UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Unidad Académica de los Ciclos Profesional y de Posgrado del Colegio de Ciencias y Humanidades

Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas

"UNA APLICACION DE LA INGENIERIA DE SOFTWARE AL DESARROLLO DE UN SISTEMA INTEGRADO PARA EL CONTROL ESCOLAR".

T E S I S

Que para obtener el Grado de:

Maestro en Ciencias de la Computación

P r e s e n t a :

Josefina Lourdes De la Mora Jiménez

MEXICO D. F. Marzo de 1990.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

INDICE DE FIGURAS

CAPITULO 1. INTRODUCCION

1.1 EL DESARROLLO DEL SOFTWARE, ANTECEDENTES Y ESTADO ACTUAL.

Una de las áreas de investigación y crecimiento que han adquirido mayor importancia en los últimos años, dentro del campo de la Ciencia de la Computación, es la Ingeniería de Software, la cual representa un intento de formalizar el desarrollo de sistemas computarizados.

A partir de los años 60, con la tercera generación de computadoras, el Hardware ha adquirido una relativa madurez estableciéndose muy bien las técnicas de diseño y los métodos de manufactura. Desafortunadamente en el desarrollo del Software todavía se sufre de lo que Maquiavelo dijo hace cuatrocientos cincuenta años: "No hay nada más difícil de adquirir, más peligroso de dirigir o más incierto en sus resultados, que encaminar el comienzo de un nuevo ordenamiento de cosas...".

Como una respuesta a la crisis de los 60's, en las conferencias de Alemania e Italia de 1968 y 1969, respectivamente, se vió la necesidad de crear un conjunto de técnicas de programación, a lo cual se le llamó **INGENIERIA DE SOFTWARE.**

Es muy difícil unificar criterios al definirla, debido a que, a diferencia de las otras ramas de la ingeniería, no existen leyes físicas que la rijan. Para unos autores la Ingeniería de

Software es un arte [Dennis 75], otros la ven desde un plano económico y administrativo [Boehm 81], hay quienes le dan mayor importancia al tamaño de los sistemas [Parnas 74] o al aspecto matemático y científico de los mismos [Pomberger 84]. Se puede considerar que la definición dada por Ratcliff [Ratcliff 87] encierra la idea principal de la mayoría de los autores:

> "Ingeniería de Software es la utilización disciplinada de un conjunto sistemático y coherente de principios, herramientas y técnicas dentro de una estructura organizada y planeada, cuyo objetivo es obtener dentro del tiempo programado, el Software de calidad con un desarrollo efectivo en costo para un rango diverso de ambientes y aplicaciones no triviales."

La definición anterior permite observar que la Ingeniería de Software es interdisciplinaria, ya que utiliza fundamentos de la Ciencia de la Computación, Administración, Economía, Comunicación e Ingeniería. De ella se espera que proporcione métodos, herramientas y normas para hacer posible la solución a los problemas que se presentan en la producción de Software.

El principal objetivo de las técnicas de Ingeniería de Software es proporcionar las bases para la planeación, análisis, diseño, implementación, validación y mantenimiento de los programas, así como un conjunto de componentes que documenten cada paso mostrando su proceso.

1.2 OBJETIVO DE LA TESIS

El presente trabajo tiene como objetivo aplicar en forma experimental algunas de las técnicas ofrecidas por la Ingeniería de Software para la realización de un determinado sistema, a fin de observar si con la metodología de desarrollo aplicada es

posible obtener los resultados esperados o ésta podría ser mejorada de alguna forma.

El proyecto que se implementó, siguiendo los lineamientos de la Ingeniería de Software, es un Sistema Integrado para el Control Escolar (SICE). Este puede considerarse de tamaño mediano, lo cual justifica la aplicación de una metodología sistemática para su desarrollo, con la posibilidad de ser realizado por una sola persona.

1.3 ESTRUCTURA DE LA TESIS

Este trabajo puede dividirse en dos bloques. El primero establece las bases teóricas proporcionadas por la Ingeniería de Software para el desarrollo de sistemas, incluidas de forma introductoria en el capítulo 2, "Introducción a la Ingeniería de Software".

El segundo bloque comprende la aplicación de la teoría en la obtención de SICE. Para ello en el capítulo 3, "Desarrollo de un Sistema Integrado para el Control Escolar", se hace la descripción de la metodología aplicada, detallando cada paso. El capítulo 4, "Uso de SICE", proporciona tres ejemplos prácticos de la forma en que se utiliza el sistema.

El capítulo 5 corresponde a las conclusiones obtenidas de la investigación y aplicación de la tesis. El capítulo 6 contiene las "Referencias Bibliográficas" utilizadas en el proyecto.

Por último, se incluyen dos anexos: el primero proporciona el manual para el usuario y el segundo es una copia del artículo presentado en el Primer Foro de Avances de Investigación, IIMAS, UNAM, México, D.F., mayo 3, 1989.

CAPITULO 2. INTRODUCCION A LA INGENIERIA DE SOFTWARE

La disciplina de la Ingeniería de Software se basa en dos objetivos fundamentales:

- perfeccionar el proceso de la producción de Software y
- mejorar la calidad del Software.

Se han definido un gran número de métodos, técnicas y herramientas encauzadas a alcanzar dichos objetivos.

2.1 CICLO DE VIDA DEL DESARROLLO DEL SOFTWARE

Un concepto unificador importante en la Ingeniería de Software es el **CICLO DE VIDA DEL DESARROLLO DEL SOFTWARE**, el cual describe la secuencia de fases que lo componen y su evolución. Estas fases típicamente son: análisis de requerimientos, especificación de requerimientos, diseño, implementación, prueba, documentación y mantenimiento. Es importante ir de una fase a otra, de adelante hacia atrás, para conocer el progreso del trabajo en diferentes puntos. Así, los problemas pueden identificarse en un estado temprano del mismo y tomar acciones correctivas.

A continuación se hace una descripción con cierto detalle de

cada una de las fases y los recursos con que se cuenta para llevarlas a cabo de manera óptima.

2.1.1 ANALISIS DE REQUERIMIENTOS

El primer paso, al proponer cualquier tipo de proyecto de Ingeniería, es conocer con profundidad el problema a resolver para poder trazar una serie de posibles soluciones con el fin de encontrar y aplicar la mejor alternativa. En la Ingeniería de Software ésto se obtiene mediante el análisis de requerimientos, cuyo objetivo es "determinar un conjunto completo y realizable de propiedades y limitaciones para el Software pretendido, con el que quedarán satisfechas las necesidades y objetivos del usuario" [Ratcliff 87]. La forma de realizarlo va a depender de si se va a automatizar un sistema por primera vez o si va a modificarse uno ya existente.

Si se trata de elaborar un proyecto totalmente nuevo, del cual aún, no se tiene una especificación clara, es necesario primero, analizar la configuración del sistema actual para determinar los puntos a resolver dentro del mismo. Los pasos a seguir son:

a) Delimitación del sistema. Se debe decidir qué partes del sistema se van a considerar y cuáles serán excluidas. Esto sitúa la aplicación y ayuda a evitar las modificaciones una vez que se inicie el desarrollo del sistema. La delimitación permite que de la totalidad se llegue al sistema particular que se va a analizar y procesar.

b) Análisis del sistema. Constituye la esencia del análisis de requerimientos. Es común que el usuario, al principio, no sepa con claridad qué quiere obtener o que reduzca sus necesidades al definirlas y más tarde las aumente; ésto da como resultado un

sistema de baja calidad. El analista, para obtener flexibilidad en el sistema, debe distinguir detenidamente el problema, el área del mismo, las posibilidades de expansión, los puntos débiles, el flujo de información, etcétera. Por lo tanto debe ser cuidadoso y sistemático.

Se han propuesto diferentes metodologías para realizarlo, algunas como la de **Teoría General de Sistemas** (GST), han tenido mucha influencia a nivel teórico, pero por su generalidad resultan totalmente ineficientes cuando se tratan de aplicar; otras requieren de la participación activa del usuario en éste aspecto.

La metodología o enfoque más usado es el llamado **tradicional** basado en el análisis del sistema existente en términos de las funciones que realiza. Los aspectos a analizar propuestos son: la estructura, el problema, las comunicaciones, los documentos, los datos, el flujo y los puntos débiles.

Otra forma de llevar a cabo esta fase es a través del método de **Análisis de datos**, el cual se basa en el concepto de que los bloques fundamentales que forman un sistema son datos; éstos se consideran estáticos, pero se puede dar una variedad de **vistas** o **aplicaciones** de ellos. Por lo tanto, es bastante adecuado para aplicaciones donde se utiliza el concepto de base de datos, ya que una vez definidos los datos se especifican las diferentes aplicaciones funcionales de los mismos.

c) Descripción del sistema. Su propósito es ordenar, estructurar y explicar los resultados obtenidos en la delimitación y análisis del sistema, de forma que éstos puedan ser representados en una descripción cerrada y completa.

La elección del procedimiento adecuado para llevar a cabo el análisis de requerimientos depende del tipo, tamaño y metas del proyecto, así como de la experiencia y conocimientos de quien lo realice.

Entre los procedimientos más comunes se encuentran los cuestionarios, en los que el analista hace preguntas concretas a los empleados del área en cuestión; ésto supone un amplio conocimiento del tema por parte del analista para poder elaborar encuestas que le den la información necesaria.

Otra forma de obtener información precisa es solicitar reportes a especialistas; sin embargo, no es fácil encontrar a las personas capacitadas para elaborar este tipo de reportes.

Las entrevistas individuales o por grupo son otra fuente de información, más flexible pero más difícil de documentar.

2.1.2 ESPECIFICACION DE REQUERIMIENTOS

Esta fase tiene como objetivo definir, de forma concisa y sin ambigüedad, los resultados del análisis de requerimientos.

La especificación de requerimientos se considera el contrato entre el usuario y el programador, por lo que debe ser clara y completa para ambos. Pomberger [Pomberger 84] sugiere la siguiente estructura para presentarla:

a) Situación inicial y metas. Contiene una descripción general de la situación inicial con base en el análisis anterior, así como las metas del proyecto y su delimitación.

b) Operación del sistema y ambiente. Describe los requisitos previos y el ambiente en el que va a operar el sistema (número de usuarios, frecuencia de uso, preparación del usuario, etcétera).

c) Requerimientos funcionales. Define las funciones del sistema esperadas por el usuario. Es importante que sea conciso,

sin ambigüedades e incluya información detallada sobre los datos (tipo, volumen y precisión); el tipo de notación que se recomienda utilizar se mencionará más adelante.

d) Requerimientos no-funcionales. Especifica, entre otras cosas, los requisitos de confiabilidad, de seguridad contra fallas, de portabilidad y de respuesta en tiempo de ejecución.

e) Interface con el usuario. Describe la manera en que el usuario se comunicará con el sistema. Incluye formatos de impresión, manejo y presentación de pantallas y ejemplos que enseñen al usuario a comunicarse adecuadamente con el sistema.

f) Comportamiento en los errores. Describe los efectos de los diferentes tipos de errores que pueden presentarse y define el comportamiento que debe seguir sistema ante ellos.

g) Requerimientos de documentación. Establece el tipo de documentación requerida. La documentación de un sistema es, por un lado, un medio indispensable para la aplicación correcta de un producto de Software (manual para el usuario) y por otro, la base para el mantenimiento del sistema. Para elaborar los manuales debe tomarse en consideración el tipo de usuario y los cambios y expansiones previstos, ya que ésto determina el grado de detalle en que deben hacerse.

h) Criterios de aceptación. Es importante establecer los criterios por los que se aceptan, tanto los requerimientos funcionales como los no-funcionales para así verificar la especificación.

i) Glosario e Indice. Ya que la definición de requerimientos es el documento que sirve de referencia a las personas que desarrollan y utilizan el sistema, es recomendable incluir en él un glosario y un índice que faciliten su manejo.

Antes de presentar al usuario el documento definitivo, donde se especifican los requerimientos, es fundamental asegurar que la especificación sea correcta y completa, así como que lo que en él se establece sea posible técnica y económicamente. La validación de los mismos, incluye:

a) Verificar que los requerimientos estén completos, es decir, que el usuario confirme que todos los requisitos funcionales y no-funcionales, restricciones y factores periféricos han sido considerados.

b) Verificar la consistencia de los requerimientos al cerciorarse de que no hay contradicción entre ellos.

c) Verificar la viabilidad técnica, la cual ocurre si se cuenta con el Hardware y la tecnología de Software apropiados para proporcionar el servicio deseado.

d) Investigación de los requerimientos de personal, ya que se necesita contar con personal que tenga la preparación adecuada tanto para la producción como para la operación del sistema.

e) Justificación económica, donde se demuestre la efectividad en costo del sistema antes de realizarlo.

La especificación de las funciones que se espera que realice el sistema, debe contar con un medio de comunicación que las defina en forma concisa y no ambigua. La inexactitud y ambigüedad de los lenguajes naturales, así como el volumen y complejidad de las especificaciones, hacen muy difícil, si no imposible, el poder verificar si este tipo de descripciones son completas y consistentes.

La Ingeniería de Software proporciona una serie de herramientas que ayudan a describir las especificaciones con una notación clara y de forma completa; localizan errores fácilmente; admiten una estructura para ayudar en la

implementación; auxilian en la planeación de costos, tiempo y personal. Con estos medios pueden representarse procesos (algoritmos), estados (datos y situaciones), flujo de datos, relaciones jerárquicas y abstracciones.

Hasta el momento no se ha desarrollado un método que logre reunir todas las características antes mencionadas; sin embargo, existe gran variedad de herramientas que ayudan en la obtención de algunas de las cualidades deseadas.

Uno de los métodos elementales para expresar especificaciones son los diagramas de flujo donde, por medio de símbolos, se muestra la trayectoria de las funciones del sistema; aunque no son ambiguos, dificultan la medición de si la especificación es completa y usándose indisciplinadamente dan como resultado una estructura pobre.

Además de los diagramas de flujo se han desarrollado técnicas más elaboradas que intentan proporcionar una ayuda más eficaz en la especificación; entre las principales se encuentran:

a) SADT (Técnica de Análisis Estructurado y Diseño). Es un método gráfico que, por medio de diagramas o modelos, representa la jerarquía de las estructuras del sistema. Cada caja en el diagrama, a su vez, puede ser definida con más detalle en otro diagrama. Se presenta un modelo en relación al proceso o a los datos. La notación y metodología de SADT pueden utilizarse tanto en la especificación de requerimientos como en el diseño del sistema ([Ross 77, 85]).

b) PSL/PSA (Lenguaje de declaración de problema/Analizador de declaración del problema). Es una combinación de un lenguaje (PSL) para especificar requerimientos y un procesador (PSA) para validar las declaraciones en PSL. Dicho lenguaje contiene un conjunto de declaraciones que permiten al usuario definir objetos, así como las propiedades y relaciones de éstos. El procesador genera una base de datos que describe los

requerimientos del sistema y realiza pruebas para determinar si los datos son completos y consistentes. Por ser un método demasiado general hay problemas al aplicarlo en algunos sistemas; su uso es efectivo en sistemas grandes, sin embargo, es muy costoso para aplicaciones pequeñas. Fue desarrollado en la Universidad de Michigan ([Teichrow 77] y [Winters 79]).

c) HIPO (Jerarquía del Proceso Entrada - Salida). Se caracteriza por hacer una distinción clara entre la notación de datos y de funciones; su representación es jerárquica [IBM 75].

d) SSA (Análisis Estructurado de Sistemas). Contiene una colección de herramientas, aplicables a las primeras fases del ciclo de vida, para hacer descripciones y modelos. Cuenta con una notación gráfica adecuada para la descripción de sistemas en general [DeMarco 78].

e) Análisis de datos. Esta técnica desarrolla modelos no funcionales del mundo real; adquiere información y la modela en términos de **entidades** (objetos del mundo real) y sus relaciones y atributos (propiedades), para construir un **esquema conceptual** que más tarde puede convertirse en una base de datos que soporte los requerimientos funcionales [Perkinson 84].

f) Especificación formal. Requiere un lenguaje de especificación matemáticamente preciso, desde el punto de vista sintáctico y semántico. Este permite la verificación formal de la misma. Es necesario hacer la traducción de la especificación formal a una representación informal para que pueda entenderla el usuario [Gehani 86].

g) GAL (Lenguaje Gráfico Abstracto) es una notación informal que, haciendo uso de un conjunto de símbolos, permite la representación gráfica de programas estructurados; proporciona medios para facilitar la modularización y la estructuración [Albizuri 84].

Todos los métodos mencionados anteriormente enfatizan sólo ciertas facetas de la especificación, son inflexibles y resuelven determinado tipo de aplicaciones. No existe aún un método general, por lo tanto, la especificación es una tarea creativa que no puede ser sustituida por reglas o metodologías.

2.1.3 DISEÑO

"El arte de la programación empieza con el diseño" [Pomberger 84]. El diseño de Software es un proceso creativo que hace uso de diferentes herramientas y se rige por ciertas normas, sin que por ello, se limite a un esquema general y único, aplicable a todos los sistemas. Debe llevar a obtener una solución algorítmica que satisfaga los requerimientos de calidad para un problema específico. Es posible distinguir entre diseño estructural (componentes, módulos, estructuras, etcétera) y diseño detallado (algoritmos, representación de datos, etcétera).

Uno de los cambios sustanciales a partir de la crisis del Software en los años 60, es el concepto de un buen diseño. Se hicieron a un lado los programas personales (sólo la persona que los hacía podía entenderlos) con una arquitectura bastante compleja y confusa para buscar la producción de programas que, por la claridad de su estructura, fueran legibles, flexibles y fáciles de mantener, aún por personas ajenas a su desarrollo original.

De este cambio de mentalidad surgió el concepto de Programación Estructurada, formalizado por Edsger Dijkstra [Dahl 72], que propone el uso de un número limitado de constructores lógicos que tienden a:

1. Minimizar la complejidad del flujo del programa,
2. reducir la susceptibilidad de error,
3. facilitar la verificación del algoritmo y,

4. conservar cada elemento del programa suficientemente pequeño para que pueda manejarse.

Tales constructores son los propuestos a mediados de los años 60 por Böhm y Jacopini [Böhm 66], quienes probaron que cualquier lógica de procedimientos puede derivarse a partir de la combinación de los tres constructores lógicos mostrados a continuación:

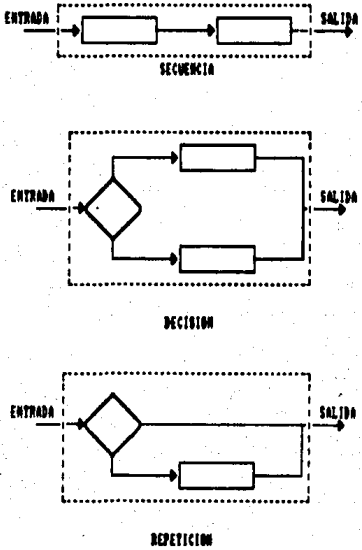


Figura 2.1 Constructores lógicos.

Estos tienen un sólo punto de entrada y uno de salida, lo que permite tomar cualquier diagrama de flujo, construido con estas formas básicas, y examinar cada parte de él como una caja negra independiente del resto.

De igual forma, Dijkstra señala que un punto clave para lograr lo anterior es la eliminación de la proposición GOTO, que permite hacer saltos incondicionados y de cuyo uso indisciplinado resultan estructuras de flujo complejas. Yourdon [Yourdon 89] recopila opiniones a favor y en contra del uso del GOTO.

Afortunadamente, los criterios para el uso del GOTO se han modificado y se encuentran posturas moderadas como la de Mills [Mills 72]: "Los programas deberían caracterizarse no por la ausencia del 'GOTO' sino por la presencia de una estructura".

Es importante no dejarse llevar por el mito de la programación sin GOTO y afirmar que por el simple hecho de no incluir esta sentencia de salto incondicional el programa ya tiene una buena estructura o que la presencia de una instrucción de este tipo convierte automáticamente al código en un programa sin estructura. El objetivo último es dar al programa una estructura que facilite su legibilidad y mantenimiento, lo demás son medios para lograrlo. El problema no está en la herramienta, sino en el mal uso de la misma.

El programar de forma estructurada trae como consecuencia algunas desventajas: invertir más tiempo en la fase de diseño; el sistema puede resultar menos eficiente puesto que el criterio central de diseño no es la eficiencia sino la estructuración y la legibilidad; se obtiene mayor cantidad de código. Sin embargo, a cambio se logran los beneficios ya mencionados (por ejemplo, minimizar complejidad).

Existen diversas teorías para llegar a la estructuración adecuada de los programas, de entre ellas, se podrían mencionar la abstracción y la modularización. Aún cuando estas teorías son presentadas y defendidas por autores diferentes (citados posteriormente), es difícil delimitarlas, ya que tienen algunos puntos de coincidencia, incluso Ratcliff [Ratcliff 87] afirma: "La modularidad es una consecuencia natural de aplicar la

abstracción a través de varias formas de estructuración, jerarquización, etcétera". A continuación se les describe brevemente:

a) Abstracción. Facilita la disminución de la complejidad de un problema al enfatizar lo que es relevante y suprimir lo que no es. Esto puede hacerse de tres maneras: se descarta lo irrelevante, se ocultan o indeterminan los detalles y se generaliza. Liskov [Liskov 88] propone desarrollar los programas por medio de la descomposición del problema, basada en un reconocimiento de abstracciones útiles. Entendiendo por abstracciones útiles las que comprendan los aspectos claves dentro del contexto del programa.

La descomposición es usada para seccionar el programa en elementos que puedan ser combinados a fin de dar solución al problema original; la abstracción asiste en la selección adecuada de los componentes. Se alternan los dos procesos, descomposición y abstracción, hasta tener reducido el problema original a un conjunto de cuestiones que de antemano tienen solución.

La meta, al dividir un programa, es crear módulos que sean en sí mismos pequeños programas que interactúen en formas simples y bien definidas, de tal manera que cada módulocumpla con los siguientes requisitos:

1. Esté al mismo nivel de detalle.
2. Pueda ser resuelto y probado independientemente.
3. Pueda combinarse con otros para resolver el problema original.

Los lenguajes de alto nivel, como Pascal, C y Ada, proporcionan elementos adecuados para realizar la abstracción; uno de ellos es el procedimiento; al separar la definición y la invocación de éste, se obtienen dos posibles métodos de abstracción:

- Por parametrización: a través de la importación y exportación de parámetros permite representar un conjunto, potencialmente infinito, de diferentes ejecuciones con un sólo código de programa, el cual es una abstracción de todos ellos.

- Por especificación: permite abstraer del funcionamiento, descrito por el cuerpo de un procedimiento, el fin para el que éste fue diseñado.

