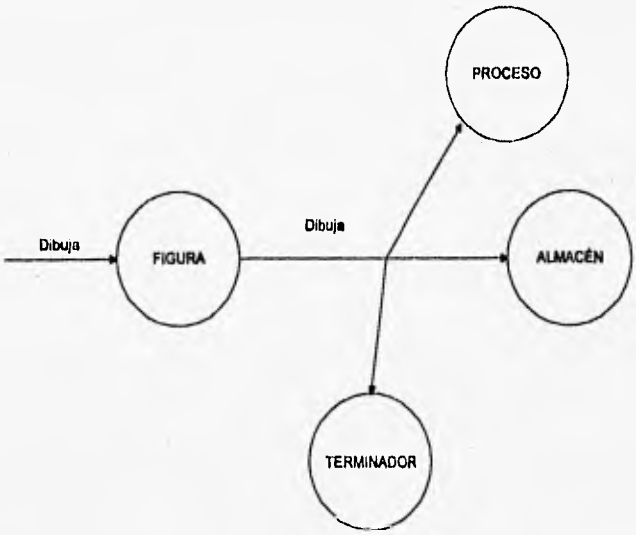


Figura 2.4 Flujo Divergente

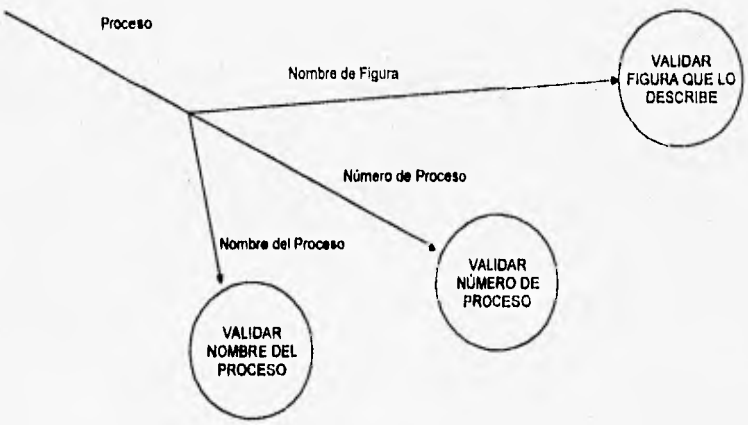
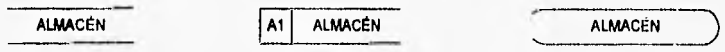


Figura 2.5 Otro Ejemplo de Flujo Divergente

Los flujos convergentes, representan datos que se integran para formar un paquete de datos más complejo.

2.5.1.3 Almacenes

El almacén se utiliza para modelar una colección de paquetes de datos en reposo. Se denota por dos líneas paralelas, como lo muestra la figura 2.6 (a) ; una alternativa de notación se muestran en la figura 2.6 (b) y (c).



(a) (b) (c)

Figura 2.6 Representaciones Gráficas de un Almacén

De modo característico el nombre que se utiliza para identificar al almacén es el plural del que utiliza para los paquetes que entran y salen del almacén por medio de flujos.

Los almacenes se conectan por flujos a los procesos. Por lo que tenemos:

- Un flujo desde un almacén. Normalmente se interpreta un flujo que procede de un almacén como una lectura o un acceso a la información de éste. Lo que significa que:
  - Se recupera del almacén un solo paquete de datos.
  - Se ha recuperado más de un paquete del almacén.
  - Se tiene una porción de un paquete del almacén.
  - Se tienen porciones de más de un paquete del almacén.

  Si el flujo no está etiquetado, significa que todo el paquete de información se está recuperando, si la etiqueta del flujo es la misma que la del almacén significa que se recupera todo un paquete (o múltiples instancias de uno completo), si la etiqueta del flujo es diferente del nombre del almacén, entonces se están recuperando uno o más componentes de uno o más paquetes.

  El almacén no cambia cuando un paquete se mueve del almacén a lo largo del flujo. Es una lectura no destructiva o , en otras palabras, del almacén se recupera una copia del paquete y el almacén mantiene su condición original.

- Un flujo hacia un almacén. Habitualmente se describe como una escritura, una actualización o posiblemente una eliminación. Específicamente, sólo puede significar que se tiene una de las situaciones siguientes:

  - Se están guardando uno o más paquetes nuevos en el almacén.
  - Uno o más paquetes se están borrando o retirando del almacén.
  - Uno o más paquetes se están modificando o cambiando.

  En todos estos casos es evidente que el almacén cambió como resultado del flujo que ingresa. El proceso (o procesos) conectado con el otro extremo del flujo es el responsable de realizar el cambio al almacén.

  Los flujos conectados a un almacén sólo pueden transportar paquetes de información que el almacén sea capaz de guardar.

2.5.1.4 Terminadores

El siguiente componente del diagrama de flujo de datos es el terminador; gráficamente se representa como un rectángulo, como se muestra en la figura 2.7. Los terminadores representan entidades externas con las cuales el sistema se comunica. Comúnmente, un terminador es una persona o grupo, que están fuera del control del sistema que se está modelando. En algunos casos, un terminador puede ser otro sistema computacional.

Figura 2.7 Representación Gráfica de un Terminador

Existen tres cosas que se deben tomar en cuenta de los terminadores:

1. Son externos al sistema que se está modelando; los flujos que conectan los terminadores a diversos procesos (o almacenes) en el sistema representan la interfaz entre él y el mundo externo.

2. El terminador está fuera del dominio del cambio. El analista de sistemas no puede modificar los contenidos, la organización ni los procedimientos internos asociados con los terminadores.

3. Las relaciones que existan entre los terminadores no se muestran en el modelo de diagrama de flujo de datos, ya que nos son parte del sistema que se está estudiando.

En un sistema real pueden existir docenas de terminadores diferentes interactuando con él. Identificar los terminadores y su interacción con el sistema es parte del proceso de construir el modelo del ambiente (ver sección 2.3).

2.5.2 Guía para la Construcción del Diagrama de Flujo de Datos

Existen reglas que nos ayudan a no elaborar diagrama de flujo de datos erróneos (por ejemplo, incompletos o lógicamente inconsistentes). Algunas de las reglas tienen la finalidad de ayudar a dibujar un diagrama de flujo de datos grato a la vista.

Las reglas son las siguientes:

1. Escoger nombres con significado para los procesos, flujos, almacenes y terminadores.
2. Numerar los procesos.
3. Redibujar el diagrama de flujo de datos tantas veces como sea necesario estéticamente.
4. Evitar los diagrama de flujo de datos excesivamente complejos.
5. Asegurarse de que el diagrama de flujo de datos sea internamente consistente y que también lo sea con cualesquiera diagrama de flujo de datos relacionados con él.

2.5.2.1 Escoger Nombres con Significado para los Procesos, Flujos, Almacenes y Terminadores

Como ya se vio, un proceso en un diagrama de flujo de datos puede representar una función que se está llevando a cabo, o pudiera indicar cómo se está llevando a cabo, identificando a la persona, grupo o mecanismo involucrado. En el último caso, es importante etiquetar con precisión el proceso, de modo que quienes leen el diagrama de flujo de datos, especialmente los usuarios, puedan confirmar que se trata de un modelo preciso. Sin embargo, si el proceso lo hace una persona, se recomienda que identifique el papel que dicha persona está representando, más que su nombre o identidad.

Para etiquetar los procesos de tal manera que se puedan identificar las funciones que el sistema está llevando a cabo hay que evitar nombrarlos con los nombres de personas o grupos.

Un buen sistema que se puede utilizar para nombrar procesos es usar un verbo y un objeto. Es decir, elegir un verbo activo y un objeto apropiado para formar una frase descriptiva para el proceso. Eliminar nombres ambiguos como HACER, MANEJAR y PROCESAR.

Los nombres de los procesos (al igual que los nombres de flujos y de terminadores) deben provenir de un vocabulario que tenga algún significado para el usuario. Esto sucederá de manera muy natural si el diagrama de flujo de datos se dibuja como resultado de una serie de entrevistas con los usuarios y si el analista tiene algún entendimiento mínimo de la materia de aplicación subyacente.

2.6.2.2 Numerar los Procesos

Como una forma conveniente de referirse a los proceso en un diagrama de flujo de datos se enumeran. No importa mucho cómo se haga esto, de izquierda a derecha, de arriba abajo o cualquier otra manera servirá, mientras haya constancia en la forma de aplicar los números.

La numeración no significa que exista una secuencia de ejecución de los proceso. El modelo de diagrama de flujo de datos es una red de procesos asincronos que se intercomunica, lo cual es una representación de la manera en la que en realidad muchos sistemas operan.

Los números se convierten en base para numeración jerárquica cuando se introduzcan los diagramas de flujo por niveles.

2.6.2.3 Evitar los Diagramas de Flujo de Datos Demasiado Complejos

El propósito de un diagrama de flujo de datos es modelar de manera precisa las funciones que debe llevar a cabo un sistema y las interacciones entre ellas. Pero otro propósito del diagrama de flujo de datos es ser leído y comprendido, no sólo por el analista que construyó el modelo. sino por los usuarios que sean los expertos en la materia de aplicación. Lo que significa que el diagrama debe ser fácilmente entendido, fácilmente asimilado y placentero a la vista.

Una regla importante para realizar el diagrama de flujo de datos es no llenarlo con demasiados procesos, flujos, almacenes y terminadores. En la mayoría de los casos, esto significa que no debe haber más de media docena de procesos y almacenes, flujos y terminadores relacionados en un solo diagrama. Otra manera de decir esto es que el diagrama de flujo de datos debe caber cómodamente en una hoja normal.

Con excepción del diagrama de contexto, que representa el sistema entero como un solo proceso y destaca las interfaces entre el sistema y los terminadores externos. Este no se pueden simplificar, pues describen con el más alto detalle, una realidad que es intrínsecamente compleja.

2.6.2.4 Volver a Dibujar el Diagrama de Flujo de Datos

En un proyecto real de análisis de sistemas, el diagrama de flujo de datos tendrá que dibujarse y volverse a dibujar antes de ser técnicamente correcto, ser aceptable para el usuario y estar bien dibujado para ser mostrado a la organización. Esto resulta mucho

trabajo, por lo cual se hace necesario usar una herramienta computacional que nos permita realizar cambios de manera rápida y sencilla.

2.6.2.6 Diagrama de Flujo de Datos Lógicamente Consistentes

Existen reglas que ayudan a asegurar la consistencia del diagrama de flujo de datos con los otros modelos del sistema: el diagrama entidad-relación, el diagrama de transición de estados, el diccionario de datos, y la especificación de procesos. Sin embargo, existen reglas para asegurar la consistencia de un diagrama de flujo de datos. Tales son:

- Sumideros infinitos. estos son procesos que tienen entradas pero no salidas.
- Procesos de generación espontánea, que tienen salidas sin tener entradas, generalmente incorrectos. Un ejemplo factible es un generador de números aleatorios.
- Flujos y procesos no etiquetados. Un flujo no etiquetado podría significar que diversos datos elementales no relacionados se agruparon arbitrariamente.
- Almacenes de "sólo lectura" o "sólo escritura". Esta regla es análoga a la de los procesos de "únicamente entradas" o "únicamente salidas". Un almacén típico debiera tener tanto entradas como salidas. La única excepción a esta regla es el almacén externo, que sirve de interfaz entre el sistema y algún terminador externo.

2.6.3 Diagrama de Flujo de Datos por Niveles

Para evitar un diagrama de flujo de datos complejo, se organiza en una serie de niveles de modo que cada uno proporcione sucesivamente más detalles sobre una porción del nivel anterior.

El diagrama de flujo de datos del primer nivel se conoce como diagrama de contexto y consta sólo de un proceso, que representa el sistema completo; los flujos de datos muestran las interfaces entre el sistema y los terminadores externos (junto con los almacenes externos que pudiera haber).

El diagrama de flujo de datos que sigue del diagrama de contexto se conoce como la figura 0. Representa la vista de más alto nivel de las principales funciones del sistema, al igual que sus principales interfaces.

Los números de cada proceso sirven como una manera adecuada de relacionar un proceso con el siguiente nivel del diagrama de flujo de datos que la describe más a fondo.

Si el proceso tiene nombre (que debe tenerlo) entonces dicho nombre se vuelve a utilizar para nombrar a la figura de nivel inmediato inferior.

Para asegurar que cada figura sea consistente con su figura de nivel más alto se sigue la siguiente regla: los flujos de datos que salen y entran de un proceso en un nivel dado deben corresponder con los que entran y salen de toda la figura en el nivel inmediato inferior que la describe.

Para mostrar los almacenes en diversos niveles la regla es la siguiente: mostrar un almacén en el nivel más alto donde primeramente sirve de interfaz entre dos o más procesos; luego, mostrarlo de nuevo en cada diagrama de nivel inferior que describa más a fondo dicho proceso. Los almacenes locales, que utilizan sólo procesos en una figura de nivel menor, no se mostrarán en niveles superiores, dado que se incluyen de manera implícita en un proceso del nivel inmediato superior.

Para desarrollar el diagrama de flujo de datos puede comenzarse de forma descendente, lo cual es, iniciar con el diagrama de contexto, después con la figura 0, y así sucesivamente, pero también podría comenzarse de manera ascendente, por los diagramas de nivel más bajo y así hasta el diagrama de contexto.

2.5.4 Extensiones del Diagrama de Flujo de Datos para Sistemas de Tiempo Real

Los diagrama de tiempo real están formados por flujos de control (señales o interrupciones) y procesos de control (procesos cuya única labor es coordinar y sincronizar las actividades de otros procesos).

Un flujo de control porta una señal binaria (de encendido o apagado).Los flujos de control que salen del proceso de control se utilizan para activar procesos normales los cuales llevan acabo la labor que se describe en su especificación de procesos.

Los flujos de control que entran al proceso de control generalmente indican que uno de los procesos ha terminado su labor o que se ha presentado alguna situación extraordinaria, de la cual necesita informarse al proceso de control.

Un proceso de control coordina las actividades de otros procesos en el diagrama; sus entradas y salida consisten sólo de flujos de control. Por lo común sólo hay un proceso de control en un diagrama de flujo de datos. El interior del proceso de control se modela con un diagrama de transición de estados, que muestra los diferentes estados en los que se puede encontrar todo el sistema y las circunstancia que lo llevan a cambiar de estado.

2.6 Diagramas de Contexto

El diagrama de contexto es un caso especial de diagrama de flujo de datos, en el cual se representa el sistema entero como un solo proceso, cuyo nombre dentro suele ser el nombre del sistema completo o un acrónimo convenido, y destaca las interfaces entre el sistema y las entidades externas. Este diagrama no se puede simplificar, pues describe, al más alto nivel una realidad que es intrínsecamente compleja.

