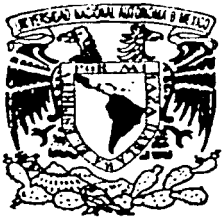
41



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA TUTORIAL PARA LA ENSEÑANZA DE LA ASIGNATURA DE MATEMÁTICAS III EN EL TERCER GRADO DE LA ESCUELA SECUNDARIA DE LA S.E.P.

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN

PRESENTAN

FERNÁNDEZ CERVANTES ADRIANA
GONZÁLEZ SÁNCHEZ MA. PIEDAD
MELCHOR REAL MARÍA ESTHER
RUBIO FÉLIX ALEJANDRO ERIC

DIRECTOR DE TESIS: ING. ORLANDO ZALDÍVAR ZAMORATEGUI

MÉXICO, D. F., 2002

PAGINACIÓN DISCONTINUA

En primer lugar quiero agradecer a quienes me enseñaron a nadar contra corriente, mis amados padres ***Filemón*** *y* ***Felisa****.*

A mis hermanos ***Gilberto, Leticia*** *y* ***Margarita****, por su cariño y apoyo incondicional.*

A mi hermana ***Marisol*** *por todo el apoyo y compañía que me brindó a lo largo de mi carrera y en especial ahora.*

Mi más profundo agradecimiento a mi familia: a ***Octavio*** *mi esposo, a mi hijo* ***David*** *y al* ***bebé*** *que viene en camino, por su amor, apoyo, entusiasmo y compañía en cada etapa del camino recorrido juntos, además de ser una motivación para poder seguir preparándome.*

A la ***Universidad Nacional Autónoma de México*** *por permitir tener mi formación en esta gran casa de estudios.*

A la ***Facultad de Ingeniería*** *y a todos y cada uno de los* ***profesores*** *que a lo largo de la carrera me enseñaron y dirigieron para poder concluir mis estudios.*

Al ***Ing. Orlando Zaldívar Zamorategui*** *por su guía, paciencia, apoyo dirección y entrega en el desarrollo de este trabajo, muchas gracias.*

Finalmente quiero agradecer a todas aquellas personas que de alguna manera hicieron posible la terminación de este trabajo de tesis y que no las mencioné, gracias a todos.

Adriana Fernández Cervantes

A mis padres:

Adelina y **Eleuterio**
que me dieron vida, amor, comprensión y que constituyen
la base de mis conocimientos.

A mis hermanos:

Francisco, Juan, Mario, Martha y **Marcela**
por ser mi mejor ejemplo a seguir y la motivación para alcanzar mis metas.
Gracias por todo el apoyo y cariño que siempre me han brindado.

A mis amigos:

Miguel, Ma. Dolores, Claudia, Jeanett, Alejandro, Marco, Jorge, Marcela, Lisandro y **Alejandrino**
que me enseñaron lo que no sabía y me despertaron a lo que ya sabía.

Un infinito agradecimiento a la Universidad Nacional Autónoma de México, a los profesores de la Facultad de Ingeniería y a todas las personas que han guiado mis pasos por el camino del conocimiento.

Ma. Piedad González Sánchez

"Árbol de sangre, el hombre siente, piensa, florece y da frutos insólitos: palabras.
Se enlazan lo sentido y lo pensado, tocamos las ideas: son cuerpos y son números".

A mi madre, **Ma. del Rosario Góngora G.** por su paciencia, su apoyo y exhortación constante e incondicional, muchas gracias.

A mi hija **Sofía** por su comprensión, su amor, por todos los momentos maravillosos que he vivido con ella, también agradezco su paciencia durante el tiempo que en lugar de dedicarlo a ella, dedique a este trabajo con lo que pude concluir esta etapa profesional.

Sofía eres:

El motivo de todas mis acciones.

El apoyo más fuerte y desinteresado.

La niña que sabe el momento justo para sonreír,
antes de hacerme perder la paciencia.

La persona más justa cuando ve un desacuerdo.

La muñeca más hermosa aún en sus caprichos.

Es la ventana a un mundo lleno de ilusión.

Te amo

Al **Ing. Orlando Zaldívar Zamorategui** por la orientación que nos proporcionó durante el desarrollo de este trabajo, gracias.

Ma. Esther Melchor Real

En primer lugar le doy gracias a Dios por darme las fuerzas necesarias para que no me diera por vencido y permitirme alcanzar esta meta tan importante en mi vida.

Le dedico este trabajo a mi mamá, ***Rosalía Félix Agüero****, quien hizo todo lo necesario por ayudarme a crecer como una persona de bien y darme una sólida educación, así como todo lo que tuvo que soportar en nuestra lejanía.*

A mi papá, ***Daniel de Jesús Rubio Cabrera****, que desde el cielo celebra este triunfo, porque dio todo para que tuviéramos una mejor condición de vida, por los años que vivimos juntos y porque algún día nos volveremos a encontrar.*

A ***Mónica*** *por darme su apoyo y por animarme a alcanzar este objetivo. Te quiero mucho.*

A mis hermanos, ***Daniel****,* ***Iván****,* ***Hugo****,* ***Ricardo*** *y* ***Javier****, para que este logro los motive a alcanzar sus metas, para que no se den por vencidos y se superen constantemente.*

Agradezco a la Universidad Nacional Autónoma de México y a todos los profesores de la Facultad de Ingeniería, por darme la oportunidad de llegar a ser un profesionista competente.

Alejandro Eric Rubio Félix

"Si ante la insistente gota de agua la roca se perfora, ante la tenacidad del hombre la palabra imposible, se evapora".

Índice

Introducción

La educación secundaria obligatoria responde a una necesidad nacional de suma importancia, ya que nuestro país atraviesa por un proceso de cambio y modernización que afecta los ámbitos de la vida de la población.

Los procesos de modernización deben consolidarse en el futuro inmediato y para ello el país requiere de una población mejor educada. Por lo tanto, seis grados de enseñanza obligatoria no han sido suficientes para cubrir la formación básica de las nuevas generaciones. Por tal motivo, surge la necesidad de extender el periodo de educación general a nueve grados, para garantizar que, en dicho proceso educativo, los jóvenes logren adquirir los conocimientos, las capacidades y los valores que son necesarios para alcanzar un aprendizaje de manera permanente, en la educación básica.

La enseñanza de las matemáticas tiene como propósito fundamental, que el alumno aprenda a utilizarlas para resolver problemas, no sólamente los que se resuelven con procedimientos y técnicas aprendidas en la escuela, sino también aquellos cuya solución requiere de la curiosidad y la imaginación creativa.

Con la integración de la computadora en el aprendizaje de las matemáticas, se han desarrollado nuevas técnicas de enseñanza. Por este medio, se busca que el alumno despierte su interés y se sienta motivado de tal manera que logre una educación de mayor calidad formativa y su desarrollo cognoscitivo.

Al hacer uso de un software educativo, la interactividad que se efectúa entre el alumno y la computadora es el aspecto más importante, ya que es una forma de aprendizaje diferente a los otros medios didácticos como los libros, videos, etc. Entonces, esto nos marca la pauta para entender qué tipo de software debe desarrollarse y tener presente que siempre debe cumplir con la función educativa.
Para cubrir el proceso de enseñanza-aprendizaje, el sistema SECUMATIC se basa en tres teorías psicopedagógicas: la de Piaget, centrada en el desarrollo cognoscitivo del adolescente; la de Vigotsky, concede al docente

un papel esencial al considerarle facilitador del desarrollo de estructuras mentales en el alumno, para que sea capaz de construir aprendizajes más complejos, y la de Ausubel, que introduce el concepto de aprendizaje significativo y señala el papel que juegan los conocimientos previos del alumno en la adquisición de nuevas informaciones.

SECUMATIC es un software que tiene como objetivo ofrecer a los estudiantes de tercer grado de educación secundaria y a los profesores de la materia de matemáticas, un apoyo didáctico a través del uso de la computadora, aprovechando los recursos de interactividad que ésta puede ofrecer con la integración de animaciones, videos, imágenes y colores.

Este proyecto se describe en tres capítulos, analizando la situación educativa en México a nivel secundaria y los aspectos psicopedagógicos, el uso de herramientas de ingeniería para la elaboración del software SECUMATIC para proceder finalmente a su implementación.

En el capítulo 1, se hace un análisis de la situación educativa en México a nivel secundaria, definiendo cuáles son sus objetivos y en qué están basados los planes y programas de estudio. Además de considerar cómo se lleva a cabo el proceso de enseñanza-aprendizaje de las matemáticas actualmente en el tercer grado de secundaria. Al final de este capítulo concluimos el fundamento curricular y psicopedagógico en que se apoya el sistema.

En el capítulo 2, tratamos los aspectos relacionados con la ingeniería de software, analizando los paradigmas más utilizados, para poder seleccionar el más adecuado a nuestro proyecto. Más adelante se describe todo el proceso para el desarrollo del software, las herramientas multimedia, el software de programación y las bases de datos.

En el capítulo 3, se hace la justificación de las herramientas y metodología utilizadas para el desarrollo de SECUMATIC, el tipo de programación y la descripción detallada de la codificación, las principales pantallas que definen el sistema, las pruebas realizadas y las evaluaciones en el entorno educativo.

En el capítulo 4, concluimos que el uso del software educativo ha contribuido a mejorar la enseñanza de las matemáticas en la secundaria y que es indispensable hacer uso de la ingenieria de software para su desarrollo, asi como de la integración de multimedia.

Finalmente, se decriben los beneficios que SECUMATIC ofrece en el ámbito educativo, así como la experiencia que obtuvimos al desarrollar un sistema de este tipo.

CAPÍTULO 1

LA EDUCACIÓN SECUNDARIA EN MÉXICO

1.1 El significado de la educación secundaria en México

La escuela es la célula fundamental del proceso educativo, en la que concurren todos los elementos que permiten concretar la acción educativa, en donde los alumnos, maestros, directores, planes y programas de estudio, padres de familia, autoridades civiles y en general el entorno social, interactúan para fortalecer y modificar la calidad educativa del país.

En mayo de 1992, al suscribirse el Acuerdo Nacional para la Modernización de la Educación Básica, la Secretaría de Educación Pública inicia otra reforma a los planes y programas de estudio de la educación básica, que consistió en la realización de acciones inmediatas para fortalecer los contenidos educativos básicos y por último la elaboración de otro currículo.

El Acuerdo Nacional para la Modernización de la Educación Básica, plantea la necesidad de reformar los procesos para mejorar el funcionamiento de las escuelas de educación básica en México.

Durante la primera mitad de 1993 se formularon versiones completas de los planes y programas de estudio, se incorporaron las precisiones requeridas para la elaboración de libros de texto y se definieron los contenidos para los materiales con sugerencias didácticas que más tarde se distribuyeron a los maestros de secundaria para apoyar su labor docente.

Con la reforma hecha al artículo Tercero de la Constitución Política de los Estados Unidos Mexicanos, el 4 de marzo de 1993, se establece como obligatoria la educación secundaria.

Esta nueva reforma compromete al gobierno federal y a las autoridades educativas, a realizar un esfuerzo mayor para que todos tengan acceso a la educación secundaria. Se establece que toda la educación que el estado imparta será gratuita, además de promover y atender todas las modalidades educativas, incluyendo la educación superior, mismas que estarán destinadas tanto a la población joven como a cualquier adulto que aspire a mejorar su formación básica.