Además, es posible extender las capacidades básicas del lenguaje por medio de tres tipos de abstracción:

- De procedimientos. Puede definir nuevas operaciones en el lenguaje.

- De datos. Permite el uso de nuevos tipos de datos mediante un conjunto de objetos y de operaciones que caracterizan el comportamiento de éstos.

- Iteración. Permite la repetición sobre todos los elementos de un multiconjunto, sin restringir el orden en el cual éstos deben ser procesados. Es una generalización de los métodos de iteración disponibles en la mayoría de los lenguajes de programación (por ejemplo Pascal, C, Modula-2, Ada).

La abstracción de datos permite aplazar las decisiones sobre las estructuras de éstos hasta que su uso sea totalmente entendido. En vez de definirla de inmediato, se introduce el tipo de datos abstractos con sus objetos y operaciones.

En la teoría de la abstracción a las partes obtenidas de la descomposición se les llama módulos. En ella no se mencionan sus características, sólo se habla de la necesidad de que realicen una función y sean independientes entre sí. Liskov [Liskov 86] hace una descripción detallada de esta teoría.

b) Modularización. Consiste en descomponer el sistema en partes o módulos para reducir su complejidad y hacerlo flexible. Módulo es: "un segmento de programa que puede comunicar sus propiedades al mundo exterior sólo a través de interfaces bien definidas" [Goos 73]. Se busca que éstos presenten las siguientes características:

- Realizar una sola función.
- Tener una interface simple y que se especifique explícitamente.
- Determinar, con facilidad, si cada módulo es correcto, considerando sólo las interfaces y no su comportamiento en el sistema completo.
- No ejercer influencia interna entre ellos y por tanto, ofrecer la posibilidad de reemplazarse por otro que respete la interface original sin afectar a todo el sistema.
- Su tamaño no puede definirse cuantitativamente respecto al número de líneas o páginas de código fuente; sino por la claridad y consistencia del mismo.
- Su especificación y diseño debe realizarse de tal manera que la información que contiene sea inaccesible para quienes no la necesitan. Esto se conoce como información oculta.

La independencia entre módulos es esencial -como ya se explicó anteriormente- y se mide por medio de dos criterios cualitativos:

- Cohesión. Medida de la relación que existe entre los elementos de un módulo. Puede ser:
  - Funcional. Todos los elementos se relacionan en el desarrollo de una sola función.
  - Secuencial. La salida de un elemento del módulo es la entrada de otro.

- Comunicacional. Los elementos del módulo hacen referencia a un mismo conjunto de datos.
- Temporal. Hay una relación de tiempo entre los elementos (inicialización, actualización, etcétera).
- Lógica. La relación entre los elementos de un módulo es ordenada sistemáticamente (por ejemplo, agrupa todos los procedimientos de impresión).
- Coincidental. Los elementos del módulo están reunidos al azar.

- Acoplamiento. Mide la relación que hay entre los módulos. Mientras sea menor la conexión entre ellos menor será la probabilidad y el riesgo de propagación de errores. El acoplamiento debe hacerse únicamente a través de procedimientos y con el mínimo de parámetros posible.

Para integrar un sistema, los módulos se estructuran de acuerdo a una jerarquía tipo árbol; en ésta, los nodos están formados por módulos, los de nivel inferior definen las estructuras de datos, sus tipos y las operaciones de acceso en elementos individuales; en el siguiente nivel, ascendiendo hacia la raíz, se encuentran aquellos cuyas funciones manipulan grupos de elementos de una estructura de datos; el nivel superior inmediato, contiene funciones que combinan elementos de varias estructuras de datos orientadas a problemas. De esta forma, se asciende en la arquitectura hasta llegar a los nodos que contienen funciones de control para relacionar los nodos inferiores de manera que se cumpla la especificación del sistema.

Se hace posible la modularización gracias a elementos tales como: las estructuras de control de programas, que varían dependiendo del lenguaje de programación (por ejemplo procedimientos, módulos, procesos, corrutinas); las estructuras de control de los datos; las cápsulas de datos (una cápsula de

datos es un módulo que consta de una declaración de datos y los procedimientos para manejarlos).

"Un método de diseño de Software es una colección de técnicas basadas en un concepto" [Peters 77]. La programación estructurada, la abstracción y la modularización son las teorías básicas de donde parten las técnicas que se mencionan a continuación:

a) Diseño de arriba hacia abajo. La idea básica se debe a Dijkstra [Dijkstra 69] y Wirth [Wirth 71] quienes afirman que el diseño debería iniciarse con el análisis de la especificación del sistema ignorando los detalles de la implementación. A partir de éste se descompone el problema, de arriba hacia abajo, en subproblemas hasta obtener funciones que puedan traducirse fácilmente en algoritmos. También se le conoce como descomposición funcional ya que su tarea principal es reestructurada en funciones más simples. Dependiendo del método que se utilice puede dividirse respecto al tiempo, al flujo de datos, etcétera. Esta técnica es un medio para realizar sistemas completamente nuevos.

b) Diseño de abajo hacia arriba. En esta estrategia se procede de manera opuesta a la anterior; en ella el Hardware y cada nivel del Software se considera como una máquina abstracta ("conjunto de operaciones con las cuales se puede expresar el funcionamiento de un sistema o de un subsistema a cualquier nivel de abstracción" [Gewald 77]). El diseño se inicia con las propiedades de una máquina concreta y se va desarrollando una tras otra a través de la adición sucesiva de nuevas propiedades hasta llegar a la máquina necesaria para llevar a cabo las funciones requeridas por el usuario. Este método es apropiado para adaptar, a nuevos requerimientos, el Software existente.

c) Refinamiento por pasos sucesivos. Para varios autores ([Liffick 85], [Ratcliff 87], [Zelkowitz 79]) es la forma estándar de la programación estructurada de arriba hacia abajo.

Se basa en el principio general dado por Wirth [Wirth 71], en el que establece que, para un problema complejo, antes de resolver los detalles se deben descomponer en subproblemas más sencillos que el original. Si se aplica de forma consistente, ésta práctica conduce, paso a paso, a una solución cada vez más precisa del problema inicial. El programa es estructurado jerárquicamente y descrito por refinamientos sucesivos. El proceso termina cuando todos los detalles quedan clarificados.

Existen diversos métodos que hacen uso de las técnicas antes expuestas, de entre ellos se podrían mencionar:

a) Diseño estructurado. Fue definido por Constantine y Yourdon [Yourdon 79]. Tiene su origen en la programación modular clásica donde la estructura del programa se ve en términos de un módulo de control en el nivel más alto que dirige una jerarquía de módulos subordinados. A partir del diagrama del flujo de datos se define el esquema de la estructura del sistema utilizando los criterios de modularización.

b) Diseño orientado a flujo de datos. Este modelo amplía los conceptos básicos (como programación estructurada y modularidad) al integrar explícitamente el flujo de información en el proceso de diseño y considerar la transformación de los datos durante el proceso. Es una descomposición funcional con respecto al flujo de datos presentada por Yourdon y Constantine [Yourdon 79]. En el libro "Software Engineering: A Practitioner's Approach" de Pressman [Pressman 82] se puede encontrar una descripción detallada de éste método.

Algunos autores establecen una diferencia entre los dos métodos antes mencionados aunque no es muy clara.

c) Diseño orientado a la estructura de datos. También se le conoce como método de Jackson. Su premisa básica es que un programa ve el mundo a través de sus estructuras de datos y que por lo tanto un modelo de éstas puede ser transformado en un

programa que incorpore una muestra correcta del mundo. Jackson [Jackson 75] afirma: "los problemas pueden descomponerse en estructuras jerárquicas de partes que pueden ser representadas por árboles de formas estructuradas". Las formas estructuradas a las que alude son secuencia, condición y repetición, que constituyen el fundamento de la filosofía de la programación estructurada. Pressman [Pressman 82] detalla un poco más este concepto.

d) Construcción lógica de programas. Este método fue definido por Warnier [Warnier 74, 81] y se presenta como una serie de reglas o leyes que gobiernan la estructura de la información y la organización del Software resultante. Parte de la estructura de datos, sigue una representación formal de los procedimientos y culmina con los métodos sistemáticos para la generación, verificación y optimización del pseudocódigo. Peters [Peters 77] lo compara con otras metodologías de diseño.

e) Métodos formales. Buscan establecer fundamentos matemáticos para el desarrollo del Software en el aspecto de la especificación y la verificación; ésta última permite probar si un Software es correcto, es decir, que no se desvía del comportamiento expresado en su especificación cuando se ejecuta en la máquina y lenguaje para los que fue realizado. Ratcliff [Ratcliff 87] lo profundiza.

Un aspecto fundamental en el diseño es la notación que se utiliza para definir las decisiones tomadas y hacerlas legibles, cuantificables y verificables, ya que de ésto depende en gran parte la simplificación de las fases de prueba y mantenimiento. Para ello se cuenta con las herramientas ya mencionadas en la especificación de requerimientos (Sección 2.1.2).

Después de analizar los conceptos, técnicas y métodos, propuestos por la Ingeniería de Software, para llevar a cabo la fase de diseño, se puede apreciar que la meta de todos ellos es una: obtener Software organizado, que refleje con claridad cuál

es su estructura, con el fin de no invertir horas en descifrar que es lo que hace una función, con el objeto de dar calidad al Software. Esto se consigue al aplicar ciertas normas que regulan y sistematizan el proceso. Un principio elemental en la resolución de cualquier tipo de problema es: **divide y vencerás.**

2.1.4 IMPLEMENTACION

La implementación de un sistema de Software comprende la transformación de las especificaciones de diseño en programas que puedan ser ejecutados en una máquina determinada. Zelkowitz [Zelkowitz 79] asegura que es la etapa más sencilla, puesto que los lenguajes de alto nivel y la programación estructurada facilitan la tarea.

"La simplicidad, claridad y elegancia son la marca de los buenos programas" [Fairley 85]. Esto se logra a través de las técnicas de codificación estructurada, un buen estilo de codificación, documentos de soporte apropiados, buenos comentarios internos y las características que proporcionan los lenguajes de programación modernos.

Dentro de las técnicas de codificación estructurada se encuentran los constructores lógicos mencionados en la fase de diseño (secuencia, decisión y repetición), así como la eliminación de la sentencia **GOTO** y el uso del principio de recursividad.

No existe un conjunto de reglas que definan qué es un buen estilo de programación para todas las aplicaciones; sin embargo, existen principios generales que pueden servir de guía en la obtención de código legible. Fairley [Fairley 85] indica lo que se debería y lo que no se debería hacer para obtener un buen estilo de programación:

Se debería:

- Usar pocos constructores de control.
- Introducir tipos de datos definidos por el usuario para modelar entidades en el dominio del problema.
- Ocultar las estructuras de datos detrás de funciones de acceso.
- Aislar las dependencias de la máquina en unas pocas rutinas.
- Proporcionar prólogos de documentación para cada subprograma y/o unidad de compilación.
- Examinar cuidadosamente las rutinas que tengan menos de 5 o más de 25 sentencias ejecutables.
- Utilizar indentación, paréntesis, espacios, líneas en blanco y bordes alrededor de los bloques de comentarios para mejorar la legilidad.

Se debe evitar:

- Ser **mañoso** (para que nadie entienda lo escrito).
- Usar la sentencia **Then** nula.
- Usar la sentencia **Then-If**.
- Anidar a demasiada profundidad.
- Usar un identificador para múltiples propósitos.

Además de lo antes propuesto, Pomberger [Pomberger 84] señala como indispensable:

- Utilizar nombres descriptivos de longitud razonable (algunos lenguajes la limitan) que expresen su significado.
- Separar los comentarios del resto del texto del programa para que la estructura del programa permanezca visible.

Otro aspecto que se menciona respecto al buen estilo de programación es la documentación adecuada del código por medio de comentarios. Pomberger [Pomberger 84] afirma: "La ausencia de comentarios indica una deficiencia en la calidad del programa...

Los buenos son aquellos que no repiten lo que dice el código, sino que dan información adicional y los ambiguos o incorrectos son peor que nada".

Se pueden distinguir dos tipos de comentarios:

- Los que describen la historia, versión y función del programa, localizándose en la parte superior del mismo y
- los que explican los objetos, secciones del programa y sentencias.

La elección del lenguaje de implementación adecuado para cada aplicación es determinante, ya que ésto facilita o entorpece la traducción de las especificaciones de diseño. Debería buscarse que el lenguaje proporcione recursos como el contar con estructuras de control, la definición de estructuras y datos abstractos, recursión, compilación por separado, etcétera.

2.1.5 PRUEBA

El Software de calidad busca que el sistema sea correcto y confiable, por lo que es indispensable hacer una evaluación cuidadosa del mismo.

El objetivo de la fase de prueba es averiguar si un programa satisface los requerimientos establecidos y descubrir tantos errores como sea posible.

Myers [Myers 79] define:

1. Prueba es el proceso mediante el cual se ejecuta un programa con el fin de localizar un error.

2. Un buen caso de prueba es aquel que tiene una alta probabilidad de detectar los errores y aún de ubicarlos.
3. Una prueba exitosa es aquella que es capaz de encontrar los errores hasta entonces no descubiertos.

La figura 2.2 muestra el flujo de información de prueba propuesto por Pressman [Pressman 82]:

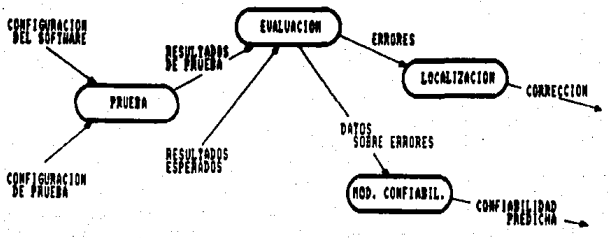


Figura 2.2 Flujo de información de prueba.

En este proceso se necesitan dos entradas:

a) La configuración del Software que incluye las especificaciones de requerimientos y diseño y el código fuente.
b) La configuración de la prueba que consta del plan, proyecto, casos de prueba y resultados esperados.

Se realizan las pruebas, se comparan los resultados obtenidos con los esperados y si aparecen errores, se inicia el proceso de detección, ubicación y eliminación de las causas de los mismos; el tiempo empleado en ello es impredecible.

Los resultados ordenados y evaluados de las pruebas indican la confiabilidad del Software.

Un producto se puede probar desde dos puntos de vista:

a) Al demostrar que funciona como un todo y que para una determinada entrada produce la salida correspondiente (prueba de caja negra) o,

b) al asegurar que su operación interna se desarrolla de acuerdo a lo especificado (prueba de caja blanca).

Ambos enfoques pueden realizarse por medio de:

a) Pruebas estáticas. Actividades que buscan errores sin necesidad de ejecutar el programa. Se relacionan con el análisis sintáctico, estructural y semántico del código. Pomberger [Pomberger 84] menciona cinco elementos importantes en ella:

1. Verificación del programa.
2. Inspección del código.
3. Análisis de complejidad.
4. Análisis de la estructura.
5. Análisis del flujo de datos.

b) Pruebas dinámicas. En ellas los objetos medidos son ejecutados o simulados. Pomberger [Pomberger 84] afirma que constituyen un aspecto indispensable en el ciclo de vida del Software e indica los siguientes pasos:

1. Preparar el código para la localización de errores.
2. Proporcionar un ambiente adecuado.
3. Seleccionar los datos y casos apropiados.
4. Ejecutar y evaluar la prueba.

Dentro del contexto de la Ingeniería de Software, la prueba podría dividirse en cuatro pasos, implementados secuencialmente:

a) Prueba por unidad. Se enfoca al análisis de cada módulo de forma individual asegurando que funciona como una unidad. Corresponde al tipo caja blanca y puede realizarse paralelamente en los diferentes módulos.

Pressman [Pressman 82] propone cinco características evaluables en este nivel:

1. Interface del módulo.
2. Estructuras de datos locales.
3. Trayectorias de ejecución importantes.
4. Trayectorias que manejan errores.
5. Condiciones límite que afectan los puntos anteriores.

Dado que los módulos no son un programa que se ejecuta individualmente, en la mayoría de los casos es necesario implementar programas de prueba que pasen datos e impriman los resultados relevantes. Myers [Myers 79] enlista algunos puntos, que considera importantes, para probar los aspectos antes mencionados.

b) Prueba de integración. Se orienta a verificar el resultado de interconectar los módulos. Es una técnica sistemática para ensamblar el Software, al mismo tiempo que se llevan a cabo pruebas para detectar errores asociados con la interface. Su objetivo es tomar los módulos probados individualmente y construir la estructura del Software que fue aceptada en el diseño.

A menudo la integración se hace de forma jerárquica siguiendo un esquema de árbol. Existen dos estrategias principales para realizarla: de arriba hacia abajo y de abajo hacia arriba.

En la primera, los elementos se estructuran a partir del módulo de control principal (raiz del árbol) en linea descendente, ya sea en la forma primero profundidad en la que se

evalúan una por una las ramas del árbol hasta llegar a su nivel más bajo; o primero anchura en la que se pasa al nivel inferior hasta que se hayan probado todas las ramas en ese nivel.

En la integración de abajo hacia arriba se procede de forma inversa a la anterior, puesto que se inicia la prueba e integración con los módulos atómicos, es decir, los del nivel más bajo en la estructura, hasta llegar al de control principal.

No se puede establecer cuál de las formas es mejor, puesto que las ventajas de una son las desventajas de la otra. La mayor desventaja de la estrategia de arriba hacia abajo es la necesidad de programas que simulen las funciones de los módulos de nivel inferior al que se está probando y la de la estrategia de abajo hacia arriba es, según Myers [Myers 79], que el programa como entidad no existe hasta que se le ha agregado el último módulo.

c) Prueba de validación. Proporciona la garantía final de que el Software, completamente ensamblado y sin errores (detectados y no corregidos), cumple todos los requerimientos funcionales y de desarrollo. En la especificación de requerimientos obtenida en la segunda fase del ciclo de vida se incluyen los criterios de validación que constituyen las bases para hacer la prueba. Esta se realiza a través de una serie de pruebas de caja negra que demuestran la conformidad con los requerimientos.

d) Prueba del sistema. Verifica que todos los elementos se acoplen adecuadamente y que la función total del sistema sea alcanzada. El último paso es la prueba de aceptación realizada con datos reales y tiene el propósito de detectar los errores que puedan surgir eventualmente cuando el usuario lo emplee. Puede prolongarse durante meses.

Para llevar a cabo los cuatro pasos antes descritos, es necesario implementar casos de prueba, cuyo primer objetivo es definir una combinación de datos que tengan la mayor probabilidad

de descubrir errores o clases de errores. Dado que es casi imposible verificar todas las trayectorias del programa, se busca encontrar la mayor cantidad de errores con el menor número posible de pruebas.

Dichas pruebas se pueden realizar a diferentes grados de profundidad:

a) Ejecutar todas las sentencias cuando menos una vez.
b) Valuar todas las condiciones como verdaderas o falsas y así conocer los efectos producidos en cada caso.
c) Probar las trayectorias más complejas para evaluar los cambios de condiciones.

Después de realizada una prueba, si se detecta un error, se sigue un proceso que lleva a la ubicación de la causa del mismo; una vez encontrada se procede a su eliminación. Aunque ésto puede verse como un proceso ordenado, es más bien una actividad intuitiva. Un Ingeniero de Software al evaluar los resultados de una prueba es, normalmente, confrontado con una indicación sintomática de un problema, es decir, la manifestación externa del error y sus causas pueden no tener relación entre sí.

Shneiderman [Shneiderman 80] asegura que:

> "La ubicación y corrección de errores es una de las partes más frustrantes de la programación. Tiene elementos para resolver problemas o 'rompecabezas', unido al reconocimiento molesto de que hemos cometido un error. El aumento de ansiedad y la mala gana de aceptar la posibilidad de error incrementa la dificultad de la tarea. Afortunadamante, es motivo de gran alivio y baja de tensión cuando el error es al fin corregido".

Myers [Myers 79] propone tres enfoques para realizar la ubicación y corrección:

1. Fuerza bruta. Es el método más común. Se utiliza la filosofía de dejar a la máquina encontrar el error. Consiste en

hacer un trazado en el momento de la ejecución introduciendo en el programa varias proposiciones write. Aún cuando el volumen de información recibida puede llevar al éxito, normalmente ocasiona pérdida de tiempo y esfuerzo.

2. Eliminación de las causas. Se lleva a cabo por medio de inducción o deducción. Los datos relacionados con el error se organizan para aislar las causas posibles. Se establece una causa hipotética y los datos organizados se utilizan para probar o desaprobar esta hipótesis.

3. Volver hacia atrás (Backtracking). Se traza manualmente el código fuente en retroceso en el punto en donde se presenta la manifestación del error hasta encontrar el lugar de la causa. Generalmente se utiliza en programas pequeños.

Es necesario documentar la fase de prueba. Los puntos importantes que debería contener el documento serán detallados como parte de la siguiente fase del ciclo de vida del Software.

Es posible concluir que el hecho de contar con un buen diseño e implementación, por el uso de los principios de programación estructurada, simplifica la etapa de prueba y reduce la probabilidad de cometer errores.

Esta fase, dentro del ciclo de vida del Software, representa el control de calidad del producto, ya que permite compararlo con los parámetros establecidos. El resultado de la evaluación indica que lo que en el inicio era una intuición o idea de posible solución, ahora es algo concreto que satisface las necesidades o sirve de retroalimentación para volver a fases previas con el fin de obtener una solución óptima.

2.1.6 DOCUMENTACION

Un sistema de Software, normalmente, no es desarrollado por una sola persona, ni es ella misma quien lo utiliza o mantiene; por consiguiente es necesario tener un medio de comunicación entre la gente que trabaja en el proyecto y quienes lo van a utilizar o a mantener. Para alcanzar este objetivo es fundamental la documentación.

A diferencia de las otras fases, la de documentación no se lleva a cabo en un sólo momento dentro del ciclo de vida, sino que se realiza a lo largo de éste y la información se recopila y afina al concluir la etapa de prueba.

Como medio de comunicación que es, tiene por objeto transmitir todo el conocimiento adquirido en la elaboración del sistema a quienes harán uso de él y lo mantendrán en funcionamiento.

Para su elaboración es necesario cuestionarse:

1. ¿Qué información debe incluirse en el documento y el nivel de detalle de la misma?
2. ¿Cómo organizarla?
3. ¿Cómo presentarla para que sea entendible?.

Pomberger [Pomberger 84] menciona tres tipos de usuarios y por lo tanto igual número de documentos con diferentes enfoques y contenidos:

a) Documentación del usuario. Debería garantizar que el producto puede utilizarse en forma óptima sin necesidad de mayor información. Debería constar de:

- Una descripción general del sistema que incluya información sobre: su propósito, los recursos de Hardware y Software necesarios para su funcionamiento, forma y estructura de los datos de entrada, contenido y forma de los datos producidos, prerrequisitos de organización, restricciones dependientes de la implementación y flexibilidad y portabilidad del sistema.