Los componentes de este diagrama son los mismo que para un diagrama de flujo de datos, y en el caso de existir almacenes, estos son externos al sistema y sirven como interface entre el sistema y uno o más terminadores.

Los flujos que aparecen en el diagrama de contexto modelan datos que entran y salen del sistema, además de las señales de control que recibe o genera las cuales representan la interface del sistema con el medio ambiente.

Capítulo 3

Diseño Orientado a Objetos

3.1 Objetos y Clases

3.1.1 Objeto

Un objeto representa un individuo, elemento identificable, unidad o entidad, real o abstracta con un rol bien definido en el dominio del problema. Un objeto tiene estado, comportamiento e identidad; la estructura y el comportamiento de objetos similares son definidos en su clase común; los términos instancia y objeto son intercambiables.

Estado

El estado de un objeto abarca todas las propiedades del objeto (usualmente estáticas) más los valores actuales (usualmente dinámicos) de cada una de estas propiedades.

Comportamiento

El comportamiento es como un objeto acciona y reacciona, en términos de sus cambios de estado y paso de mensajes. El comportamiento de un objeto es completamente definido por sus acciones. Las operaciones que los clientes pueden ejecutar sobre un objeto son comunmente declaradas como métodos. Los clientes normalmente ejecutan cinco clases de operaciones sobre un objeto:

- **Modificador** Una operación que altera el estado de un objeto; es una operación de escritura o acceso.
- **Selección altera** Una operación que accesa al estado de un objeto, pero no el estado, es una operación de lectura.
- **Iterador** Una operación que permite que todas las partes de un objeto sean accesadas en algún orden bien definido.
- **Constructor** Una operación que crea un objeto y/o inicializa sus estado.
- **Destructor** Una operación que libera el estado de un objeto y/o destruye al objeto.

Identidad

Identidad es aquella propiedad de un objeto por la cual se distingue de otros objetos. Un objeto tiene un identificador para poder accesarlo

3.1.1.1. Interrelación entre objetos

La interrelación entre dos objetos asume que hace cada uno respecto a otro, incluyendo que operaciones pueden ser ejecutadas y que funcionamiento resulta.
Hay dos clases de interrelación de objetos de interés para el diseño orientado a objetos:

- **Interrelación de Uso**

  Decimos que existe interrelación de uso cuando un objeto A usa un objeto B, y donde el primer objeto puede enviar mensajes al segundo. El paso de mensajes entre dos objetos es normalmente unidireccional, sin embargo, la comunicación bidireccional es posible.
  En la interrelación de uso, cada objeto puede jugar uno de los siguientes roles:

  - **Actor** Es un objeto que puede operar sobre otros objetos pero otros objetos nunca operan sobre él.
  - **Servidor** Es un objeto que nunca opera sobre otros objetos: éste sólo lo pueden operar otros objetos.
  - **Agente** Es un objeto que puede operar sobre otros objetos y lo pueden operar otros objetos.

- **Interrelaciones de Contención.**

  La interrelación de contención es cuando un objeto está forrmado por otros objetos.

3.1.2 Clases

Un objeto es una entidad concreta que existe en el tiempo y el espacio, una clase representa sólo una abstracción, la esencia de un objeto. Una clase es un conjunto de objetos que

comparten una estructura común y un comportamiento común. Un objeto es simplemente una instancia de una clase. Un objeto no es una clase, pero una clase puede ser un objeto.

3.1.2.1 Interrelación entre Clases

Tenemos tres tipos de interrelación de clases:

- **Generalización** Denota una interrelación de "tipo de".
- **Agregación** Denota una interrelación de "parte de".
- **Asociación** Denota alguna conexión semántica entre clases no relacionadas.

Las combinaciones de interrelaciones entre clases soportadas por los lenguajes orientados a objetos son:

Interrelación de Herencia

La herencia es quizá la más poderosa de estas interrelaciones y puede ser usada para expresar generalización y asociación. Es una interrelación entre clases donde una clase comparte la estructura y funcionamiento definidos en otra clase (herencia simple) o en otras clases (herencia múltiple). La clase desde la cual otras clases heredan es una superclase, y las que heredan son subclases.

Polimorfismo

Polimorfismo es un concepto en teoría de tipos en el cual un nombre puede denotar objetos de algunas clases diferentes que están relacionadas por alguna superclase en común. Así algún objeto denotado por este nombre puede responder algún conjunto común de operaciones en forma diferente. También se podría decir que polimorfismo es tener métodos con el mismo nombre que ejecutan diferentes funciones y varían por el número de argumentos y tipo. El polimorfismo es útil cuando hay clases con los mismos protocolos. Es posible tener la herencia sin polimorfismo, pero ciertamente no es muy útil.

Interrelaciones de Uso

La interrelación de uso indica que la clase cliente depende de otras clases para proporcionar ciertos servicios. Normalmente, una interrelación de uso indica que las operaciones de la clase cliente invocan operaciones de otra clase.

En las interrelaciones de uso, podemos expresar la cardinalidad de las interrelaciones de la siguiente manera:

- 1:1 (de uno a uno) Esto es cuando una clase utiliza una instancia de otra clase.
- 1:n (uno a n) Cuando una clase utiliza más de una instancia de otra clase.
- m:n (de m a n). Cuando varias instancias de una clase utilizan varias instancias de otra clase.

Interrelación de Contención

Una interrelación de contención muestra una interrelación entre dos clases, donde una clase contiene una instancia de otra clase. Esta interrelación es denominada una interrelación de **agregación**.

La interrelación de contención puede ser de dos tipos:

- **Por Valor** Indica que la clase cliente tiene una instancia de otra clase.
- **Por Referencia** Indica que la clase cliente tiene una apuntador o una referencia a una instancia de otra clase.

Interrelaciones de Instanciación

Las interrelaciones de instanciación soportan la generalización y la asociación pero de manera diferente. La relación de instanciación es definir clases con instancias del mismo tipo. Idealmente nos gustaría definir un conjunto de clase de tal manera que podamos presentar la clase de objetos permitidos en el conjunto y entonces permitir las reglas de tipificación de nuestro lenguaje prohibiéndonos hacer esto de otra manera.

Un conjunto es una clase contenedora, la cual es una clase cuyas instancias son colecciones de otros objetos. Una clase contenedora puede denotar colecciones homogéneas, donde los objetos son de la misma clase o colecciones heterogéneas, en la cual los objetos pueden ser de diferentes clases, sin embargo, estas deben ser parte de alguna superclase común.

Hay básicamente cuatro formas de construir clases contenedoras:

- La primera es usando macros.
- En la segunda se utiliza la herencia y el ligado dinámico. Con esta aproximación quizá sólo construyamos clases contenedoras heterogéneas.
- La tercera podemos construir clases contenedoras generalizadas pero entonces utiliza código de verificación de tipo explícito para hacer cumplir la convención que todo los contenidos sean de la misma clase, lo cual es afirmado cuando el objeto contenedor se crea.
- La cuarta aproximación provee un mecanismo para clases parametrizadas. Una clase parametrizada (también conocida como una clase genérica) es una que sirve como un patrón (template) para otras clases, un patrón que puede ser parametrizado por otras clases, objetos y/o operaciones. Una clase parametrizada debe ser instanciada antes que los objetos se puedan crear. Desde la perspectiva de diseño, las clases parametrizadas son útiles para capturar ciertas decisiones de diseño acerca de la interface de la clase.

Mientras que una definición de clase, puede nombrar las operaciones que uno puede ejecutar sobre las instancias de esa clase, la parte genérica de un patrón se puede usar para declarar las operaciones que las instancias de esa clase permite para ejecutar sobre otros objetos.

3.2 La Interface e Implementación de una Clase

La interface de una clase provee sólo su vista superficial y de esta manera enfatiza la abstracción mientras oculta su estructura y secretos de su comportamiento. Declara las operaciones aplicables a las instancias. La implementación de una clase es su vista interna, la cual abarca los secretos de su comportamiento es la implementación de todas las operaciones declaradas en la interface de la clase.

En la interface tenemos tres tipos de declaraciones:

- **Pública** Una declaración que forma parte de la interface de una clase y es visible para todos los clientes.
- **Protegida** Una declaración que forma parte de la interface de una clase pero no es visible para otras clases excepto para sus subclases.
- **Privada** Una declaración que forma parte de la interface de una clase pero no es visible para ninguna otra clase.

3.3 El Rol de las Clases y Objetos en el Diseño

Durante el análisis y las primeras etapas del diseño el desarrollador tiene pricipalmente dos tareas.

- Identificar las clases y objetos que forman el vocabulario del dominio del problema.
- Inventar la estructura con la cual el conjunto de objetos trabajen juntos para proveer el comportamiento que satisfaga los requerimientos del problema.

Colectivamente, a las clases y objetos los denominaremos abstracciones llave del problema, y a las estructuras cooperativas las denominaremos los mecanismos de la implementación.

3.4 Clasificación

Clasificación es con lo cual nosotros ordenamos conocimientos. No hay como tal una estructura de clase perfecta, ni un conjunto de objetos correcto. Cuando clasificamos, buscamos agrupar cosas que tengan una estructura común o presenten un funcionamiento en común. La clasificación nos ayuda a identificar jerarquías de generalización/especialización y agregación entre clases.

3.5 El Modelo a Objetos

La programación orientada a objetos es un método de implementación en el cual los programas son organizados como una colección cooperativa de objetos, los cuales representan una instancia de alguna clase, y cuyas clases son miembros de una jerarquía de clases unidas mediante una relación de herencia.

Hay tres puntos importantes en la definición de programación orientada a objetos:

1. Usa objetos.
2. Cada objeto es una instancia de una clase.
3. Las clases se relacionan entre ellas por una relación de herencia.

Un lenguaje es orientado a objetos si y sólo si satisface los siguientes requerimientos:

- Que soporte objetos que son abstracción de datos con una interface de operaciones nombradas y un estado local oculto
- Objetos que tienen un tipo asociado.
- Los tipos pueden heredar atributos de supertipos.

El diseño orientado a objetos es un método de diseño conjugando el proceso de descomposición orientada a objetos y una notación para representar los modelos lógicos y físicos así como también los modelos estáticos y dinámicos del sistema a diseñar.

En esta definición tenemos dos puntos importantes:

1. Descomposición orientada a objetos.
2. Usa diferentes notaciones para expresar diferentes modelos de diseño lógico (estructuras de clases y objetos) y físico (módulos y arquitectura del proceso de un sistema).

El soporte para la descomposición orientada a objetos es lo que hace al diseño orientado a objetos completamente diferente al diseño estructurado: el primero utiliza la abstracción de clases y objetos para estructurar lógicamente los sistemas, y el segundo utiliza la abstracción de algoritmos.

3.5.1 Modelos Lógicos versus Modelos Físicos

Para el diseño orientado a objetos se deben tener las siguientes consideraciones:

- ¿Qué clases existen y como están relacionadas?
- ¿Qué mecanismos se utilizan para regular la colaboración entre clases?
- ¿Dónde se deben declarar cada clase y objeto?
- ¿A que procesador se debe asignar un proceso, y para un procesador dado, como se debe calendarizar sus procesos múltiples?

La respuesta a cada una de estas preguntas se puede expresar en cada uno de los siguientes diagramas, respectivamente:

- Diagramas de Clases
- Diagramas de Objetos
- Diagramas de Módulos
- Diagramas de Procesos

Estos cuatro diagramas forman la notación básica del diseño orientado a objetos. Los primeros dos diagramas son parte de la vista lógica del sistema, porque sirven para describir la existencia y significado de las abstracciones llave que forman el diseño. Los dos diagramas restantes son parte de la estructura física del sistema, porque son utilizados para describir el software concreto y los componentes de hardware de una implementación. En este capítulo sólo explicaremos los diagramas de clases y los diagramas de objetos debido a que sólo estos se utilizaron en el trabajo de tesis.

3.5.2 Semántica Estática versus Semántica Dinámica

Los cuatro diagramas que se han descrito son en gran parte estáticos. Sin embargo, los eventos suceden dinámicamente en todo sistema de software: los objetos se crean y se destruyen, los objetos se envían mensajes de uno a otro de una manera ordenada, y en algunos sistemas, los mensajes se envían simultáneamente. En el diseño orientado a objetos, la semántica dinámica de un diseño se expresa a través de dos diagramas adicionales:

- Diagramas de Transición de Estados
- Diagramas de Tiempos

Un diagrama de tiempos se puede utilizar en conjunción con cada diagrama de objetos para mostrar el ordenamiento de tiempo de mensajes de como estos son enviados y evaluados. Los diagramas de transición de estados no serán tratados en este capítulo.

3.6 Diagramas de Clases

Un diagrama de clases es usado para mostrar la existencia de clases y sus interrelaciones en el diseño lógico de un sistema. Un solo diagrama de clases representa todo o parte de la estructura de clases de un sistema. Los dos elementos más importantes de una estructura de clase son las clases y las interrelaciones de clases.

3.6.1 Clases

La figura 3.1 muestra el icono utilizado para representar una clase en un diagrama de clases.

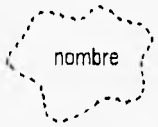


Figura 3.1 Icono de una clase.

Este icono representa una abstracción con algunas fronteras bien definidas. La línea punteada que forma el contorno de este icono indica que los clientes generalmente operan sólo sobre las instancias de una clase, y no en la clase misma. El nombre de la clase es requerido, y es colocado dentro de la nube. Cada nombre de clase debe ser único para la categoría de clase abarcada.

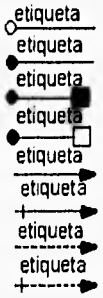
3.6.2 Interrelaciones de Clases

Los iconos para cada tipo de interrelación de clases y los iconos que se utilizan para expresar cardinalidad se muestran en la figura 3.2. Cada interrelación puede incluir una etiqueta (la cual se puede omitir) para documentar el nombre o rol de la interrelación.

Una interrelación de uso es indicada por una línea con un círculo blanco colocado en un extremo para designar la clase que utiliza los recursos de otra. Por ejemplo si usamos una línea para indicar una interrelación entre las clases A y B, y colocamos un círculo blanco cerca de la clase A, estamos asegurando que la clase A utiliza los recursos de la clase B.

La interrelación de contención es indicada con una línea con un círculo negro para denotar que una clase tiene instancias de otra clase. Para indicar la interrelación de contención por valor se coloca un cuadro negro al final de la interrelación. Y para indicar una interrelación de contención por referencia se coloca un cuadro blanco al final de la interrelación.