Los alumnos, los padres de familia y la sociedad entera, están obligados a contribuir para alcanzar los objetivos propuestos por la reforma educativa y así elevar el nivel educativo de la población del país.

El establecimiento de carácter obligatorio de la educación secundaria, surge como una respuesta a la necesidad nacional de formar ciudadanos mejor educados, ya que nuestro país atraviesa por un proceso de cambio y modernización que repercuten directamente en la vida de los mexicanos. Para lograr estas exigencias se propuso como posible alternativa, satisfacer las necesidades de formación básica de las nuevas generaciones, extendiendo el periodo de educación general a nueve grados, para que los alumnos adquieran más conocimientos, capacidades y valores que son indispensables para alcanzar una formación de calidad, permitiéndoles integrarse de manera responsable a la vida adulta y al trabajo productivo.

Es hasta ahora que el desarrollo alcanzado por el sistema educativo hace posible que la escolaridad de nueve grados, sea una oportunidad real y tangible para la mayoría de los ciudadanos. De tal manera que los docentes enfrentan una gran responsabilidad ya que en gran parte, depende de ellos la formación de los alumnos, especialmente en el medio rural, que es donde no existen los medios propicios para aplicar las nuevas técnicas de enseñanza-aprendizaje. Sin embargo, es de vital importancia seguir ampliando el nivel educativo en general.

1.2 Plan de estudios de la educación secundaria básica de la Secretaría de Educación Pública

El plan de estudios de la educación y los programas que lo conforman, son el resultado de un largo proceso de consulta y diagnóstico, en el cual se integraron los niveles de educación preescolar, primaria y secundaria. Esto se logró con la ayuda de maestros, directores, padres de familia, algunos centros de investigación, organismos sociales y sindicales.

Se realizó una consulta amplia, que permitió identificar los principales problemas educativos del país, se establecieron los siguientes objetivos por prioridades y se propusieron estrategias y alternativas de solución.

- Renovación de los contenidos y los métodos de enseñanza.
- Mejoramiento de la formación de maestros.
- Integración de los niveles educativos que conforman la educación básica.

A lo largo del proceso de consulta y discusión del plan de estudios, surgió como alternativa de solución a las necesidades que demandaba la educación básica obligatoria, el hecho de reforzar los conocimientos y habilidades básicos, tales como las capacidades de lectura y escritura, el uso de las matemáticas en la solución de problemas aplicados a la vida práctica, la vinculación de los conocimientos científicos con la preservación de la salud y la protección del medio ambiente y finalmente ampliar el conocimiento de la historia y la geografía de México.

De acuerdo a los estudios hechos en relación a la educación secundaria, se identificó que uno de los principales problemas organizativos radica en la existencia de dos estructuras académicas, una es por asignaturas y la otra por áreas. Se definió que la organización por áreas propicia una deficiente formación disciplinaria en los alumnos. Este problema se deriva tanto de la organización de los estudios, como de la dificultad que representa para los maestros la enseñanza de contenidos de diferentes campos de conocimiento.

En este nivel educativo intervienen muchos factores para su desarrollo y operación; sin embargo, los resultados concretos se dan en las escuelas, en donde el trabajo de los profesores, directores y el apoyo de los padres de familia son finalmente los que favorecen o limitan el aprovechamiento escolar, ya que existen escuelas en todas las regiones del país con condiciones muy diversas en cuanto a su infraestructura y tradiciones, a pesar de que los planes y programas de estudio son los mismos para la educación básica en todo el territorio nacional.

1.3 Propósitos y prioridades del plan de estudios de la educación secundaria básica

El propósito central del plan de estudios de la educación secundaria básica, para los tres grados, es contribuir a elevar la calidad de formación de los alumnos que han terminado la educación primaria, esto se logra mediante el fortalecimiento de aquellos contenidos que responden a las necesidades básicas de aprendizaje de los jóvenes y que sólo la escuela puede ofrecer. Estos contenidos son el medio fundamental para que los alumnos logren los objetivos de la formación integral, permitiéndoles continuar con su aprendizaje con un alto grado de independencia, dentro o fuera de la escuela, además de estimular las habilidades necesarias para el aprendizaje permanente. Se ha procurado en su totalidad, que la adquisición de conocimientos esté íntimamente relacionada con el ejercicio de habilidades intelectuales y que esto conlleve a la reflexión.

El nuevo plan de estudios es un medio que sirve para organizar el trabajo escolar y lograr el avance cualitativo. Para que estos propósitos se cumplan, es necesario difundir información entre profesores y padres de familia sobre la manera en que se cubrirán los objetivos educativos que se persiguen en los tres grados, apoyar a los profesores en el conocimiento y la puesta en práctica de los planes y programas de estudio. De igual manera, los libros de texto gratuitos y el material de apoyo didáctico, serán objeto de revisión y renovación continua.

Se propone establecer la congruencia y continuidad del aprendizaje entre la educación primaria y secundaria para que no sea éste un obstáculo, y se logren alcanzar niveles satisfactorios de aprendizaje promedio en la escuela secundaria. Para desarrollar la formación adquirida durante la enseñanza secundaria, se han establecido las siguientes prioridades en la organización del plan de estudios, así como la distribución del tiempo de trabajo.

- Asegurar que los estudiantes ejerciten el español, para poder expresarse en forma oral y escrita.

- Ampliar los conocimientos, habilidades matemáticas y las capacidades para aplicar la aritmética, el álgebra y la geometría en la resolución de problemas en la vida práctica.
- Fortalecer los conocimientos científicos y superar los problemas de aprendizaje correspondientes a esta área.
- Aprendizaje de una lengua extranjera y orientación educativa como asignatura, ante la necesidad de ofrecer una educación integral.

1.4 Enseñanza de las matemáticas en el tercer grado de educación secundaria básica

Las matemáticas junto con otras ciencias, tienen como objetivo fundamental, comprender y explicar el universo y las cosas que en él ocurren. Su enseñanza no debe reducirse a la transmisión de un conocimiento fijo y abstracto, sino que debe fomentar en el alumno la curiosidad y el razonamiento para aplicarlos en todos los aspectos de la vida cotidiana.

El propósito central de los programas de estudio de matemáticas, es que el alumno aprenda a utilizarlas para resolver problemas, no necesariamente los que se resuelven con procedimientos y técnicas adquiridas en la escuela, sino también aquellos problemas cuyo descubrimiento y solución requieran de la curiosidad y la imaginación creativa.

La enseñanza de las matemáticas en la escuela secundaria tiene como finalidad el desarrollo de las habilidades operatorias, comunicativas y de descubrimiento de los estudiantes, tales como:

- Seguridad y destreza en el empleo de métodos y procedimientos para resolver un problema.

- Reconocer y analizar los aspectos principales que definen un problema, para elaborar conclusiones, comunicarlas y validarlas de manera clara y concisa.
- Reconocer situaciones análogas.
- Adaptar estrategias adecuadas en la resolución de problemas.
- Capacidad para predecir y generalizar resultados.
- Desarrollar gradualmente el razonamiento deductivo.

Los temas del programa de educación básica a nivel secundaria del tercer grado están agrupados en seis áreas:

- Aritmética
- Álgebra
- Geometría
- Trigonometría
- Presentación y tratamiento de la información
- Nociones de probabilidad

El programa no está estructurado como una sucesión de temas, por lo que es el profesor quien decide la forma de organizar los contenidos, aunque se le recomienda que procure integrarlos en diferente orden y así los alumnos pueden percibir las relaciones que existen entre las diferentes partes de las matemáticas y tengan la posibilidad de practicar y reafirmar sus conocimientos adquiridos; de esta manera el aprendizaje de algunos temas no se ve aislado de los otros.

Analizando el tema de aritmética, se hace énfasis en la comprensión de las operaciones con números naturales y decimales, ya que juegan un papel muy importante en la vida cotidiana, así como en otros temas. El trabajo en clase favorece la comprensión de las nociones aritméticas a partir de la solución de problemas y permiten el desarrollo de estrategias de conteo, cálculo mental, estimación de resultados y el uso correcto de la calculadora. En el tema de fracciones algebraicas, se revisan las operaciones con

fracciones comunes, con lo que se reafirma la comprensión de conceptos alcanzada por los alumnos.

Es de suma importancia que a lo largo del estudio de los temas anteriores, se diseñen actividades con las cuales se favorezca la práctica permanente de las operaciones con números naturales, decimales y fraccionarios, sin que dichas actividades se reduzcan al ejercicio rutinario de los algoritmos.

De igual manera, se propone el cálculo de la raíz cuadrada por diversos métodos, así como el tema de errores de aproximación, que fomentarán en el alumno, ideas importantes de las matemáticas, tales como la recurrencia, el error de aproximación, su cálculo y estimación en situaciones sencillas.

Uno de los temas centrales en la enseñanza de las matemáticas es el álgebra. Se proponen desde el primer grado algunos conceptos de preálgebra, con el propósito de aprovechar las oportunidades que ofrecen la aritmética y la geometría, para que los alumnos se vayan familiarizando con el uso de literales, construyan tablas de valores y su representación en forma gráfica.

También se estudian las principales propiedades algebraicas y las ecuaciones lineales. Se revisan algunas operaciones con monomios y polinomios, así como la solución de sistemas de ecuaciones lineales. Estos temas representan el primer contacto de los alumnos con los procedimientos fundamentales del álgebra, que más adelante serán indispensables para la comprensión de otros temas complejos que serán analizados.

La idea en general, es iniciar operaciones con expresiones en una variable de grados pequeños e ir avanzando paulatinamente hacia expresiones más complejas.

El programa de estudio fue concebido de tal manera que los alumnos tengan la oportunidad de revisar y utilizar constantemente las nociones y los procedimientos básicos del álgebra y así resolver problemas de aplicaciones sencillas.

La geometría se estudia a lo largo de los tres grados, para que se logre explorar y conocer las propiedades y características de las figuras geométricas, propiciando con esto el razonamiento deductivo. También, el conocimiento, la manipulación y representación plana de sólidos permiten el desarrollo de la imaginación espacial del alumno.

La trigonometría es importante por sus numerosas aplicaciones en la ciencia y la tecnología, además de tener un estrecho vínculo con la geometría, aritmética y álgebra. Muestra de ello es el tema de razones trigonométricas de un triángulo, estos conceptos se utilizan en la solución de problemas donde esta disciplina es muy extensa.

En general, el programa de estudio está diseñado para que los alumnos logren desarrollar sus habilidades operatorias, comunicativas y de descubrimiento; además de fomentar la motivación en base a las necesidades e intereses de los alumnos, las cuales deben estar encaminadas hacia actividades pertinentes.

"La tarea principal de los docentes no es transmitir conocimientos sino fomentar el desarrollo y práctica de los procesos cognoscitivos del alumno".[1]

Entonces, para alcanzar un aprendizaje concreto y permanente, el profesor juega un papel muy importante, ya que él es quien debe tomar en cuenta los intereses e inquietudes de los alumnos, al momento de seleccionar el material que le sirve de apoyo didáctico. Debe presentar el material instruccional de manera organizada, interesante y coherente. Su función principal es la de identificar los conocimientos previos que los alumnos ya tienen acerca del tema o contenido a enseñar, para relacionarlos con lo que se va a aprender. Debe procurar hacer amena y atractiva la clase teniendo presente que su principal objetivo es lograr el aprendizaje significativo en el alumno.