- Un manual de instalación con: información sobre cómo se instala el sistema, prerrequisitos de instalación (como archivos y recursos), descripción completa y no ambigua de todas las funciones del sistema y las respuestas del usuario necesarias para su uso, ejemplos de aplicaciones de las funciones, información sobre el formato de los datos de entrada, representación y explicación de los resultados y lista de mensajes de error con indicaciones de las causas y medios para eliminarlos.

b) Documentación del sistema. Debería describir todos los detalles de la construcción de un sistema de Software, la estructura de sus componentes y las actividades de prueba realizadas. Incluye los documentos obtenidos en cada fase:

1. Descripción del problema como resultado de la etapa de especificación de requerimientos.

2. Descripción de la implementación en general efectuado en la fase de diseño.

3. Descripción de la implementación en detalle presentado como encabezado del módulo y escrito durante la codificación del mismo.

4. Descripción de los archivos usados donde se incluye

el nombre, estructura y organización de éste. Se elabora durante la implementación.

5. Reporte de prueba con los medios, datos y resultados de los casos de prueba; se recaba en las fases de especificación y prueba.

6. Tablas y diagramas que muestran, por orden alfabético, los módulos, funciones, datos y estructura del sistema. Se trazan al finalizar la etapa de codificación.

7. Listado de los programas como resultado de la implementación.

c) Documentación del proyecto. Consta de la información necesaria para quien dirige y controla el proyecto. Su enfoque es la eficiencia en cuanto a costo, incluyendo organización del personal, tiempo empleado en el desarrollo y resultados obtenidos.

Se recomienda que estos documentos sean precisos y claros, sin ambigüedades ni redundancias, de manera que no confundan al lector; breves, puesto que una documentación demasiado extensa es difícil de mantener actualizada; fáciles de usar por la utilización de paginación, encabezados, índices, referencias y tablas de contenido; capturados conforme va evolucionando el sistema y no al final, ya que pueden perderse aspectos importantes detectados durante el desarrollo.

De forma especial, en el documento dirigido al usuario debería evitarse el empleo de palabras y notaciones técnicas que puedan confundir en vez de ayudar.

Aún cuando parezca tedioso, es necesario hacer hincapié en que la documentación es un aspecto clave para que el desarrollo de un sistema no resulte una Torre de Babel por la falta de

comunicación entre quienes lo producen y pueda, a su vez, ser utilizado correctamente, de lo contrario resultará obsoleto en un periodo de tiempo demasiado corto.

2.1.7 OPERACION Y MANTENIMIENTO

Un sistema, después de que se entrega para su uso normal, no permanece estático, puede haber una ampliación o cambio en los requerimientos, descubrirse errores, necesitarse instalaciones diferentes, etcétera. A esta serie de cambios, llevados a cabo una vez instalado el sistema, se le conoce como mantenimiento.

Algunos autores ([Pomberger 84], [Lientz 80], [Pressman 82], [Ramamoorthy 86]) definen tres tipos de mantenimiento:

a) Perfectivo. Incorporación de los nuevos recursos ofrecidos por la ciencia de la computación en busca de la optimización del sistema.

b) Correctivo. Proceso que incluye el diagnóstico y eliminación de uno o más errores no detectados en la fase de prueba.

c) Adaptivo. Cambios debidos a nuevos requisitos.

Hay quienes ([Pomberger 84] y [Bendifallah 87]) agregan a los anteriores el preventivo, que es aquel que se realiza con el fin de proporcionar bases para posibles cambios futuros.

Parece casi imposible producir sistemas que no requieran cambios posteriores por lo que es importante que los programas se obtengan de tal manera que faciliten esta tarea, es decir, que posean capacidad de manutención (posteriormente, al tratar de calidad del Software, se hablará con más detalle de esta

cualidad). Como se mencionó en la descripción de las fases anteriores, el desarrollo adecuado de cada una de ellas ayuda al mantenimiento del sistema y prolongando así la duración de la vida del mismo.

Después de observar las fases del ciclo de vida del desarrollo del Software se puede concluir que este método proporciona un medio sistemático para su producción y resulta de gran ayuda en la obtención de sistemas capaces de funcionar de acuerdo a unos requisitos planteados.

En un ciclo cada etapa es importante, aunque hay puntos en los que se recomienda prestar especial atención por los efectos que pueden producir en fases posteriores.

2.2 SOFTWARE DE CALIDAD

La calidad del Software puede definirse como la utilidad o capacidad de uso de un producto. Existen diversos factores que influyen en la calidad y la productividad del Software, como son: la habilidad individual, la comunicación entre el grupo de trabajo, la complejidad del producto, las anotaciones adecuadas, el desarrollo sistemático, el nivel de tecnología, el control de cambio de requerimientos, el nivel de confiabilidad, la claridad con que se entienda el problema, una planeación adecuada, el tiempo disponible, los recursos con que se cuenta, el buen entrenamiento, tener metas alcanzables, etcétera.

Se han establecido pautas para evaluar la capacidad de uso del Software. En la figura 2.3 se muestra, en forma gráfica, las características del Software de calidad, de acuerdo a lo propuesto por Chin-Kuei [Chin-Kuei 80], Pomberger [Pomberger 84] y Ratcliff [Ratcliff 87]:

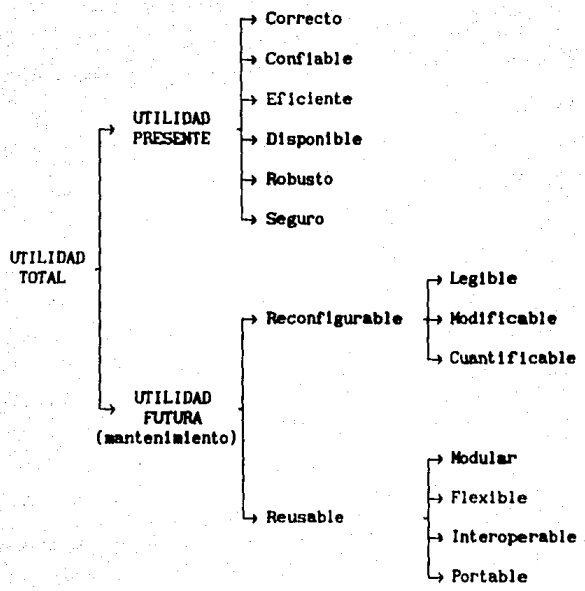


Figura 2.3 Características del Software de calidad.

Correcto. Un sistema es correcto si satisface las especificaciones funcionales dentro de las tolerancias requeridas.

Confiable. Esta característica se alcanza cuando el Software es capaz de comportarse sin desviar su especificación durante todo el tiempo que opera en condiciones consistentes con la misma.

Eficiente. Si el Software es capaz de llevar a cabo sus funciones con los recursos mínimos. Considerando como recursos el espacio de memoria, tiempo de CPU, canales de entrada y salida, mecanismos periféricos, etcétera.

Disponible. Un programa está disponible si es amigable y fácil de usar para el usuario; si está protegido contra errores debidos al mal uso, con mensajes claros y concisos, documentos fáciles de entender y sin errores.

Robusto. Es cuando el Software es capaz de lograr que los efectos de datos erróneos, las fallas en el funcionamiento y/o en el Hardware sean mínimos.

Seguro. La seguridad del Software se obtiene si cuenta con protección en el acceso a datos y/o áreas confidenciales para tener un mayor control de la información y de los usuarios de ella.

Reconfigurable. Es la característica que permite al Software ser modificado exitosamente con un costo y esfuerzo mínimos. Para ello necesita ser:

a) Legible. El usuario debe poder leer y entender el programa con facilidad. Es importante que los nombres de los identificadores sean mnemotécnicos, que haya consistencia en la indentación de todo el código, que el uso de las notaciones sea coherente y estable, que se evite la redundancia en los datos, se deben utilizar comentarios claros y concisos y, por último, que sea fácil de hacer el seguimiento de las interfaces entre los diferentes módulos y su flujo de datos.

b) Modificable. Un programa es modificable si, cuando requiere cambios, es posible y menos costoso modificarlo que volverlo a hacer.

c) Cuantificable. La especificación, documentación y diseño del Software deben permitir la aplicación efectiva de procedimientos para su validación y verificación.

Reusable. Es la capacidad por la cual el Software puede ser usado en diferentes aplicaciones y ambientes operacionales con alteraciones mínimas. Es uno de los factores más importantes para mejorar la productividad y la calidad del mismo. Además de que un programa sea legible, es importante que sea:

a) Portable. Debe ser lo más independiente posible del Hardware en que se ejecuta.

b) Modular. Cada función del sistema debe corresponder a un módulo o subprograma y éstos deben estar interfaceados entre si. Cuando la modularidad está bien estructurada proporciona el beneficio de que las modificaciones al sistema afectan a algunos módulos pero no al sistema completo.

c) Interoperable. El Software o sus componentes deben poder ser fácilmente incorporados como componentes de otro sistema.

d) Flexible. El Software es flexible cuando puede crecer o decrecer en sus funciones con facilidad.

Las características de calidad antes mencionadas pueden dividirse a su vez, dado que algunas afectan al producto en si, en: corrección, exactitud y utilidad; otras, sin embargo, se refieren al sistema completo, como son: eficiencia, legibilidad, confiabilidad, robustez, estructuración, consistencia, ingeniería humana, cuantificabilidad, modificabilidad, mantenibilidad y portabilidad.

2.3 MODELOS PARA EL DESARROLLO ESTRUCTURADO DEL SOFTWARE

El Ingeniero de Software cuenta con una serie de principios, técnicas y métodos que constituyen sus herramientas de trabajo para ayudarle a mejorar la calidad y productividad del Software que desarrolla. Inicialmente estaban enfocadas sólo a la codificación, pero en la actualidad cubren todas las fases del ciclo de vida del desarrollo del Software. Entre las principales se encuentran las diferentes metodologías para el desarrollo estructurado de sistemas.

Algunos de los factores que deberían considerarse al adaptar una metodología son:

- Las aplicaciones que se desean resolver.
- El tamaño aproximado de éstas, incluyendo el tamaño del código, número de personas involucradas en el proyecto y tiempo requerido para la elaboración.
- El número de versiones de las aplicaciones que se realizarán o el tiempo de vida que se espera.

Además de estas características organizacionales, una buena metodología debería:

- Cubrir todo el ciclo de vida del desarrollo del Software.
- Permitir una fácil transición entre fase y fase.
- Ser aplicable a una gran variedad de proyectos.
- Ser fácil de comprender y utilizar.
- Contar con un soporte automatizado.

Aunque existen varios modelos que enfatizan algunos de los aspectos del ciclo de vida, ninguno de ellos es apropiado para

todas las aplicaciones. Es importante definir el modelo de ciclo de vida para cada proyecto, ya que éste da las bases para evaluar y controlar las distintas actividades requeridas para desarrollar y mantener un producto de Software. A continuación se presentan algunos de los principales modelos para el desarrollo del ciclo de vida.

2.3.1 MODELO DE FASES O CASCADA

El modelo de fases o cascada divide el ciclo de vida en una serie de fases y el Software va secuencialmente de una a otra. Cada fase requiere una información de entrada, misma que utiliza en un proceso, dando como resultado un producto bien definido. Las fases que establece este modelo son: análisis, definición, diseño, codificación, prueba, documentación y mantenimiento. La figura 2.4 muestra el modelo básico.

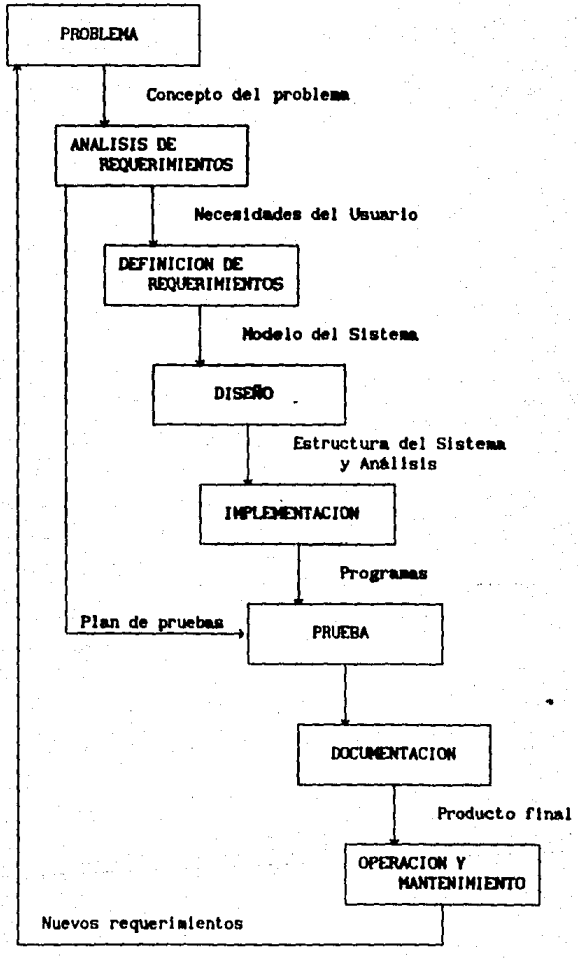


Figura 2.4 Modelo básico de fases o cascada del ciclo de vida del Software.

Hay un aspecto muy importante en la solución de un problema real con ayuda de la computadora, el cual no se menciona explícitamente como elemento del ciclo de vida y, sin embargo, es el principio obligado para el desarrollo del Software. Para que se dé una solución primero tiene que haber un problema.

El objetivo de la fase de análisis de requerimientos es determinar detalladamente las características del problema a fin de poder proponer soluciones.

Los resultados del análisis anterior quedan definidos en la fase de especificación de requerimientos produciendo un documento, el cual debe describir con precisión qué es lo que hará el sistema deseado. Dicho documento puede presentarse como contrato al usuario y, a su vez, servir de base para quienes desarrollen el Software.

El diseño del sistema consiste en determinar cómo se cumplirán los requerimientos especificados.

La fase de implementación radica en la traducción de las especificaciones del diseño a un lenguaje de programación, para así producir el código fuente.

En la fase de prueba se intenta encontrar tantos errores como sea posible determinando su procedencia y garantizar que el sistema cumple con sus especificaciones. Una vez que se ha realizado la instalación del mismo, comienza la fase de mantenimiento, en la que se hacen las correcciones necesarias, bien sea por fallas en la ejecución o por cambios en los requerimientos.

Una modificación del modelo de fases enfatiza la elaboración de puntos de prueba o **milestones** ("milestone es la especificación de un evento demostrable en el desarrollo de un proyecto. Son puntos para medir el progreso" [Zelkowitz 79]), documentos y reportes que mejoran la visión del estado de desarrollo en el que

se encuentra el proyecto y, por lo tanto, facilitan su evaluación. Fairley [Fairley 85] detalla cuáles son los puntos de prueba, documentos y reportes necesarios y el momento en que deben ser elaborados.

Otro aspecto del modelo de fases es el basado en los costos involucrados en cada uno de los pasos. Este enfoque permite observar, entre otras cosas, la importancia, en relación al costo, que tiene un buen análisis y definición de requerimientos, ya que un cambio o error en los mismos implica un costo bastante elevado en las fases siguientes.

El modelo de cascada plantea una fase más, de la cual no se ha hecho mención aún, muy importante y determinante en la vida futura del sistema, la documentación. Aunque en la figura 2.4 se localiza después de las pruebas y se menciona que las fases siguen un proceso secuencial, en el caso de la documentación esto no sucede. Para que un documento reporte correctamente un sistema y sea de utilidad durante la operación y mantenimiento del mismo, es indispensable que se comience a elaborar desde la primera fase del desarrollo. Más aún, a pesar de que el sistema ya esté en operación, la documentación debe ser actualizada constantemente, para que en todo momento refleje con exactitud las características, posibles deficiencias, correcciones y ampliaciones que el sistema contiene.

2.3.2 MODELO DE PROTOTIPO

Este modelo propone el desarrollo de un prototipo del sistema antes de la elaboración total del mismo. Esto permite una mejor comunicación con el usuario, entender a mayor profundidad el problema, así como tomar decisiones de diseño con bases experimentales.

Para el desarrollo del prototipo se sigue el esquema del modelo de fases, considerando que se trata de un sistema preliminar. Cuando el prototipo del sistema está terminado y es aceptado por el usuario se lleva a cabo el desarrollo total del sistema, basándose en las experiencias y resultados obtenidos [SEN 82].

2.3.3 MODELO DE VERSIONES SUCESIVAS

Es una extensión del modelo de protopipo. El producto inicial se va refinando hasta llegar al sistema deseado. Una característica importante es que cada versión es un producto capaz de realizar ciertas funciones correctamente [Fairley 85].

2.3.4 MODELO OPERACIONAL

Sigue los pasos del modelo de fases modificando la forma de llevarlos a cabo. La diferencia principal entre ambos es la forma de especificar los requerimientos y el diseño: el modelo de fases trata el sistema como una caja negra describiendo qué hace pero no cómo lo hace, mientras que el modelo operacional especifica el sistema en términos de una estructura independiente de la implantación. Zave [Zave 84] hace una comparación detallada entre el modelo de fases y el operacional.

2.3.5 MODELO DE ENTREGA EVOLUTIVA

Toma como base el modelo de cascada y lo modifica

al considerar tres nuevos principios: entregar algo al usuario, recibir la retroalimentación del mismo y ajustar el diseño y los objetivos de acuerdo a la realidad observada. Es similar al modelo de prototipos, pero en este caso no se piensa en la posibilidad de hacer a un lado el prototipo, sino en ir refinando poco a poco el producto [Spinrad 85].

2.3.6 MODELO EN ESPIRAL

Considera el desarrollo del Software como la composición de varias fases. En cada fase se producen prototipos y el análisis de riesgos [Boehm 86].

2.3.7 COMPARACION DE LOS MODELOS

El modelo de fases se considera la base teórica sobre la que se fundamentan los modelos, para el desarrollo estructurado del Software, mencionados anteriormente. Sin embargo, para algunos autores ([McCracken 82] y [Gladden 82]) presenta ciertas inconveniencias por lo que no ha sido completamente aceptado.

Dichos autores aseveran que el modelo de fases es muy rígido al obligar al sistema a seguir una trayectoria lineal en su desarrollo; sin embargo, gracias a ello, se puede evaluar el progreso del sistema comparándolo con un plan bien definido. Por otra parte, la trayectoria que sigue el proceso no es lineal sino iterativa; el desarrollo de una fase, en ocasiones, puede tener consecuencias en fases anteriores, lo que significa interrumpir la secuencia del ciclo de vida y regresar a modificar las fases previas. Otra característica es que el mismo esquema de ciclo de vida puede aplicarse a diferentes sistemas haciéndolo flexible,

ya que permite utilizar diferentes herramientas en las distintas fases dependiendo del tipo de aplicación.

Se puede concluir que el criterio de diseño planteado por el modelo en cascada es aplicable y que la clave de su uso es el empleo de herramientas adecuadas en cada una de sus fases. Se cuenta con poca información que oriente en la selección de las metodologías óptimas en cada fase. Tripp junto con otros autores, proponen en su artículo "Evaluation of Software Development Cycle Methodology Implementation" [Tripp 82] un proyecto orientado a definir este tipo de información.

La mayoría de las técnicas que se han desarrollado hasta el momento son aplicables a las últimas fases del ciclo de vida. La fase de prueba cuenta con las técnicas más avanzadas, le sigue la fase de implementación o codificación; en la actualidad se está dando gran importancia al desarrollo de herramientas adecuadas para el análisis y especificación de requerimientos.

CAPITULO 3. DESARROLLO DE UN SISTEMA INTEGRADO PARA EL CONTROL ESCOLAR

3.1 ANTECEDENTES DE LA APLICACION

La experimentación es una forma de verificar si un principio propuesto es correcto y aplicable a determinados problemas. De ahí que se pensó utilizar este método con el fin de evaluar las técnicas que la Ingeniería de Software presenta para mejorar la productividad y calidad de los sistemas.

Yourdon [Yourdon 89] describe una serie de condiciones que deben considerarse al elegir un proyecto piloto:

a) Representar un riesgo y costo bajos para quien lo realiza. Por tratarse de una prueba no hay garantía de alcanzar los resultados deseados.

b) Tener una dimensión proporcionada, es decir, ni demasiado pequeño que no produzca resultados significativos ni demasiado grande que resulte complejo su desarrollo y evaluación.

c) Dar solución a problemas reales de manera que pueda utilizarse y probarse en forma sistemática como parte de ambientes de trabajo concretos.

d) Ser secundario para el funcionamiento general de la empresa, de tal modo que un fallo no acabe con el proyecto y constituya un descrédito para la computación.

e) Ser evaluable en términos del tiempo establecido para su desarrollo, eficiencia, errores encontrados después de su distribución, etcétera.

El proyecto piloto descrito en esta tesis intenta cumplir los puntos anteriores. En especial el de responder a una necesidad concreta, como es el fomentar el uso de las computadoras en las instituciones escolares mexicanas; con este propósito se desarrolló un sistema para el control escolar de los aspectos académicos (calificaciones, datos de alumnos y profesores, etcétera).

Basándose en una experiencia personal se detectó la necesidad de desarrollar un sistema que manejara información de tal forma que permitiera llevar el control escolar de manera eficiente, sencilla y no demasiado costosa. Los sistemas con que hasta entonces se contaba resultaban lentos, no se adaptaban a las necesidades o su costo era muy elevado.

Para poder conocer si la necesidad percibida hasta entonces era algo generalizado en el país o se reducía a un determinado grupo de instituciones, se elaboró un cuestionario que se envió a algunas instituciones, además de concertar entrevistas con empresas que distribuyen este tipo de Software y con quienes lo utilizan. Los cuestionarios resueltos aportaron la siguiente información:

* En muy pocas instituciones escolares se utiliza la computadora para llevar este tipo de control. Las principales razones que se enumeran son: falta de recursos, poco conocimiento de los beneficios que se obtienen y experiencias de resultados negativos.

* En general, se detectó éxito en las instituciones que a través de los años han ido elaborando y depurando su propio sistema, además de contar con personal capacitado en el área.

* Los programas elaborados en otros países, generalmente, no proporcionan los resultados esperados por manejar criterios de evaluación diferentes a los de México y por su costo excesivo.

* Se observó falta de interés ya que sólo el 33.3% de las instituciones encuestadas respondieron. Sin embargo, las que atendieron a la solicitud de información, aportaron opiniones favorables respecto al uso de la computadora en este campo.