Una interrelación de clases, en donde la clase A instancie la clase B es indicada por una línea punteada, con una flecha indicando la clase que esta siendo instanciada. Este tipo de interrelación es útil sólo para lenguajes que soportan algún ordenamiento de clase parametrizada o mecanismo genérico.



de uso	1	uno
de contención	n	número ilimitado
de contención por valor	0..n	de cero a n
de contención por referencia	1..n	de uno a n
de herencia (tipo compatible)	0..1	de cero a uno
de herencia (nuevo tipo)		
de instanciación (tipo compatible)		
de instanciación (nuevo tipo)		

Figura 3.2 Interrelaciones de Clase y Cardinalidad

Una interrelación de herencia aparece igual que una interrelación de instanciación, excepto que se usa una línea sólida. Si las instancias de una subclase no son de tipo compatible con instancias de la superclase, entonces se coloca una línea perpendicular atravesando la línea de interrelación opuesta a la flecha.

3.7 Categorías de Clases

Para solucionar el problema de que una estructura de clase contenga bastantes clases y sea imposible poner todas las clases en un diagrama de clases surge la idea de una categoría de clases. En pocos lenguajes de programación orientada a objetos, las clases pueden ser anidadas en otras clases, esto es utilizado en programaciones muy grandes. En la práctica un sistema grande tiene un nivel alto en el diagrama de clases, consistiendo de categorías de clases en el nivel más alto de abstracción. Cada categoría de clases denota otro diagrama de clases así si cortamos alguna de las categorías de clases de lo más alto del diagrama y ampliamos en su implementación, podemos encontrar un diagrama de clases simple constituido de clases e interrelaciones de clases o (en problemas muy grandes) otro diagrama de clases conteniendo otras categorías de clases.

El icono para una categoría de clases aparece en la figura 3.3 un rectángulo con el nombre de la categoría adentro.

Figura 3.3 Icono de Categorías de Clases

3.8 Diagramas de Objetos

Un diagrama de objetos es usado para mostrar la existencia de objetos y sus interrelaciones en el diseño lógico de un sistema. Un solo diagrama de objetos representa todo o parte de la estructura de objetos de un sistema, comunmente el diseño de un sistema requiere un conjunto de diagramas de objetos.

Son importantes las interrelaciones entre diagramas de clases y diagramas de objetos. Específicamente cada objeto en un diagrama de objetos denota alguna instancia (específica o arbitraria) de una clase. Las operaciones en un diagrama de objetos deben ser consistentes con las operaciones definidas en las clases asociadas. La consistencia es en ambas direcciones. Necesitamos tanto los diagramas de clases como los diagramas de objetos para documentar el diseño lógico de un sistema, debido a que estos diagramas muestran diferentes decisiones de diseño. Los diagramas de clases documentan las abstracciones llave en nuestro sistema, y los diagramas de objetos subrayan los mecanismos importantes que manipulan estas abstracciones.

3.8.1 Objetos

En la figura 3.4 se muestra el icono que representa un objeto en un diagrama de objetos. El objeto en un diagrama de objetos es representado de forma similar a la clase, excepto que se utiliza una línea sólida para mostrar que es una instancia. El nombre del objeto es requerido pero no indispensable y no necesita ser único.

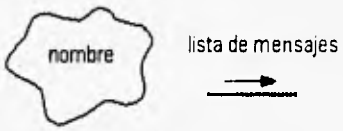


Figura 3.4 Iconos de Objetos e Interrelación de Objetos

3.8.2 Interrelación de Objetos

Una interrelación entre dos objetos simplemente significa que los objetos pueden mandar mensajes a otros. Como los mensajes son normalmente bidireccionales se utilizan líneas sin flechas para indicar esta interrelación como se muestra en la figura 3-4. Se utiliza una línea sólida para interrelaciones de objetos que podemos modelar en el software dentro del sistema, y líneas grises para las que están fuera del sistema.

3.9 Diagramas de Tiempos

Eventos Ordenados por Tiempos

Por sí mismos, los diagramas de objetos son estáticos, muestran una colección cooperativa de objetos que pasan mensajes de uno a otro, pero no muestran el flujo de control ni tampoco muestran el ordenamiento de eventos. Por tal motivo se utilizan los diagramas de tiempos para documentar las dinámicas de paso de mensajes en un diagrama de objetos:

Capítulo 4

Descripción del Sistema

4.1 Características de los Proyectos

En un proyecto de ingeniería de software además de estar incluidas todas las etapas del desarrollo de un sistema, se debe llevar el control y la administración del proyecto. Existen herramientas para el desarrollo de software denominadas CASE (Computer-Aided Software Engineering), que nos facilitan el desarrollo de estos proyectos. Algunas de las características de una herramienta CASE son las siguientes: manejo de gráficos para los diversos tipos de modelos y verificación de consistencia internamente en cada modelo y a nivel general entre los diversos modelos.

El desarrollo de estas herramientas requiere una serie de recursos humanos y tecnológicos sumamente especializados, debido a que se trata de una tarea bastante compleja y de amplios alcances, implicando varios años/hombre de trabajo. Estas características hacen de éste, un producto de alto costo, el cual sólo puede ser adquirido por grandes compañías.

En este sistema se manejarán proyectos pero de una manera muy primitiva, ya que sólo cubrirá dos herramientas del análisis de sistemas:

- Diagrama de Flujo de Datos
- Diagrama de Contexto

y no se desarrollará ninguna herramienta para el control y administración de proyectos. Sin embargo el sistema está desarrollado de tal forma que su ampliación y modificación puedan llevarse acabo de manera muy rápida y sencilla en trabajos posteriores.

Cada proyecto, en este sistema, además de tener un diagrama de flujo de datos y un diagrama de contexto, contiene información propia del proyecto, que le es útil al usuario para llevar un control muy básico de su proyecto. A continuación se enlista la información general que cada proyecto contiene :

Número de Proyecto: Se trata de un número con el cual se pueda identificar el proyecto de una manera más sencilla y rápida. Este número es propuesto por el usuario.

Nombre del Proyecto: Es el nombre asociado a dicho proyecto, el cual es otorgado por el usuario.

Nombre del Responsable: Es el nombre de la persona que esta al frente del proyecto, el usuario lo proporciona.

Puesto del Responsable: Es el cargo que tiene en la empresa el responsable del proyecto, y también es dado por el usuario.

Compañía: Corresponde al nombre de la empresa la cual desarrolla el proyecto, o a la cual se le lleva acabo el proyecto, y es proporcionado por el usuario.

Fecha de creación: Nos indica en que fecha fué creado el proyecto. Esta fecha es dada automáticamente por el sistema.

4.2 Características de los Diagramas de Flujo de Datos

En el desarrollo de un proyecto normalmente se encuentra involucrado un equipo de trabajo, el cual cuenta con una persona responsable que se encargará de supervisar el trabajo de cada uno de sus integrantes con el fin de obtener resultados satisfactorios en el menor tiempo posible. En este caso la persona responsable del proyecto supervisará que se realice adecuadamente el diagrama de contexto y el diagrama de flujo de datos, evitando diagramas erróneos es decir incompletos ó lógicamente inconsistentes y procurando que la presentación de estos diagramas sea grata a la vista del usuario.

Considerando que el análisis estructurado es un proceso iterativo, los diagramas de flujo de datos se modifican constantemente, siendo incluso necesario retomar diagramas que habían sido desechados. Bajo estas consideraciones es necesario manejar diferentes versiones de los diagramas. Dado que en el desarrollo de un diagrama de flujo de datos pueden resultar bastantes versiones, es conveniente llevar un control de las mismas para evitar confusiones. Para lograr este control es de gran utilidad conocer la última versión con la que se trabajó, para mantener una continuidad en nuestro trabajo, y no detenernos en buscar dentro de todas las versiones del diagrama de flujo de datos cual fué la última versión con la que estuvimos trabajando. La última versión con la que se trabajó corresponde a la versión actual, esta versión actual podría ser la última versión que sufrió modificaciones ó alguna versión que el usuario eligió libremente. También es importante manejar una versión aprobada y la fecha de aprobación; la versión aprobada corresponde a la versión que aprueba la persona responsable del desarrollo del diagrama de flujo de datos, para ser

presentada al cliente. De todas las versiones que pueda tener una diagrama de flujo de datos sólo podrá haber una versión aprobada.

En este sistema se manejan versiones de diagrama de flujo de datos. Un diagrama de flujo de datos tiene por lo menos una versión. Además se lleva el control de la versión actual y la versión aprobada, por lo cual el sistema guardará el status de estas versiones. La versión actual es la versión que se abre por omisión al inicializar el módulo de diagrama de flujo de datos, existen tres maneras de asignar la versión actual en el sistema:

- El usuario puede cambiar explícitamente la versión actual, eligiendo cualquiera de las versiones existentes.
- Al momento de modificar una versión y guardarla como una nueva versión, automáticamente esta nueva versión pasa a ser la versión actual.
- Al elegir una versión como la versión aprobada ésta será también la versión actual.

Existen diversas simbologías que se pueden utilizar para representar gráficamente los componentes de un diagrama de flujo de datos. Debido a que cada empresa tiene diferentes estándares de trabajo, para realizar la representación gráfica de los diagramas, el analista debe adecuarse a estos estándares y utilizar la simbología indicada. En el caso de que una empresa se dedique a realizar proyectos de diversas empresas, por lo general el equipo de trabajo encargado de desarrollar dichos proyectos tiene ya una normatividad en su plan de trabajo, y por lo tanto se encuentran familiarizados en el uso de una simbología, por tal motivo sería conveniente que para el desarrollo de sus diagramas de flujo de datos utilizaran esta simbología y si en un momento dado alguno de sus clientes solicitara otra simbogía, automáticamente crear otra versión de diagrama de flujo de datos con la nueva simbología. Para satisfacer estas necesidades en el sistema se le permite al usuario modificar la simbología por versión de diagrama de flujo de datos, y manejar diversas simbologías. Aunque en este trabajo de tesis sólo se manejan dos simbologías, el sistema está diseñado de tal manera que se puedan agregar más simbologías en trabajos posteriores de manera sencilla. Y en el caso de que el usuario no elijá una simbología en especial se tomará la simbología por omisión.

Una versión de diagrama de flujo de datos, se conforma de figuras que a su vez contienen procesos, terminadores, almacenes y flujos. Para cada figura se podrá definir el tamaño que se manejará para representar cada uno sus elementos, por ejemplo, si elegimos en la simbología representar a un proceso en nuestro diagrama de flujo de datos como un círculo, podríamos tener en la figura 1 un proceso con un radio de tres centímetros y en la figura 2 un proceso con un radio de dos centímetros, con fines de organización y estética.

Es importante conservar información general del diagrama de flujo de datos que nos ayude a tener un mayor control de nuestro proyecto. En este sistema se maneja información general del diagrama de flujo de datos, de las figuras y de los procesos.

Información del Diagrama de Flujo de Datos:

> *Nombre del Proyecto:* Es el nombre que el usuario le asignó al proyecto al cual pertenece el diagrama de flujo de datos

Nombre del Responsable: Es el nombre de la persona responsable de realizar el diagrama de flujo de datos, esta información también la proporciona el usuario.

Número de Versiones: Es el número de versiones que constituyen el diagrama de flujo de datos, siendo uno el número mínimo de versiones. Este número lo asigna el sistema automáticamente.

Versión Aprobada: Es la versión que aprueba la persona responsable del diagrama de flujo de datos para ser entregada al cliente.

Fecha de Aprobación: Esta fecha es dada por el sistema cuando una versión ha sido aprobada.

Versión Actual: Esta versión la otorga el sistema de acuerdo a las diversas maneras que existen para asignar la versión actual y que ya se explicaron anteriormente.

Información de la Figura:

Nombre de la Figura: Es el nombre que el usuario le asocia a la figura, y en el caso de que la figura represente un proceso debe llevar el mismo nombre del proceso.

Número de la Figura: Es el número ó indentificador que el usuario le asigna a la figura.

Nombre del Responsable: Es el nombre de la persona responsable de realizar la figura.

Fecha de Creación: Esta fecha la asigna automáticamente el sistema y corresponde a la fecha de creación de la figura.

Fecha de Actualización: Es la fecha que el sistema asigna cada vez que se hacen modificaciones a la figura y estos se guardan, por lo cual esta fecha se actualizará constantemente.

Información del Proceso:

Nombre del Proceso: Es el nombre que el usuario le asigna a un proceso, dicho nombre debe identificar la función que lleva acabo el proceso en el sistema.

Número del Proceso: Es el número ó identificador que el usuario le asigna al proceso. Esta numeración servirá para relacionar las diversas figuras por niveles.

Flujos de Entrada al Proceso: El sistema arrojará la lista de flujos de entrada al proceso automáticamente.

Flujos de Salida del Proceso: El sistema arrojará la lista de flujos de salida al proceso automáticamente.

Documentación: El usuario proporcionará una breve documentación sobre la función que realiza el proceso en el sistema.

Para evitar un diagrama de flujo de datos complejo, éste se organiza en una serie de niveles de modo que cada uno proporcione sucesivamente más detalle, sobre una porción del nivel anterior. La manera para relacionar las figuras entre los diversos niveles, es gracias a los números y nombres de procesos, debido a que de esta manera, un proceso de una figura de un nivel dado se relaciona con la figura que lo describe más a fondo del nivel inmediato inferior llevando esta figura el mismo número y nombre del proseso que describe.

Para realizar un diagrama de flujo de datos existen tres estrategias de desarrollo, la primera consiste en comenzar de forma descendente, lo cual es, iniciar el diagrama a partir del diagrama de contexto y posteriormente ir desarrollando los niveles inferiores hasta llegar a los procesos más simples, la segunda es comenzar de forma ascendente, es decir iniciar desde los procesos más simples hasta llegar al diagrama de contexto y la tercera consiste en combinar ambas estrategias. En nuestro sistema se podrán utilizar cualquiera de estas estrategias, lo que implica que si se realiza un diagrama de flujo de datos de forma ascendente ó combinada tengamos figuras aisladas, que no tengan relación con otras figuras, por lo que también ofrecemos mecanismos para asociar un proceso a una figura y de esta manera ir haciendo la nivelación del diagrama de flujo de datos.

Además le proporciona al usuario operaciones, para borrar toda una versión de diagrama de flujo de datos ó solo una figura, operaciones de edición como deshacer y rehacer un comando, operaciones para desplegar una lista de terminadores y almacenes, estas listas nos ayudarán a tener presente los nombres de los almacenes y terminadores que ya hemos utilizado y si necesitamos repetir algún terminador o almacén, evitaremos equivocarnos con el nombre.

Las principales verificaciones de consistencia que se deben llevar acabo en un diagrama de flujo de datos son:

- Todos los procesos deberán llevar un nombre
- No se podrán repetir los nombres de los procesos en la misma versión de diagrama de flujo de datos.
- El nombre del proceso debe corresponder con el nombre de la figura que lo representa.
- No se permitirán procesos con entradas pero sin salidas.
- No se permitirán procesos con salidas sin entradas.
- Tampoco se permitirán almacenes de sólo lectura ó sólo escritura.
- Los flujos de datos que salen y entran de un proceso de un nivel dado deben corresponder a los flujos que entran y salen de toda la figura en el nivel inmediato inferior que lo describe.