Pero es aquí donde se ve desfavorecido el docente, porque no cuenta con el material necesario y muchas veces la enseñanza se imparte basada únicamente en el libro de texto de matemáticas. Por tal motivo, los alumnos

[1]Guzmán Jesús, Carlos. Implicaciones educativas de sus teorías psicológicas. El cognoscitivismo. UNAM, CONALTE, 1994, México. Pág. 29

muestran notablemente cierto desinterés por la asignatura, ya que se estudian conceptos muy abstractos que resultan difíciles de comprender y sólo son aprendidos de manera mecánica, limitando en gran medida la capacidad de razonamiento y deducción.

Por lo tanto, la enseñanza debe ser un proceso placentero, fascinante y no algo tedioso, mecánico y aburrido, sino una actividad tan atractiva como para lograr que los estudiantes descubran su adquisición de conocimientos por sí mismos y no porque el profesor se lo imponga o para obtener una calificación, de esta manera se ayudará al alumno a mejorar su rendimiento académico.

1.5 Justificación curricular

La reforma a la educación se hizo considerando las exigencias que para los ciudadanos implica directamente el desarrollo social y económico del país. Se busca dotar a los jóvenes de una educación permanente, desarrollando sus capacidades y habilidades que sólo en nueve grados de estudio obligatorios, pueden alcanzar.

Ha sido indispensable actualizar los planes de estudio, así como la edición de nuevos libros de texto gratuitos, en los que se basa la enseñanza de la educación secundaria. Estos cambios tienen como finalidad el propiciar cada día, una cultura más sólida en los alumnos que la cursan.

Por lo tanto, se siguen dando cambios en la educación y en los métodos de enseñanza, por lo que surge la necesidad de integrar la tecnología que la computación nos ofrece. Muestra de ello es que la SEP ha implementado el proyecto de "Red Escolar".

"Red Escolar propone llevar a las escuelas de educación básica y normal un modelo tecnológico flexible, que pueda adaptarse fácilmente a las necesidades particulares de cada entidad federativa. El modelo está basado en el uso de la televisión y la informática educativas, principalmente a través de la RED EDUSAT y de la conexión a Internet. Tiene el fin de proveer a la escuela con información actualizada y relevante y con un sistema de

comunicación eficiente que permita a estudiantes y maestros compartir ideas y experiencias".[2]

También se ha distribuido software de tipo multimedia que sirve como material de apoyo didáctico a los profesores, los cuales han contribuido a nuevos métodos de enseñanza, despertando el interés y la motivación de los alumnos.

Dado que el software provisto por la SEP, sólo contempla temas específicos de algunas materias, especialmente en el área de matemáticas, los profesores se ven limitados a su uso y tienen que basar la mayor parte de la enseñanza en el libro de texto; propiciando con ello que los alumnos no se sientan motivados y consideren que aprender matemáticas es un proceso difícil y aburrido.

En este sentido, se deben rediseñar e integrar actividades escolares que proporcionen al profesor información sobre las características de los alumnos de este nivel, así como la forma en que conciben los conceptos y materiales de apoyo didáctico actualizados, tales como el software interactivo, los videos, las animaciones, etc., de esta manera el alumno se sentirá motivado y mostrará mayor interés por la información y comprenderá los conceptos que integran la asignatura de matemáticas en el tercer grado de secundaria.

"Es indispensable, que la enseñanza y el aprendizaje de las matemáticas se realicen a través de materiales y actividades que propicien el análisis, la reflexión y la comprensión, en lugar de la memorización de datos aislados."[3]

Para lograr este tipo de enseñanza-aprendizaje, es fundamental el papel que juega el profesor, porque de él depende el nivel educativo que pueda alcanzarse. Además, debe conocer a fondo las ideas e inquietudes de los alumnos, así como los factores que repercuten en su conducta.

---

[2] http://redescolar.ilce.edu.mx/

[3] Libro para el maestro. Secretaría de Educación Pública. 1994. México. Pág. 10.

Bajo esta perspectiva, son los profesores quienes se han visto en la necesidad de apoyarse en el uso del software interactivo, para despertar el interés y lograr un aprendizaje significativo en el alumno, además de mantener una fuerte motivación al recibir las clases por medio de una computadora.

Al realizar algunas visitas de campo a tres secundarias de la SEP, los profesores nos hicieron la observación de que no existe software de tipo multimedia que cubra el contenido temático de la asignatura de matemáticas del tercer grado, que sirva como reforzador de conocimientos y de apoyo didáctico.

Ante esta problemática fue necesario revisar en primera instancia, las condiciones de trabajo tanto de los profesores como de los alumnos en el aula; además de conocer más a fondo sus ideas e inquietudes acerca de cómo conciben el uso del software en la computadora, para poder dar una alternativa de solución en base a sus requerimientos.

Se realizaron entrevistas de dos tipos, una dirigida a los profesores para conocer cuáles son los temas más representativos y abstractos que les resulta difíciles de ensenar, así como la manera de cómo se estructurarían los contenidos de la materia. En la entrevista dirigida a los alumnos nos dimos cuenta de sus preferencias por los colores, la música, la lectura, el software multimedia y la forma en que perciben las matemáticas a ese nivel, tanto en complejidad como en abstracción.

Tomando en cuenta todas estas características y como posible alternativa de solución, se propone el desarrollo de un software de tipo multimedia interactivo que contemple los temas más representativos de la asignatura de matemáticas del tercer grado de secundaria de la SEP, al cual a partir de este momento llamaremos SECUMATIC, por estar dirigido a estudiantes de nivel secundaria (SECU) en la materia de matemáticas (MATIC).

1.6 Enfoque psicopedagógico

Todo software debe estar apoyado en alguna teoría psicopedagógica con el fin de entender el pensamiento de los estudiantes de 12 a 15 años y así

tener información sobre cómo aprenden y cuándo enseñarles un conocimiento nuevo.

"El software educativo permite el desarrollo de la formación de los educandos, proporcionándoles seguridad y confianza en lo que van construyendo sus estructuras intelectuales. De ahí la importancia de que el software que se ocupe sea el indicado para cada asignatura y para cada eje temático, con especificaciones del grado al cual va dirigido, ya que los intereses de los alumnos varían de acuerdo con su grado de escolaridad." [4]

El psicólogo Piaget constituye una importante aportación para explicar cómo se produce el conocimiento en general y el científico en particular, ya que considera que lo más importante es el individuo en conjunto con su campo vital, su estructura cognoscitiva y sus propias expectativas. Definió que los niños pasan a través de una serie de cuatro etapas en un orden fijo y que la diferencia entre éstas, no sólo es en cuanto a la cantidad de información adquirida en cada etapa, sino también en relación con la cantidad del conocimiento y la comprensión de la etapa.

"El desarrollo cognoscitivo supone la adquisición sucesiva de estructuras mentales cada vez más complejas; dichas estructuras se van adquiriendo evolutivamente en sucesivas fases o estadios, caracterizados cada uno por un determinado nivel de su desarrollo" [5]

El paso de una etapa a la siguiente ocurre cuando el niño alcanza un nivel apropiado de maduración y se le ha expuesto a tipos relevantes de experiencias.

Las cuatro etapas de Piaget son conocidas como: *sensoriomotor, preoperatorio,* de las *operaciones concretas* y de las *operaciones formales.*

[4] Esquivel Granados, Leticia. Software Identidad Nacional Letysoft. UPN-SEP, 1996, México, Pág. 21.
[5] Guzmán Jesús, Carlos; Implicaciones educativas de seis teorías psicológicas. El cognoscitivismo. UNAM, CONALTE, 1994, México, Pág. 77

Analizaremos las características de las dos últimas etapas *operaciones concretas y operaciones formales* por ser, según Piaget, en la que se encuentran los alumnos de entre 7 a 15 años de edad.

Los alumnos interpretan la realidad estableciendo relaciones de comparación, seriación y clasificación. Precisan manipular la realidad y tienen dificultades para razonar de manera abstracta.

En la adolescencia, a partir de los 12 años, se empieza a razonar de una manera más abstracta y se tienen que utilizar representaciones de la realidad sin manipularla directamente, comienza el pensamiento formal. Las habilidades intelectuales que caracterizan a esta etapa están íntimamente relacionadas con los requerimientos que se exigen para el aprendizaje de las matemáticas. Se es capaz de comprobar hipótesis, controlar variables o utilizar el cálculo combinatorio.

Para Piaget el mecanismo básico de adquisición de conocimientos consiste en usar un proceso en el que las nuevas informaciones se incorporen a los esquemas o estructuras preexistentes en la mente de las personas, que se modifican y reorganizan según un mecanismo de asimilación y acomodación facilitado por la actividad del alumno.

Según esta teoría, el desarrollo cognitivo del alumno en un momento determinado o a lo largo de un estadio condiciona en gran medida el tipo de tareas que puede resolver y en definitiva, lo que es capaz de aprender. Por lo tanto, se deduce que hay que adaptar los conocimientos que se pretende que aprenda el alumno en su estructura cognoscitiva.

Vigotsky estudió el impacto del medio y de las personas que rodean al niño en el proceso de aprendizaje.

El concepto básico aportado por Vigotsky es el de *zona de desarrollo próximo*. Cada alumno es capaz de aprender una serie de aspectos que tienen que ver con su nivel de desarrollo, pero existen otros fuera de su alcance que pueden ser asimilados con la ayuda de un adulto con más experiencia, como lo es el profesor.

Este concepto es de gran interés, ya que define una zona donde la acción del profesor es de especial incidencia. En este sentido, la teoría de Vigotsky concede al docente un papel esencial al considerarle facilitador del desarrollo de estructuras mentales en el alumno, para que sea capaz de construir aprendizajes más complejos.

La idea sobre la construcción de conocimientos evoluciona desde la concepción piagetiana de un proceso fundamentalmente individual con un papel más bien secundario del profesor, a una consideración de construcción social donde la interacción con los demás a través del lenguaje es muy importante. Por consiguiente, el profesor se convierte en un agente que facilita la superación del propio desarrollo cognitivo personal.

Vigotsky propone también la idea de que se aprende en interacción con los demás y se produce el desarrollo cuando internamente se controla el proceso, integrando las nuevas competencias a la estructura cognitiva, además considera que el aprendizaje contribuye al desarrollo; esta consideración asigna al profesor y a la escuela un papel relevante, al conceder a la acción didáctica la posibilidad de influir en el mayor desarrollo cognoscitivo del alumno.

El aprendizaje cooperativo como estrategia de aprendizaje promueve la reflexión sobre la necesidad de propiciar interacciones en las aulas, más ricas, estimulantes y saludables.

La teoría de Ausubel introduce el concepto de *aprendizaje significativo* para distinguirlo del repetitivo o memorístico y señala el papel que juegan los conocimientos previos del alumno en la adquisición de nuevas informaciones. La significatividad sólo es posible si se relacionan los nuevos conocimientos con los que ya posee el alumno. Estima que aprender significa comprender y para ello es condición indispensable, tomar en cuenta lo que el alumno ya sabe, para partir de ahí hacia nuevos conocimientos.