A pesar del optimismo, no deja de verse la falta de conocimiento en esta área, ya que varias instituciones se sienten satisfechas de elaborar sus boletas de calificaciones en la computadora utilizando un procesador de palabras. Esto manifiesta la urgencia de una preparación a todo nivel que amplíe los horizontes de la computación en México y lleve al máximo aprovechamiento de los recursos disponibles.

Los motivos anteriores han originado y dirigido esta investigación que llevó a la implementación del Sistema Integrado para el Control Escolar (SICE).

3.2 DESARROLLO DE SICE

La práctica demuestra que un sistema de Software puede desarrollarse de muy diversas formas; la variedad de herramientas y aplicaciones existentes proporcionan al ingeniero el campo propicio para hacer uso de su capacidad creadora y le ayudan en la tarea de concretizar aquello que ha concebido en su mente.

SICE no es la excepción, existen infinidad de posibilidades que pueden llevar a una solución adecuada; sin embargo, en la práctica, es indispensable elegir una de ellas. En este caso se

optó por seguir los pasos propuestos por el modelo de cascada del ciclo de vida del Software y aplicar las técnicas de programación estructurada en el desarrollo del sistema, con el fin de evaluar si ésto realmente ayuda a mejorar la productividad y calidad del mismo.

Para dar una solución antes tiene que existir un problema. Con el objeto de que SICE fuera una aplicación real, que resultara útil al responder a una necesidad concreta y que pudiera probarse en la práctica por personas ajenas a su desarrollo, se pidió la colaboración del Instituto Cultural, A.C. (Miguel Angel de Quevedo 1190, México, D.F.), el cual proporcionó la información y apoyo necesarios para la realización de este proyecto.

El Instituto Cultural es un centro de enseñanza que en la actualidad cuenta con mil alumnas aproximadamente, desde el nivel preescolar hasta preparatoria.

Otro aspecto favorable para este trabajo, además de poder contar incondicionalmente con la información requerida, es el hecho de que quien lo realiza tiene a la vez conocimientos de computación y del funcionamiento y organización de este tipo de instituciones.

A continuación se describen brevemente las fases del ciclo de vida del Software, seguidas en el desarrollo de SICE:

3.2.1 ANALISIS DE REQUERIMIENTOS

Para profundizar en un problema con el fin de darle solución es fundamental hacer un análisis previo del mismo.

Si se asumen todos los aspectos en los que puede utilizarse un sistema de cómputo como solución se corre el riesgo de querer abarcar tanto que resulte imposible obtener resultados favorables.

En una institución escolar es factible el uso de la computadora en infinidad de campos; cabe mencionar el control de aspectos administrativos y de aspectos académicos, además de servir de apoyo a la tarea educativa.

Cada una de ellas requiere tal cantidad de recursos humanos y materiales para su implementación que resulta prácticamente absurdo incluirlas todas en un sólo proyecto; de ahí la importancia de ser realistas desde el principio, conocer las capacidades y limitaciones existentes y abarcar únicamente aquello para lo que es posible dar una respuesta, mediante dos pasos.

a) Delimitación del problema. Se han mencionado algunos campos en los que puede incidir la computación en una institución escolar; sin embargo, en este punto ya no se habla de problemas en general, es preciso darles una identidad específica. La figura 3.1 muestra un organigrama sintetizado del Instituto Cultural, A.C.

INSTITUTO CULTURAL, A. C.

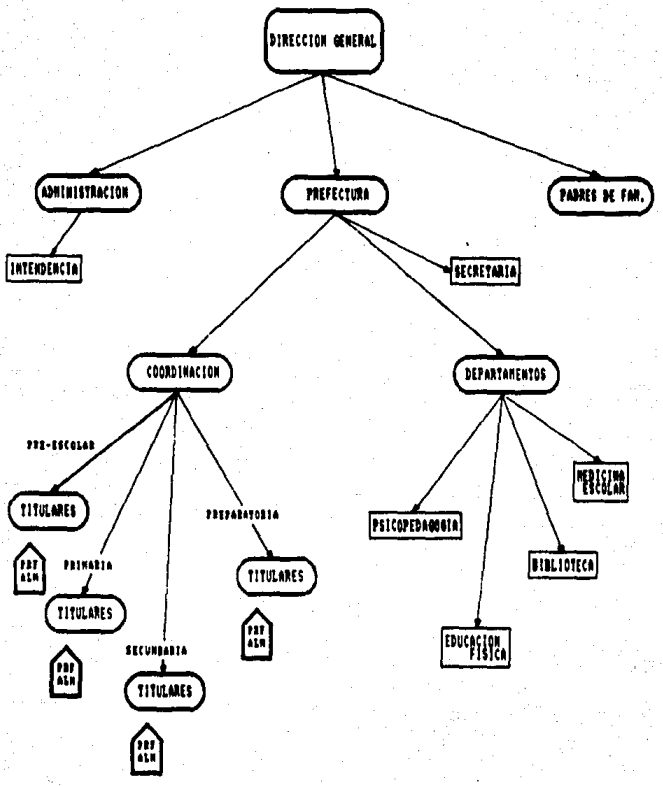


Figura 3.1 Organigrama sintetizado del Instituto Cultural, A.C.

En primera instancia, la institución mostró interés en un sistema que permitiera el manejo de la información relacionada con alumnos, maestros y materias en el aspecto académico, desde la sección de preescolar hasta preparatoria.

Posteriormente, al evaluar los recursos de personal y tiempo con que se cuenta para este proyecto, se limitó a la sección de secundaria. En la figura 3.2 se muestra el área de trabajo de SICE dentro de la institución.

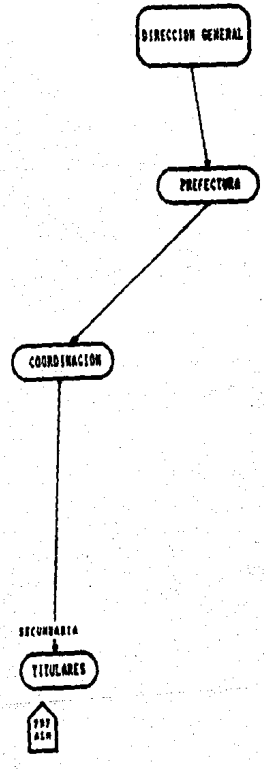


Figura 3.2 Área de trabajo de SICE dentro del Instituto Cultural, A.C.

b) Análisis del sistema. Se orienta a definir, con la mayor exactitud posible, las funciones que se pretende que realice y los datos que deben incluir éstas; para obtener la información que permita establecer ésto, se emplearon los siguientes métodos:

- Las entrevistas individuales con personal de la institución y de compañías productoras de sistemas similares,
- la recopilación de los documentos que se producen manualmente en la actualidad y
- la experiencia personal.

Obtenida la información se analizó y sintetizó de acuerdo al método que propone Perkinson [Perkinson 84] en el que a partir de los datos incluidos en algunos de los documentos significativos, se definen las funciones que deben implementarse; se enlistan los documentos considerados en el análisis y los datos contenidos en cada uno de ellos se desglosan al elaborar la especificación de requerimientos:

- Boleta de calificaciones (mensual, bimestral).
- Lista de asistencia mensual (por grupo).
- Lista de calificaciones mensuales (por materia, por grupo)
- Concentrado mensual de calificaciones (por grupo).
- Directorio de alumnos.
- Directorio de maestros.
- Datos generales de alumnos.
- Datos generales de maestros.
- Reporte estadístico mensual.

Como resultado de las entrevistas y la experiencia personal se llegó a la conclusión de que SICE debe ser un sistema fácil de usar, de bajo costo, que simplifique y agilice el trabajo y que, con sus resultados eficaces, sirva de estímulo para ampliar el uso de la computadora en los campos que aún no se atienden.

3.2.2 ESPECIFICACION DE REQUERIMIENTOS

A continuación se describen los requisitos que debe cumplir el sistema de manera que puedan ser entendidos y aceptados por quienes representan a la institución que hará uso de él y los que van a producirlo.

De acuerdo a la estructura propuesta por Pomberger [Pomberger 84] el documento resultante es el siguiente:

a) Situación actual. En el presente, la institución utiliza la computadora para el manejo de ciertos aspectos administrativos. Se quiere ampliar el campo de aplicación de ésta y abarcar, progresivamente, otras áreas de la institución, entre las que destacan:

- Aspectos académicos.
- Departamento de psicopedagogía.
- Departamento de medicina escolar.
- Biblioteca.

Como proyecto inicial se piensa en utilizar la computadora para mejorar la eficiencia en el procesamiento de información del área académica en la sección de secundaria; las restantes aplicaciones se posponen en futuras expansiones, teniendo como primera prioridad la ampliación a las otras secciones (preparatoria, primaria y jardín de niños).

Se pretende que el sistema sea eficaz en cuanto al tiempo de impresión de los documentos requeridos, por presentarse limitaciones en los períodos disponibles para su elaboración.

La institución cuenta con 270 alumnos en la sección de secundaria. Todos ellos cursan las 12 materias designadas como

obligatorias por la SEP. La materia de tecnológicas representa un caso excepcional, ya que los alumnos pueden elegir de entre tres talleres, intregrando para ésto nuevos grupos.

b) Forma y ambiente de operación del sistema. Estará instalado y centralizado en una computadora personal. Quienes harán uso de él pueden clasificarse en:

- Administración. Los usuarios son el administrador y el secretario quienes cuentan con acceso ilimitado a todo el sistema.
- Consulta. Lo constituyen el coordinador, los titulares y los maestros pudiendo utilizar sólo determinadas partes del sistema.

Se calcula que se tendrán aproximadamente dos usuarios del primer tipo y veinticinco del segundo. El uso del sistema se concentra en la semana previa a cada evaluación mensual.

El administrador del sistema requiere amplios conocimientos de computación, pues será quien se encargue de su operación y mantenimiento; es recomendable que el secretario entienda el uso del sistema operativo y el funcionamiento elemental de la máquina; el resto de usuarios pueden utilizarlo si conocen lo elemental para cargarlo y responder a las órdenes del mismo.

c) Requerimientos funcionales. El sistema debe permitir que se realicen con la información las siguientes funciones: captura, almacenamiento, asignación, modificación, eliminación, filtrado, ordenamiento, búsqueda, lectura, cálculo, impresión y consulta.

Los datos que SICE utiliza son:

- Datos personales de alumnos y profesores.
- Directorio de padres de familia.
- Número de alumnos por grupo.
- Calificaciones por alumno, por grupo y por materia.
- Control de asistencias de alumnos y maestros.

Los cálculos comprenden:

- Promedios de alumnos por periodo de evaluación, siguiendo el criterio de la SEP.
- Promedios o medias por materia por grupo.
- Promedios o medias por evaluación por grupo.
- Porcentajes de aprobación por grupo.
- Edad promedio del grupo.
- Alumnos con mayor rendimiento.
- Alumnos con menor rendimiento.

Proporciona los siguientes reportes impresos:

- Boleta de calificaciones mensual por alumno. (1)
- Boleta de calificaciones bimestral por alumno. (2)
- Lista de asistencia mensual por grupo. (3)
- Lista de calificaciones mensuales por materia y por grupo. (4)
- Concentrado mensual de calificaciones por grupo. (5)
- Directorio de padres de familia. (6)
- Directorio de maestros. (7)
- Datos generales de alumnos. (8)
- Datos generales de maestros. (9)
- Reporte estadistico mensual. (10)

En la siguiente figura se muestra la tabla de los datos incluidos en cada uno de los reportes:

DATOS	1	2	3	4	5	6	7	8	9	10
Nombre Institución	•	•	•	•	•	•	•	•	•	•
Dirección Instit.	•	•								
Teléfono Instit.	•	•								
Fecha	•	•	•	•	•	•	•			
Nombre Alumno	•	•	•	•	•	•		•		•
Núm. Expediente Al.	•	•			•			•		
Lugar Nacimiento								•		
Fecha Nacimiento								•		
Sexo								•		
Nombre Padre						•		•		
Ocupación Padre								•		
Nombre Madre						•		•		
Ocupación Madre								•		
Dirección	•	•				•		•		
Código Postal	•	•				•		•		
Teléfono	•	•				•		•		
Sección	•	•	•	•	•	•		•		
Grado	•	•	•	•	•	•		•		
Grupo	•	•	•	•	•	•		•		
Núm. Lista	•	•	•	•	•					
Núm. Evaluación	•	•		•	•					
Nombre Materia	•	•	•	•	•					
Núm. Materia		•	•	•	•					
Calificación	•	•			•					
Promedio Men. Al.	•	•			•					
Promedio Men. Gpo.	•	•			•					
Promedio Bim. Al.		•			•					
Promedio Bim. Gpo.		•			•					
Promedio Materia G.	•	•			•					
Promedio Final Al.		•			•					
Promedio Final Gpo.		•			•					
Faltas Asistencia	•	•			•					
Retardos	•	•			•					

Figura 3.3 Tabla de los datos proporcionados en los reportes.

DATOS	1	2	3	4	5	6	7	8	9	10
Nombre Maestro			•	•			•		•	
Sexo									•	
Estado Civil									•	
Tùtulo									•	
RFC									•	
Direcciön Mtro.							•		•	
Cödigo Postal Mtro.							•		•	
Teléfono Maestro							•		•	
Año Ingreso Instit.									•	
Nom.Materia Imparte							•		•	
% Reprobaciön Mat.										•
% Reprobaciön Gpo.										•
% Reprobaciön Secc.										•
Nom.Alum.> Prom.Mat.										•
Nom.Alum.> Prom.Gpo.										•
Nom.Alum.> Prom.Sec.										•
Nom.Alum.< Prom.Mat.										•
Nom.Alum.< Prom.Gpo.										•
Nom.Alum.< Prom.Sec.										•

Figura 3.3 Continuación.

En la gráfica que se muestra a continuación (figura 3.4) se observa la frecuencia de elaboración de los reportes.

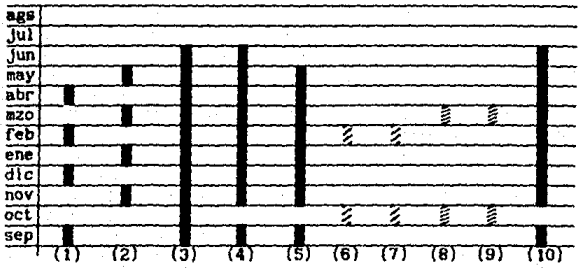


Figura 3.4 Frecuencia de elaboración de reportes.

La figura 3.5 presenta la normativa de la SEP para calcular los promedios finales del curso.

Suma de las calificaciones de los 4 periodos.	Calif final
40 39 38	10
37 36 35 34	9
33 32 31 30	8
29 28 27 26	7
25 24	6
23 22 21 20	5

* Además puede obtenerse la calificación reprobatoria (5) si se cumple alguna de las siguientes condiciones:
  - que el alumno tenga reprobados los dos últimos periodos.
  - que el alumno tenga reprobados tres periodos indistintamente.

Figura 3.5 Normativa de la SEP para calcular promedios finales.

d) Requisitos no-funcionales. El sistema debe incluir alguna forma de protección de los datos que permita controlar el acceso a la información y restringir el manejo confidencial según sea el caso, limitándola de acuerdo al puesto desempeñado por el usuario. Las funciones para cada tipo de usuario son:

- Administrador:
  - Inicializar el sistema.
  - Establecer claves de seguridad para el acceso al mismo.
  - Accesar ilimitadamente toda la información, con posibilidades de captura, consulta, impresión, respaldo y/o modificación de la misma sólo para fines de mantenimiento.
  - Respaldar archivos de datos.
  - Mantener el sistema.
- Secretario:
  - Accesar ilimitadamente toda la información, con posibilidades de captura, consulta, impresión, respaldo y/o modificación de la misma.
- Coordinador de la sección:
  - Consultar toda la información referente a su sección.
- Titular de grupo:
  - Consultar toda la información referente a su grupo.
- Maestros:
  - Consultar la información respecto a sus materias.

Si alguno de los usuarios intenta entrar a las áreas que no le están permitidas, el sistema le da tres oportunidades, por considerar los posibles errores al introducir la clave, después de lo cual le niega por completo el uso del mismo.

El sistema debe ofrecer facilidad para realizar cambios, especialmente en la forma de calcular los promedios y en el número de periodos de evaluación.

e) Interface con el usuario. La comunicación con el usuario, a través de la pantalla y el teclado, debe ser sencilla, protegiéndolo contra los errores que pueden ocurrir por falta de conocimiento del sistema o de las funciones permitidas a cada usuario. Para la impresión se utilizarán hojas en blanco y el sistema se encargará de darles el formato (este formato aún está sujeto a la aprobación de la institución).

f) Comportamiento en los errores. Debe rechazar errores al contestar las órdenes del sistema; señalar que es una respuesta inadecuada y permanecer en el estado previo al error.

Si recibe una clave de seguridad equivocada, dará tres oportunidades para corregirla, si después de esto aún no se obtiene la correcta, rechazará por completo el acceso al sistema; el objetivo es anular el ingreso utilizando el método de prueba y error hasta encontrar una clave autorizada.

Si falta algún archivo de datos, hacerlo notar y salir del sistema. Debe tener filtros de datos que no permitan introducir caracteres equivocados, dependiendo de si están limitados a caracteres numéricos, alfabéticos o alfanuméricos; si ocurre este tipo de error, lo señala y no lo acepta, permaneciendo en el estado anterior.

g) Requerimientos de documentación. Debe incluirse un manual dirigido a los usuarios comprendidos en el tipo consulta, con las indicaciones necesarias para la operación del sistema y otro para la administración con todo lo necesario para el mantenimiento del sistema, conteniendo un listado del código debidamente comentado.

h) Criterios de aceptación. El sistema será aceptado si: cumple con los requisitos en el tiempo de impresión; el equipo necesario para su operación se reduce a una computadora personal y una impresora; su manejo es accesible a personas sin conocimientos de computación y proporciona seguridad para el control del acceso a la información.

Aún cuando en la fase de especificación de requerimientos se indicó la utilidad de incluir un glosario y un índice como parte de la misma, por la extensión que en este caso tiene tal documento no se consideró necesario añadir esa información.

Una vez realizada la especificación, antes de considerarla definitiva, se presentó a diferentes miembros de la institución con el fin de asegurar que todos los requisitos estuvieran incluidos para ser aceptado por ellos. Los requisitos planteados fueron analizados, no encontrando contradicción en ellos.

Para el desarrollo del sistema se requerirán los siguientes recursos técnicos:

- Computadora personal de 512 Kb. de memoria mínima.
- Dos drives para discos flexibles.
- Impresora.
- Discos flexibles de 5.25".
- Sistema operativo MS-DOS versión 2.0 o posterior.
- Compilador Turbo Pascal 4.0
- Procesador de palabras.

En esta aplicación, el personal que llevará a cabo el desarrollo del sistema es la sustentante asistida por su asesora. El personal preparado para operar el mismo ya forma parte de la institución.

En este caso no se realiza una justificación económica de la aplicación ya que su objetivo no es comercializarlo.

El definir los requerimientos utilizando las herramientas facilitadas por la Ingeniería de Software, se dejó para la fase de diseño, con el fin de incluirlas en esa sección, para dar claridad a su estructura y funcionamiento y evitar repeticiones.

3.2.3 DISEÑO

El diseño de SICE se basa en la especificación de requerimientos obtenida en la fase previa; se realizó siguiendo el método de refinamiento por pasos sucesivos, que incluye los

conceptos de programación estructurada, en especial de modularidad.

Por las características que presenta SICE, al contar con un gran número de datos, que deben ser manejados con ciertas restricciones de seguridad, se consideró que una forma adecuada para su desarrollo era seguir los lineamientos trazados para el diseño de sistemas de bases de datos.

En este tipo de proyectos, por medio de una determinada forma de organización y manejo de los datos, se obtiene el control centralizado de los mismos. Esto permite, entre otras cosas:

- Evitar la redundancia impidiendo que cada aplicación o usuario tenga sus propios archivos privados.
- Disminuir inconsistencia en los datos al no permitir que haya dos entradas diferentes para el mismo dato y eliminar así la posibilidad de que éstas no concuerden o que el cambio hecho a una de ellas no se propague a la otra.
- Compartir datos entre varias aplicaciones con el fin de proporcionar a cada una lo que necesita sin tener que crear nuevos archivos.
- Asegurar que el único modo de acceso a la base de datos sea por los canales establecidos y restringir el acceso a datos confidenciales.
- Conservar la integridad de los datos aplicando filtros en la captura de ellos y procesos de validación.

Existen tres modelos principales para estructurar y manejar los datos:

- Modelo jerárquico. Los datos se representan por medio de una estructura de árbol. En el nivel más alto se encuentra el nodo raíz, formado por un sólo registro,

del cual se deriva el resto. Cada nodo sólo puede tener un padre y las ligas entre ellos deben apuntar en dirección de padre a hijo. Así, para buscar un nodo específico es necesario iniciar en la raíz y recorrerlo de manera descendente.

- Modelo de red. Es una representación no jerárquica de los datos ya que un registro puede ser hijo de más de un padre. Esto permite que el acceso a un nodo determinado pueda iniciarse en cualquier punto de la red y buscarle indistintamente.

- Modelo relacional. Los datos se organizan como una serie de tablas, llamadas relaciones, que pueden ser descritas en columnas y renglones. Cumplen con las siguientes características:
  - cada tabla representa un tipo de registro,
  - cada relación contiene un número específico de atributos (columnas) y un campo identificador único,
  - los atributos están normalizados,
  - no hay renglones duplicados.

Cada uno de estos modelos se ha ampliado y perfeccionado a través de los años. Para obtener una mayor información al respecto se pueden consultar los siguientes textos: [Codd 70], [Albizuri 89], [Date 86], [Ullman 82].

Los manejadores de Bases de Datos, diseñados siguiendo alguno de los modelos antes mencionados, son una de las herramientas más utilizadas debido a que casi todas las aplicaciones requieren de información. Por esta razón, se puede encontrar una gran variedad de manejadores comerciales, tanto para uso específico como para uso general. En el caso concreto de SICE no se utilizó ninguno de estos sistemas dado que, basándose en la experiencia y en los resultados obtenidos en las encuestas realizadas a instituciones escolares, se concluyó que manejar la

información con ayuda de un sistema comercial implicaba, entre otras cosas, un aumento en el costo inicial y mayores requerimientos en espacio de memoria y tiempo de ejecución.

SICE sigue los principios del modelo relacional sin ceñirse por completo a ellos por el tipo de funciones que realiza; SICE podría considerarse como un pequeño manejador de bases de datos cuya función principal es permitir el uso y administración compartida de la información de manera eficiente. Su diseño se divide en manejo de la información y organización de la misma.