El sistema maneja todos los puntos anteriores con la excepción del último.

4.3 Características del Diagrama de Contexto

El diagrama de contexto es una especialización del diagrama de flujo de datos, por tal motivo sus características son muy similares. Sus principales diferencias son:

- El diagrama de contexto sólo maneja una figura. Por lo cual no maneja la nivelación.
- El diagrama de contexto contiene un solo proceso.

Por los mismos motivos que en el diagrama de flujo de datos, en el diagrama de contexto se manejan versiones y se puede llevar el control de la versión actual y la versión aprobada. Se puede hacer la selección de la simbología deseada.

Es importante conservar información general del diagrama de contexto para tener información de sus versiones. En este sistema se maneja información general del diagrama de contexto.

Información del Diagrama de Contexto:

Nombre del Proyecto: Es el nombre que el usuario le asignó al proyecto al cual pertenece el diagrama de contexto.

Nombre del Responsable: Es el nombre de la persona responsable de realizar el diagrama de contexto, esta información también la proporciona el usuario.

Número de Versiones: Es el número de versiones que constituyen el diagrama de contexto, siendo uno el número mínimo de versiones. Este número lo asigna el sistema automáticamente.

Versión Aprobada: Es la versión que aprueba la persona responsable del diagrama de contexto para ser entregada al cliente.

Fecha de Aprobación: Esta fecha es dada por el sistema cuando una versión ha sido aprobada.

Versión Actual: Esta versión la otorga el sistema de acuerdo a las diversas maneras que existen para asignar la versión actual y que ya se explicaron anteriormente.

4.4. Interface con el Usuario

4.4.1 Interface del Proyecto

Para trabajar en el sistema es necesario crear un proyecto. Al accesar al proyecto se presenta una ventana la cual se muestra en la figura. 4.1. La ventana está formada por un menú, una barra de herramientas y una barra de estados. El menú cuenta con diversas opciones para manejar la información del proyecto y para accesar al módulo de diagrama de

flujo de datos y al módulo del diagrama de contexto. La barra de herramientas nos permite en forma rápida realizar operaciones sobre el proyecto y accesar al módulo de diagrama de flujo de datos y al módulo del diagrama de contexto. En la parte inferior se encuentra una barra de estados en la que se describe en forma textual la función de las opciones del menú.

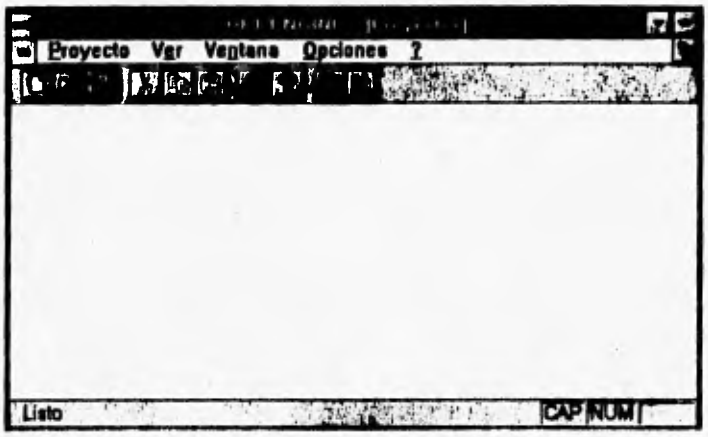


Figura 4.1 Ventana Principal del Módulo Proyecto

Crear Proyecto

1. Elegir del menú **Proyecto** la opción **Nuevo**.
2. Se muestra una caja de diálogo para capturar la información del proyecto.
3. Una vez capturada la información del proyecto, dar *click* al botón de **Aceptar** si se desea almacenar, de lo contrario dar *click* al botón de **Cancelar.**

Modificar Información del Proyecto

1. Del menú **Opciones** elegir **Información del Proyecto.**
2. Se mostrará una caja de diálogo con la información del proyecto como la que se muestra en la figura 4.2. Con la tecla de TAB se desplazará a través de los campos, o posicione el cursor donde desee alguna modificación y realice los cambios. Dar *click* al botón **Acepter.**

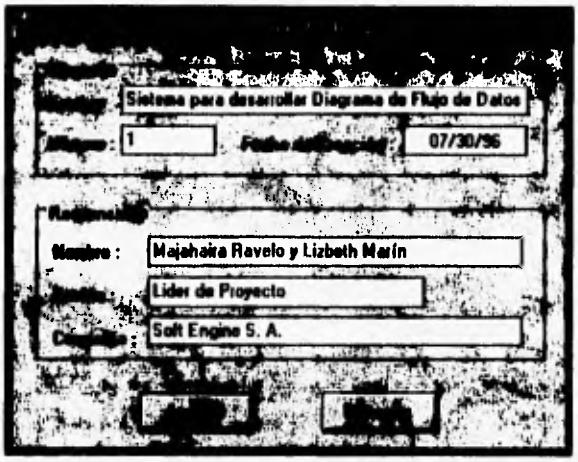


Figura 4.2 Caja de Diálogo de la Información del Proyecto

Abrir el Módulo de Diagrama de Flujo de Datos

1. Del menú **Opciones** elegir **DFD** o de la barra de herramientas dar *click* al icono .
2. La aplicación del diagrama de flujo de datos se mostrará.

Abrir el Módulo de Diagrama de Contexto

1. Del menú **Opciones** elegir **DC** o de la barra de herramientas dar *click* al icono .
2. La aplicación del diagrama de flujo de datos se mostrará.

4.4.2 Interface de Diagrama de Flujo de Datos

Al accesar al Diagrama de Flujo de Datos a través de su proyecto asociado, se presenta una ventana que nos permite trabajar sobre las versiones del diagrama de flujo de datos, esta ventana se muestra en la figura. 4.3.

La ventana está formada por un menú, una barra de herramientas, una barra de estados y una barra de elementos. El menú cuenta con diversas opciones que son útiles para llevar a cabo las versiones de un diagrama de flujo de datos. La barra de herramientas nos permite en forma rápida realizar operaciones sobre las figuras y sobre los elementos que contienen las figuras. En la parte inferior se encuentra una barra de estados en la que se describe en forma textual las opciones del menú. Del lado izquierdo se localiza la barra de elementos

que está formada por cuetro botones con los cuales se puede dibujar en una figura procesos, almacenes, terminadores y flujos.

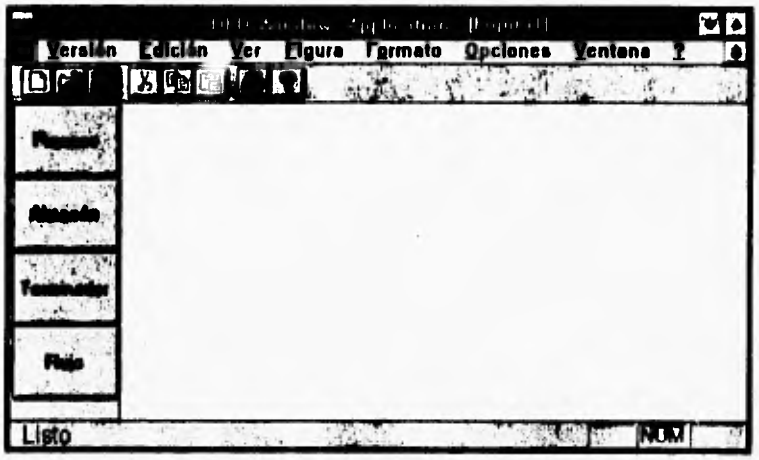


Figura 4.3 Ventana Principal del Diagrama de Flujo de Datos.

En el diagrama de flujo de datos se pueden llever acabo diferentes comandos a través del menú. Entre las opciones del menú con que se cuenta tenemos las siguientes:

- **Verelón** Este menú proporcione operaciones para manejar las versiones de diagrama de flujo de datos.
- **Edición** Este menú permite llevar a cabo comandos de edición.
- **Ver** Permite visualizar u ocultar las barras de herramientas y estados.
- **Figure** Este menú proporciona operaciones para manejar las figuras de una versión de diagrama de flujo de datos.
- **Formato** Precenta comandos para especificar los parámetros gráficos de los elementos de una figura.
- **Opcionee** Permite realizar modificaciones a la información generel del diagrama de flujo de datos, de las figuras y de los procesos.

- **Ventana** Proporciona operaciones para el manejo de ventanas.

Estas opciones de menú las podemos ver en la figura 4.3.

4.4.2.1 Manú Versión

Las operaciones que podemos llevar acabo con las Versiones de Diagrama de Flujo de Datos se muestran en la Figura 4.4 y son las siguientes:

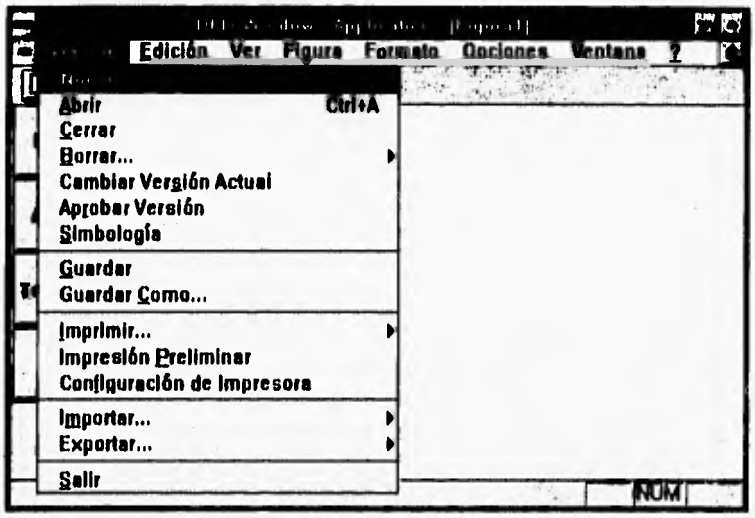


Figura 4.4 Opciones del Menú de Versión

Abrir Versión de Diagrama de Flujo de Datos.

1. Elegir del menú **Versión** la opción **Abrir**.
2. De las versiones disponibles que se muestran en la caja de diálogo (ver figura 4.5)seleccionar la versión deseada. Dar *click* al botón de **Abrir**.
3. La figura actual será desplegada.

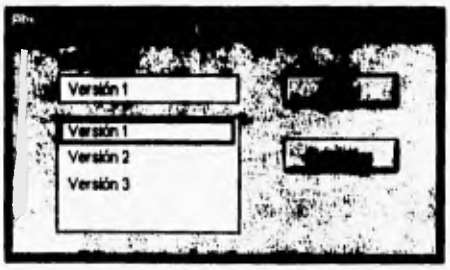


Figura 4.5 Caja de Diálogo para Abrir una Versión

Cerrar Versión de Diagrama de Flujo de Datos.

Seleccionar del menú **Versión** la opción **Cerrar**. El sistema verifica si existen cambios sin guardar, de ser así pregunta si se desean guardar los cambios.

Borrar Versión de Diagrama de Flujo de Datos.

1. Elegir del menú **Versión** la opción **Borrar...**
2. Seleccionar la opción **Borrar Versión DFD**.
3. De las versiones disponibles seleccionar la versión que se desea borrar. Dar *click* al botón de **Aceptar**.
4. Se mostrará un mensaje de advertencia preguntando si se está seguro de que se desea borrar la versión seleccionada. En caso de estar seguro dar *click* al botón de **Aceptar**

Borrar Figura.

1. Elegir del menú **Versión** la opción **Borrar...**
2. Seleccionar la opción **Borrar Figura**.
3. De las figuras disponibles seleccionar la figura que se desea borrar. Dar *click* al botón de **Aceptar**.
4. Se mostrará un mensaje de advertencia preguntando si se está seguro de que se desea borrar la figura seleccionada. En caso de estar seguro dar *click* al botón de **Aceptar**.

Cambiar Versión Actual

1. Elegir del menú **Versión** la opción **Cambiar Versión Actual**.
2. Se despliega una caja de diálogo como la de la figura 4.6 presentando una lista de las versiones de diagrama de flujo de datos existentes, donde se selecciona la versión que se desee como versión actual. Dar *click* al botón de **Cambiar Versión**.

3. Esta versión será abierta por omisión cada vez que se inicialice el sistema o mientras no se cree una nueva versión.

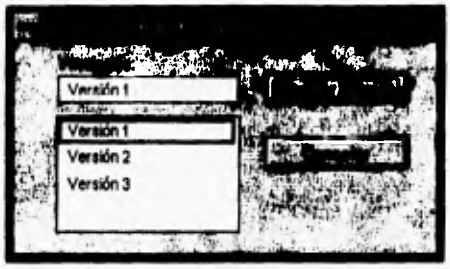


Figura 4.6 Caja de Diálogo para Cambiar la Versión Actual

Cambiar Simbología.

1. Elegir del menú **Opciones** la opción **Simbología.**
2. Se mostrará una caja de diálogo con las simbologías que el sistema maneja (ver figura 4.7). Seleccionar la simbología que se desee. Si se quiere utilizar la simbología por omisión dar *click* al botón **Default**.
3. Esta simbología se mantiene para todos los elementos de las figuras que pertenecen a la versión actual.

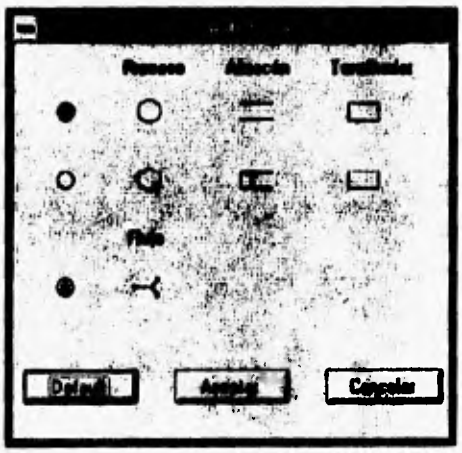


Figura 4.7 Caja de Diálogo para Seleccionar Simbología

4.4.2.2 Menú Edición

Las opciones de **Menú Edición** se muestran en la figura 4.8.