Coincide con Piaget en la necesidad de conocer los esquemas de los alumnos y considera que lo más importante es la cantidad y calidad de los

conceptos relevantes, así como las estructuras cognoscitivas que ya posee el alumno.

Ausubel definió tres condiciones básicas para que se produzca el aprendizaje significativo:

- Que los materiales de enseñanza estén estructurados lógicamente con una jerarquía conceptual, situándose en la parte superior los más generales, inclusivos y poco diferenciados.
- Que se organice la enseñanza respetando la estructura psicológica del alumno, es decir, sus conocimientos previos y sus estilos de aprendizaje.
- Que los alumnos estén motivados para aprender.

Entonces, debemos respetar las etapas de evolución de los alumnos, así como sus posibilidades educativas, es decir, que al momento de incrementar su capacidad de aprendizaje, se está ampliando su entorno, es por eso que es de vital importancia estimularlo, motivarlo y sobre todo, dejar que el mismo haga sus propios descubrimientos y conclusiones, mas no presionarlo a dar respuestas inmediatas.

No hay que olvidar que el alumno vive en armonía con el medio que le rodea, por ello, es necesario diseñar y desarrollar software que le sea interesante, atractivo y de intensa interactividad. Estas características son principalmente, el sonido, el color, las imágenes en movimiento, entre otras.

Sin embargo, los programas educativos poseen características muy diversas. Unos aparentan ser un laboratorio o una biblioteca; otros se limitan a ofrecer una función instrumental de tipo máquina de escribir o calculadora; otros se presentan como juego o como un libro; algunos pueden ser usados como evaluadores, pero todos ellos tienen como objetivos, despertar el interés de los alumnos, modificar su razonamiento de tal manera que le facilite el aprendizaje y mejore su desempeño académico.

Por otra parte, no se puede afirmar que un software educativo por sí mismo sea bueno o malo, todo dependerá del uso que de él se haga, de la manera en cómo se utilice en cada situación particular. En última instancia, su funcionalidad y las ventajas e inconvenientes que pueda tener, serán el resultado del material, de su integración al contexto educativo al que está dirigido y principalmente, de la manera en que el profesor organice su aplicación.

Entonces, para mejorar la enseñanza, son los profesores quienes deben seleccionar el software didáctico y los métodos más eficaces, para lograr el fin, ya que ellos son los que conviven de manera directa con los alumnos y, por lo tanto, conocen de manera más cercana las inquietudes y expectativas de éstos. Los directores y los padres de familia actúan como apoyo externo para lograr los fines del sistema educativo.

Para el desarrollo de un software educativo deben considerarse los aspectos, tanto pedagógicos como de carácter personal, logrando así una combinación de ambos y que den como resultado un software de calidad, cuya finalidad sea la de motivar y despertar una actitud más participativa por parte del alumno. De esta forma verán a la computadora como algo nuevo y fascinante, que los llevará a un aprendizaje más concreto y permanente; además, tendrán la oportunidad de investigar y descubrir mecanismos no antes explorados. Estos son los objetivos con los cuales se diseñará el software SECUMATIC, que está dirigido a los alumnos del tercer grado de la materia de matemáticas, de la educación secundaria de la SEP. De esta forma se pretende que el aprendizaje de los temas resulte sencillo, atractivo y estimulante para los alumnos.

CAPÍTULO 2

INGENIERÍA DE SOFTWARE

A finales de los años 60 se introduce la tercera generación de hardware, que provoca una crisis en el software debido a que no existía una planificación adecuada en los proyectos, nula estimación de costos y un diseño poco estructurado. Bajo estas condiciones, el software que se generaba era de muy baja calidad, dificultando el mantenimiento de los programas, por lo que surgió la necesidad de buscar métodos, herramientas y procedimientos para poder desarrollar software confiable y eficaz que estuviera a la vanguardia del hardware propio de la tercera generación de computadoras. Estos métodos, herramientas y procedimientos dieron origen al término Ingeniería de software.

Los problemas principales que afectan y limitan el desarrollo del software son los siguientes:

- Planificación y estimación de costos imprecisos.
- Baja calidad y confiabilidad en el software.
- Baja productividad contra la demanda de requerimientos.
- Elevados costos de mantenimiento.
- Dependencias de los desarrolladores y nula solución de errores.

Algunas causas que han dado origen a estos problemas son:

- Ambigüedad en el planteamiento de los objetivos.
- Procesos de administración deficientes.
- No existe un método formal para el desarrollo.
- Falta de herramientas específicas para la realización de pruebas y mantenimiento.

Una forma de evitar estos problemas es utilizar métodos completos para todas las fases del desarrollo de software y herramientas para automatizar dichos métodos logrando una mejora en la calidad del software, todo esto nos lo proporciona la Ingeniería de software.

2.1 Ingeniería de software

En la actualidad, la Ingeniería de software es una disciplina de la informática que comprende modelos conceptuales y herramientas de trabajo que hacen posible la solución de cualquier problema en el proceso de desarrollo y mantenimiento de software.

Revisando algunas definiciones formales del concepto de Ingeniería de software tenemos las siguientes:

"La Ingeniería de software es el establecimiento y uso de sólidos principios de ingeniería orientados a obtener software económico, fiable y que funcione de una manera eficiente en un equipo de cómputo".[6]

"Ingeniería de software trata del establecimiento de los principios y métodos de la ingeniería a fin de obtener software de modo rentable que sea fiable y trabaje en máquinas reales."[7]

El uso de la Ingeniería de software es multidisciplinaria, utiliza modelos matemáticos para el análisis de algoritmos, ingeniería para estimar costos y definir los métodos, herramientas y procedimientos para el desarrollo, administración para definir la planeación, organización, requerimientos, riesgos y control en los procesos.

Los objetivos que persigue la Ingeniería de software son:

- Mejorar la calidad de los productos de software.
- Aumentar la productividad y trabajo de los ingenieros del software.
- Facilitar el control del proceso de desarrollo de software.
- Suministrar a los desarrolladores las bases para construir software de alta calidad en una forma eficiente.

---

[6] Pressman, Roger S. Ingeniería de Software Un enfoque práctico. Mc Graw Hill, 5ª Edición, 2002, México, Pág. 17

[7] Bauer, Fredrich L. Software Engineering. North Holland Publishing Co. 1993, Amsterdam, Pág. 71

- Definir una disciplina que garantice la producción y el mantenimiento de los productos software desarrollados en el plazo fijado y dentro del costo estimado.

La ingeniería de software es una tecnología multicapa, Fig. 1. Los enfoques de esta tecnología se basan en una organización de calidad, para mejorar los procesos y conducir a un desarrollo más robusto que constituye la base de la Ingeniería de software.

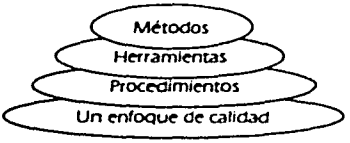


Fig.1 Tecnología multicapa.

La forma en que la Ingeniería de software nos apoya en la informática es conociendo sus cuatro elementos principales.

Métodos. Nos enseñan cómo construir técnicamente el software y abarcan los siguientes puntos:

- Planificación y estimación de proyectos.
- Análisis de requisitos del sistema y del software.
- Diseño de estructuras de datos.
- Diseño de programas y procedimientos algorítmicos.
- Codificación.
- Prueba.
- Mantenimiento.

Herramientas. Son el soporte automático o semiautomático de aplicación de los métodos. Existen herramientas para cada uno de los métodos anteriores. Cuando la información creada por una herramienta puede ser utilizada por otra, se establece un sistema para el desarrollo del software llamado Ingeniería del Software Asistida por Computadora (CASE: Computer Aided Software Engineering). Cada método tiene su propia herramienta.

Procedimientos. Son la aplicación de los métodos utilizando las herramientas; son los elementos de unión entre los métodos y las herramientas y definen la secuencia en la que se aplican los métodos.

Enfoques de calidad. La calidad nos indica la secuencia en que se aplicarán los métodos, procedimientos y herramientas; podemos decir que se puede asegurar la calidad y coordinar los cambios en el sistema.

Cuando todos estos procesos se integran se establece un sistema de soporte para el desarrollo de software llamado Ingeniería de software Asistida por Computadora.

2.2 Paradigmas de la ingeniería de software

La ingeniería de software propone utilizar una metodología para el diseño de software que sea flexible, de bajo costo y confiable. A estas metodologías se les conoce como paradigmas y nos permiten situar el proceso de desarrollo del sistema en diferentes etapas, cada metodología tiene características diferentes ya que se emplean de acuerdo a la naturaleza del proyecto.

2.2.1 Ciclo de vida clásico

Este modelo exige un enfoque sistemático y secuencial es decir, que cada una de sus partes se relacionan entre sí para lograr un software eficiente y en consecuencia coherente. Después de una etapa sigue otra que se basa en la información de la etapa anterior y durante el recorrido de cada etapa se tiene un inicio y un fin.

Se consideran cinco etapas para este modelo:

Análisis, Diseño, Codificación, Prueba y Mantenimiento, Fig. 2.

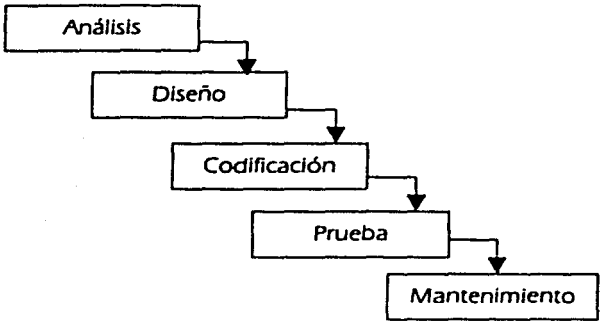


Fig.2 Modelo de ciclo de vida clásico.

Análisis. Se establecen los requisitos de todos los elementos del sistema y se asigna algún subconjunto de estos requisitos al software. Este planteamiento es necesario porque el software tiene que interactuar con el hardware, con personas y posiblemente con bases de datos o alguna otra fuente de información. Se documentan y se revisan los requisitos del sistema con el cliente.

Diseño. Aquí se traducen los requisitos en una representación del software considerando la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz, a fin de obtener la calidad requerida antes de que se comience con la codificación.

Codificación. Se traduce el diseño en forma legible para la computadora por medio de código.

Prueba. Una vez generado el código, la prueba se centra en la lógica interna del software, asegurando que todas las sentencias se han probado, y en las funciones externas verificando que la entrada definida produce los resultados requeridos.

Mantenimiento. El software, indudablemente, sufrirá cambios después de que se entregue al cliente. Los cambios ocurrirán debido a que se hayan encontrado errores, a que se tiene un nuevo sistema operativo o dispositivo periférico, o debido a que el cliente requiera ampliación de funciones o del rendimiento. El mantenimiento del software aplica cada uno de los pasos precedentes del ciclo de vida a un programa existente en vez de a uno nuevo.

El modelo de ciclo de vida clásico presenta las siguientes ventajas:

- Se tiene un seguimiento secuencial del proyecto, esto permite tener un orden durante el desarrollo del software.
- Evita un crecimiento incontrolable, debido a que no se pueden añadir requerimientos una vez finalizada la etapa de análisis.