3.2.3.1 MANEJO DE LA INFORMACION EN SICE

El manejo de la información puede considerarse como la función principal de SICE, a partir de la cual se hace la descomposición, en subfunciones, hasta llegar a componentes elementales de fácil solución. La figura 3.6 muestra la función general del sistema.

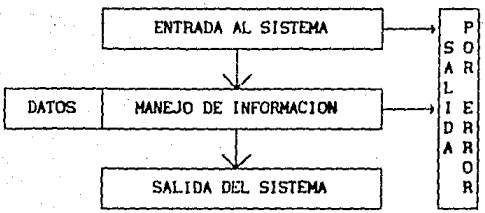


Figura 3.6 Función general de SICE.

Cada uno de estos bloques puede dividirse en funciones más simples. La figura 3.7.a muestra las subfunciones para entrar a SICE; en la figura 3.7.b se contemplan las que manejan la información; la figura 3.7.c presenta la división de funciones

para salir del sistema y la figura 3.7.d indica la secuencia ejecutada para salir del sistema por error irreversible.

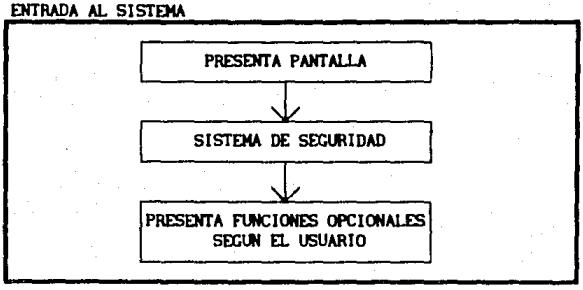


Figura 3.7.a Subfunciones para entrar al sistema.

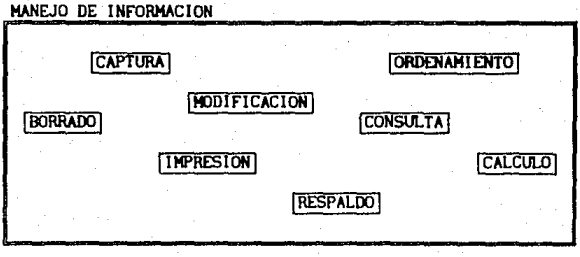


Figura 3.7.b Partes para el manejo de información.

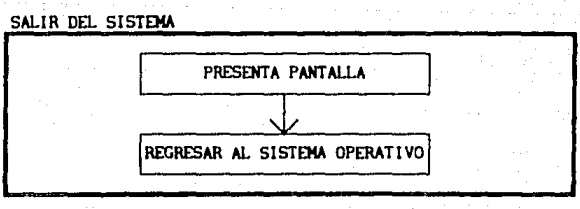


Figura 3.7.c División de funciones para salir del sistema.

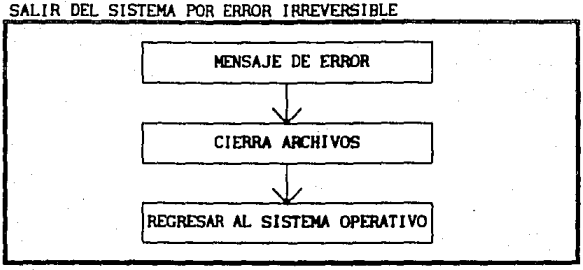


Figura 3.7.d Salir del sistema por error irreversible.

Las subfunciones mencionadas se subdividen en:

a) Entrada al sistema.

1. Presentar pantalla:
   - Dibujar pantalla de presentación.
   - Borrar pantalla de presentación.
2. Sistema de seguridad:
   - Pedir la clave de acceso.
   - Leer la clave de acceso.
   - Codificar la clave de acceso.
   - Leer de archivo las claves autorizadas.
   - Comparar las claves autorizadas con la de acceso.
   - Permitir la entrada si la clave de acceso es correcta (o salir de SICE por error irreversible).
3. Presentar funciones opcionales según el usuario:
   - Dibujar pantalla de opciones de acuerdo a la clave de acceso.
   - Leer información del teclado.
   - Llamar a la función seleccionada.

b) Manejo de información.

1. Captura:
   - Leer datos de la terminal.

- Asignar datos.
- Almacenar en archivos.

2. Ordenamiento lógico:
   - Leer información de archivos.
   - Ordenar información.
   - Almacenar en archivos la información ordenada.
3. Búsqueda secuencial:
   - Leer del teclado el dato a buscar.
   - Leer información del archivo secuencialmente.
   - Comparar con el dato buscado.
   - Desplegar mensaje de éxito o fracaso en la búsqueda.
4. Búsqueda binaria (cfr. [Wirth 87]).
   - Leer del teclado el dato a buscar.
   - Leer información del archivo en forma binaria.
   - Comparar con el dato buscado.
   - Desplegar mensaje de éxito o fracaso en la búsqueda.
5. Consulta:
   - Leer información de archivos.
   - Presentar información en pantalla.
6. Modificación:
   - Leer información de archivos.
   - Presentar información en pantalla.
   - Leer datos de la terminal.
   - Asignar datos.
   - Almacenar en archivos.
7. Borrado lógico:
   - Leer información de archivos.
   - Dar de baja la información.
   - Reordenar el archivo.
8. Impresión:
   - Leer información de archivos.
   - Imprimir de acuerdo a un formato.
9. Cálculo:
   - Leer información de archivos.
   - Realizar un cálculo determinado.

- Almacenar resultados en un archivo.

10. Respaldo:
    - Copiar la información de un archivo en otro.
11. Recolección de basura:
    - Crear un nuevo archivo.
    - Copiar la información que se quiere conservar.
    - Borrar el archivo de datos anterior.

c) Salida del sistema.

1. Presentar pantalla:
    - Recibir señal de salida.
    - Dibujar pantalla de salida.
    - Borra pantalla de salida.
2. Regresar al sistema operativo.

d) Salida del sistema por error irreversible.

1. Presentar pantalla de error:
    - Recibir señal de error.
    - Desplegar el mensaje de error correspondiente.
2. Cerrar archivos.
3. Regresar al sistema operativo.

Como se puede apreciar, algunas funciones no necesitan dividirse (por ejemplo: regresar al sistema operativo), otras pueden simplificarse aún más y de entre ellas, algunas se repiten (por ejemplo: lectura de archivo). A continuación se enlistan las funciones obtenidas después de subdividir hasta llegar a niveles adecuados de simplificación:

FUNCIONES

Archivos

- Crear
- Abrir
- Cerrar
- Posicionar
- Leer
- Escribir
- Borrar

Interface

Presentar en pantalla:
- menús de opciones jerarquizados
- preguntas
- mensajes de error
- instrucciones
- información de ayuda
- formato de captura de datos
- cambio de color del fondo y texto.

Borrar la pantalla.
Emitir sonidos indicando error.
Imprimir datos según el formato.

Datos

Leer del teclado.
Filtrar según pertenezca a:
- un conjunto de opciones
- un conjunto de caracteres:
  - numéricos
  - alfabéticos
  - alfanuméricos.

Asignar
Comparar
Declarar tipos
Almacenar en forma de registro
Modificar
Borrar
Codificar las claves de acceso

Otras

Crear un árbol binario.
Recorrer un árbol binario enorden.
Borrar árbol binario (liberar memoria).
Calcular:
- promedios
- promedios finales (SEP).
- porcentajes.

Regresar al sistema operativo.

Con el fin de indicar con mayor claridad cómo se desempeñan algunas de las funciones antes mencionadas, a continuación se les describe haciendo uso de GAL:

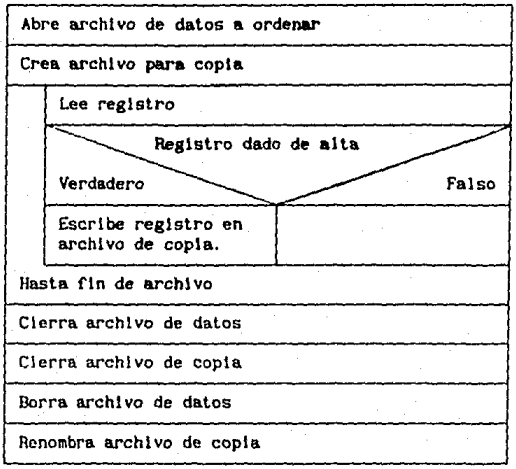


Figura 3.8a Recolección de basura.

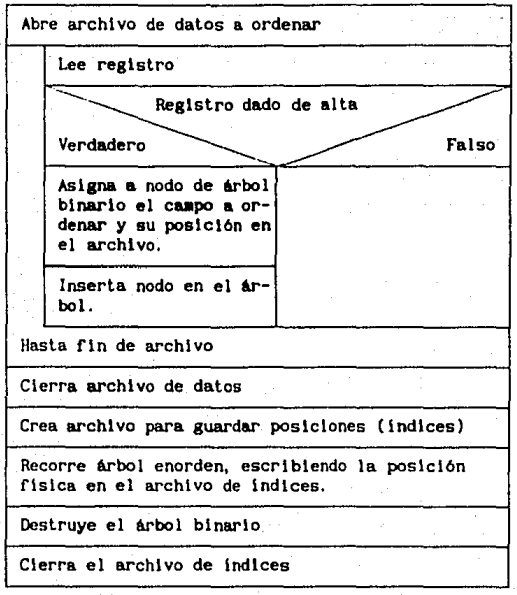


Figura 3.8b Ordenamiento lógico.

3.2.3.2 ORGANIZACION DE LA INFORMACION EN SICE

En la Base de Datos de SICE la información se encuentra organizada de acuerdo a una estructura, para ello se siguió el modelo relacional. Los datos se presentan en forma de relaciones, cada renglón representa un registro o tupla y cada columna un atributo o campo del registro.

Para toda relación debe existir una llave única que permita identificar cada uno de los registros, además debe haber un solo dato en cada atributo del registro; en este aspecto, por

cuestiones de eficiencia en la búsqueda, se hizó una modificación en SICE en los campos de calificaciones, promedios y asistencias en los que se incluye una matriz en vez de un dato atómico.

De acuerdo a la información obtenida del análisis y especificación de requerimientos se organizaron los datos en seis relaciones, cada una constituye un archivo cuyo contenido se muestra a continuación:

Archivo de datos académicos-alumno:

NOMBRE	CARACTERISTICA
Vivo	Booleano; indica si el registro está dado de alta o no.
Número de expediente	Cadena de caracteres numéricos; es la LLAVE, se asigna consecutivamente al inscribir al alumno.
Nombre del alumno	Cadena de caracteres alfabéticos.
Nivel y Grado	Cadena de 2 caracteres numéricos; el primero indica el nivel (1=Pre-Esc; 2=Prim; 3=Sec; 4=Prep); el segundo indica el grado (1o., 2o., etc.).
Grupo	Caracter alfabético (A, B, C, ...)
Materia optativa	Caracter numérico (1=corte; 2=tejido; 3=taqui-mecanografía).
Calificaciones	Matriz cuadrada de 14 x 13 números reales; contiene las calificaciones de 14 materias en 13 evaluaciones.
Ausencias	Matriz cuadrada de 14 x 13 números enteros; contiene las ausencias a 14 materias en 13 evaluaciones.
Apuntador	Entero largo; se utiliza para el manejo interno de la información; apunta a la dirección del alumno en el archivo de datos generales-alumno.

Archivo de datos generales-alumno:

NOMBRE	CARACTERISTICA
Sexo	Caracter alfabético (F,M);
Lugar de nacimiento	Cadena de caracteres alfabéticos;
Fecha de nacimiento	Cadena de caracteres numéricos; no acepta fechas menores a 01/01/70, ni mayores 12/31/99.
Nombre del padre	Cadena de caracteres alfabéticos.
Nombre de la madre	Cadena de caracteres alfabéticos.
Direccción	Cadena de caracteres alfanuméricos.
Código postal	Cadena de caracteres numéricos.
Teléfono	Entero largo.
Ocupación del padre	Cadena de caracteres alfabéticos.
Ocupación de la madre	Cadena de caracteres alfabéticos.

Archivo de datos académicos-maestro

NOMBRE	CARACTERISTICA
Vivo	Booleano; indica si el registro está dado de alta o no.
Clave del maestro	Cadena de caracteres numéricos; es la LLAVE, se asigna consecutivamente al registrar al maestro.
Nombre del maestro	Cadena de caracteres alfabéticos.
Apuntador	Entero largo; se utiliza para el manejo interno de la información; apunta a la dirección del maestro en el archivo de datos generales-maestro.

Archivo de datos generales-maestro:

NOMBRE	CARACTERISTICA
Sexo	Caracter alfabético (F,M).
RFC	Cadena de caracteres alfanuméricos.
Direccción	Cadena de caracteres alfanuméricos.
Código postal	Cadena de caracteres numéricos.
Teléfono	Entero largo.
Titulo	Cadena de caracteres alfabéticos.
Estado civil	Caracter alfabético (S,C,V).
Fecha de ingreso	Cadena de caracteres numéricos.

Archivo de materias:

NOMBRE	CARACTERISTICA
Clave de la materia	Cadena de caracteres alfanuméricos; es la LLAVE, está formada por los caracteres correspondientes al nivel, grado, grupo y número de materia.
Clave del maestro	Cadena de caracteres numéricos.
Total de alumnos	Número entre 0 y 256 (byte); indica el total de alumnos que cursan la materia.
Apuntador	Entero largo; se utiliza para el manejo interno de la información; apunta a la dirección del archivo de datos académicos-alumno, del primer alumno, por orden alfabético, que cursa la materia.
Clases impartidas	Vector de números enteros; contiene las clases impartidas por el maestro, en 8 periodos de evaluación.

Archivo de grupos:

NOMBRE	CARACTERISTICA
Clave del grupo	Cadena de caracteres alfanuméricos; es la LLAVE; está formada por los caracteres correspondientes al nivel, grado y grupo.
Promedio del grupo	Matriz de números reales; contiene los promedios del grupo por materia y general por evaluación.
Total de alumnos	Número entre 0 y 256 (byte); indica el total de alumnos que pertenecen al grupo.
Apuntador	Entero largo; se utiliza para el manejo interno de la información; apunta a la dirección del archivo de datos académicos-alumno del primer alumno, por orden alfabético, que forma parte del grupo.

El registro de datos por grupo podría parecer redundante con respecto a la información contenida en el archivo de materia; sin embargo, es necesario definirlo debido al caso excepcional presentado por la materia de tecnológicas, para la que se requiere formar grupos diferentes a los constituidos para las materias restantes, donde los alumnos de un grupo toman las mismas materias.

Además de los archivos antes mencionados, existen los directorios de índices utilizados para hacer búsquedas de acuerdo a un orden descendente establecido. La figura 3.9, muestra las claves respecto a las cuales se pueden hacer ordenamientos, mientras que la figura 3.10 presenta la interconexión entre los archivos y los directorios de índices.

DATOS DE:	CLAVE
Alumno	Número de expediente Nombre del alumno Nivel + grado + grupo Nivel + grado + grupo + Nombre del alumno
Maestro	Clave del maestro Nombre del maestro
Materia	Clave de la materia Número del maestro
Grupo	Clave del grupo

Figura 3.9 Claves de ordenamiento.

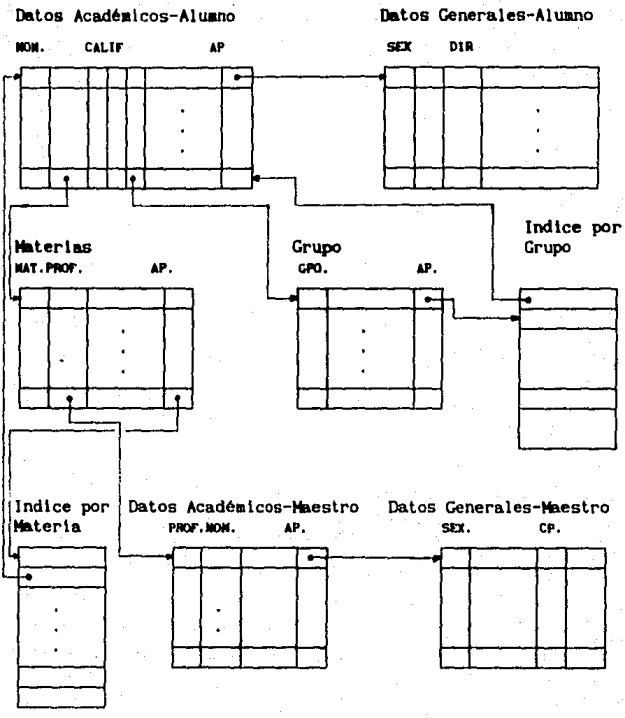


Figura 3.10 Interconexión entre los archivos y los directorios de índices.

Las claves de acceso al sistema también son almacenadas en un archivo, codificadas de acuerdo al algoritmo de Rivest, Shamir y Adleman [Rivest 78]; en éste, la clave se codifica de la siguiente forma: se le representa como un número M, elevado a una potencia específica e y se toma el residuo obtenido al dividir el resultado entre el producto n de dos números primos determinados. La seguridad del sistema reside en parte en la dificultad para descubrir los números primos utilizados.

Las modificaciones a la información provocan, durante la sesión, actualizaciones lógicas en la Base de Datos. Debido a que es mínima la cantidad de basura que puede acumularse en los archivos, las actualizaciones físicas de éstos, sólo se realizan al finalizar el curso escolar y el resultado se almacena en disco flexible para guardar la información como historia. Los directorios son reordenados cada vez que sufren cualquier modificación.

El sistema está dividido en módulos, situados a diferentes niveles dentro de la estructura del mismo donde cada nivel puede incluir más de un módulo. La figura 3.11 muestra la estructura modular de las funciones del sistema, indicando los niveles y componentes de éste.

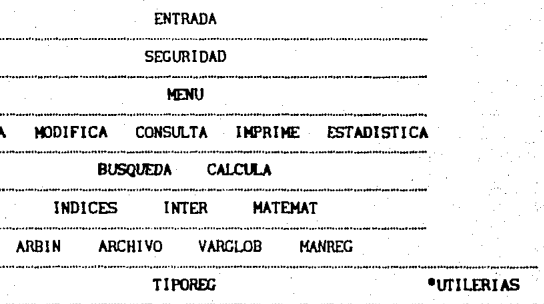


Figura 3.11 Estructura modular de las funciones de SICE.

De entre los módulos de SICE, se pueden considerar como principales, de acuerdo a su función, los siguientes:

1) SEGURIDAD- Restringe el acceso al sistema y proporciona a cada usuario la vista de información que le corresponde.

2) MENU- Presenta al usuario menús jerarquizados que le ofrecen, como opciones, las operaciones que puede solicitar al sistema.

3) CAPTURA- Recibe datos del usuario y los almacena ya filtrados en el archivo correspondiente.

4) CONSULTA- Permite al usuario consultar la base de datos mostrándose en la pantalla los datos que le solicitan.

5) MODIFICA- Actualiza los datos capturados anteriormente.

6) ESTADISTICA- Realiza el cálculo de los promedios y datos estadísticos.

7) IMPRIME- Elabora los reportes en papel.

8) BUSQUEDA- Con base en una clave proporcionada por el usuario, o por el sistema, realiza búsquedas secuenciales y binarias, para actualizar y consultar la base de datos.

9) INDICES- Crea archivos que contienen la posición lógica que ocupan los registros en los archivos de datos correspondientes, de acuerdo a una clave de ordenamiento determinada. Para ordenar la información utiliza un árbol binario.

10) INTER- Proporciona medios para facilitar la interface entre usuario y sistema, como son filtros de datos, formatos de presentación, etcétera.

11) ARCHIVO- Define los tipos de archivos de datos y los procedimientos necesarios para su manejo.

12) TIPOREG- Define los seis tipos de registros de la base de datos; conteniendo datos de alumnos, profesores, materias y grupos.

13) MANREG- Permite el manejo de los registros, limitando su uso a través de determinados procedimientos.

14) UTILERIAS- Incluye procedimientos, independientes del tipo de aplicación, para llevar a cabo el manejo de la pantalla y de la impresora; desplegar mensajes de error; cambios de color en el texto, etcétera.

De acuerdo a su papel dentro del sistema, cada módulo presenta diferentes grados de cohesión; por ejemplo, el módulo de búsqueda secuencial es funcional (cohesión fuerte) ya que todos sus elementos están orientados a la ejecución de la misma función; sin embargo, el módulo de utilería es coincidental puesto que agrupa procedimientos aislados que pueden utilizarse en funciones diferentes.

Un aspecto importante dentro del diseño es lo que se conoce como Ingeniería Humana. En ella se busca que el sistema sea desarrollado pensando en quienes van a utilizarlo, de manera que resulte agradable y sencillo su manejo.

Para la producción de SICE se tomó en consideración este aspecto y por ello la interface con el usuario se facilita con el uso de:

- Menús jerarquizados que presentan las opciones permitidas.
- Sonidos y mensajes que indican error.
- Filtros de datos que evitan la introducción de caracteres equivocados.
- Ayuda en pantalla.
- Un formato en pantalla para ayudar a la captura de los datos.
- Cambios de color en el fondo de la pantalla y en el texto.
- Preguntas sencillas que no contienen términos poco familiares para él.

3.2.4 IMPLEMENTACION

La implementación de SICE se llevó a cabo siguiendo los lineamientos planteados en el diseño del sistema. Como se mencionó en la sección 2.1.4, la implementación es la traducción del diseño a un lenguaje de programación.

En el caso de SICE, el lenguaje utilizado es Turbo Pascal 4.0. La razón por la cual se eligió este lenguaje es porque facilita el desarrollo de programas estructurados al permitir la definición de procedimientos, funciones y tipos de datos abstractos y la compilación de unidades por separado.

Para llevar a cabo esta fase se utilizó la técnica de programación de abajo hacia arriba. Se inició por la declaración de los tipos de registros de datos y se fue ascendiendo hasta llegar a la parte de seguridad del sistema.

Para facilitar el desarrollo, primero se implementaron las funciones necesarias para el manejo de los datos referentes a los alumnos y una vez que éstas fueron probadas, se ampliaron progresivamente las capacidades del sistema a los tipos de datos restantes.

SICE ha sido implementado para ejecutarse en una computadora IBM-PC o compatible con 512 Kb. de memoria y dos drives para discos flexibles. Es recomendable, aunque no necesario, contar con disco duro. Se requiere una impresora para los reportes de información. Se ejecuta bajo el sistema operativo MS-DOS versión 2.0 o posterior.

3.3.5 PRUEBA

En la fase de prueba de SICE se sigue el esquema propuesto en la sección 2.1.5 de este trabajo.