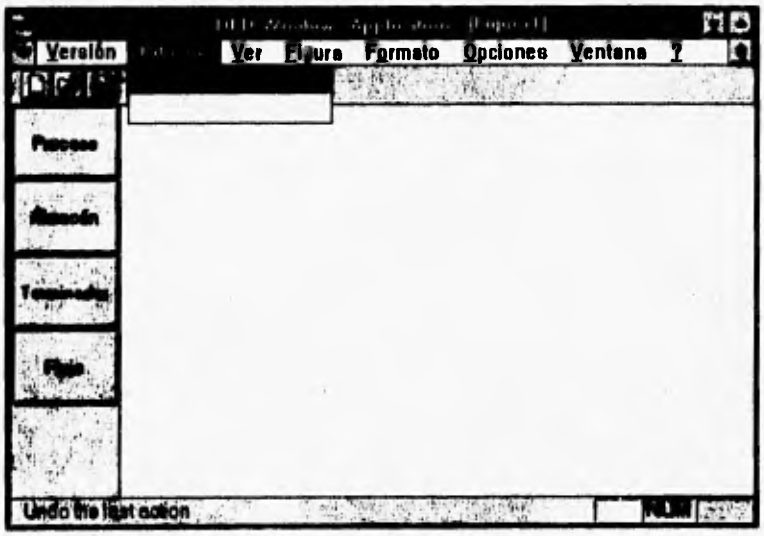


Figura 4.8 Opciones del Menú de Edición

4.4.2.3 Menú Ver

Las opciones de **Menú Ver** se pueden ver en la figura 4.9.

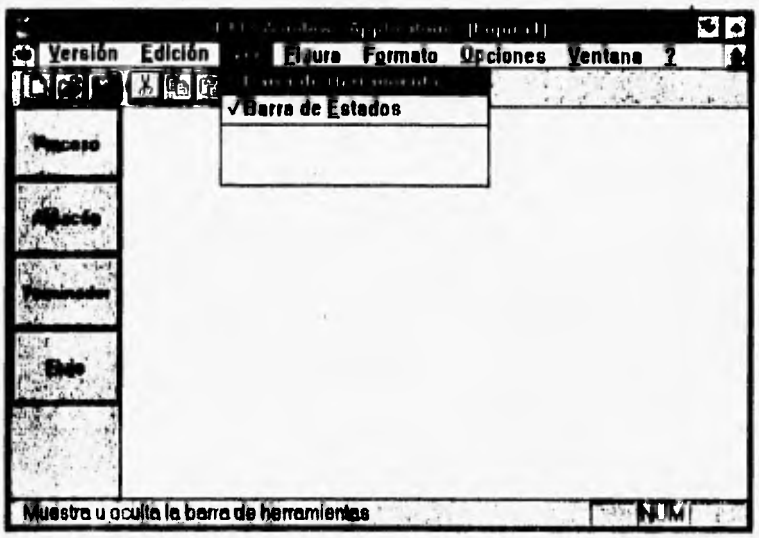


Figura 4.9 Opciones del Menú Ver

Ver u Ocultar Barra de Herramientas

1. Elegir del menú **Ver** la opción **Barra de Herramientas.**
2. La opción se marcará con una palomita (√) y la barra de herramientas será mostrada.
3. Para ocultar la barra de herramientas darle *click* nuevamente a la opción barra de herramientas y esta se ocultará.

Ver u Ocultar Barra de Estados

1. Elegir del menú **Ver** la opción **Barra de Estados.**
2. La opción se marcará con una palomita (√) y la barra de estados será mostrada.
3. Para ocultar la barra de estados darle *click* nuevamente a la opción barra de estados y esta se ocultará.

4.4.2.4 Menú Figura

Las opciones de **Menú Figura** se presentan en la figura 4.10.

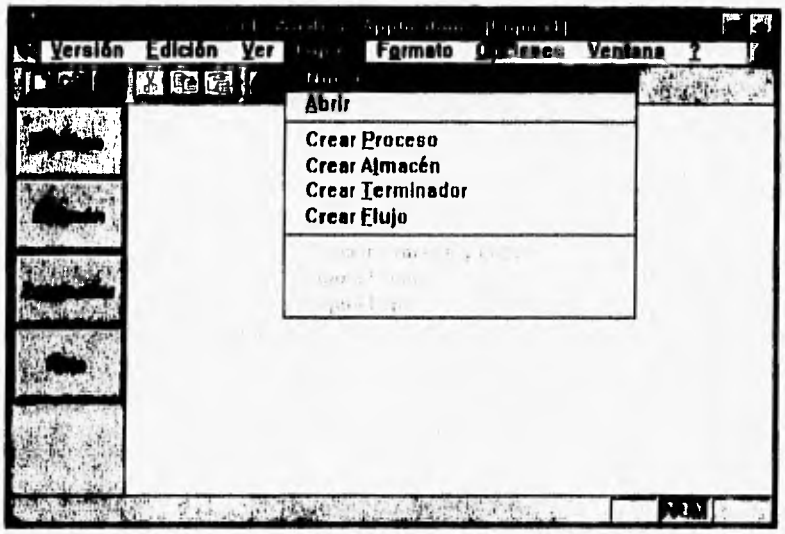


Figura 4.10 Opciones del Menú Figura

Crear Figura.

1. Elegir del menú **Figura** la opción **Nuevo**.
2. Introducir la información de la nueva figura, si se desea , en la caja de diálogo que se mostrará. Dar *click* al botón de **Aceptar**
3. Las nueva figura se mostrará en la pantalla con el nombre que se le haya otorgado.

Abrir Figura.

1. Elegir del menú **Figura** la opción **Abrir**.
2. Se mostrará una caja de diálogo como la mostrada en la figura 4.11, que presente las figuras de la versión actual del diagrama de flujo de datos, donde se seleccionará la figura que se desea abrir. Dar *click* al botón de **Abrir**.
3. La figura será desplegada.

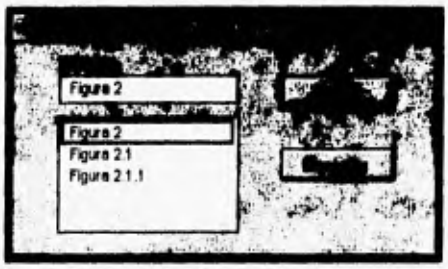


Figura 4.11 Caja de Diálogo para Abrir una Figura

Creación de Elementos

La creación de los elementos que integran una figura (procesos, almacenes, terminadores y flujos) se puede llevar acabo de dos maneras diferentes, una es por medio del menú, la otra es a través de la barra de elementos que se localiza del lado izquierdo en la ventana de Diagrama de Flujo de Datos.

Creación de Elementos con la Barra de Elementos

1. Seleccionar el botón del elemento que se desee dibujar (proceso, almacén, terminador o flujo).
2. Posicionar el cursor donde se deseé dibujar el elemento.
3. Dar *click* al botón izquierdo del mouse y el elemento se dibujará en esa posición.

Creación de Elementos con el Menú

Crear Proceso.

1. Elegir del menú **Figura** la opción **Crear Proceso.**
2. El cursor del mouse se tornará en forma de cruz, indicando que está en modo de crear elemento. Dar *click* al botón izquierdo del mouse cuando éste se encuentre en la posición donde se desea colocar el proceso.
3. El proceso se dibujará, en dicha posición, según la representación gráfica seleccionada.

Crear Almacén.

1. Elegir del menú **Figura** la opción **Crear Almacén.**
2. El cursor del mouse se tornará en forma de cruz, indicando que está en modo de crear elemento. Dar *click* al botón izquierdo del mouse cuando éste se encuentre en la posición donde se desea colocar el almacén.
3. El almacén se dibujará, en dicha posición, según la representación gráfica seleccionada.

Crear Terminador.

1. Elegir del menú **Figura** la opción **Crear Terminador**.
2. El cursor del mouse se tomará en forma de cruz, indicando que está en modo de crear elemento. Dar *click* al botón izquierdo del mouse cuando éste se encuentre en la posición donde se desea colocar el terminador.
3. El terminador se dibujará, en dicha posición, según la representación gráfica seleccionada.

Crear Flujo.

1. Elegir del menú **Figura** la opción **Crear Flujo**.
2. El cursor del mouse se tomará en forma de cruz, indicando que está en modo de crear elemento. Dar *click* al botón izquierdo del mouse cuando éste se encuentre en la posición donde se desea colocar el punto inicial del flujo, arrastrar el mouse manteniendo el botón presionado hasta el punto final.
3. El flujo se dibujará, en dicha posición, según la representación gráfica seleccionada.

Ir a Figura Padre.

1. Elegir del menú **Figura** la opción **Figura Padre**.
2. Se mostrará la figura que contiene al proceso que es descrito por la figura actual.

Ir a la Figura Hijo.

Esta operación puede llevarse acabo de dos maneras: una es por medio del menú y la segunda es automática.

De manera automática

1. Seleccionar el proceso y dar *doble click*.
2. Se mostrará la figura que describe el proceso seleccionado.

Por Menú

1. Seleccionar el proceso.
2. Elegir del menú **Figura** la opción **Figura Hijo**.
3. Se mostrará la figura que describe el proceso seleccionado.

4.4.2.5 Menú Formato

Las opciones de **Menú Formato** son mostradas en la figura 4.12.

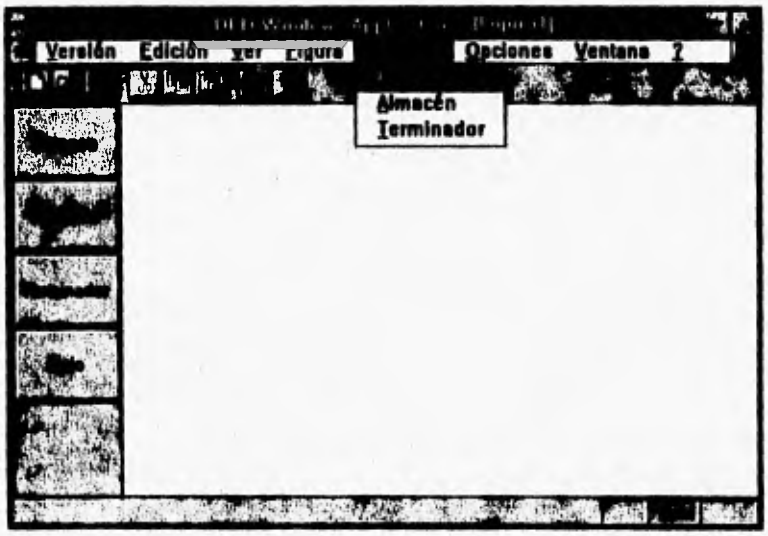


Figura 4.12 Comandos del Menú Formato

Especificar Dimensiones del Proceso.

1. Elegir del menú **Formato** la opción **Proceso.**
2. Se mostrará una caja de diálogo con los últimos valores, proporcionar los nuevos valores que se deseen (ver figura 4.13). Dar *click* al botón **Aceptar.** En caso de desear los valores por omisión dar *click* al botón **Default.**. Estas dimensiones se mantienen solo en la figura actual.
3. Todos los proceso que se encuentren en la figura se dibujarán con las nuevas dimensiones.

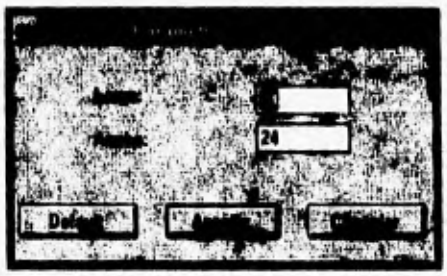


Figura 4.13 Caja de Diálogo para Seleccionar los Parámetros del Proceso

Especificar Dimensiones del Almacén.

1. Elegir del menú **Formato** la opción **Almacén.**
2. Se mostrará una caja de diálogo como la de la figura 4.14 con los últimos valores, proporcionar los nuevos valores que se deseen. Dar *click* al botón **Aceptar.** En caso de desear los valores por omisión dar *click* al botón **Default.**. Estas dimensiones se mantienen solo en la figura actual.
3. Todos los almacenes que se encuentren en la figura se dibujarán con las nuevas dimensiones.

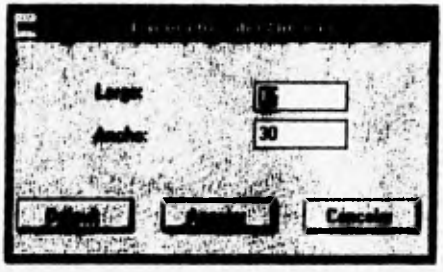


Figura 4.14 Caja de Diálogo para Seleccionar los Parámetros del Almacén

Especificar Dimensiones del Terminador.

1. Elegir del menú **Formato** la opción **Terminador.**
2. Se mostrará una caja de diálogo con los últimos valores, proporcionar los nuevos valores que se deseen (ver figura 4.15). Dar *click* al botón **Aceptar.** En caso de desear los valores por omisión dar *click* al botón **Default.**. Estas dimensiones se mantienen sólo en la figura actual.
3. Todos los terminadores que se encuentren en la figura se dibujará con las nuevas dimensiones.

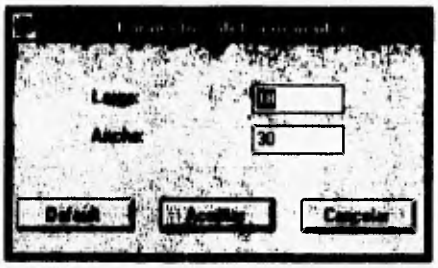


Figura 4.15 Caja de Diálogo para Seleccionar los Parámetros del Terminador

4.4.2.6 Menú Opciones

Los comandos del **Menú Opciones** se muestran en la figura 4.16.

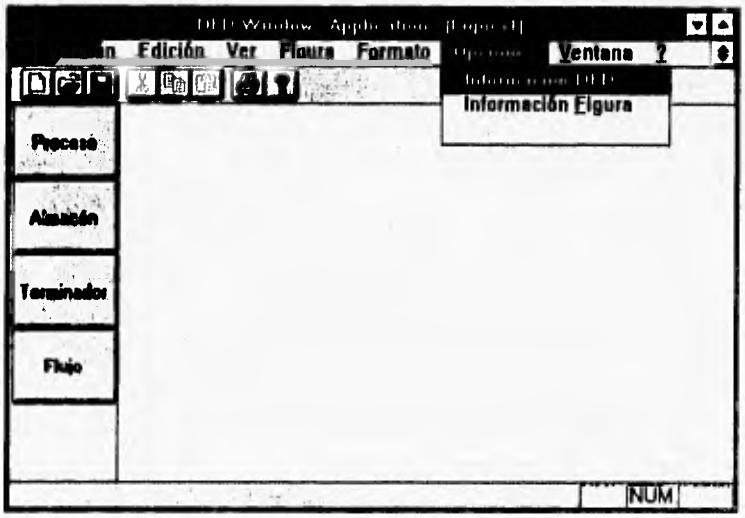


Figura 4.16 Comandos del Menú Opciones

Modificar Información de Diagrama de Flujo de Datos.