Pero como todo modelo, también presenta desventajas como las siguientes:

- El desarrollo de un proyecto raramente sigue la secuencia que el modelo propone.
- Siempre ocurre la iteración y crea problemas en la aplicación de esta metodología.
- El método de ciclo de vida clásico requiere con claridad la definición de requisitos, lo que el cliente no expone generalmente de manera precisa. Esto último genera cierta incertidumbre en cuanto a lo que se espera del sistema y hace difícil el inicio del proyecto.

- *Una versión desarrollada del programa no estará disponible hasta mucho tiempo después de haberse iniciado el proyecto, por lo que el cliente debe esperar.*
- *Una falla o error detectado hasta el final del desarrollo y no durante el mismo, puede ser desastroso.*

Tomando en cuenta los elementos del ciclo de vida clásico y con la intención de hacerlo aplicable a la mayoría de las situaciones, surge el ciclo de vida clásico modificado.

2.2.2 Ciclo de vida clásico modificado

Este modelo nos proporciona más flexibilidad entre cada una de las fases, permitiendo interacciones entre ellas, según sean necesarias. Se efectúan cambios en los requerimientos y se disminuyen los efectos de errores en cada etapa del desarrollo.

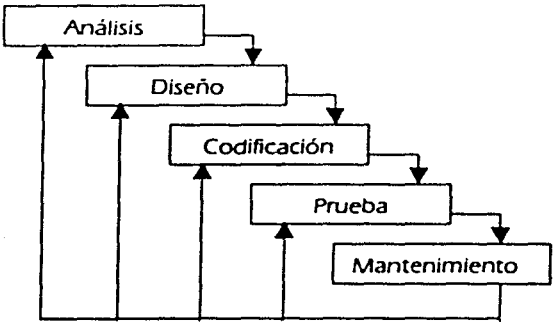


Fig.3 Modelo de ciclo de vida clásico modificado.

Es notable que los pasos de este modelo son muy similares a los pasos genéricos, aplicables a todos los paradigmas de la ingeniería de sistemas, por lo que resulta ser uno de los más completos.

Por otro lado, este modelo nos permite representar los procesos de concepción y producción en una forma gráfica y lógica. Así, todas las actividades que se realicen en cada etapa, serán construidas de una manera disciplinada y con retroalimentación, permitiendo la interacción entre cada una de ellas.

2.2.3 Construcción de prototipos

Esta metodología presenta un diseño rápido sobre la representación de los aspectos del software visibles al usuario, el modelo que se crea puede diseñarse de la siguiente forma.

- Un prototipo en papel o un modelo basado en computadora que describa la interacción hombre-máquina.
- Un prototipo que implemente algunos subconjuntos de funciones requeridas en el programa.
- Un programa existente que ejecute parte o toda la función deseada, pero que tenga características que deban ser mejoradas en el programa final que se dará al cliente.

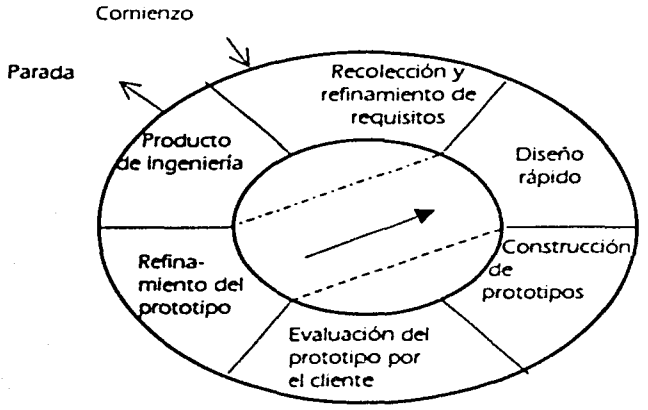


Fig.4 Modelo de construcción de prototipos.

Como en todos los métodos de desarrollo de software, la construcción de prototipos comienza con la recolección de requisitos. Luego se produce un diseño rápido, el cual muestra al usuario los métodos de entrada, los formatos de salida y las diversas funciones del programa, según los requisitos. A partir de este diseño se construye el prototipo, el cual es evaluado por el cliente, y con base en esta evaluación se mejora el software. De esta manera, se genera un proceso interactivo en el que el prototipo es mejorado una y otra vez hasta que satisfaga las necesidades del cliente.

Este paradigma de prototipo se utiliza muchas veces con la finalidad de establecer un conjunto de requerimientos formales, que más adelante son traducidos en la producción de programas mediante el uso de métodos y técnicas de ingeniería de programación.

Las ventajas de utilizar este método:

- El cliente puede ver a corto plazo una primera versión quizá muy elemental de su sistema, pero ésta le puede ayudar para comenzar a familiarizarse con la computadora.
- Al tener un prototipo del proyecto de software, se pueden detectar con mayor facilidad las características faltantes, proponerse nuevas ideas, mejorar el diseño y en general ampliar la perspectiva del proyecto que se quiere desarrollar.

Las desventajas de utilizar este método:

- El cliente ve una versión del sistema en la cual no se consideran los aspectos de calidad ni mantenimiento de software a largo plazo.
- Cuando se le informa que el producto debe ser reconstruido, el cliente cree que presenta fallas porque no comprende que haciendo pequeñas modificaciones al prototipo, éste se ajustará a las nuevas especificaciones y será puesto en ejecución.

2.2.4 Modelo en espiral

Este modelo inicialmente fue propuesto por Boehm, con el objetivo de conjuntar las mejores características tanto del modelo de ciclo de vida clásico modificado como las de creación de prototipos, añadiendo una nueva etapa de análisis de riesgo.

Este modelo define cuatro actividades principales que se representan en los cuatro cuadrantes que lo caracterizan, Fig. 5.

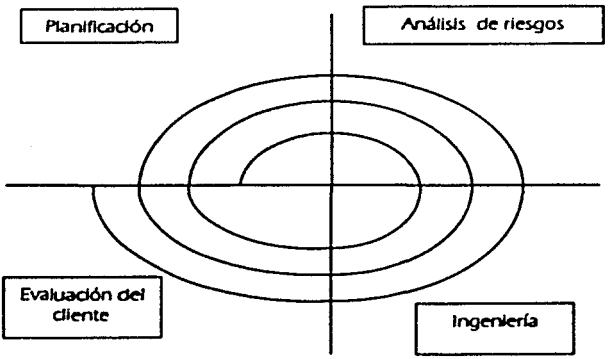


Fig.5 Modelo en espiral.

Planificación. En esta etapa se determinan los objetivos, alternativas y restricciones. Al igual que en los métodos anteriores, esta actividad involucra la recolección de requisitos, donde el cliente debe exponer sus necesidades y dar una idea general de lo que espera; una vez recopilada la información se procede a realizar el estudio inicial para el diseño del software.

Análisis de riesgo. Aquí se hace un análisis de alternativas y se identifican y resuelven los posibles riesgos existentes. Esta actividad permite realizar un análisis del diseño que se ha definido para el desarrollo del software; así, se pueden detectar los inconvenientes y/o limitaciones que se tuvieran antes de comenzar la programación del sistema. También esta etapa permite predecir las expectativas a futuro del proyecto, ya que se pueden visualizar anticipadamente las fallas que pudieran ocurrir por actualizaciones en hardware y/o software, lo cual permite proponer soluciones a los problemas que muy probablemente se presentarán en el futuro.

Ingeniería. Se desarrolla lo que será el producto de "siguiente nivel". En este caso, la espiral va girando y creciendo hasta que obtenemos el producto final, un software de calidad. Si comenzamos la primera actividad del modelo en el centro de la espiral y se va avanzando hacia afuera, el desarrollo del producto de siguiente nivel se refiere a la creación de un prototipo del sistema final, el cual ya consideró todas las características en su totalidad al cubrir la primera curva de la espiral.

Evaluación del cliente. Una vez teniendo el prototipo de la primera vuelta se hace una evaluación de los resultados de la ingeniería. Esta actividad sugiere ir revisando con el cliente tanto el diseño como el programa que se están empleando para la realización de su sistema, esto con la finalidad de que si vamos bien, continuemos por el mismo camino y si no, analizar qué está sucediendo, las causas de que algún proceso no funcione y contar con la oportunidad de corregir y/o cambiar aquello que sea necesario.

En este modelo las actividades se realizan conforme se va desenvolviendo la espiral. Debe considerarse el tiempo de desarrollo del software puesto que la espiral puede crecer desordenadamente, y lo que el modelo pretende es tener un sistema más completo y satisfactorio para el cliente, conforme se avanza en la espiral y no un desarrollo que crezca indefinidamente.

Este modelo nos ayuda a que a en cada vuelta de la espiral, se construyan sucesivas versiones del software, cada vez más depuradas y completas, hasta llegar a obtener el producto final.

Las ventajas del modelo en espiral son:

- Se desarrolla un sistema que, por las diferentes etapas en las que estuvo envuelto, promete ser el óptimo y adecuado para las necesidades del cliente.
- Al tenerse una actividad de análisis de riesgo, el producto final garantiza un acoplamiento a los cambios de software o de hardware que pudieran presentarse en el futuro, es decir, la etapa de mantenimiento llegaría automáticamente sin necesidad de realizar un largo análisis para evaluar las alternativas de solución.

Las desventajas del modelo en espiral son:

- Es difícil convencer al cliente de que el enfoque evolutivo es controlable.
- Se requiere de cierta habilidad para la valoración de los riesgos.
- Si no se delimita bien el número de iteraciones, la espiral crecerá de manera desordenada y sin saber en qué momento se detendrá.

2.2.5 Técnicas de cuarta generación

En el área de la computación existe el término "generación" que se refiere a la evolución de los lenguajes de alto nivel, de máquina y ensambladores. El proceso de cambio de lenguajes de máquina de la primera generación y los lenguajes ensambladores de la segunda generación produjeron una mejora muy importante en la calidad de los programas, algo similar ocurrió cuando se introdujeron los lenguajes de alto nivel. Actualmente existen varias herramientas de desarrollo de aplicaciones que mejoran aun más la productividad de los sistemas; dichas herramientas son conocidas en su conjunto como lenguajes de cuarta generación.

Un lenguaje de cuarta generación interactúa con programas de sistema de manejo de base de datos (SMBD) a fin de almacenar, manipular y recuperar los datos que se necesitan para satisfacer los requerimientos del usuario.

Las técnicas de cuarta generación facilitan el desarrollo de software porque toman en cuenta las siguientes herramientas: lenguajes no estructurados para consulta a bases de datos, generación de informes, manipulación de datos, interacción y definición de pantallas, generación de código, facilidades gráficas de alto nivel y facilidades de hoja de cálculo; sin embargo, se utilizan en problemas de aplicación muy específicos donde la información a manejar es muy extensa.

Esta técnica al igual que las anteriores comienza con la recolección de requisitos para proponer a continuación un prototipo. Posterior a este paso se realiza una estrategia de diseño para después implementar el sistema y realizar las pruebas que sean necesarias.

Su principal ventaja es:

- Disminuye el tiempo de diseño y análisis de las aplicaciones o proyectos de software.

Sus desventajas son:

- Es una técnica relativamente nueva por lo cual no existe una teoría específica para el desarrollo del sistema.
- Las técnicas de cuarta generación están limitadas a software de determinadas características.