El primer paso a establecer es la prueba por unidad desde el enfoque de **caja blanca**, es decir, asegurar que la operación interna del módulo se lleva a cabo de acuerdo a lo especificado. El proceso de evaluación de las unidades consta de:

- Pruebas dinámicas. Se aplicaron a todas las unidades; incluyendo su compilación y la ejecución de pequeños programas que demuestran el funcionamiento correcto de éstas.

- Pruebas estáticas. Se realizaron únicamente en los casos de error, por medio del análisis del código para facilitar la detección y corrección de fallas.

Después de probar las unidades es necesario llevar a cabo su integración; para ello se siguió, como en la fase de implementación, la estrategia de abajo hacia arriba; a partir del módulo de definición de registros se ensamblaron los módulos y se aplicó, sucesivamente, el mismo proceso de evaluación que para las unidades.

De igual forma se procedió en la prueba del sistema considerándolo como un todo. En los casos específicos donde existía un requisito preestablecido por el usuario se llevaron a cabo un mayor número de pruebas, con el fin de cumplir la condición deseada.

Para la ubicación de los errores se aceptaron los tres enfoques propuestos por Myers [Myers 79] (fuerza bruta, eliminación de causas y retroceso), dependiendo del caso detectado.

Como resultado de esta fase de prueba se concluyó que la estructura modular facilita el proceso ya que permite limitar el área de propagación del error. El uso de pocas variables globales también es clave a favor de la optimización del tiempo y resultados obtenidos.

3.3.6 DOCUMENTACION

La documentación de SICE se elaboró de manera incremental y simultánea al desarrollo del sistema. Está constituida por el manual para el usuario y la documentación del proyecto. El manual para el usuario corresponde al Anexo A; mientras que el presente trabajo puede considerarse como la documentación del proyecto.

Se consideró conveniente mencionar los criterios que se siguieron para la documentación del sistema ya que en la tesis no se incluye el código del mismo.

- Se utiliza una indentación consistente mostrando la modularidad y anidamiento de las estructuras y facilitando, de esta forma, su lectura.

- Las palabras reservadas de Pascal comienzan con una letra mayúscula.

- Una línea punteada señala claramente el principio y fin de cada función y procedimiento.

- No se incluyen procedimientos ni funciones anidadas.

- Se utilizan nombres claros y significativos para identificadores. Los descriptores de los módulos, procedimientos y funciones empiezan con una letra mayúscula, mientras que el primer caracter de las variables es una letra minúscula.

- En cada procedimiento y función principal se incluye un encabezado indicando la función que realiza.

- En la parte superior de cada módulo se indica su nombre, interfaces y objetos que importa y exporta.

- En los puntos del código donde se considera necesario se incluyen comentarios para aclarar su función.

Así mismo, se busca utilizar un estilo de programación consistente que muestre, en su aspecto externo, que cada módulo forma parte del mismo sistema y no es un parche programado ajeno al ambiente.

3.3.7 OPERACION Y MANTENIMIENTO

El prototipo de SICE se encuentra en operación en la institución escolar para la que fue creado.

De momento se encuentra en etapas de mantenimiento perfectivo y correctivo. Su operación aún no comprende todo un curso escolar, de forma que puedan ser evaluados los resultados en ese periodo de vida normal.

Se le ha ido perfeccionando, especialmente en la parte de la interface con el usuario.

Es importante que el sistema siga creciendo, puesto que, como se mencionó en la fase de análisis de requerimientos, son muchos los aspectos que pueden ser atendidos por una computadora.

En la actualidad, el mantenimiento del sistema no es un problema ya que es realizado por la misma persona que lo desarrolló; se espera que, en el futuro, cuando ésto no sea posible, los medios proporcionados para ello sean suficientes y la operación y mantenimiento de SICE resulte una tarea sencilla.

CAPITULO 4. USO DE SICE

Desde el inicio del presente trabajo se ha hecho hincapié en que el sistema resultante sea aplicable y dé solución a una necesidad real; desde este punto de vista, la facilidad de uso del mismo adquiere gran importancia.

Con el fin de mostrar algunas de las características relevantes en la forma de uso de SICE, en este capítulo se incluyen tres ejemplos en los que se indica el procedimiento correspondiente a la ejecución de algunas de las funciones principales:

1) Captura de datos generales del alumno.
2) Consulta de datos académicos del alumno.
3) Impresión de boletas de calificaciones.

Estas descripciones pueden considerarse un complemento del manual para el usuario (anexo A) que detalla los pasos a seguir para el uso adecuado del sistema.

4.1 CAPTURA DE DATOS GENERALES DEL ALUMNO

USUARIO:

Secretario.

CONDICIONES ACTUALES:

SICE presenta al usuario la pantalla de opciones.

CARACTERISTICAS:

El alumno será dado de alta; para ello se requiere proporcionar los datos generales del alumno (nombre, dirección, teléfono, etcétera). Si la información no está completa puede incluirse posteriormente.

SECUENCIA EN LA COMUNICACION SICE-USUARIO:

1. SICE presenta el menú principal.

MENU PRINCIPAL
1. Consulta. 2. Captura. 3. Ordena. 4. Calcula. 5. Actualiza. 6. Imprime. 7. Termina. Su opción por favor:
Teclee el número deseado o seleccione con la flechas

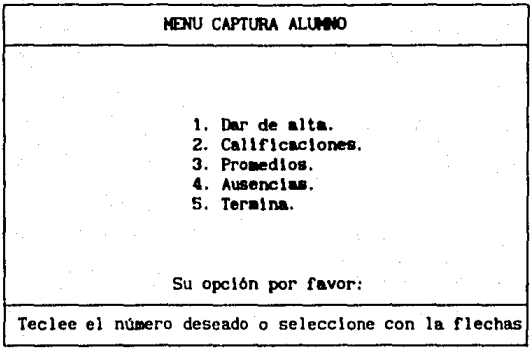
2. El usuario selecciona la opción 2. Captura.

3. SICE presenta el menú de captura.

MENU DE CAPTURA
1. Alumno. 2. Maestro. 3. Materia. 4. Grupo. 5. Termina. Su opción por favor:
Teclee el número deseado o seleccione con la flechas

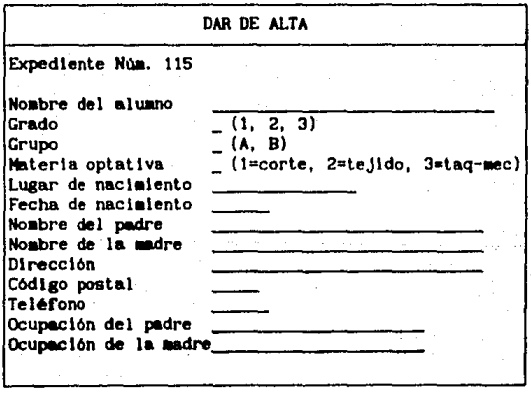
4. El usuario selecciona 1. Alumno.

5. SICE presenta el menú de captura de datos alumno.



6. El usuario selecciona opción 1. Dar de alta.

7. SICE presenta pantalla para dar de alta.



* Además de los datos proporcionados por el usuario, SICE asigna automátiamente:

- Indicador de que está dado de alta el registro.
- Número de expediente (consecutivo que corresponde a

la posición del registro en el archivo de datos académicos-alumno).

- Nivel (Secundaria).
- Sexo (Femenino).
- Apuntador al archivo de datos generales-alumno.

8. El usuario teclea los datos correspondientes. Deja en blanco aquellos que desconoce, los cuales podrán ser introducidos posteriormente, por medio de la función ACTUALIZA.

9. SICE pregunta al usuario si desea dar de alta a otro alumno o terminar y regresar al menú anterior.

10. El usuario opta por regresar al menú anterior.

11. SICE presenta el menú captura alumno.

12. El usuario elige la opción 5. Termina.

13. SICE presenta el menú captura.

14. El usuario elige la opción 5. Termina.

15. SICE presenta el menú principal, dando por terminada la sesión de captura.

4.2 CONSULTA DE DATOS ACADEMICOS DEL ALUMNO.

USUARIO:

Titular del grupo.

CONDICIONES ACTUALES:

SICE presenta al usuario la pantalla de opciones.

CARACTERISTICAS:

El usuario desea consultar las calificaciones de un determinado alumno. Como clave para localizar la información tiene el número de expediente del mismo. Además necesita proporcionar el grado y grupo al que pertenece.

SECUENCIA EN LA COMUNICACION SICE-USUARIO:

1. SICE presenta el menú principal.

MENU PRINCIPAL
1. Consulta. 2. Termina. Su opción por favor:
Teclee el número deseado o seleccione con la flechas

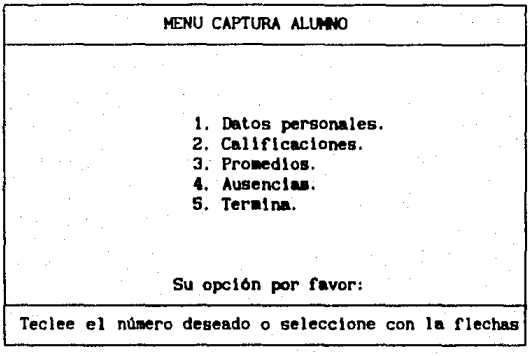
2. El usuario selecciona la opción 1. Consulta.

3. SICE presenta el menú de consulta.

MENU DE CAPTURA
1. Alumno. 2. Maestro. 3. Materia. 4. Grupo. 5. Termina. Su opción por favor:
Teclee el número deseado o seleccione con la flechas

4. El usuario selecciona 1. Alumno.

5. SICE presenta el menú de consulta de datos alumno.

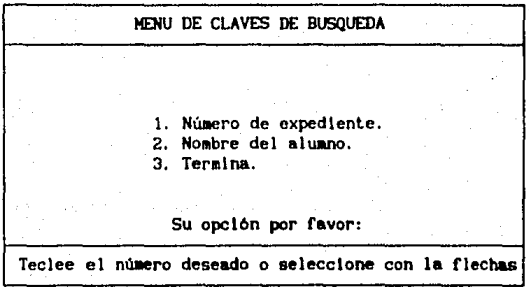


6. El usuario selecciona opción 2. Calificaciones.

7. SICE pregunta el grado y grupo del alumno.

8. El usuario responde 2A.

9. SICE presenta la pantalla de claves.

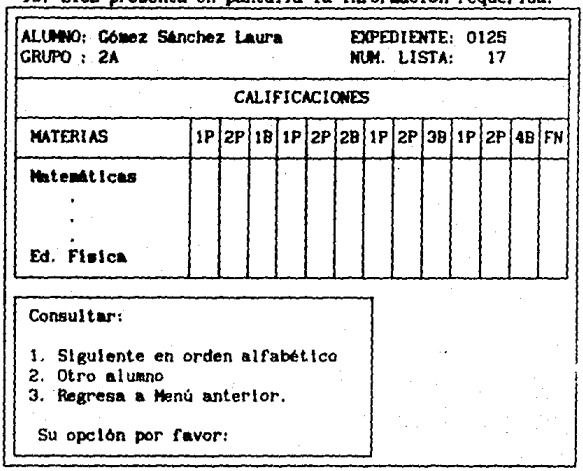


10. El usuario selecciona la opción 1. Número de expediente.

11. SICE solicita al usuario el número del expediente que desea consultar.

12. El usuario teclea 0125.

13. SICE presenta en pantalla la información requerida.



ALUMNO: Gómez Sánchez Laura EXPEDIENTE: 0125
GRUPO : 2A NUM. LISTA: 17

CALIFICACIONES

MATERIAS	1P	2P	1B	1P	2P	2B	1P	2P	3B	1P	2P	4B	FN
Matemáticas													
.													
.													
.													
Ed. Física													

Consultar:

1. Siguiente en orden alfabético
2. Otro alumno
3. Regresa a Menú anterior.

Su opción por favor:

14. El usuario elige la opción 3. Regresa a menú anterior. Seguidas de la opción **Termina** en los siguientes menús que se le presentan, hasta salir del sistema y concluir la sesión.

4.3 IMPRESION DE BOLETAS DE CALIFICACIONES.

USUARIO:

Secretario.

CONDICIONES ACTUALES:

SICE presenta al usuario la pantalla de opciones.

CARACTERISTICAS:

El usuario desea imprimir las boletas de calificaciones de un determinado grupo de alumnos. Para ello necesita contar

con la siguiente información:

a) Grado.

b) Grupo.

c) Tipo de evaluación (mensual o bimestral).

d) Periodo de evaluación.

SECUENCIA EN LA COMUNICACION SICE-USUARIO:

1. SICE presenta el menú principal.

MENU PRINCIPAL
1. Consulta. 2. Captura. 3. Ordena. 4. Calcula. 5. Actualiza. 6. Imprime. 7. Termina. Su opción por favor:
Teclee el número deseado o seleccione con la flechas

2. El usuario selecciona la opción 6. Imprime.

3. SICE presenta el menú imprime.

MENU IMPRIME
1. Alumno. 2. Maestro. 3. Materia. 4. Grupo. 5. Termina. Su opción por favor:
Teclee el número deseado o seleccione con la flechas

4. El usuario selecciona 1. Alumno.

5. SICE presenta el menú imprime alumno.

MENU IMPRIME ALUMNO
1. Datos generales. 2. Directorio. 3. Calificaciones mensuales. 4. Calificaciones bimestrales. 5. Estadísticas. 6. Termina. Su opción por favor:
Teclee el número deseado o seleccione con la flechas

6. El usuario selecciona opción 3. Calificaciones mensuales.

7. SICE pregunta si se desea imprimir todo el grupo.

8. El usuario responde afirmativamente.

9. SICE pregunta el grado y grupo que se van a imprimir.

10. El usuario contesta 2A.

11. SICE pregunta periodo de evaluación.

12. El usuario responde 2.

13. SICE inicia la impresión de las boletas solicitadas, mientras que en pantalla presenta al usuario el número de éstas impresas hasta el momento, las que faltan de elaborar y el tiempo transcurrido hasta entonces. Si el usuario desea interrumpir el proceso puede hacerlo oprimiendo la tecla Esc.

14. SICE termina la impresión y pregunta si se desea seguir imprimiendo o terminar la sesión.

15. El usuario teclea sucesivamente las opciones que le permiten terminar la sesión, hasta llegar al menú principal.

CAPITULO 5. CONCLUSIONES.

Como se afirmó al inicio del presente trabajo, la Ingeniería de Software, en su intento por formalizar el desarrollo de sistemas, proporciona métodos, técnicas y herramientas para ayudar a mejorar la calidad y productividad del Software. Hasta hoy se ha caracterizado por ser subjetiva, ya que permite la iniciativa y creatividad individual. Ciertamente, se pueden observar reglas, pero la variedad de aplicaciones no admite la suficiente generalización.

Se podría decir que la Ingeniería de Software se encuentra en una fase inicial; es una ciencia en pleno desarrollo. Sin embargo, los avances que se han dado últimamente son significativos: cada vez se producen sistemas de mayor calidad con menos costos permitiendo así incrementar la productividad.

Puede considerarse como uno de los mayores logros el hecho de que ya se superó el momento de programar con elementos complicados. Ya no interesan los jeroglíficos y se opta por lo más sencillo, se considera importante que los programas se puedan entender y sean reutilizables.

El campo de aplicación y expansión con que cuenta la Ingeniería de Software es muy amplio, por lo que es necesario ir de lo general a lo particular con el fin de obtener resultados prácticos que muestren la validez de sus propuestas.

En el caso específico de SICE, el sistema representa una concretización del análisis teórico en el desarrollo de un sistema para el manejo de información escolar. De esta aplicación

se podría concluir que utilizar técnicas de programación estructurada favorece la obtención de un sistema modular presentando las ventajas ofrecidas por este tipo de programación, como reducir la complejidad y facilitar el mantenimiento sin que por ello se haya aumentado excesivamente el tamaño del código.

Asumir algunos de los aspectos propuestos por la Ingeniería Humana, como el desarrollo de los sistemas pensando en el usuario, ha propiciado que SICE resulte fácil de manejar aún para quienes no cuentan con amplios conocimientos de computación, alcanzando así uno de los objetivos establecidos al inicio del proyecto. De igual forma, la documentación proporcionada refuerza la ayuda al usuario y simplifica su comunicación con el sistema.

Este sistema, por su bajo costo y su eficiencia en el manejo de información, da respuestas a las necesidades detectadas en la encuesta que se aplicó en algunas instituciones escolares de nuestro país, como es el hecho de evaluar siguiendo los lineamientos de la SEP, agilizar la elaboración de boletas de calificaciones, ofrecer sencillez en la forma de operación, proteger la información confidencial, proporcionar datos estadísticos, etcétera.

En la actualidad la primera versión de SICE se encuentra funcionando experimentalmente (no se ha dejado el sistema manual anterior) en el Instituto Cultural, A.C. La prueba con datos reales ha permitido ir depurando el sistema para darle mayor robustez y confiabilidad.

Después de transcurrido el periodo de operación suficiente para garantizar su buen funcionamiento, se piensa utilizar en otras institutciones similares, representando así un impulso al uso de las computadoras en México.

Es posible afirmar que programar, teniendo en cuenta el concepto de **Ciclo de vida de desarrollo del Software**, da un nuevo enfoque para la obtención de sistemas cuya vida útil se ve

prolongada, al considerárseles objetos que siempre tienen oportunidades de ser mejorados.

Hay que subrayar la importancia de aplicar una metodología que ordene y sistematice la producción de Software evitando la improvisación que suele ser frecuente.

En el presente trabajo se trata de mostrar que es factible mejorar la calidad y productividad del Software haciendo uso de las herramientas proporcionadas por una ciencia joven que invita a la creatividad organizada para optimizar resultados sin desperdiciar esfuerzos.

CAPITULO 6. REFERENCIAS.

[Albizuri 84]

Albizuri-Romero, M. B., "A Graphical Abstract Programming Language", ACM SIGPLAN Notices, vol. 19, núm. 1, enero 1984, pp. 14-23.

[Albizuri 89]

Albizuri-Romero, M. B., "Estructuras de datos e introducción a bases de datos", Ed. Limusa, 1a. edición, México, 1989.

[Bendifallah 87]

Bendifallah, S. y W. Scacchi, "Understanding Software Maintenance Work", IEEE Transactions on Software Engineering, vol. 13, núm. 3, marzo 1987, pp. 311-323.

[Boehm 81]

Boehm, B., "Software Engineering Economics", Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981.

[Boehm 86]

Boehm, B., "A spiral model of software development and enhacement", in Proc. IEEE 2nd Software Process Workshop, 1986.

[Böhm 66]

Böhm, C. y G. Jacopini, "Flow Diagrams, Turing Machines, and Languages with Only Two Formation Rules", Comm. ACM, vol. 9, núm. 5, mayo 1966, pp. 366-371.

[Chin-Kuei 80]

Chin-Kuei, Cho, "An Introduction to Software Quality Control", John Wiley and Sons, E.U.A., 1980.

[Codd 70]

Codd, E.F., "A Relational Model of Data for Large Shared Data Banks", Communications of the ACM, vol. 13, núm. 6, junio 1970, pp. 377-387.

[Dahl 72]

Dahl, O., E. Dijkstra y C. Hoare, "Structured Programming", Academic Press, London, 1972.

[Date 86]

Date, C. J., "Introducción a los sistemas de bases de datos", Addison-Wesley Iberoamericana, México, 1986.

[DeMarco 78]

DeMarco, T., "Structured Analysis and System Specification", Yourdon Press y Prentice-Hall, E.U.A., 1978.

[Dijkstra 69]

Dijkstra, E., "Structured Programming", Software Engineering Techniques, NATO Scientific Affairs Division, Brussels 39, Bélgica, 1968, pp. 84-88. (Report on a Conference, Roma, 1969).

[Dennis 75]

Dennis, J., "The Design and Construction of Software Systems", In: Software Engineering As An Advanced Course, Springer, 1975.

[Fairley 85]

Fairley, R., "Software Engineering Concepts", McGraw-Hill, Inc., E.U.A., 1985.

[Gehani 86]

Gehani, N. y A. McGettrick, "Software Specification Techniques", Addison Wesley, Wokingham, Berks.

[Gewald 77]

Gewald, K. y otros, "Software Engineering: Grundlagen und Technik rationeller Programmentwicklung", Oldenbourg, 1977.

[Gladden 82]

Gladden, G., "Stop Life Cycle, I Want To Get Off", ACM Software Engineering Notes, vol. 7, núm. 2, abril 1982, pp. 35-39.

[Goos 73]

Goos, G., "Systemprogrammiersprachen und strukturietes Programmieren", Lecture Notes in Computer Science Nr. 23, Springer, 1973.

[IBM 75]

IBM: "HIPO, A Design Aid and Documentation Technique", GC 20-1851-11, White Plains, New York, 1975.

[Jackson 75]

Jackson, M., "Principles of Programs Design", Academic Press, 1975.

[Lientz 80]

Lientz, B. y E. Swanson, "Software Maintenance Management", Addison-Wesley, Reading, Mass., 1980.

[Liffick 85]

Liffick, B., "The Software Developer's Sourcebook", Addison-Wesley, E.U.A., 1985.

[Liskov 86]

Liskov, B. y J. Guttag, "Abstraction and Specification in Program Development", MIT Press y McGraw-Hill, E.U.A., 1986.

[McCracken 82]

McCracken, D. y M. Jackson, "Life Cycle Considered Harmful", ACM Software Engineering Notes, vol. 7, núm. 2, abril 1982, pp. 29-32.

[Mills 72]

Mills, H., "Mathematical Foundations for Structured Programming", IBM Technical Report FSC-72-6012, 1972.

[Myers 79]

Myers, G., "The Art of Software Testing", John Wiley and Sons, New York, 1979.

[Parnas 74]

Parnas, D., "Software Engineering or Methods for the Multi-Person Construction of Multi-Version Programs", Lecture Notes in Computer Science, Programming Methodology, Springer, 1974.

[Perkinson 84]

Perkinson, R., "Data Analysis: The Key to Data Base Design", North-Holland, E.U.A., 1984.

[Peters 77]

Peters, L. y L. Tripp, "Comparing Software Design Methodologies", Datamation, noviembre 1977.

[Pomberger 84]

Pomberger, G., "Software Engineering and Modula-2", Prentice-Hall International, Gran Bretaña, 1984.

[Pressman 82]
Pressman, R., "Software Engineering: A Practitioner's Approach", McGraw-Hill, E.U.A., 1982.

[Ramamoorthy 86]

Ramamoorthy, C. y otros, "Programming in the Large", IEEE Transactions on Software Engineering, vol. 12, núm. 7, julio 1986, pp. 768-783.

[Ratcliff 87]

Ratcliff, B., "Software Engineering: Principles and Methods", Blackwell Scientific Publications, Londres, 1987.