1. Del menú **Opciones** elegir **Información DFD.**
2. Se mostrará una caja de diálogo con la información del Diagrama de Flujo de Datos (ver figura 4.17). Con la tecla de TAB se desplazará a través de los campos, o posicione el cursor donde desee alguna modificación y realice los cambios. Dar *click* al botón **Aceptar.**

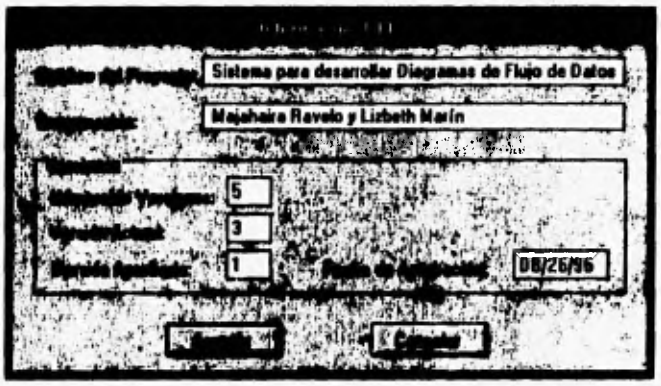


Figura 4.17 Caja de Diálogo de la Información del Diagrama de Flujo de Datos

Modificar Información de la Figura.

1. Elegir del menú **Opciones** la opción **Información Figura.**
2. Se mostrará una caja de diálogo con la información de la figura (ver figura 4.18). Posicione el cursor donde desee realizar una modificación, o desplácese con la tecla de TAB a través de los campos y haga los cambios. Dar *click* al botón **Aceptar .**

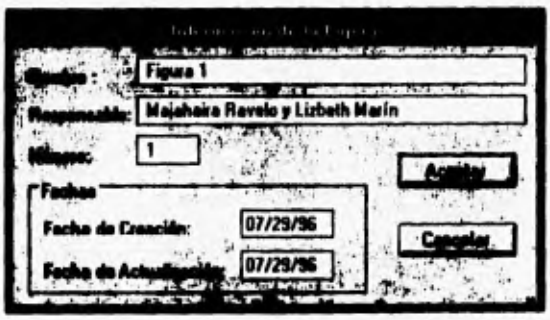


Figura 4.18 Caja de Diálogo de la Información de la Figura

Modificar Información del Proceso.

1. Elegir del menú **Opciones** la opción **Información Proceso.**
2. Se mostrará una caja de diálogo con la información del proceso como se muestra en la figura 4.19. Posicione el cursor donde desee realizar una modificación o desplácese con la tecla de TAB a través de los campos y haga los cambios. Dar *click* al botón **Aceptar** .

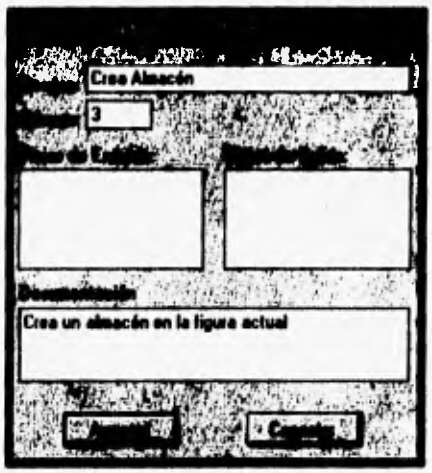


Figura 4.19 Caja de Diálogo de la Información del Proceso

4.4.3 Interface del Diagrama de Contexto

La ventana principal del diagrama de contexto se muestra en la figura 4.20. Como se puede observar es muy parecida a la del diagrama de flujo de datos con la diferencia que la barra de elementos (barra del lado izquierdo de la pantalla), no contiene el botón para crear procesos, sino que por omisión éste se crea ya que sólo debe existir un proceso en el diagrama.

Las operaciones que se llevan acabo en el diagrama de contexto son las mismas que las del diagrama de flujo de datos, a diferencia que éste se conforma sólo de una figura.

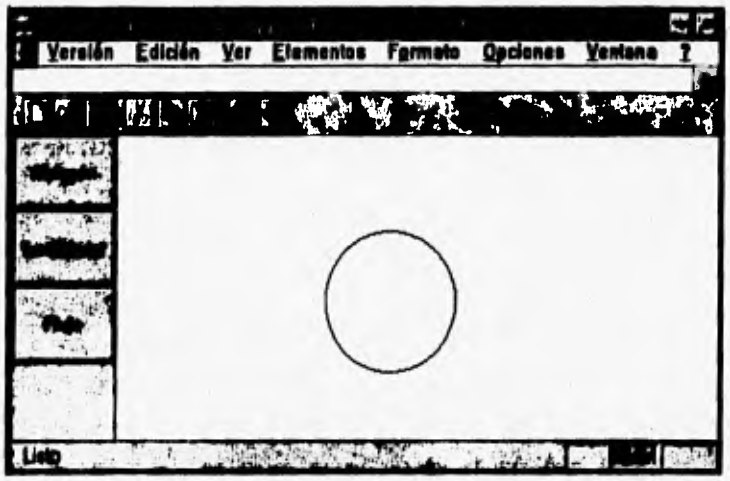


Figura 4.20 Ventana Principal del Módulo de Diagrama de Contexto

Las opciones de menú son semejantes a las del diagrama de flujo de datos por lo que no se explicarán.

Capítulo 5

Implementación del Sistema

Con el objeto de poder ampliar el sistema en forma flexible se programó éste en forma modular de tal manera que está conformado por tres programas:

- Un programa para el manejo de proyectos.
- Un programa para el manejo de diagramas de contexto.
- Un programa para el manejo de diagramas de flujo de datos.

Esta estructura permite ampliar y modificar cada uno de los componentes del sistema en forma indipendiente.

5.1 Estructura de Clases Proporcionada por el Framework de Visual C++

En esta sección se describe la relación entre la estructura de clases proporcionada por Visual C++ y las clases desarrolladas particularmente para el diagrama de flujo de datos la cual se muestra en la figura 5.1.

En la estructura de clases proporcionada por Visual C++ a través de la creación de una aplicación con la herramienta AppWizard (descrita en el apéndice B), se observa que la clase de la aplicación CDFDApp tiene dos objetos de las clases: CMainFrame y CMultiDocTemplate, los cuales los contiene por referencia.

La clase CMainFrame contiene dos objetos por valor uno que pertenece a la clase CToolBar y otro que pertenece a la clase CStatusBar. La clase CMultiDocTemplate contiene por

referencia a tres objetos: uno de la clase CMDIChild, otro de la clase CDFDView y el último de la clase CDFDDoc. La clase DFDView usa a las clases CPrintInfo y CClientDC. Mientras que la clase CDFDDoc contiene por referencia un objeto de la clase SEDFD la cual es parte principal del diagrama de flujo de datos.

Esta estructura de clases coincide tanto para el proyecto como para el diagrama de contexto.

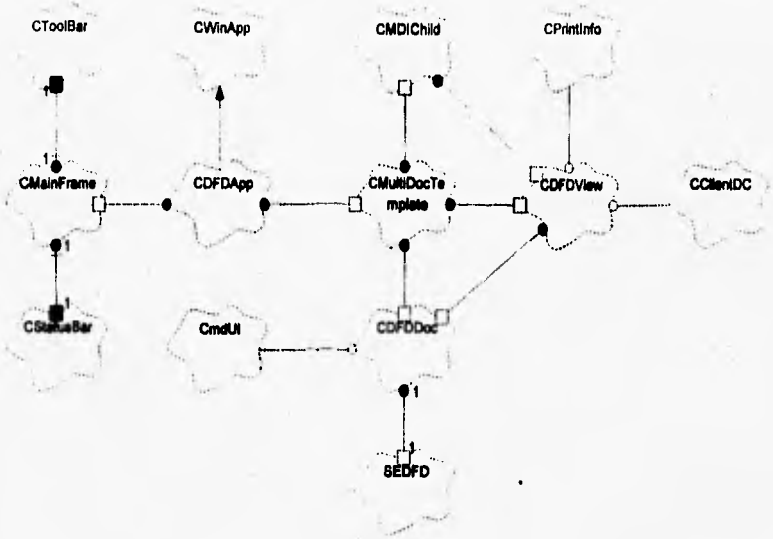


Figura 5.1 Diagrama de Clases Proporcionadas por Visual C++

5.2 Representación Interna de un Proyecto

En la figura 5.2 se muestra la estructura de clases de un proyecto, en donde un proyecto contiene fundamentalmente dos objetos: un diagrama de contexto (dc), representado por medio de la clase SEDC y un diagrama de flujo de datos (dfd), representado por medio de la clase SEDFD. Como se muestra en dicha figura, el proyecto contiene tanto al diagrama de contexto como al diagrama de flujo de datos por referencia. De igual manera dichos diagramas contienen por referencia al proyecto (proy).

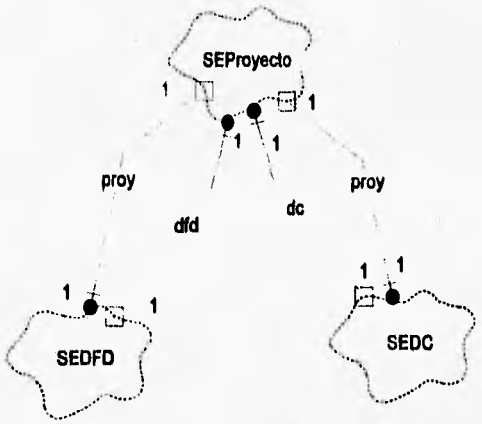


Figura 5.2 Diagrama de Clases del Proyecto

5.3 Representación Interna de un Diagrama de Flujo de Datos

La representación interna de un diagrama de flujo de datos se muestra en la figura 5.3, donde se observa que un objeto de la clase SEDFD está formado principalmente por una secuencia de versiones de diagrama de flujo de datos (versec) y contiene por referencia a la versión actual (currver), a la versión aprobada (apprver) y a la simbología por omisión (simdef) la cual se aplica en caso de que no se elija alguna otra.

La secuencia de versiones de diagrama de flujo de datos se representa por la clase GlyphSecuence, y está implementada como una lista ligada de nodos, en donde cada nodo contiene por referencia a un objeto de la clase Glyph. Glyph es una clase abstracta de la cual hereda la clase DFDVersion. Esta secuencia nos permite manejar diferentes versiones del diagrama de flujo de datos, en donde cada versión contiene por referencia al diagrama de flujo de datos al que pertenece (parent).

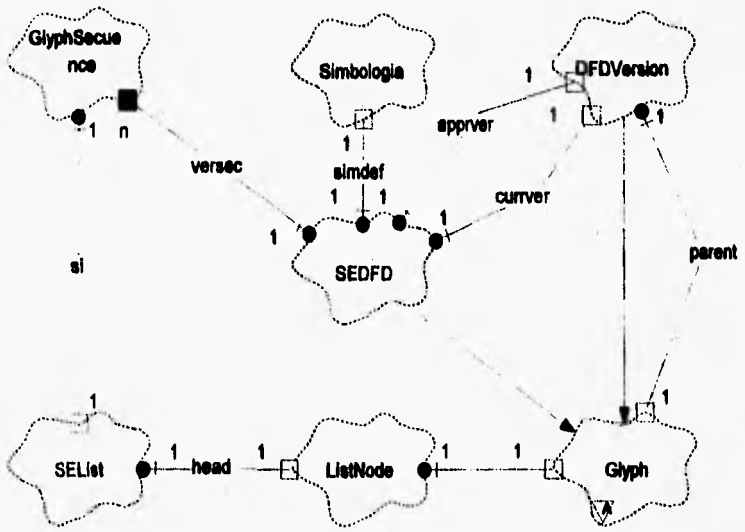


Figura 5.3 Diagrama de Clases del Diagrama de Flujo de Datos

Una versión de diagrama de flujo de datos se compone principalmente por:

- **Una escuencia de figuras (figsec)**
  Se utiliza para desarrollar el diagrama de flujo de datos en forma descendente, ascendente o combinada.
- **Una secuencia de figuras modificadas (figmod)**
  La cual nos permite saber cuales son las figuras que deberán ser almacenadas, ya que el almacenamiento se hará desde la versión y no por figura.
- **Una secuencia de procesos (procsec)**
  Sirve para verificar por versión que los procesos no se dupliquen
- **Una secuencia de almacenes (almasec), una secuencia de terminadores (termsec) y una secuencia de flujos (flujsec)**
  Las tres secuencias sirven para llevar el control de los elementos repetidos

También contiene por referencia a los objetos de la figura actual (currfig), la figura 0 (figura0) y la simbología propia de la versión (dfdgr). Esta estructura de clase se muestra en la figura 5.4

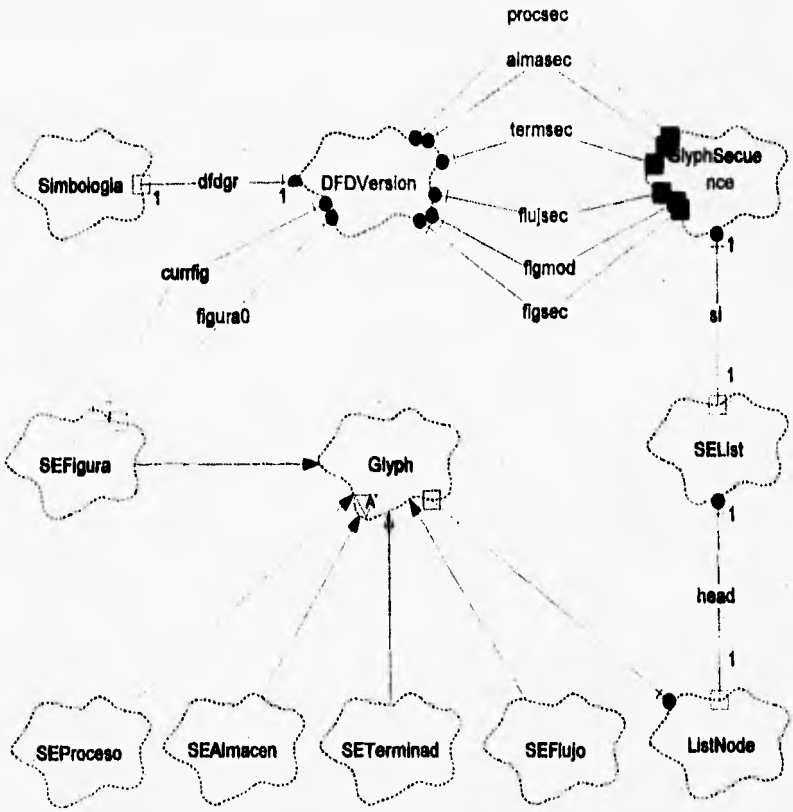


Figura 5.4 Diagrama de Clases de las Versiones de Diagrama de Flujo de Datos.

Todas las secuencias utilizadas en esta tesis se implementaron de la misma forma como se especificó en la clase SEDFD, por lo que no se presentará esta implementación en estructuras de clases subsecuentes.

5.4 Representación Interna de una Figura

Una figura se representa mediante la clase SEFigura y se compone fundamentalmente por una secuencia de procesos (procsec), una secuencia de almacenes (almasec), una secuencia de terminadores (termsec) y una secuencia de flujos (flujsec), además contiene

por referencia al elemento seleccionado (selobj) y a la versión a la que pertenece. En la figura 5.5 se puede observar la estructura de clase SEFigura.