Tomando en consideración las etapas que cada paradigma nos ofrece podemos ver que el modelo en espiral resulta ser uno de los más completos en comparación con los otros, ya que cubre con las etapas del modelo de ciclo de vida clásico, así como el modelo de creación de prototipos; además, agrega la etapa de análisis de riesgo que nos permite hacer una evaluación real de los riesgos existentes, así como de sus posibles soluciones.

2.3 Herramientas para el desarrollo de software

2.3.1 La programación estructurada

En programación, uno de los problemas más graves es el de la comunicación, en el sentido de que un programa realizado por un programador debe, muchas veces, ser interpretado y modificado por otro programador. Los problemas de comunicación llevan a una falta de fiabilidad en los programas y a una pérdida de eficiencia considerables, así como a una disminución de la flexibilidad. Para aumentar la eficiencia en la programación y mantenimiento se creó la Programación Estructurada.

La programación estructurada es una metodología de organización y programación de sistemas que mantiene una lógica ordenada, simple y directa, para facilitar su entendimiento y su modificación. Con esta metodología el programador puede elaborar el algoritmo que se va a implementar, de una manera más lógica, clara y comprensible, lo cual hará que el programar sea un proceso coherente y disciplinado, disminuyendo el número de errores y el tiempo invertido.

A la programación estructurada se le denomina así porque en ella se hace uso de estructuras lógicas de programación, las cuales son simplemente, un conjunto de instrucciones iterativas y de decisión, que indican la manera lógica en que se comporta el programa.

Todo programa y su diseño, para ser llamado estructurado debe respetar los siguientes principios:

a) Estructuras básicas

Para el diseño de cualquier programa son suficientes tres estructuras: secuencia, iteración y selección.

- Secuencia

Puede ser conceptualizada como una o más instrucciones de un lenguaje o cualquier secuencia computacional con una entrada y una salida.

```
Begin
  acción 1;
  acción 2;
  ...
  acción n
End;
```

- Iteración (o mecanismo de loop generalizado).

Corresponde a una estructura de control cíclica que permite cualquier tipo de expresión lógica como control de iteración.

```
While  expresión lógica
   instrucciones ...
Do   secuencia;
```

- Selección (o mecanismo de decisión binario).

Corresponde a una estructura de control de alternativas en base a una expresión lógica.

```
If expresión lógica  Then
    secuencia
Else
      secuencia
End if
```

b) Recursos abstractos

Consisten en concebir una acción compleja no en términos de instrucciones de la computadora (o de lenguaje de programación), sino en términos de entidades naturales al problema, deducidos en una forma adecuada. Es decir, se descompone el programa abstracto en acciones más elementales

desarrolladas por instrucciones de una "super máquina". Se continúa este proceso de descomposición de elementos abstractos a niveles más simples, hasta alcanzar un nivel que puede ser ejecutado por una "máquina real". Esta máquina real corresponde al lenguaje o ambiente de programación utilizado.

c) Estructura top-down

Las construcciones básicas pueden ser concebidas como cajas de proceso, en la medida que tienen una entrada y una salida. Todo programa estructurado está compuesto por secuencias, iteraciones y selecciones. Por lo tanto, usando las transformaciones es posible reducir cualquier programa a una sola caja de proceso. Esta secuencia de transformaciones puede ser utilizada para entender y corregir un programa estructurado.

La secuencia de transformaciones corresponde a la estructura top-down del programa. Es decir, se comienza a partir de una única caja de proceso y gradualmente se va expandiendo a una estructura compleja de componentes básicos.

En la medida que se utilizan sólo las estructuras básicas, la secuencia de descomposicion es única, tanto top-down como bottom-up (en sentidos inversos obviamente). Sin embargo, el sentido top-down da cuenta no sólo de la estructura en sí, sino también de la forma en que ésta ha sido concebida y diseñada.

Ventajas y desventajas de la programación estructurada.

"Las ventajas que generalmente se encuentran en la literatura especializada son bastante obvias, pero debe tenerse en cuenta que ellas comparan las ventajas de este enfoque con respecto a lo que ocurría en lo que podríamos llamar la "era pre-estructurada". [8]

[8] Shooman, M. L.; Software Engineering. Mc Graw Hill. 1990. New York. Pág. 60.

Ventajas:

- Claridad. Los programas estructurados tienen generalmente patrones claros y lógicos en sus estructuras de control, lo que es tremendamente ventajoso a través del proceso de diseño.
- Productividad. Los programadores que usan técnicas estructuradas muestran un incremento significativo en la cantidad de instrucciones codificadas por horas/hombre de trabajo. Se habla de ventajas similares en el proceso de pruebas.
- Estilo estándar. La programación estructurada tiende a limitar el código a unos pocos enfoques de diseño simples. Esto ayuda al diseñador, a sus colegas y sucesores a entender el diseño.
- Mantenimiento. La claridad y la modularidad del diseño estructurado es una gran ayuda en la localización de errores y el rediseño de la parte del código afectada.
- Rediseño. La mayoría de los grandes productos de software están sujetos a ocasionales rediseños (frecuentemente llamados mejoramientos). La claridad y la modularidad del diseño estructurado maximiza la cantidad de código que puede ser reutilizado.

Desventajas:

- Actitudes reaccionarias. Muchos programadores y administradores se oponen a aprender nuevas técnicas que consideran aún no lo suficientemente probadas.
- Ineficiencia. En algunos casos puede mostrarse claramente que programas estructurados requieren más memoria y tiempo de ejecución que sus equivalentes no estructurados.

De lo anterior, podemos decir que la programación estructurada presenta más características favorables para la solución de problemas de generación y modificación del software que desfavorables. Sin embargo, aunque la programación estructurada contribuye directamente a mejorar las

características de claridad, productividad, estilo, mantenimiento y rediseño, ella no es la única que incide en estas propiedades. Otros dos elementos que inciden fuertemente en la programación son la tipificación y la modularización.

Se dice que un lenguaje tiene un manejo estricto de tipos si 1) cada objeto pertenece a uno y sólo un tipo, y 2) la transformación de un objeto de un tipo a otro sólo puede realizarse a través de un operador explícito de conversión de tipos. Está demostrado que la tipificación débil es una fuente importante de errores en la codificación y dificulta la legibilidad de los programas.

Un ciclo correctamente estructurado que consta de muchas operaciones y que no puede ser visualizado en una página es más difícil de entender que uno más pequeño que puede ser visualizado completamente en una página. Vemos pues, que el tamaño y la funcionalidad de los "bloques estructurados" son también importantes en la calidad de la programación.

Esta problemática ha sido abordada por la modularización, la que también hace uso intensivo de la abstracción y el razonamiento top-down. La modularización ha tenido un desarrollo más estrecho con la programación estructurada, y puede decirse que el concepto de módulo es la piedra angular del análisis y diseño estructurado, que en un sentido amplio son parte de la programación estructurada.

2.3.2 Programación orientada a eventos

Los lenguajes visuales orientados al evento y con manejo de componentes dan al usuario la posibilidad de construir sus propias aplicaciones utilizando interfaces gráficas sobre la base de ocurrencia de eventos.

Para soportar este tipo de desarrollo interactúan dos tipos de herramientas, una que permite realizar diseños gráficos y un lenguaje de alto nivel que permite codificar los eventos. Con dichas herramientas es posible desarrollar cualquier tipo de aplicaciones basadas en el entorno.

Para entender mejor los conceptos que se están manejando se describe a continuación algunos conceptos básicos sobre la programación orientada a eventos:

Eventos. Son las acciones que un usuario realiza sobre un programa. Por ejemplo, algunos eventos típicos son el clic sobre un botón, el arrastrar un icono, el pulsar una tecla o combinación de teclas, hacer doble clic sobre el nombre de un archivo para abrirlo, el menú contextual que aparece con el botón derecho del ratón, elegir una opción sobre un menú, etc.

Propiedades. Una propiedad es una característica de un objeto, como un formulario o un botón. Según la propiedad de que se trate se puede asignar en tiempo de diseño o en tiempo de ejecución.

Métodos. Los métodos son funciones que manda llamar el usuario, que a diferencia de los procedimientos en un programa no son programados por el usuario, sino que se encuentran ya preprogramados por el lenguaje, con el fin de realizar tareas típicas. Sólo pueden ser llamados en tiempo de ejecución, no en tiempo de diseño. Como cualquier rutina que son, pueden contener argumentos.

Orden de disparo de eventos. Para controlar con éxito la aparición y el comportamiento de formularios (y también de controles) en tiempos de ejecución, debe comprenderse en qué orden se disparan los eventos.

Los eventos de formularios pueden ser divididos en los siguientes grupos:

- Inicio
- Respuesta a una acción
- Vinculación
- Cierre

También es importante comprender que un evento inicia automáticamente con frecuencia a otro evento, produciendo un efecto en cascada. Para trabajar con esta clase de situaciones, debe tenerse una comprensión clara

de qué es lo que dispara cada evento en la secuencia; el peligro de la codificación es iniciar una cadena sin fin de llamadas a eventos circulares recursivos.

Al momento que se dispara un evento el programa entra a un "procedimiento de evento", el cual se ejecuta inmediatamente debido a un evento ejecutado por el usuario o por el código del programa. Lo que se tiene que agregar es el código necesario en respuesta a un evento o control. Por ejemplo, cuando el usuario pulsa un botón de comando o cuando después de un proceso de código se dispara un evento, o también en el caso de que exista un contador que después de cierto tiempo ejecute un evento.

Un procedimiento es una secuencia de instrucciones que se encuentran contenidas en un módulo que representa a un proceso específico. Los procedimientos son llamados o invocados desde expresiones; otros procedimientos o macros, para realizar una operación o calcular un valor ejecutando tareas y acciones que pueden utilizarse en varias ocasiones.

Algunos programas orientados a eventos son los programas típicos de Windows, tales como Word, Excel, PowerPoint y otros. Cuando uno de estos programas ha arrancado, lo único que hace es quedarse a la espera de las acciones del usuario, que en este caso son llamadas a eventos.

Estos programas pasan la mayor parte de su tiempo esperando las acciones del usuario. Las acciones que el usuario puede realizar en un momento determinado son variadas, y exigen el tipo especial de programación orientada a eventos. La programación orientada a eventos utiliza en sus procedimientos la lógica de la programación estructurada, con el fin de manejar orden y claridad de comprensión en su código.

2.3.3 Metodología de lenguaje unificado de modelado (UML)

El UML o lenguaje unificado de modelado, apareció por primera vez en 1997 y ha sido aceptado por la industria del software como un lenguaje gráfico estándar. Es un lenguaje que sirve para escribir los planos del

software, puede utilizarse para visualizar, especificar, construir y documentar todos los artefactos que componen un sistema con gran cantidad de software. Puede usarse para modelar desde sistemas de información hasta aplicaciones distribuidas basadas en Web. UML es simplemente un lenguaje por lo que sólo es una parte de un método de desarrollo de software; es independiente del proceso, aunque para que sea óptimo debe usarse en un proceso dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

Debido a su estandarización y su definición completa no ambigua, UML se puede conectar de manera directa a lenguajes de programación como Java, C++ o Visual Basic. Esta correspondencia permite lo que se denomina como ingeniería directa (obtener el código fuente partiendo de los modelos); pero además, es posible reconstruir un modelo en UML partiendo de la implementación, o sea, la ingeniería inversa.