[Rivest 78]

Rivest, R., A. Shamir y L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", CACM, vol. 21, núm. 2, febrero 1978, pp. 120-126.

[Ross 77]

Ross, D. y K. Schoman, "Structured Analysis for Requeriments Definition", IEEE Transactions on Software Engineering, vol. 3, núm. 1, enero 1977, pp. 6-15.

[Ross 85]

Ross, D., "Applications and Extensions of SADT", Computer, vol. 18, núm. 4, pp. 25-34.

[Schneiderman 80]

Schneiderman, B., "Software Psychology", Winthrop Publishers, Inc., Cambridge, MA., 1980.

[SEN 82]

"Software Engineering Notes: Special Issue on Rapid Prototyping", ACM Software Engineering Notes, vol. 7, núm. 5, diciembre 1982.

[Spinrad 85]

Spinrad, M. y C., Abraham, "The Wild-West Lifecycle (WILI)", ACM Software Engineering Notes, vol. 10, núm. 3, julio 1985, pp. 47-61.

[Teichrow 77]

Teichrow, D. y E. Hershey, "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems", IEEE Transactions on Software Engineering, vol. 3, núm. 1, pp. 41-48.

[Tripp 82]

Tripp, Bosch, Ellis, Freeman, Johnson, McClure, Robinson, Scacchi, Scheff y Stas, "Evaluation of Software Development Cycle Methodology Implementation", ACM Software Engineering Notes, vol. 7, núm. 1, enero 1982, pp. 45-60.

[Ullman 82]

Ullman, J., "Principles of Database Systems", Computer Science Press., 2a. edición, E.U.A., 1982.

[Warnier 74]

Warnier, J.D., "Logical Construction of Programs", Van Nostrand, 1974.

[Warnier 81]

Warnier, J.D., "Logical Construction of Systems", Van Nostrand, 1981.

[Winters 79]

Winters, E., "An Analysis of the Capabilities of Problem Statement Language: A Language for System Requeriments and Specifications", In Preceedings of the 3rd International Conference on Computer Software and Applications, IEEE Computing Society Press, New York, 1979, pp. 283-288.

[Wirth 71]

Wirth, N., "Program Development by Stepwise Refinement", CACM, vol. 14, núm. 4, 1971, pp. 221-227.

[Wirth 87]

Wirth, N., "Algoritmos y Estructuras de datos", Prentice Hall Hispanoamericana, México, 1987.

[Yourdon 79]

Yourdon, E. y L. Constantine, "Structured Design", Prentice Hall, 1979.

[Yourdon 89]

Yourdon, E., "Managing The Structured Techniques", Prentice Hall, 4a. Ed., E.U.A., 1989.

[Zave 84]

Zave, P., "The Operational Versus The Conventional Approach To Software Development", CACM, febrero 1984, vol. 27, núm. 2, pp. 104-118.

[Zelkowitz 79]

Zelkowitz, M., A., Shaw y J.,Gannon, "Principles os Software Engineering and Design", Prentice Hall, Inc., Englewood Cliffs, NJ., 1979.

ANEXO A.

MANUAL PARA EL USUARIO DEL

SISTEMA INTEGRADO

DE

CONTROL ESCOLAR

SICE

I N D I C E

INTRODUCCION

SICE es el nombre abreviado que se utiliza para hacer referencia al Sistema Integrado de Control Escolar cuyo objetivo es el manejo de información académica básica, en la sección de secundaria dentro de una institución escolar incorporada a la Secretaría de Educación Pública (SEP).

El sistema se basa en un diseño estructural en el que los diferentes módulos o sub-sistemas están relacionados entre sí por medio de menús escritos completamente en español, evitando el uso de comandos difíciles de recordar; ésto permite que cualquier persona, con un mínimo de conocimientos de computación, opere el paquete con excelentes resultados invirtiendo poco esfuerzo y tiempo.

SICE registra información básica correspondiente a los alumnos, profesores y materias de una escuela como son, entre otros aspectos: nombre del alumno, grado que cursa, calificaciones, ausencias, nombre del profesor, materias que imparte, datos personales, nombre y clave de la materia.

Asimismo permite la captura de datos, consulta y actualización de los mismos, corrección de errores, cálculos estadísticos, impresión de reportes y conservación de la información. Estas operaciones están limitadas según el usuario con el fin de restringir el acceso a información confidencial.

1. REQUERIMIENTOS MINIMOS.

En la actualidad existe gran variedad de equipo periférico para las computadoras personales, por lo quees importante establecer los requerimientos mínimos de equipo para el funcionamiento correcto de SICE.

Tales requisitos son:

- Una computadora IBM o compatible,
- 512K de memoria RAM,
- sistema operativo MS/DOS versión 2.0 o posterior,
- dos manejadores (drives) de disco flexible,
- una impresora compatible.

Por el aspecto de economía en el costo inicial del equipo, el sistema está diseñado para funcionar adecuadamente con dos manejadores de disco flexible; sin embargo, si el presupuesto lo permite, se recomienda usar el sistema con un disco duro, para lograr mayor eficiencia en el control de grandes volúmenes de información.

No se aconseja el empleo de una configuración con un sólo manejador de disco flexible porque su operación resulta bastante lenta y el constante cambio de discos que requiere puede llegar a dañar la integridad de los datos.

Además, se recomienda que el usuario cuente con un conocimiento adecuado del equipo y de SICE, lo que posiblemente implicaría una capacitación previa.

2. DESCRIPCION GENERAL DEL SISTEMA.

2.1 COMANDOS GENERALES.

Existen ciertos comandos, teclas y palabras claves que se utilizarán frecuentemente en la operación de SICE. A continuación se da una explicación detallada de cada uno de ellos.

2.1.1 TECLAS

RETURN O ENTER

Debe oprimirse después de teclear cualquier información con el propósito de que ésta llegue a la computadora y el programa continúe la ejecución.

En algunas computadoras la palabra **RETURN** o **ENTER** está escrita en la tecla, mientras que en otras aparece la siguiente flecha:  ; los tres tipos de indicación se refieren a la tecla en cuestión.

ESC

Generalmente se localiza en el extremo superior izquierdo del teclado; en SICE se utiliza para interrumpir la operación que se esté realizando en el momento de oprimir la tecla; ya sea de captura, en cuyo caso no serán almacenados los datos en el archivo, o de impresión.

BACK SPACE

Al oprimir esta tecla se borra de la pantalla el caracter situado a la izquierda del cursor. SICE permite su utilización únicamente durante la operación de captura o la de modificación.

FLECHAS

Se usan para situar el cursor, ya sea para seleccionar una opción en los menús o para posicionarse dentro de la pantalla en el dato que se desea capturar o modificar.

2.1.2 MENU

Es un listado de las diferentes funciones que en ese momento permite realizar el sistema. Cada opción va precedida de un número que servirá para indicar cuál de ellas es la que se desea ejecutar.

Los menús están estructurados jerárquicamente. Se tiene acceso a ellos por medio de un menú de nivel más alto; para identificar con facilidad el nivel en que se encuentra el usuario en ese momento, en la parte superior de la pantalla se indican los niveles de los que proviene; por ejemplo: MENU PRINCIPAL\ CAPTURA\ ALUMNO\ CALIFICACIONES... representa un menú de cuarto nivel.

Existen dos maneras de seleccionar una opción dentro de un menú. La primera consiste en teclear el número de la opción SIN necesidad de oprimir a continuación la tecla RETURN, ya que en este caso la información se envía directamente a la computadora.

La segunda consiste en mover la barra luminosa que señala uno de los renglones, hasta colocarla en la opción que se desea, por medio de las teclas con las flechas que indican movimiento

hacia arriba o hacia abajo. Una vez localizada la opción se ejecuta con sólo teclear **RETURN**.

En todos los menús la última opción permite regresar al menú anterior, o terminar con la sesión si se trata del menú principal.

2.1.3 CLAVE DE ACCESO

Para controlar el acceso a la información, cada usuario de SICE tendrá una clave diferente compuesta siempre por 4 caracteres. Esta clave determina las funciones que le serán permitidas al usuario dentro del sistema.

El administrador del sistema es el encargado de asignar a los diferentes usuarios su clave y de registrarla adecuadamente en SICE, de lo contrario no se le permitirá el acceso.

Al teclearla aparece el caracter "*" en la pantalla, para protección, sin embargo la clave se registra en el sistema tal como se envió. Si existe un error al teclear la clave, SICE da tres oportunidades para corregirla, si el usuario no lo hace, el sistema le niega por completo el acceso al mismo.

2.1.4 FECHAS

Cuando SICE solicite una fecha se debe escribir en el formato **DDMMAA**, es decir, dos cifras para el dia, dos para el mes y dos para el año. En algunos casos, para eliminar posibles errores, se han establecido límites en las fechas que el sistema acepta; por ejemplo en la fecha de nacimiento de los alumnos no puede haber ninguno que haya nacido antes de 1960 y en la actualidad tenga más de 30 años de edad.

2.1.5 ERRORES AL ESCRIBIR LA INFORMACION

El sistema cuenta con filtros para protegerse contra algunos errores que puede cometer el usuario al momento de teclear la información; como el no aceptar números cuando se escribe un nombre o letras en una fecha; sin embargo, no evita los errores implícitos en una mala información (por ejemplo ortográficos).

Cuando se detecta un error, el sistema no lo acepta, emite un sonido para alertar al usuario y permanece en el estado previo a éste, hasta que se teclee el dato correcto.

3. INSTALACION DE SICE

Antes de que SICE pueda utilizarse normalmente, es necesario instalar el sistema. Esta tarea corresponde al administrador del mismo.

La instalación inicial del sistema comprende:

- Creación de archivos para almacenar la información y
- registro de claves de acceso autorizadas.

Los usuarios deben cerciorarse, con el administrador, de que el sistema ha sido instalado adecuadamente antes de intentar utilizarlo.

4. ACCESO AL SISTEMA

4.1. INICIO DE OPERACIONES.

El uso de SICE resulta bastante sencillo si se siguen una serie de pasos en forma sistemática.

El sistema requiere de dos discos flexibles, uno de ellos conteniendo el programa y el otro los archivos donde se almacenan los datos. La secuencia a seguir para iniciar la operación de SICE es la siguiente:

1. Coloque el disco del programa en el manejador de disco flexible (drive) A.

2. Inserte el disco con los archivos de datos en el manejador (drive) B.

3. Encienda la computadora y espere a que ésta le solicite los datos correspondientes a la fecha y hora en que se inicia la sesión.

4. Proporcione los datos que le solicitan.

5. A continuación aparecerá en la pantalla el símbolo A> seguido de la palabra **SICE** esto significa que el sistema empieza a ejecutarse automáticamente.
   Si por alguna razón la computadora ya se encontraba encendida y en este momento presenta el símbolo A> basta introducir los discos como se menciona en los pasos 1 y 2 y teclear la palabra **SICE**, se pueden utilizar letras mayúsculas o minúsculas indistintamente, seguida de la tecla **RETURN**.

6. Después de unos segundos aparecerá la pantalla de presentación del sistema en la que se pide la clave de acceso. Teclee detenidamente su clave, si ésta no fuera correcta, se le darán dos oportunidades más para escribirla, si aún después de ésto no proporciona una clave autorizada NO podrá hacer uso del sistema.

7. Una vez que SICE acepta la clave la pantalla, presenta el menú principal en el que podrá seleccionar la opción requerida.

4.2 FIN DE OPERACIONES

Una sesión con el sistema puede terminar porque el usuario así lo indica o porque hubo algún error en la ejecución del mismo.

En el primer caso, dada la estructura del sistema formado de MENUS anidados, la salida del programa únicamente podrá efectuarse a través de la opción 'SALIR DE SICE' presentada en el MENU PRINCIPAL. Para ello, si se está en un nivel interno, habrá de utilizarse la opción 'SALIR DE ...' para subir de nivel hasta alcanzar el principal.

En caso de error, el sistema es abortado cuando alguno de los archivos en los que se quiere leer o escribir no existe. Si esto ocurre, antes de intentar de nuevo el acceso al sistema verifique que el disco con los archivos de datos esté bien colocado en el manejador B, si es así, entonces consulte al administrador de SICE para que solucione el problema.

Al terminar la sesión por cualquiera de las causas anteriores, la computadora regresa al sistema operativo y aparece en la pantalla el símbolo A>.

5. OPERACIONES

SICE ofrece diversos servicios al usuario. Las funciones que éste pueda realizar dependen del tipo de autorización de acceso que posea. La figura 5.1 muestra la clasificación de los usuarios; mientras que en la figura 5.2 se mencionan las funciones principales del sistema y los usuarios que pueden ejecutarlas.

TIPO	USUARIOS	TIENE ACCESO A:
1	Administrador	Toda la información
2	Secretario	Toda la información
3	Coordinador	Toda la información de la sección
4	Titular	Calificaciones de su grado
5	Profesor	Calificaciones de su(s) materia(s)

Figura 5.1 Clasificación de usuarios de SICE.

OPERACION	USUARIO
1. Consulta	1, 2, 3, 4, 5
2. Captura	1, 2
3. Actualización	1, 2
4. Ordenamiento	1, 2
5. Cálculos	1, 2
6. Impresión de reportes	1, 2

Figura 5.2 Operaciones principales y usuarios correspondientes.

El usuario no debe preocuparse ante la posibilidad de ejecutar operaciones que no le corresponden ya que SICE le presentará únicamente aquellas que le son permitidas.

A continuación se detallan las principales operaciones del sistema. Si Usted corresponde al tipo 3, 4 ó 5, sólo necesita leer el inciso 5.1 que corresponde a la consulta de la información.

5.1 CONSULTA

Como ya se indicó anteriormente, todos los usuarios de SICE pueden realizar la función de consulta; sin embargo, el tipo de información al que tienen acceso es diferente. La figura 5.3 presenta la información disponible y los usuarios que pueden consultarla.

INFORMACION	USUARIO
Datos generales del alumno	1, 2, 3
Calificaciones por alumno	1, 2, 3, 4
Datos generales del profesor	1, 2, 3
Calificaciones por materia	1, 2, 3, 4, 5
Promedios por grupo	1, 2, 3, 4

Figura 5.3 Información disponible y usuarios.

A partir del supuesto de que el sistema ha autorizado el acceso, a continuación se describe lo que sucede en cada una de las opciones de la operación de consulta.

5.1.1 CONSULTA DE DATOS GENERALES DE ALUMNOS

Es la primera opción del menú. Se dá un diálogo secuencial entre el usuario y SICE. Para llevar a cabo la operación de consulta el usuario debe proporcionar al sistema el grado y grupo al que pertenece el alumno en cuestión. Una vez que el sistema obtiene estos datos pregunta si la consulta será por alumno, por grupo o por grado.

Si la consulta se refiere a la información de un alumno, SICE pregunta si ésta se va a buscar de acuerdo al número del alumno o a su nombre. Según la respuesta del usuario, solicita la información correspondiente y presenta en la pantalla los datos que le fueron solicitados con la opción de continuar con ejecución o de terminar la sesión.

Si la consulta es por grupo o por grado, aparece en pantalla la información correspondiente al primer alumno siguiendo el orden alfabético y se podrá solicitar la información de los siguientes alumnos en forma consecutiva.

5.1.2 CONSULTA DE CALIFICACIONES DE ALUMNOS

Se sigue la misma secuencia en el diálogo con el sistema que en la opción anterior, pero la información presentada es la que corresponde a las calificaciones obtenidas por el alumno durante el curso.

5.1.3 CONSULTA DE DATOS GENERALES DE PROFESORES

Como se indica en la figura 5.3, sólo tienen acceso a esta información el administrador, el secretario y el coordinador de sección. Para ello, SICE pregunta el número de profesor o el

nombre, según lo indicado por el usuario y aparece en pantalla la información requerida.

5.1.4 CONSULTA DE CALIFICACIONES POR MATERIA

El sistema pregunta el grado, grupo y número de materia. Una vez obtenida esta información presenta en pantalla los datos correspondientes; éstos aparecen con el siguiente formato:

MATERIA:
PROFESOR:
GRUPO : GRADO:

CALIFICACIONES

NOMBRE DEL ALUMNO	1P	2P	1B	1P	2P	2B	1P	2P	3B	1P	2P	4B	FN
...													
...													
...													
...													
...													

Pulse cualquier tecla para ver la siguiente página

Se presenta a quince alumnos por página.

5.1.5 CONSULTA DE PROMEDIOS

Se sigue el mismo proceso que en el punto anterior, sólo que en vez de calificaciones aparecen los promedios del grupo.

5.2 CAPTURA

Al elegir esta función, el sistema ofrece las siguientes alternativas:

1. Inscripción de alumnos
2. Inscripción de profesores
3. Calificaciones de alumnos
4. Dar de alta materias
5. Dar de alta grupos
6. Dar de baja alumnos
7. Dar de baja profesores
8. Regresar al menú principal

Cada una de ellas registra determinada información dentro del sistema. Los datos que se almacenan presentan cierto formato y características que serán detalladas posteriormente.

Las pantallas de captura muestran el nombre de los datos que se requieren junto con una barra luminosa cuyo tamaño corresponde a la longitud máxima permitida para ese campo.

Al llegar al último espacio del campo, automáticamente, se pasa al siguiente dato. En caso de que la información proporcionada tenga menos caracteres que los previstos será necesario teclear <RETURN> para pasarlo.

5.2.1 INSCRIPCION DE ALUMNOS

El sistema asigna al alumno que se va a registrar un número consecutivo como identificador. Aparecen en la pantalla los datos que deben proporcionarse al sistema de acuerdo a un formato y a las siguientes características:

DATOS SOLICITADOS	LONGITUD MAXIMA	TIPO CARACTER (a)	RANGO
Nombre del Alumno	30	1	
Sexo	1	1	(M, F)
Lugar de Nacimiento	15	1	
Fecha de Nacimiento	6	6	010160-010199
Nombre del Padre	30	1	
Nombre de la Madre	30	1	
Dirección	30	2	
Código Postal	5	3	
Teléfono	8	2	
Ocupación del Padre	20	1	
Ocupación de la Madre	20	1	
Grado	1	3	
Grupo	1	1	
Materia Optativa	1	3	(1, 2, 3)

(a) TIPO DE CARACTER

1 = Sólamente letras

2 = Letras y Números

3 = Sólamente Números

5.2.2 INSCRIPCION DE PROFESORES

Al igual que con los alumnos, SICE asigna un número consecutivo a cada profesor al registrarlo. Las caracteristicas de los datos almacenados respecto a los profesores son las siguientes:

DATOS SOLICITADOS	LONGITUD MAXIMA	TIPO CARACTER (a)	RANGO
Profesor	30	1	
RFC	16	2	
Dirección	30	2	
Código Postal	5	3	
Teléfono	8	2	(XXX-XXXX)
Título Profesional	25	1	
Sexo	1	1	(M/F)
Estado Civil	1	1	(S/C/V/D)
Fecha Ingreso	6	3	

(a) TIPO DE CARACTER

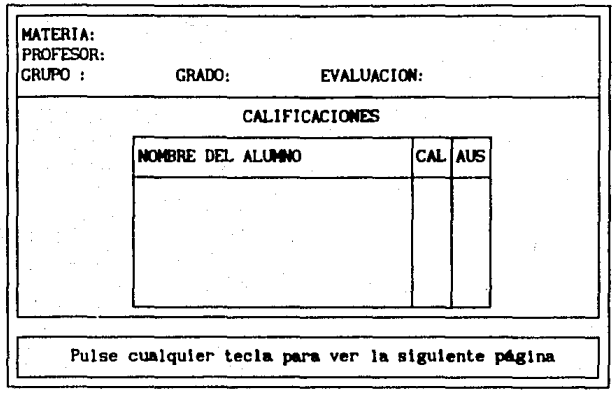
1 = Sólamente letras

2 = Letras y Números

3 = Sólamente Números

5.2.3 CALIFICACIONES DE ALUMNOS

Para ejecutar esta alternativa, SICE solicita el grado, grupo, materia y número de evaluación a los que corresponde la información que será registrada y a continuación presenta la siguiente pantalla:



El sistema escribe los nombres de los alumnos y el usuario sólo tiene que añadir el número real correspondiente a la calificación de cada uno y el número de ausencias durante el periodo de evaluación. Si no se tiene alguno de estos datos, se deja el espacio en blanco y se procede con el siguiente alumno. Posteriormente, por medio de la función de actualización podrán completarse los datos, si así se requiere.

En este caso cada página o pantalla solicita los datos de los alumnos del grupo por bloques de quince, para capturar la información de los quince siguientes basta oprimir cualquier tecla y así obtener otra pantalla lista para recibir los datos.

5.2.4 DAR DE ALTA LAS MATERIAS

Esta operación sólo se requiere realizar una vez por materia al inicio del curso. En ella se establece el profesor que impartirá por grupo cada una de las asignaturas. Los datos que se solicitan y las características de éstos son las siguientes:

DATOS SOLICITADOS	LONGITUD MAXIMA	TIPO CARACTER (a)	RANGO
Grado	1	3	(1, 2, 3)
Grupo	1	1	(A, B)
Número de la materia	2	3	
Número de Profesor	3	3	

(a) TIPO DE CARACTER

1 = Sólamente letras

2 = Letras y Números

3 = Sólamente Números

5.2.5 DAR DE ALTA LOS GRUPOS

Una vez que al inicio del curso se dan de alta todas las materias se procede a dar de alta o formar los grupos. Mediante la información de grado y grupo. SICE consulta la información y automáticamente asigna al grupo el total de alumnos que lo componen.

5.2.6 DAR DE BAJA ALUMNOS

Esta operación no es muy común; se utiliza para retirar la información de los alumnos dados de baja durante el curso. Los datos NO son eliminados del sistema, sino que simplemente no están activos dentro del mismo. El sistema pregunta el número o nombre del alumno que será dado de baja, presenta en la pantalla la información que le corresponde y pide que se le confirme si procede en la ejecución.

El número de expediente del alumno dado de baja NO será asignado a ningún otro durante el año escolar en curso. Al final

del periodo, esta información será parte del archivo muerto y se reinscribirá a los alumnos que así lo requieran.

5.2.7 DAR DE BAJA PROFESORES

Es el mismo proceso que en el apartado anterior. Se retira la información del profesor indicado. El sistema podrá seguir funcionando adecuadamente, pero es necesario reasignar, a un nuevo profesor, las materias impartidas por el anterior para que la información esté completa.

5.3 ACTUALIZACION

Es la operación que se utiliza para modificar o actualizar la información que ya se encuentra almacenada en el sistema. Para ello se sigue la misma secuencia que para la captura inicial de datos.