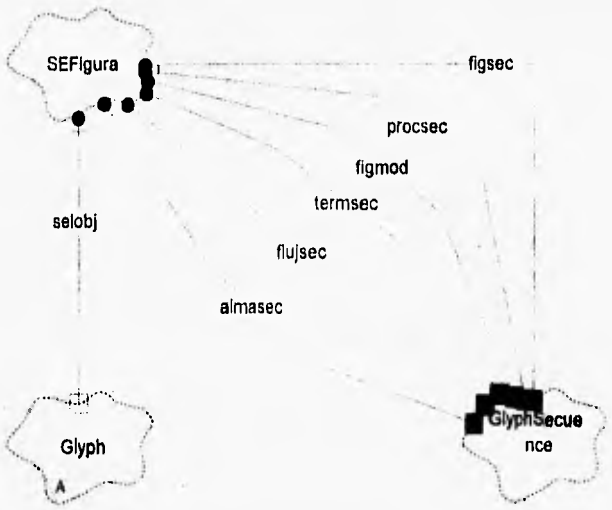


Figura 5.5 Diagrama de Clases de Las Figuras.

5.5 Representación Interna de un Diagrama de Contexto

En la figura 5.6 se presenta la representación interna de un diagrama de contexto, en donde la clase SEDC representa un diagrama de contexto; la cual tiene fundamentalmente una secuencia de almacenes, una secuencia de terminadores, una secuencia de flujos, un objeto de la clase SEProceso (proc), además contiene por referencia al elemento seleccionado (selobj).

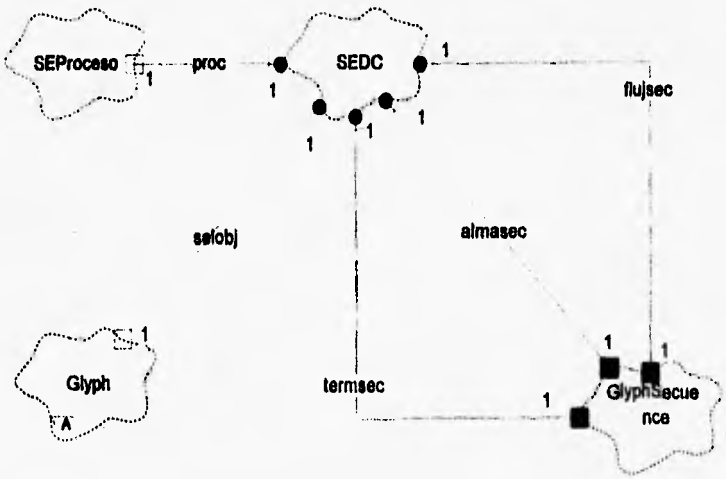


Figura 5.6 Diagrama de Clases del Diagrama de Contexto.

Capítulo 6

Evaluación de Resultados

6.1 Estrategias de Diseño e Implementación

Como ya se mencionó en el primer capítulo, este sistema forma parte de un proyecto para el desarrollo de un CASE. Un CASE debe integrar todas las etapas del desarrollo de software, en donde cada etapa tiene sus propios instrumentos de desarrollo, dado que estos instrumentos pueden evolucionar en forma independiente se diseñó y programó el sistema de manera que cada instrumento pueda modificarse y ampliarse en forma autónoma. Para obtener esto se implementaron tres programas:

- El programa de proyecto.
- El programa de diagrama de flujo de datos.
- El programa de diagrama de contexto.

Observando que existen funcionalidades comunes en las diversas herramientas de desarrollo de un CASE, se hace posible el volver a utilizar el diseño y el código. En este sistema en particular se encontraron similitudes tanto en el aspecto gráfico como en algunas reglas de verificación, entre el diagrama de flujo de datos y el diagrama de contexto. Por tal motivo se diseñaron algunas clases que se emplearon en estas herramientas y que también podrán ser utilizadas en otras herramientas que se creen posteriormente para completar el desarrollo del CASE.

La codificación se llevo acabo utilizando la herramienta de desarrollo Visual C++. La cual proporciona una estructura de clases básica para la aplicación a desarrollar. Sin embargo

para obtener un sistema que permita la portabilidad a otras plataformas de hardware como de software se tomaron las siguientes consideraciones:

- Las clases de nuestra aplicación se localizan en archivos independientes de los proporcionados por Visual C++.
- Ninguna de las clases de nuestra aplicación hereda de las clases del framework de Visual C++.
- Dado que el lenguaje de programación C++ es un estándar de factor, hará más sencilla la portabilidad.
- Se trató de no mezclar objetos de las clases proporcionadas por el MFC (Microsoft Fundation Class) de Visual C++, con los objetos de las clases creadas para nuestra aplicación.

6.2 Características del Software Utilizado

Visual C++ permite desarrollar sistemas bajo ambiente MSWindows, utilizando el lenguaje de programación C++. Para este fin Visual C++ proporciona herramientas como el AppWizard, el AppStudio y ClassWizard.

La herramienta AppWizard nos proporcionó un esqueleto de la aplicación sobre del cual pudimos construir el código específico de nuestra aplicación. También nos ofreció opciones que nos permitió incorporar la barra de herramientas y la barra de estados para cada módulo del sistema.

El AppStudio nos facilitó diseñar la interface con el usuario de la aplicación y nos permitió crear los recursos de aplicación tales como menús, cajas de diálogo, teclas aceleradoras, iconos, cursores y cadenas de caracteres.

Por último el ClassWizard nos hizo posible realizar de manera más sencilla la conexión del código con los objetos de la interface al usuario tales como los menús, cajas de diálogo, etc.

Entre las desventajas observadas se encuentra la generación automática de clases las cuales no se encuentran documentadas. Como también la falta de claridad al indicar los errores en el sistema. En cuanto su ambiente de trabajo hace un mal manejo de las ventanas dejando abiertas algunas que pertenecen a otro proyecto y que crean confusión. La ayuda en línea proporcionada no es suficientemente explícita.

6.3 Ampliaciones al Sistema

Para que este sistema alcance la funcionalidad requerida para que sea utilizado por el usuario final necesita de las siguientes ampliaciones:

- En el sistema se llevaron a cabo algunos mecanismos para la verificación de consistencia, sin embargo, hizo falta realizar la verificación de consistencia entre los diversos niveles del diagrama de flujo de datos.

- Tanto en el diagrama de flujo de datos como en el diagrama de contexto se permiten utilizar sólo dos tipos de simbología, pero dado a que existen diferentes simbologías para satisfacer los requerimientos de los usuarios es necesario disponer de más simbologías las cuales se pueden añadir de manera sencilla en trabajos posteriores.

- Debido a que el sistema está planteado para usar una base de datos la cual será común para todo el CASE. El desarrollo de esta base de datos, sería una aplicación a este sistema. Por tal motivo todas las operaciones relacionadas con el almacenamiento de información no se implementaron en este trabajo.

Para poder completar la etapa del análisis de requerimientos siguiendo la metodología de Eduard Yourdon se podrían desarrollar las siguientes herramientas de modelado:

- El Diagrama Entidad-Relación.
- El Diagrama de Transición de Estados.
- El Diccionario de Datos.
- La Especificación de Procesos.
- Balanceo de Modelos.

Algunas partes del diseño y del código de nuestro sistema se podrían reutilizar en el desarrollo de estos nuevos módulos, básicamente las operaciones gráficas.

Ya que un CASE debe cubrir todas las etapas del desarrollo de software, será necesario añadir herramientas para cada una de las etapas y una herramienta más para el control y administración de proyectos.

6.4 Conclusiones

Llevar a cabo correctamente cada una de las etapas del desarrollo de software, aplicando las metodologías adecuadas, es sumamente importante para obtener un sistema de calidad. Y si contamos con una herramienta computarizada que nos ayude a aplicar estas metodologías de una manera más sencilla, proporcionando al profesional responsable de esta tarea, mecanismos de verificación automática de consistencia y manejo de gráficos, la labor de desarrollar un sistema resulta menos compleja.

El uso de estas herramientas computarizadas para el desarrollo de software, provee la flexibilidad necesaria para realizar las modificaciones pertinentes de acuerdo a los requerimientos del usuario. La presentación de los diversos modelos que se obtienen para mostrarlos al usuario, es excelente. Los tiempos y costos se minimizan, haciendo a un lado el gasto de ocupar personal en tareas de documentación repetitivas y laboriosas. Además estas herramientas mantienen el control del proyecto y coordinan las actividades del personal del equipo de trabajo destinado a realizar el proyecto.

El desarrollo de una herramienta CASE requiere una serie de recursos humanos y tecnológicos sumamente especializados, debido a que se trata de una tarea bastante compleja y de amplios alcances, implicando varios años/hombre de trabajo. Por lo cual el

propósito de este trabajo de tesis fue desarrollar un sistema que sirviera como base para posteriormente desarrollar un sistema funcional para el usuario final, ofreciendo en esta primera etapa manejo de gráficos y manejo de diferentes estrategias de desarrollo.

Todas las ventajas que presenta una herramienta computarizada, lleva a los profesionales del área de computación, a ocupar su tiempo en tareas creativas y de desarrollo intelectual, haciendo a un lado el trabajo monótono.

El desarrollo de esta tesis nos permitió reafirmar la importancia que tiene la preparación de ingenieros comprometidos con el desarrollo de México, que tengan la capacidad de resolver los problemas que aquejan a la sociedad, con los recursos disponibles.

La tecnología en al área de computación evoluciona muy rápidamente, lo cual exige a los profesionista de esta área una preparación constante. La formación académica adquirida en la Facultad de Ingeniería nos provee de las bases necesarias para poder entender las nuevas tecnologías.

Apéndice A

Lenguaje de Programación C++

En este apéndice nos enfocaremos a describir algunas de las principales características propias del lenguaje de programación C++ que nos fueron útiles para el desarrollo de la tesis. El lenguaje de programación C++ es un lenguaje orientado a objetos desarrollado por el doctor Bjarne Stroustrup, en los laboratorios Bell de la AT&T.

A.1 Clases

Una clase contiene dos tipos de componentes: las variables miembro y las funciones miembro. Los valores de las variables miembro asociadas a un objeto definen el estado del objeto, y una función miembro define el comportamiento del objeto (es decir, las acciones que el objeto pueda ejecutar). El concepto de clases provee una herramienta bastante poderosa para crear nuevos tipos de datos, donde una clase también puede verse como un tipo de datos definido por el usuario.

A.2 Objetos

Un objeto es una instancia de una clase, es decir, es una variable de un tipo definido por el usuario. Las funciones miembro manipulan variables y pueden llamarse entre sí, dentro de ciertos alcances sintácticos, estos pueden ser:

Global al programa:	Las variables son accesibles en cualquier lado.
Global a un archivo:	Sólo se les puede usar en el archivo donde están declaradas.

Locales: Las variables sólo pueden emplearse en la función que las define

Para crear un objeto en C++ primero se debe definir su forma general utilizando la palabra *class*. Sintácticamente, una clase y una estructura son similares. Una vez que se ha definido una clase, es posible crear un objeto de ese tipo usando el nombre de dicha clase, es decir, el nombre de la clase se utiliza como un nuevo especificador de tipos de datos.

A.3 Reglas de Alcance y Funciones Miembro

En C++ cuando se codifican las funciones miembro es necesario indicar precisamente a qué clase pertenece dicha función., utilizando el operador de resolución de alcance (: :).

```
Tipo   Clase::Nombre_de_la_función
       (
       Código_de_la_función
       )
```

En C++ un objeto se puede pasar a una función de la misma manera que los datos. Los objetos se pasan a las funciones en la forma convencional de paso de parámetros (llamada por valor -"call by value"); esto quiere decir que una copia del objeto (del valor del objeto) se pasa a la función y no el objeto (la dirección del objeto), por lo que es posible hacer modificaciones dentro de la función sin afectar el objeto usado en la llamada a la función.

En C++ se puede hacer referencia a un objeto directamente o mediante apuntadores, siendo esta una de las características más sobresalientes. Cuando se accesa a un objeto mediante el objeto mismo, se utiliza el operador punto (.), mientras que para accesar un elemento específico de un objeto utilizando un apuntador se emplea el operador flecha (->). Para declarar un apuntador a objeto, la declaración sintáctica es la misma que en los otros tipos de datos.

La dirección del objeto se obtiene mediante el uso del operador de dirección (&), de la misma forma que una dirección se obtiene en cualquier tipo de datos; y sigue siendo válida la aritmética de apuntadores: al incrementar un apuntador, éste se mueve al siguiente objeto (hacia adelante o atrás).

A.4 Funciones en línea

C++ provee la palabra reservada *inline* para identifcar una función en línea y se utiliza antes de la declaración de una función. Esta palabra causa una copia nueva de la función para insertarse en cada lugar donde se llame. Si se llama una función en línea desde 20 lugares en el programa, el compilador inserta 20 copias del código de la función en el archivo .EXE. Insertando copias individuales del código de las funciones elimina el gasto de las llamadas a función, así el programa se ejecuta más rápido. Sin embargo, teniendo múltiples copias del código de una función se hace el programa más grande. La función en línea se debe usar sólo cuando la función insertada sea muy pequeña o se llame de pocos lugares. Cualquier

función que se implemente dentro de una clase será automáticamente a una función en línea, y no es necesario preceder su declaración con la palabra *inline*.

A.5 Sobrecarga de funciones

A pesar de que en general una función efectúa un único trabajo (por ejemplo, los constructores se encargan de la inicialización de un objeto), ésta puede ser sobrecargada definiendo diferentes acciones para un mismo nombre de función.

A.6 Funciones Constructoras y Destructoras

Cuando se usan objetos, es muy común que requieran algún tipo de inicialización. En C++ es posible inicializar objetos cuando estos se crean, esta inicialización se ejecuta automáticamente a través de la función denominada constructora. La función constructora, o constructor, es una función miembro de la clase, con la peculiaridad de que tiene el mismo nombre que ésta y no regresa ningún valor. El constructor se llama cuando un objeto se crea (esto es, cuando la declaración del objeto se ejecuta), así que cuando un objeto es local, la función constructora se invoca cada vez que la declaración del objeto se encuentre.

Debido a que las constructoras no regresan ningún valor, no cuenta con un especificador de tipo de retorno(int, float, etc.). Si una función constructora requiere argumentos, éstos deben indicarse al menos en principio con el especificador de tipo. Las funciones constructoras adicionalmente siguen las mismas reglas para los argumentos como lo hacen las funciones sobrecargadas ("overloaded"), y si en las constructoras los argumentos difieren en sus tipos, el compilador puede seleccionar para cada caso.