UML proporciona la capacidad de modelar actividades de planificación de proyectos y de sus versiones, expresar requisitos y las pruebas sobre el sistema, representar todos sus detalles, así como la propia arquitectura. Mediante estas capacidades se obtiene una documentación que es válida durante todo el ciclo de vida de un proyecto. Es una herramienta relativamente joven y se utiliza principalmente en proyectos muy robustos.

2.4 Fases de un sistema

Todo proyecto de software debe basar su desarrollo apoyándose en las siguientes etapas:

Definición del sistema.

Esta etapa se inicia al identificar un problema, obtener y definir los requisitos clave del sistema y del software, así como las necesidades, alcances y limitaciones del proyecto. Selección de la metodología a seguir y las posibles herramientas de apoyo para su realización.

Al identificar un problema se debe pensar en qué información se necesita procesar, cómo se requiere y a qué necesidades debe dar solución. Para identificar estos aspectos es necesario tomar en cuenta los siguientes pasos:

1) "Desarrollar un enunciado definitivo del problema por resolver, incluir una descripción de la situación actual, restricciones del problema y de las metas que se lograrán. El enunciado del problema debe realizarse empleando terminología del área a la que pertenezca el sistema que se requiere desarrollar.
2) Justificar una estrategia de solución computarizada para el problema.
3) Identificar las funciones por realizar, las restricciones, el subsistema de equipo electrónico, el subsistema del producto de programación y el de personal.
4) Determinar los objetivos y requisitos en el nivel del sistema para el proceso de desarrollo y los productos finales.
5) Establecer criterios de alto nivel para la aceptación del sistema." [9]

Fase de desarrollo.

En el proceso de desarrollo de software, las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código; el código es probado, documentado y certificado para su uso operativo.

Se define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo, un plan a seguir, se especifican las características principales y se fundamentan las herramientas a utilizar.

En esta fase se realiza una lista de pasos llamado algoritmo en donde se va verificando que el problema se resuelve como se necesita. Es la parte más difícil del proceso de solución del problema, ya que se debe revisar que el algoritmo está correcto y se puede traducir a un lenguaje de programación

[9] Fairley, Richard. Ingeniería de Software. Mc Graw-Hill. 1991. México. Pág 33

antes de continuar. Se auxilia de técnicas como pseudocódigo y diagramas de flujo.

Validación, pruebas y mantenimiento.

Esta etapa consiste en probar el programa completo y verificar que trabaja como se esperaba. Se prueban cada una de las funciones, primero por separado y luego en conjunto; probar el programa completo con distintos conjuntos de datos de prueba, en caso de que haya errores repetir las pruebas hasta la satisfacción de los requerimientos.

El sistema se emplea de manera experimental para asegurarse que el software no tenga fallas, es decir, que funciona de acuerdo con las especificaciones y en la forma en que los usuarios esperan que lo haga. Se alimentan como entradas conjuntos de datos de prueba para su procesamiento y después se examinan los resultados. En ocasiones se permite que varios usuarios utilicen el sistema, para que los analistas observen si tratan de emplearlo en formas no previstas, antes de que se implante el sistema y dependan de él.

En muchas ocasiones, las pruebas son conducidas por personas ajenas al grupo que desarrolló los programas originales o para asegurarse que las pruebas sean completas e imparciales y por otra, que el software sea más confiable.

El mantenimiento está asociado a la corrección de errores, a nuevas adaptaciones requeridas por la evolución del entorno del software, así como los cambios y nuevas especificaciones propuestas por el cliente

Existen cuatro formas de llevar a cabo el proceso de mantenimiento del software. Una es cuando se corrige el software por defectos; otra cuando se producen modificaciones en el software para adaptarlo a los cambios de su entorno externo. En el mantenimiento perfectivo el software sobrepasa los requerimientos originales propuesto por el cliente y el preventivo o llamado reingeniería del software se encarga de cubrir las necesidades del usuario final.

Implantación y evaluación.

La implantación es el proceso de verificar e instalar el software desarrollado en equipos nuevos, además de entrenar a los usuarios, instalar la aplicación y construir todos los archivos de datos necesarios para su uso.

Los sistemas de información deben mantenerse siempre al día, la implantación es un proceso de constante evolución. La evaluación de un sistema se lleva a cabo para identificar puntos débiles y fuertes.

2.5 Análisis del sistema

El análisis de un sistema debe efectuarse para comprender lo que se quiere obtener como producto final. Cuando la entidad es algo físico podemos construir un modelo igual en cuanto a la forma, pero a una escala menor; sin embargo, cuando la entidad que se va a construir es un software, el modelo debe ser diferente por tratarse de modelar la información que opera en el software y las funciones que permiten que ocurran las operaciones del sistema.

2.6 Análisis de requerimientos

Durante el análisis de los requerimientos del software, se elaboran modelos del sistema a obtener. Los modelos se enfocan en lo que debe hacer el sistema, no en cómo lo hace. En los modelos pueden usarse notaciones gráficas que nos describen información, procesamiento, comportamiento del sistema y otras características con diferentes símbolos reconocibles; algunas veces los modelos pueden ser sólo texto.

El modelo ayuda al analista a interpretar la función, la información y el comportamiento del sistema, lo cual hace más sencilla y sistemática la función de análisis de requisitos. Se convierte en el centro de interés y clave para verificar consistencia de la especificación. Además, proporciona al diseñador una representación clara que le permite implementarla.

En el modelo de análisis se logran tres objetivos:

1. Proporcionar de manera clara y concreta lo que requiere el usuario (cliente).
2. Establecer un punto de partida para la creación de un diseño de software.
3. Definir un subconjunto del total de requisitos, que se puedan validar al obtener el software.

2.7 Análisis y especificación estructurada

2.7.1 Diagrama de flujo de datos

El diagrama de flujo de datos es un modelo que representa el flujo de la información y las transformaciones que se aplica a los datos al moverse desde la entrada hasta la salida.

Se puede utilizar el diagrama de flujo de datos para representar un sistema o un software a cualquier nivel de abstracción. Los diagramas de flujo de datos pueden ser divididos en niveles que representan un mayor flujo de información y un mayor detalle funcional. Los diagramas de flujo de datos proveen un mecanismo para el modelado de tipo funcional, así como el modelo de flujo de información.

Un diagrama de flujo de datos lo podemos tener en dos categorías: Un diagrama de flujo de datos de nivel cero y un diagrama de flujo de datos de nivel uno.

"Un diagrama de flujo de datos de nivel cero que también se denomina modelo fundamental del sistema o modelo de contexto, representa al elemento de software completo como una sola burbuja con datos de entrada y de salida representados por flechas de entrada y de salida, respectivamente.

Al particionar el diagrama de flujo de datos de nivel cero para mostrar más detalles aparecen representados procesos, burbujas, almacén de datos y caminos de flujo de información adicionales, lo que conlleva a un diagrama de flujo de datos de nivel uno, en este diagrama se pueden contener cinco o seis burbujas con flechas interconectadas, y cada uno de los procesos representados en el nivel uno es una subdivisión del sistema general ".[10]

Para evitar construir diagramas de flujo de datos confusos es necesario tomar en cuenta cuatro de sus componentes:

- *El proceso.*
- *El flujo.*
- *El almacén.*
- *El terminador.*

El proceso. *Representado por un círculo, un óvalo o un rectángulo, Fig. 6, se refiere a partes del sistema que llevan a cabo una función de transformación de entradas en salidas. Generalmente se nombran con una frase verbo-objeto.*

Fig. 6 Símbolo de proceso en un diagrama de flujo de datos.

El flujo. *Representado por una flecha que entra o sale de un proceso, Fig. 7, se refiere al movimiento de bloques o datos de una a otra parte del sistema. Generalmente se nombran con el significado del paquete de datos se mueve a lo largo del flujo. Puede existir flujo en ambas direcciones.*

[10] Cruz, Alejandro. Evolución de las Culturas Mesoamericanas. 2000, México. Pág. 67.

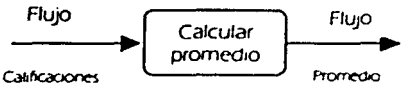


Fig. 7 Símbolo de flujo en un diagrama de flujo de datos.

El almacén. *La Fig. 8 muestra el símbolo con el cual se representa este elemento, se refiere a colecciones de paquetes de información en reposo como una base de datos. Generalmente su nombre es el plural del que se utiliza para los paquetes de datos que entran y salen del almacén por medio de flujos.*

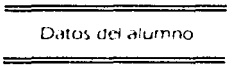


Fig. 8 Símbolo de almacén en un diagrama de flujo de datos.

El terminador. *Representado por un rectángulo Fig. 9, se refiere a entidades externas con las cuales el sistema se comunica y que se encuentran fuera del control del sistema, puede ser otro sistema o el usuario. Generalmente se nombran con una palabra significativa para el usuario.*

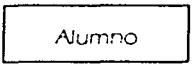


Fig. 9 Símbolo de terminador en un diagrama de flujo de datos.

2.7.2 Diccionario de datos

El modelo de análisis acompaña las representaciones de objetos (entidades) de datos, funciones y control; en cada representación, las entidades y/o elementos de control desempeñan un papel importante.

Es necesario proporcionar un enfoque organizado para representar las características de cada entidad de datos y elementos de control. Esto se realiza con el diccionario de datos; esta importante notación de modelado ha sido definida de la siguiente manera:

El diccionario de datos es un conjunto central de tablas y vistas de la base de datos de sólo lectura, contiene definiciones precisas que permite unificar criterios entre el usuario y el analista.

Un diccionario de datos proporciona la siguiente información:

- Nombre de los usuarios.
- Alias o nombre opcional.
- Privilegios (permisos) que tienen los usuarios.
- Nombre de los objetos (tablas, vistas, índices, sinónimos, clusters, secuencias, procedimientos, tigre, etc.).
- Defaults o información adicional.
- Espacio ocupado por los objetos.

Durante la operación de la Base de Datos, el DBMS (Sistema Manejador de Base de Datos) lee el Diccionario de Datos (DD) para ver los objetos y ver quién los accesa, así como para actualizar estructuras del desarrollo de la base de datos; ya que si un usuario crea una tabla, se generan registros de la nueva tabla en el diccionario de datos, sus columnas, sus segmentos y privilegios que el usuario tiene en la tabla.

2.7.3 Diagrama entidad-relación

El diagrama entidad-relación es un modelo que se basa en la percepción del mundo real y describe la distribución de los datos de un sistema.

Representa los objetos llamados entidades y las relaciones entre ellas. Además, indica la estructura lógica general de la base de datos de manera gráfica.

Hay cuatro componentes principales en un diagrama entidad-relación:

- Tipos de objetos.
- Relaciones.
- Indicadores asociativos de tipo objeto.
- Indicadores de subtipo/supertipo.

Tipos de objetos. Se representa con una caja rectangular, como se muestra en la Fig. 10.

Fig. 10 Representación del tipo de objeto en el diagrama entidad-relación.

Modela una colección o conjunto de objetos del mundo real cuyos miembros individuales tienen las siguientes características:

- Cada uno puede identificarse de manera única por algún medio.
- Cada uno juega un papel necesario en el sistema que se construye.
- Cada uno puede describirse por uno o más datos, llamados atributos.