SICE en vez de mostrar espacios en blanco presenta en la pantalla la información archivada; por medio de las flechas se localiza el punto que se desea modificar; es posible borrar (utilizando la tecla **BACK SPACE**) y/o escribir los datos actualizados.

5.4 ORDENAMIENTO

Esta operación se realiza una vez capturada la información al inicio del curso escolar. En ella se consideran todos los alumnos inscritos hasta ese momento y se les ordena alfabéticamente con respecto al nombre de acuerdo a tres opciones:

a) Por sección (en este caso Secundaria)
b) por grupo,
c) por taller (caso excepcional de la materia de Tecnológicas).

La información que establece este orden es almacenada en archivos que se utilizan como índices para la búsqueda de los datos.

Es indispensable que al inscribir un nuevo alumno se reordenen los índices ya que de no ser así el sistema IGNORARA la información recién archivada.

5.5 CALCULOS

Los cálculos que SICE realiza se dividen en:

a) Promedios. Se refiere a la obtención de este dato en base a las calificaciones obtenidas por el alumno y por el grupo, en cada evaluación, bimestre y curso. Los resultados se almacenan en el archivo correspondiente.

b) Estadísticos. Esta operación se efectúa con respecto a la sección, grado y grupo. Para cada evaluación, bimestre y curso se obtiene el promedio más alto, el más bajo y los porcentajes de aprobación y reprobación.

5.6 IMPRESION DE REPORTES

SICE proporciona la alternativa de imprimir la información que almacena. Esta es una de sus funciones principales. Los reportes en papel que pueden obtenerse son los siguientes:

1. Boleta de calificaciones mensual
2. Boleta de calificaciones bimestral
3. Lista de asistencia
4. Lista de calificaciones por materia
5. Reporte de calificaciones por grupo
6. Directorio de alumnos
7. Directorio de profesores.
8. Datos generales de alumnos
9. Datos generales de profesores
10. Reporte estadistico mensual.

En cada uno de ellos, el sistema pide los datos que le permiten identificar el tipo de reporte que va a imprimir y el grado, grupo, alumno o evaluación a que se refiere. Se ofrece la posibilidad de imprimir por bloque o de forma individual.

Si se desea interrumpir en el transcurso de la impresión, basta presionar la tecla de ESC y el sistema detendrá la ejecución.

6. RESPALDO

Por cuestiones de seguridad, es recomendable respaldar periódicamente la información almacenada en los archivos. Esta tarea corresponde al administrador del sistema. Antes de efectuar la copia de la información es necesario preparar el(los) disco(s) en donde será almacenada, de acuerdo a los siguientes pasos:

1. Formatee un disco flexible según lo indica el manual del sistema operativo.
2. Inserte el disco que contiene a SICE en el manejador A.
3. Inserte el disco formateado en el manejador B.

4. Escriba enseguida del símbolo **A>** que se le muestra en pantalla la siguiente instrucción:

   copy a:respaldo.bat b:

5. Retire los discos de los manejadores. Ya tiene un disco preparado para el respaldo.

El proceso a seguir para respaldar la información es el siguiente:

1. Inserte en el manejador A el disco que preparó para la COPIA de la información.
2. Inserte en el manejador B el disco con los datos ORIGINALES.
3. Escriba la palabra RESPALDO (en mayúsculas o minúsculas) seguido de la tecla **RETURN**. El sistema automáticamente copiará la información del disco B al A.

Al final del curso escolar es conveniente almacenar la información como parte del historial de la institución. Este respaldo se lleva a cabo de la siguiente forma:

1. Inserte en el manejador A el disco que contiene el sistema operativo.
2. Escriba enseguida del símbolo **A>** que se le muestra en pantalla la siguiente instrucción: diskcopy a: b:
   (con la cual se indica que copiará el disco colocado en A al disco colocado en B)
3. Retire del manejador A el disco del sistema operativo.
4. Inserte en el manejador A el disco que contiene los datos originales.
5. Inserte en el manejador B un disco formateado SIN datos.
6. Oprima la tecla **RETURN** y espere a que en la pantalla se le pregunte EN INGLES si desea copiar otro disco; si es así, oprima la letra **Y** y repita los pasos 4, 5 y 6; de lo contrario teclee la letra **N** para que vuelva a aparecer el símbolo **A>**.

ANEXO B.

Una aplicación de la Ingeniería de Software al desarrollo de un Sistema Integrado de Control Escolar

Ponencia presentada en el

PRIMER FORO DE AVANCES DE INVESTIGACION

IIMAS

UNAM

México, D.F., mayo 3, 1989.

UNA APLICACIÓN DE LA INGENIERÍA DE SOFTWARE AL DESARROLLO DE UN SISTEMA INTEGRADO DE CONTROL ESCOLAR

Josefina L. De la Mora Jiménez
y
M. Begoña Albizuri Romero

Instituto de Investigaciones en Matemáticas
Aplicadas y Sistemas
Universidad Nacional Autónoma de México

RESUMEN

El sistema integrado de control escolar (SICE) proporciona la posibilidad de, en una forma computarizada, llevar el control de instituciones escolares a nivel Secundaria bajo el método de evaluación establecido por la Secretaría de Educación Pública (SEP).

El sistema se ocupa de archivar la información referente a alumnos, padres de familia, profesores, materias y cursos. Con dicha información, las principales operaciones que se pueden realizar utilizando SICE son: dar de alta información nueva, consultar o modificar la existente, imprimir diversos reportes y realizar ciertos cálculos referentes a las calificaciones.

SICE fué creado siguiendo las etapas del ciclo de vida del software (análisis de requerimientos, definición de requerimientos o especificación, diseño, implantación, pruebas y mantenimiento) sugeridas por Pomberger [Pomberger 82]. Para el análisis y la definición de requerimientos se hizo uso de experiencias personales de los autores y de cuestionarios aplicados a diferentes instituciones nacionales. En el diseño se utilizó el método "de arriba hacia abajo" ([Dijkstra 69] y [Wirth 71]) de programación estructurada basándose en la modularidad. Las dos fases de las pruebas dinámicas (cada módulo individualmente y el sistema en global) se realizaron siguiendo el método de "causa y efecto" [Hughes y Michtom 78]. El mantenimiento ha consistido en corregir los errores detectados durante las pruebas, cambios al código fuente para mejorar la eficiencia funcional y de espacio y adaptación de nuevas funciones al sistema.

INTRODUCCION

Como parte de un proyecto de investigación en el IIMAS, respondiendo a la necesidad de fomentar el uso de las computadoras en las Instituciones Escolares Mexicanas, se desarrolló un sistema para llevar a cabo el control escolar.

En base a una experiencia personal se detectó la necesidad

de desarrollar un sistema que manejara información y permitiera llevar el control escolar de forma eficiente, sencilla y no demasiado costosa, ya que los sistemas con que hasta entonces se contaba resultaban lentos, no se adaptaban a las necesidades o su costo era muy elevado.

Se elaboró un cuestionario que se envió a algunas escuelas, concertándose además entrevistas con empresas que distribuyen este tipo de Software y con instituciones que lo utilizan, para poder conocer si la necesidad percibida hasta entonces era algo generalizado en el país o se reducía a un determinado grupo de instituciones. Los cuestionarios resueltos aportan la siguiente información:

* En muy pocas instituciones escolares se utiliza la computadora para llevar este tipo de control. Las principales razones que se enumeran son: falta de recursos, poco conocimiento de los beneficios que se obtendrían o se han tenido resultados negativos.

* En general, se detectó éxito en las instituciones que a través de los años han ido elaborando y depurando su propio sistema y además cuentan con personal capacitado en el área.

* Los programas importados han dado problemas por manejar criterios de evaluación diferentes a los de nuestro país y por resultar demasiado costosos.

* Existe gran interés en aprovechar la computadora en este campo.

* Se mencionó la dificultad de que, en la mayoría de los casos, los usuarios no tienen amplios conocimientos de computación y de que no están en posibilidad de utilizar equipo demasiado costoso.

En base a la experiencia y al análisis de los cuestionarios se decidió la implantación de SICE.

FUNCIONES DE SICE

En SICE se intenta ayudar a las instituciones escolares a mejorar y a hacer más eficiente el trabajo de archivar, reportar y realizar ciertos cálculos estadísticos sobre la información con que trabajan. SICE puede ser utilizado para:

* Manejar información como:
  - Datos personales de alumnos y profesores.
  - Directorio de padres de familia.
  - Calificaciones por alumno, por grupo y por materia.
  - Control de asistencias de alumnos y maestros.
  - Observaciones sobre el rendimiento de los alumnos.

* Calcular:
  - Promedios de alumnos por período de evaluación, siguiendo el criterio de la SEP.

- Promedios o medias por materia por grupo.
- Promedios o medias por evaluación por grupo.
- Porcentajes de aprobación por grupo.
- Edad promedio del grupo.
- Alumnos con mayor rendimiento.
- Alumnos con menor rendimiento.

' Reportar:
- Listas de asistencias.
- Directorio de padres de familia.
- Directorio de profesores.
- Boletas de calificaciones.
- Historia académica de grupos por materias.
- Datos estadísticos.

Dado que la información que constituye la base de datos es muy delicada (calificaciones) y a veces confidencial (direcciones) se distinguen dos grupos de usuarios, según el tipo de información que pueden accesar. Por medio de claves de acceso se tiene control de la información disponible para cada usuario así como de las funciones del sistema que puede éste. Para el "ocultamiento" de información confidencial se utiliza el algoritmo de Rivest, Shamir y Adleman [Rivest 81].

La clasificación de usuarios que se hace es:

1) Administración.

a) Administrador - Inicializa el sistema, establece las claves de seguridad y tiene acceso ilimitado a toda la Base.

b) Secretario - Tiene acceso a toda la Base, realiza la captura de la información, imprime reportes, corrige errores en los datos, da de alta y de baja a alumnos y profesores.

2) Consulta.

a) Coordinador de sección - Puede consultar toda la información referente a su sección.

b) Coordinador de grupo - Puede consultar toda la información referente a su grupo.

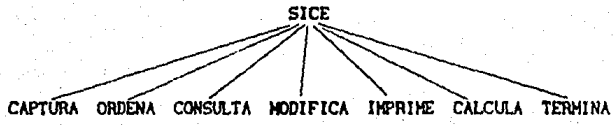
c) Profesores - Pueden consultar la información respecto a sus materias.

Al comenzar la sesión, SICE solicita al usuario que le proporcione su clave; si dicha clave no existe en el directorio, el usuario tiene dos oportunidades más para dar una clave válida, si aún la tercera clave no existe la sesión termina. Si por el contrario, alguna de las tres claves está ya dada de alta en el directorio, entonces se le autoriza utilizar el sistema. En el directorio de usuarios permitidos, además de las claves y nombres de los usuarios se indican las clasificaciones, o sea, qué tipo de usuario es.

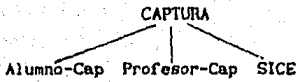
La interface del usuario con SICE se realiza por medio de menús jerarquizados en los que, al oprimir la tecla

correspondiente a una función se invoca la operación deseada. Los menús que se presentarán van a depender del tipo de usuario de que se trate, de tal forma que algunos menús y/o algunas operaciones de ciertos menús son ocultas para determinados usuarios.

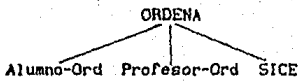
En la figura 1 se muestran, en forma gráfica, las operaciones que ofrece SICE en los menús.



a. Menú Principal.

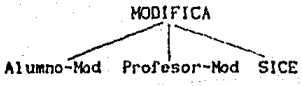


b. Menú de Captura.



c. Menú de Ordena.

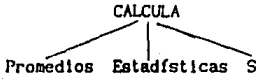
d. Menú de Consulta.



e. Menú de Modifica.

Figura 1. Representación gráfica de las operaciones de SICE.

f. Menú de Imprime.



g. Menú de Calcula.

Figura 1. continuación.

Para mostrar la diferencia entre los menús que se ofrecen a los distintos tipos de usuarios nos valdremos de un ejemplo. Primeramente, supóngase que el usuario de SICE es el coordinador del grupo 2A y desea consultar los datos académicos del alumno Gómez Sánchez Juan, cuyo expediente es 0125. Para ello seguirá los siguientes pasos:

1. SICE le presenta el Menú Principal con las opciones de Consultar y Terminar. (Figura 2.a)

2. El coordinador selecciona la opción 1 (Consulta), entonces SICE pregunta la identificación del grupo; si no corresponde al que él coordina termina la sesión, de lo contrario continúa con el paso siguiente.

3. SICE le presenta el Menú de Consulta con 5 opciones (Figura 2.b), elige la opción 1 (Alumno(s)).

4. Se le presenta el Menú de Alumnos con 3 opciones (Figura 2.c), elige la opción 1 (Datos académicos).

5. Del Menú de Datos Académicos con 3 opciones (Figura 2.d), al seleccionar la opción 1 (Expediente) se le pide el número correspondiente al alumno (en este caso 0125);

6. SICE le muestra en pantalla los datos académicos del alumno y le ofrece otras 3 opciones (Figura 2.e).

El coordinador puede continuar consultando la información del grupo o grupos que coordina u optar por terminar la sesión.

Las pantallas que reflejan este ejemplo se muestran en la siguiente figura:

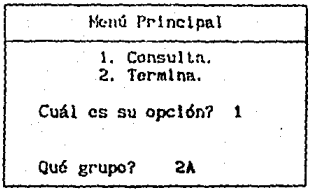


Figura 2.a

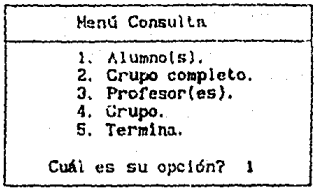


Figura 2.b

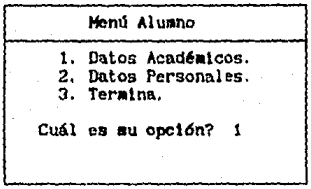


Figura 2.c

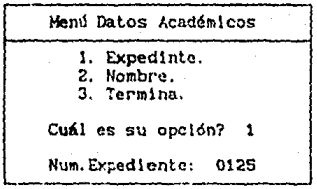


Figura 2.d

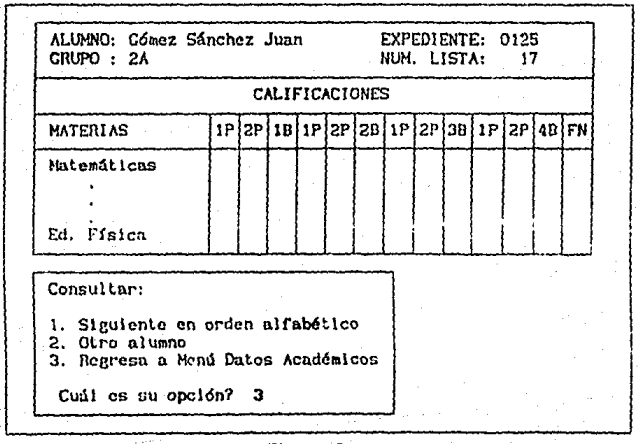


Figura 2.e

Figura 2. Ejemplo de consulta de un coordinador de grupo.

El modelo de Bases de Datos que se utiliza para el almacenamiento de la información es el Relacional. Está constituida de seis relaciones cada una de las cuales constituye un archivo: datos personales-alumno, datos académicos-alumno, datos personales-profesor, datos académicos-profesor, materias y grupos; y tres directorios: usuarios, índices por grupo e índices por materia. En la figura 3 se muestra gráficamente la interconexión de las relaciones y los directorios.

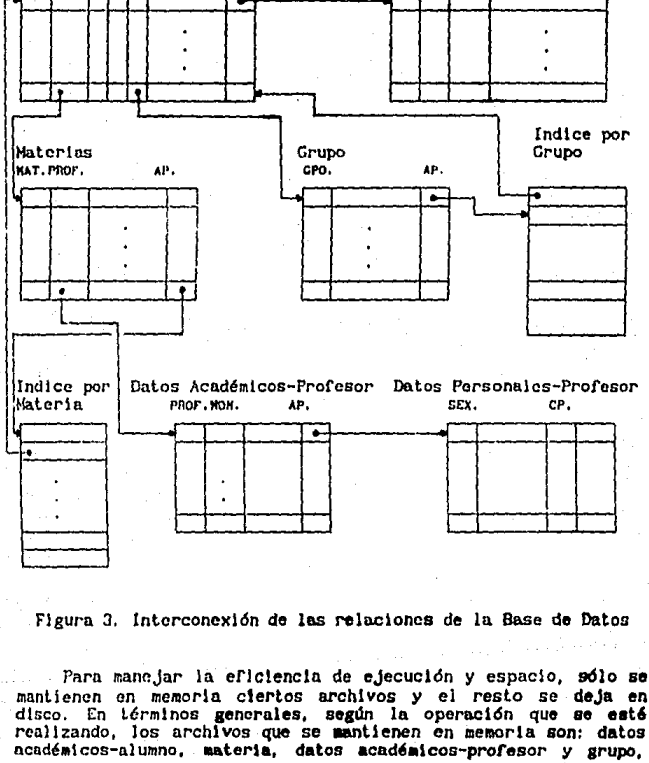


Figura 3. Interconexión de las relaciones de la Base de Datos

Para manejar la eficiencia de ejecución y espacio, sólo se mantienen en memoria ciertos archivos y el resto se deja en disco. En términos generales, según la operación que se esté realizando, los archivos que se mantienen en memoria son: datos académicos-alumno, materia, datos académicos-profesor y grupo,

mientras que los tres directorios (índice por materia, índice por grupo y usuarios) y los archivos de datos personales, tanto de alumnos como de profesores, típicamente están en el disco.

Las modificaciones a la información provocan, durante la sesión, actualizaciones lógicas en la Base de Datos. Las actualizaciónes físicas de los archivos se realizan sólo al finalizar el curso escolar y se archivan en disco flexible para guardar la información como historia. Los directorios de índice por grupo e índice por materia son recreados cada vez que sufren cualquier modificación.

Los respaldos de la información son responsabilidad del administrador quien debe realizarlos después de que se han incorporado las evaluaciones mensuales de todos los alumnos.

La información es filtrada antes de ser actualizada en los archivos para proporcionar robustez, confiabilidad y seguridad en los datos.

En SICE se contempla la posibilidad de cambios en ciertos aspectos de la especificación del sistema. Algunos cambios podrían realizarse interactivamente, mientras que otras modificaciones implican modificar en el código fuente, según sea el caso, algunos parámetros de los módulos, el código fuente en sí o la estructura de los archivos. Todas estas actualizaciones a las especificaciones deberán ser realizadas únicamente por el administrador del sistema. La flexibilidad que presenta el programa para ser actualizado se debe a que, cada función del sistema corresponde a un módulo (diseñado y programado siguiendo los criterios de la programación estructurada), la interface funcional entre ellos es la mínima necesaria y la importación y exportación de datos entre los módulos es pequeña.

Los principales módulos del sistema y la interface entre ellos es la siguiente:

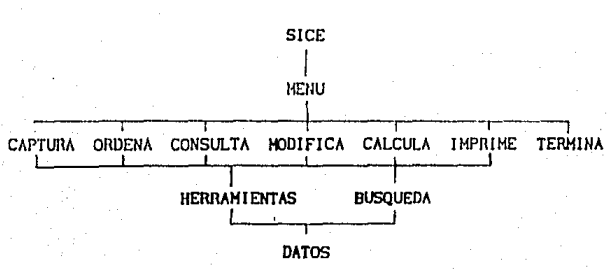


Figura 4. Principales módulos de SICE y sus interfaces.

De acuerdo a su función:

1) SICE- Programa principal para comenzar la ejecución del sistema, controla las claves de acceso al mismo y presenta el menú de entrada de acuerdo al tipo de usuario.

2) MENU- Presenta al usuario menús jerarquizados, con los que puede realizar las diferentes operaciones del sistema.

3) CAPTURA- Recibe datos del usuario y los almacena ya filtrados en el archivo correspondiente.

4) ORDENA- De acuerdo a un orden establecido, acomoda las llaves de las entidades en el índice que almacena en un archivo, mismo que utilizará más tarde para realizar las búsquedas.

5) CONSULTA- Permite al usuario consultar la base de datos, poniendo en pantalla los datos que se le piden.

6) MODIFICA- Modifica los datos capturados anteriormente.

7) CALCULA- Realiza el cálculo de los promedios y datos estadísticos.

8) IMPRIME- Impresión en papel de los diferentes reportes.

9) TERMINA- Concluye la sesión.

10) HERRAMIENTAS- Incluye los procedimientos de lectura y filtrado de datos, los manejadores de archivos, diseño y elección de menús, preguntas al usuario, etc.

11) BUSQUEDA- En base a una clave recibida del usuario o del mismo sistema, puede realizar búsquedas secuenciales y binarias, para actualizar y consultar la Base de datos.

12) DATOS- Define los registros de la Base de datos. Son seis archivos con datos de alumnos, profesores, materias y grupos.

IMPLANTACION DE SICE

SICE ha sido creado para ejecutarse en una computadora IBM-PC o compatible, con 512 Kb. de memoria y dos drives para discos flexibles. Es recomendable, aunque no necesario, contar con disco duro. Se requiere una impresora para los reportes de información. El sistema se ejecuta bajo MS-DOS versión 2.0 y el lenguaje de programación utilizado es Turbo Pascal 4.0.

CONCLUSIONES

Actualmente SICE es un prototipo, se considera que quedará concluido en el transcurso de los próximos seis meses, cuando sea implantado en una Institución Escolar para así poder probarlo con

datos reales.

Dado que el sistema ha sido programado en forma modular y estructurada, no se requiere demasiado trabajo para ampliar las operaciones de éste, de acuerdo a las necesidades de cada institución. Algunas de las posibles expansiones pueden comprender: un mayor número de reportes, directorio de exalumnos, ampliación a otras secciones (Primaria y Preparatoria) de la institución, manejo administrativo, control de la biblioteca, horarios de clase, curriculum de maestros, bitácora de quién ha usado el sistema, etc.

REFERENCIAS

[Dijkstra 69]

Dijkstra E. W., "Structured Programming: Software Engineering Technique", Reporte de una conferencia, Roma, 1969.

[Hughes y Michtom 76]

Hughes J. K. y Michtom J. I., "A Structured Approach to Programming", Prentice-Hall, 1976.

[Pomberger 82]

Pomberger G., "Software Engineering and Modula-2", Prentice-Hall Int., 1982.

[Rivest 81]

R. L. Rivest, A. Shamir y L. Aldeman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Comm. of the ACM, vol. 21, pp. 120-126, 1981.

[Wirth 71]

Wirth N., "Program Development by Stepwise Refinement", Comm. of the ACM, vol. 14, núm. 4, 1971.