Debido a que en algunas circunstancias un objeto necesita ejecutar alguna acción o acciones cuando es destruido (lo que ocurre con las variables locales al alcanzar el fin del bloque donde se definen y con las variables globales al terminar el programa), se cuenta con las funciones destructoras o destructores. En C++ la función destructora tiene el mismo nombre que la constructora, salvo que la destructura se precede por una tilde. El uso de constructores y destructores permite disminuir los errores ocasionados por inicializaciones incorrectas, y facilita la limpieza de las estructuras que ya no se necesitan (liberar memoria).

A.7 Operadores para el Manejo de Memoria Dinámica

C++ provee el operador *new* para asignar memoria dinámica. Al operador *new* se le dá un argumento que le indique la cantidad de memoria que debe asignar, es decir, la clase del objeto que se va ha asignar, y el operador *new* automáticamente llama al constructor de la clase para inicializar la memoria asignada. El operador *new* retorna una apuntador de la clase que se dió como argumento.

El operador *delete* es la contraparte del operador *new*, debido a que desasigna la memoria obtenida por el operador *new*. El operador *delete* automáticamente llama al destructor del

objeto antes de que este desasigne la memoria. Sólo se puede aplicar *delete* a apuntadores que fueron retornados por *new*, y sólo se pueden eliminar una vez.

A.8 Herencia

Como resultado de aplicar las características de herencia en C++ es posible:

- Añadir datos y código a una clase sin tener que cambiar la clase original.
- Reducir el código.
- Cambiar el comportamiento de una clase.

En C++ es posible organizar elementos de una clase en tres tipos de categorías:

- *Public:* Cuando los elementos de una clase se declaran como públicos los pueden accesar cualquier función en el programa.
- *Private:* Cuando los elementos de una clase se declaran como privados sólo los pueden accesar funciones miembro de la clase.
- *Protected:* protegidos y Cuando los elementos de una clase se declaran como sólo los puenden accesar funciones miembro de la clase funciones miembro de clases derivadas.

C++ soporta dos tipos de herencia: la herencia simple y la herencia múltiple. En este caso sólo describiremos la herancia simple ya que la herencia múltiple no se útilizó.

A.8.1 Herencia Simple

En la herancia simple, una clase derivada recibe únicamente los atributos correspondientes a una sola clase base, con lo que se crea una jerarquía de clases en forma de árbol. Cuando una clase hereda a otra, si los elementos de la clase base están declarados como privados, son entonces inaccesibles aún para la clase derivada. Cuando los elementos de una clase son protegidos, se restringe el acceso a otras funciones que no sean funciones miembro de esa clase (como en private), pero el acceso se hereda a las clases derivadas. Si el acceso se define como privado, éste no se herede.

La forma general para declarar una clase derivada es la siguiente:

```
class  Nombre_de_la_clase : Tipo_de_acceso  Clase_base
{
        public:
                variables_y_ funciones_públicas
        protectec:
                variables_y_funciones_protegidas
        private:
                variables_y_funciones_privadas
};
```

en donde el tipo de acceso puede ser privado o público (si se omite, se asume como privado). Si el tipo de acceso es público, los elementos públicos y protegidos de la clase base se heredan como públicos y protegidos respectivamente en las clase derivada, si el tipo de acceso es privado los elementos públicos y protegidos se heredan como privados en la clase derivada.

A.8.2 Polimorfismo

La habilidad para llamar funciones miembro para un objeto sin especificar el tipo exacto del objeto se conoce como polimorfismo. La palabra polimorfismo significa la habilidad para asumir varias formas, refiriendose a la característica para tener una sola declaración que invoque varias funciones diferentes.

A.9 Funciones Virtuales

Una función virtual es una función miembro que se espera se redefina en clases derivadas. Cuando se llama una función virtual a través de una apuntador a la clase base, la versión de la función de la clase derivada se ejecuta. Esto es precisamente el comportamiento opuesto de las funciones miembro ordinarias.

En la redefinición de una función virtual en una clase derivada existen las siguientes restricciones:

- Los prototipos de las funciones virtuales deben ser iguales, a diferencia de las funciones sobrecargadas (en donde el tipo regresado y el número y tipos de parámetros son diferentes en cada función). De otra manera la función simplemente es considerada sobrecargada y su naturaleza virtual desaparece.
- Las funciones virtuales deben ser funciones miembro de la clase.
- Si sólo los tipos regresados por la función difieren, entonces ocurre un error (una ambigüedad).
- Una función virtual puede ser una función destructora, pero no puede ser constructora.

Estas diferencias y restricciones entre las sobrecarga de funciones normales y la sobrecarga de funciones virtuales hacen que el término sobremontado ("overriding") se utilice para volver a describir la definición de funciones virtuales.

Una función virtual es declarada como virtual dentro de la clase base precediendo su declaración con la palabra *virtual*. Cuando una función virtual se vuelve a definir en una clase derivada, la palabra *virtual* puede no anteponerse en su declaración. Una vez que una función ha sido declarada como virtual, permanece así, no importando a cuantas clases derivadas pase.

Cuando en una clase derivada no se sobremonta una función virtual, entonces se usa la versión de la función que está en la clase base. De manera más general, cuando en una clase no se sobremonta una función virtual, C++ usa la primera definición que encuentra, en

orden inverso en la derivación de herencia. Es muy claro que el polimorfismo mediante funciones virtuales es de la forma en que la clase base dictamina las interfaces generales que cualquier objeto de una clase derivada debe tener, pero permite a la clase derivada definir una nueva función miembro. Así, cuando se usa correctamente, la clase base provee todos los elementos que las clases derivadas pueden usar, además de la estructura de aquellas funciones que en la clase derivada deben implementarse de otra manera. Con esto se logra que el programador maneje mayor complejidad, ya que se tienen interfaces consistentes con múltiples implementaciones.

A.10 Funciones Virtuales Puras

Anteriormente se expuso que, cuando una función no está sobremontada en una clase derivada y un objeto de esa clase la llama, la función definida en la clase base es usada. Sin embargo, en muchas circunstancias no existe una definición manejable de la función virtual en la clase base, por lo que en estos casos, es conveniente usar las funciones virtuales puras.

Una función virtual pura es una función declarada en la clase base que no tiene definición relativa a esa clase, por lo que cualquier tipo derivado debe definir su propia versión. La forma general de las funciones virtuales puras es la siguiente:

virtual tipo nombre_de_la_función (lista de parámetros) = 0

El único propósito de esta función es proveer una interface de polimorfismo a las clases derivadas. No se puede declarar ninguna instancia de una clase con una función que es declarada como función virtual pura. Esta restricción es necesaria para prevenir cualquier llamada de una función virtual pura a un objeto.

A.11 Clases Abstractas

Una clase que define funciones virtuales puras se conoce como una *clase abstracta*, porque no se puede declarar ninguna instancia de ella. Es legal, sin embargo, declarar apuntadores a una clase abstracta.

Si una función virtual pura no se vuelve a definir en la clase derivada ésta pasa a ser una clase abstracta. Esto no sucede con funciones virtuales ordinarias, porque cuando una clase derivada omite una definición de una función virtual ordinaria, utiliza la versión de la clase base.

Apéndice B

Programación para Ms-Windows con Visual C++

Este apéndice contiene una descripción de los elementos principales que integran una aplicación desarrollada con Visual C++ para MS-Windows.

La Librería de Clases de la Fundación Microsoft (The Microsoft Foundation Class Library) nos permite desarrollar aplicaciones para Microsoft Windows, brindando un conjunto de clases de C++ que forman una estructura de trabajo (framework), la cual provee los componentes esenciales de una aplicación para el ambiente gráfico de Windows. El propósito de la estructura de trabajo es reducir el esfuerzo requerido para diseñar e implementar aplicaciones para Windows.

Los conceptos centrales de la estructura de trabajo son los siguientes :

- **El Objeto de la Aplicación**
  Es corazón de la aplicación para Windows, maneja una lista de documentos y envía comandos a otros objetos en el programa.

- **El Documento**
  Es la unidad de datos que el usuario trabaja. El documento es responsable de almacenar mantener y recuperar sus datos. Una aplicación puede manejar un solo documento (single document interface o SDI) o múltiples documentos (multiple document interface o MDI)

- **La Vista**

  El usuario interactua con el documento a través de una vista. La vista despliega los datos del documento y recibe los mensajes del teclado y del mouse. Los mensajes los traduce en acciones de selección y edición.

- **Objetos en la Interface con el Usuario**

  Tales como menús y botonas, envían mensajes al documento, vistas y otros objetos en la aplicación para ser procesados.

Para llevar acabo una aplicación con la Librería de Clases de la Fundación Microsoft existen herramientas de apoyo tales como: AppWizard, AppStudio y ClassWizard.

AppWizard

Crea un esqueleto de la aplicación sobre del cual se puede construir el código especifico de la nueva aplicación. También el AppWizard ofrece numerosas opciones que permiten incorporar barra de herramientas, impresión e impresión preliminar, controles VBX, etc., en los archivos que crea, dejando al usuario la tarea de implementar estos controles.

AppStudio

Permite diseñar la interface con el usuario de la aplicación y crear los recursos de la aplicación: menús, cajas de diálogo, controles custom, teclas aceleradores, mapas de bits, iconos, cursores y cadenas de caracteres.

ClassWizard

Permite llevar acabo la conexión del código con los objetos de la interface al usuario tales como los menús, cajas de diálogo, etc.

Apéndice C

Archivos del Sistema

Este apéndice presenta los archivos que forman parte del sistema, los cuales fueron divididos en dos grupos: los archivos proporcionados por Visual C++ y los archivos que fueron desarrollados por nosotras para proporcionar la funcionalidad deseada del sistema.

C.1 Los Archivos Generados por Visual C++

- ***Para el Programa de Proyecto***

  proy.h
  Contiene la declaración de clase de la aplicación.

  proy.cpp
  Contiene las funciones miembro aplicables al objeto raíz de la aplicación.

  proydoc.h
  Contiene la declaración de la clase para el manejo de documentos de la aplicación.

  proydoc.cpp
  Tiene las funciones relacionadas con el documento.

  proyview.h
  La clase que maneja las vistas se encuentra declarada en este archivo.

proyview.cpp
Le funciones que permiten el manejo de las vistas estan contenidas en este archivo.

mainfrm.h
Contiene la declaración de los recursos del marco principal de la aplicación, tales como el statusbar y toolbar.

mainfrm.cpp
Tiene las funciones relacionadas con el marco principal de la aplicación.

proydlg.h
La clase relacionada con la caja de diálogo que maneja el proyecto, se encuentra en este archivo declarada.

proydlg.cpp
Las funciones relacionadas con la caja de diálogo se encuentran en este archivo

- **Para el Programa de Diagrama de Flujo de Datos**

dfd.h
Contiene la declaración de clase de la aplicación.

dfd.cpp
Contiene las funciones miembro aplicables al objeto raiz de la aplicación.

dfddoc.h
Contiene la declaración de la clase para el manejo de documentos del diagrama de flujo de datos.

dfddoc.cpp
Tiene la definición de las funciones miembro relacionadas con el documento.

dfdview.h
La clase que maneja las vistas se encuentra declaradas en este archivo.

dfdview.cpp
La funciones que permiten el manejo de las vistas estan contenidas en este archivo.

mainfrm.h
Contiene la declaración de los recursos del marco principal de la aplicación, tales como el statusbar y toolbar.

mainfrm.cpp
Tiene las funciones relacionadas con el marco principal de la aplicación.

infodlg.h
Este archivo contiene la declaración de las clases relacionadas con las cajas de diálogo del diagrama de flujo de datos.

infodlg.cpp
Las funciones relacionadas con las cajas de diálogo se definen en este archivo.

- **Para el Programa de Diagrama de Contexto**

dc.h
La declaración de la clase de la aplicación raíz se encuentra en este archivo.

dc.cpp
Contiene las funciones miembro aplicables al objeto raíz de la aplicación.

dcdoc.h
Este archivo contiene la declaración de la clase para el manejo de documentos.

dcdoc.cpp
La definición de las funciones miembro relacionadas con el documento se encuentran es este archivo.

dcview.h
Se localiza la declaración de la clase que maneja las vistas en el diagrama de contexto.

dcview.cpp
La definición de funciones que permiten el manejo de las vistas estan contenidas en este archivo.

mainfrm.h
Contiene la declaración de los recursos del marco principal de la aplicación, tales como el statusbar y toolbar.

mainfrm.cpp
Tiene la definición de las funciones relacionadas con el marco principal de la aplicación.

infodlg.h
Este archivo contiene la declaración de las clases relacionadas con las cajas de diálogo del diagrama de flujo de datos.

infodlg.cpp
Las funciones relacionadas con las cajas de diálogo se definen en este archivo.

C.2 Los Archivos Propios del Sistema

- **Para el Programa de Diagrama de Flujo de Datos**

glyph.h
Contiene la declaración de las clases que manejan las esculturas.

glyph.cpp
Contiene las funciones que manejan las esculturas.

interfac.h
Este archivo tiene las declaración de las clases relacionadas con la interface.

interfac.cpp
Tiene las funciones relacionadas con las clases de la interface.

sedfd.h
Contiene la declaración de las clases que integran un diagrama de flujo de datos.

sedfd.cpp
El archivo contiene las funciones relacionadas con las clases forman un diagrama de flujo de datos.

sutility.h
La declaración de las clases de utileria del sistema se encuentra en este archivo.

sutility.cpp
La definición de las funciones de utileria estan localizadas en este archivo.

- **Para el Programa de Digrama de Contexto**

glyph.h
Contiene la declaración de las clases que manejan las esculturas.

glyph.cpp
Contiene las funciones que manejan las esculturas.

interfac.h
Este archivo tiene las declaración de las clases relacionadas con la interface.

interfac.cpp
Tiene las funciones relacionadas con las clases de la interface.

sedc.h
Contiene la declaración de las clases que integran un diagrama decontexto.

sedc.cpp
El archivo contiene las funciones relacionadas con las clases forman un diagrama de contexto.

sutility.h
La declaración de las clases de utileria del sistema se encuentra en este archivo.

sutility.cpp
La definición de las funciones de utileria estan localizadas en este archivo.

Bibliografía

Object Oriented Design with Applications
Grady Booch
1991 The Benjamin/Cummings Publishing Company, Inc.

Metodlogías de Desarrollo
Producción Automática de Software con Herramientas CASE
Antonio López-Fuensalida
1991 Macrobit Editores, S.A. de C.V.

Análisis Estructurado Moderno
Edward Yourdon
1993 Prentice-Hall Hispanoamericana, S.A.

Reference Volume I
Class Library Reference
For the Microsoft Foundation Class Library
Microsoft Visual C++ Version 1.

Programmer's Guides
C++ Tutorial
Class Library User's Guide
Programming Techniques
Microsoft Visual C++ Version 1.

Visual C++
Funciones y Aplicaciones
1994 McGraw-Hill