Relaciones. Una relación establece el conjunto de conexiones entre dos o más objetos, y se representa por medio de un rombo. La Fig. 11 muestra la relación que existe entre la instancia ALUMNO y PROFESOR a través de GRUPO. Un alumno pertenece a un grupo, el cual está asignado a un profesor; o un alumno puede tener asignados a varios profesores.

Fig. 11 Representación gráfica de una relación en un diagrama entidad-relación.

Indicadores asociativos de tipo objeto. Representa algo que funciona como objeto y como relación. Es algo acerca de lo cual se desea mantener alguna información. En el ejemplo de la Fig. 11, el GRUPO nos permite saber algunos datos, como el nombre del grupo y el profesor al cual está asignado, estos datos no se encuentran en los atributos de ALUMNO y PROFESOR; por lo tanto, es necesario crear a GRUPO como un objeto con sus respectivos atributos, como resultado de una relación.

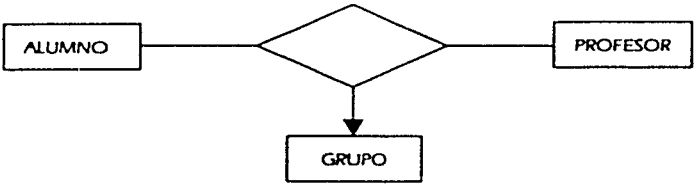


Fig. 12 Creación de un tipo objeto como resultado de una relación entre dos tipos de objetos.

Indicadores de subtipo/supertipo. Consisten en tipos de objeto de una o más categorías conectados por una relación. La Fig. 13 nos muestra un subtipo, donde la categoría general es ALUMNO y sus subcategorías son ALUMNO REGULAR y ALUMNO IRREGULAR.

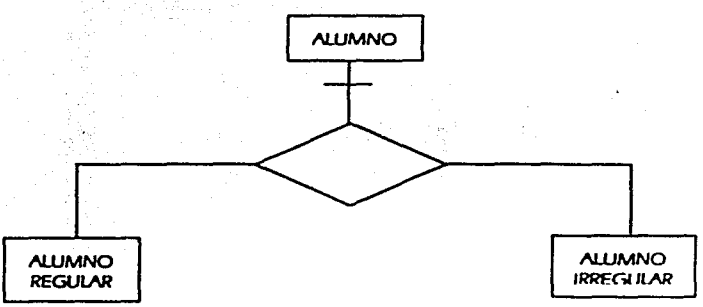


Fig. 13 Tipo de objeto subtipo/supertipo.

Podemos observar que los subtipos se conectan por medio de una relación sin nombre y el supertipo se conecta a la relación con una línea que contiene una barra. El supertipo denota los datos que se aplican a todos los subtipos.

Una vez concluido el modelo entidad-relación, se deben traducir las entidades a tablas, nombradas de acuerdo a los objetos que representan; los cuales poseen atributos (características propias).

Estas tablas se utilizan para guardar datos (campos y registros) relacionados al objeto, siendo cada columna un campo y cada fila un registro, como se muestran en la Fig. 14.

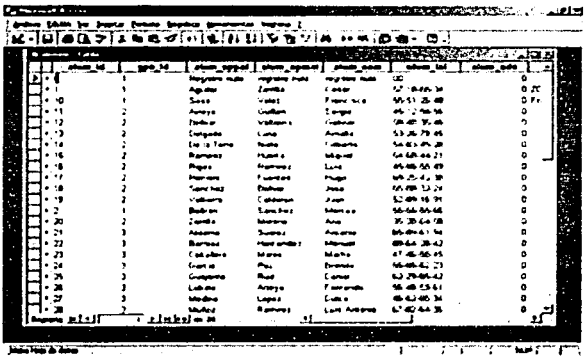


Fig. 14 Visualización de la entidad ALUMNO traducido a una tabla.

2.8 Diseño e ingeniería de software

La etapa del diseño durante el desarrollo de un sistema es importante, ya que nos da un panorama del software (producto) a obtener, mediante las especificaciones y características deseadas y expuestas por el cliente. Si no contamos con un buen diseño, tenemos el riesgo de obtener un sistema totalmente ineficiente.

El diseño estructurado es una excelente opción, ya que traduce los requerimientos del usuario a una representación gráfica del sistema, obteniendo un diagrama de módulos independientes que interactúan entre sí para lograr las funciones solicitadas por el usuario.

Algunas de las características relacionadas con la conceptualización del diseño son:

"Un diseño debe exhibir una organización jerárquica que haga uso inteligente del control entre los componentes del software.

Un diseño debe ser modular, esto es, el software debe estar dividido de forma lógica en elementos que realicen funciones y subfunciones específicas.

Un diseño debe contener representaciones distintas y separadas de los datos y de los procedimientos.

Un diseño debe llevar a módulos que exhiban características funcionales independientes.

Un diseño debe llevar a interfaces que reduzcan la complejidad de las conexiones entre los módulos y el entorno exterior.

Un diseño debe obtenerse mediante un método que sea reproducible y que esté reducido por la información obtenida durante el análisis de los requisitos del software"[11]

2.8.1 La carta estructurada

Es la representación gráfica de un sistema, que muestra de manera general el refinamiento conceptual del problema de forma general e identifica funciones internas de los procesos, la descomposición de funciones principales en subfunciones, la definición de los tipos de datos locales y el tipo de almacenamiento utilizado. Además, establece las relaciones e interconexiones entre las funciones y los datos.

Esta carta presenta una organización jerárquica que indica claramente la relación entre los elementos del sistema, define los procesos concurrentes con variables comunes y proporciona las interfaces de comunicación entre ellos.

Generalmente se utiliza un nodo principal que representa la raíz y éste a su vez puede tener nodos subordinados; además, establece la relación de subrutinas y el paso de parámetros entre los módulos.

[11] Pressman, Roger S; Ingeniería de Software. Un enfoque práctico; McGraw Hill,5ª Edición 2002, México, Pág. 332.

El uso de la carta estructurada es de gran ayuda para el desarrollo de un sistema, es una forma de visualizar el mismo ordenadamente y con criterios unificados, permitiendo el seguimiento del desarrollo sin dificultad.

2.8.1.1 Características de la carta estructurada

La abstracción es uno de los conceptos fundamentales del diseño estructurado, es un concepto intelectual que permite aislar mentalmente las cualidades de un problema en particular. La abstracción permite separar los aspectos conceptuales del sistema. Existen mecanismos para la abstracción en el diseño de sistemas de tipo procedimental, de datos y de control.

Abstracción procedimental. Se desarrolla mediante enunciados o instrucciones con una función bien definida y limitada, la cual se establece en términos del entorno del problema.

Abstracción en los datos. Aquí se especifican los tipos de datos u objetos además de las operaciones permitidas entre ellos.

Abstracción en el control. Nos muestra el efecto deseado de algún módulo sin necesidad de definir el mecanismo exacto de control.

Abstracción en el flujo de datos. Nos muestra el flujo deseado del sistema o sólo de algún módulo sin necesidad de definirlo detalladamente.

A continuación se muestra en la Fig. 15, un ejemplo de la carta estructurada del módulo SECUMATIC.

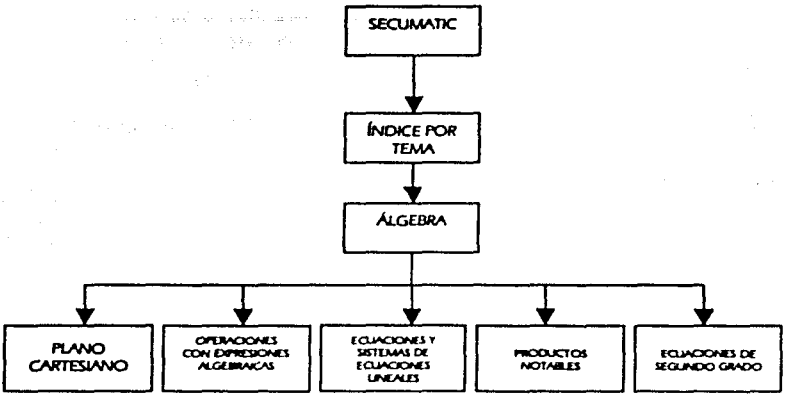


Fig.15 Carta estructurada del módulo SECUMATIC.

El refinamiento se aplica durante el desarrollo del sistema, ya que al principio tenemos una idea muy general del mismo y conforme se avanza se va desglosando en unidades más pequeñas y detalladas, por lo que la carta estructurada se va afinando constantemente.

2.8.2 Modularidad

Esta característica se refiere a la división de los sistemas en partes pequeñas, a las cuales se les llama módulos, éstos son independientes entre sí por lo que pueden ser manejados por separado, siendo sencillo seguir el flujo de control de instrucciones al igual que la localización de las variables utilizadas.

Su objetivo es reducir el tiempo de generar programas y, en consecuencia, se obtiene un beneficio en el costo, ya que éste se ve disminuido.

La modularidad se logra mediante afinaciones constantes de los procedimientos verificando de lo general a lo particular.

El ocultamiento de la información se presenta cuando diseñamos de tal forma que, los procedimientos y datos que lo integran, sean inaccesibles por otros módulos que no necesiten tal información.

Dentro del diseño estructurado existen los siguientes tipos de módulos:

Secuenciales. Su ejecución es sin interrupción aparente.

Incrementales. Su ejecución puede interrumpirse y restablecerse en el punto donde se interrumpió.

Paralelos. Su ejecución es simultánea con otros módulos.

Independientes. Su ejecución es de forma separada a los otros módulos. La independencia funcional se da como resultado de la modularidad mediante el desarrollo de módulos con una función bien definida.

2.8.3 Cohesión

Es la medición de la fuerza de asociación de los componentes internos de los módulos y se puede ver como una extensión del ocultamiento de información. Nos sirve para verificar la fuerza funcional de un módulo individual al realizar una sola función dentro del sistema y después transmitir el resultado a otro módulo.

El tipo de cohesión puede ser distinto en función de la fuerza de unión de los componentes dentro del módulo siendo la mejor la más fuerte. A continuación se muestran de la más débil (la menos deseada) a la más fuerte (la más deseada).

Cohesión coincidental. Se presenta cuando los elementos internos de un módulo aparentemente no tienen relación, ya que ésta es muy débil.

Cohesión lógica. Aquí existe una relación entre los elementos internos del módulo en cuanto a sus actividades, ya que una puede complementar o ser consecuencia de otra.

Cohesión temporal. Se da cuando se ejecutan en un momento dado todos los elementos de un módulo sin necesidad de un parámetro o lógica.

Cohesión en la comunicación. Aquí no importa la función de cada elemento, se presenta si estas funciones son alimentadas por la misma entrada de datos.

Cohesión secuencial. En este tipo de cohesión cada salida de un elemento es la entrada de otro elemento de manera ordenada.

Cohesión funcional. Es la más deseada por ser fuerte, la idea es de trabajo en equipo por parte de los elementos al estar enfocados a llevar a cabo una misma función.

Cohesión informacional. Ocurre cuando el módulo contiene una estructura de datos compleja, en sí es la abstracción de dichos datos.

2.8.4 Acoplamiento

Es la medición de la fuerza de interconexión entre los módulos, la cual está influida por la complejidad de la interfaz y el tipo de comunicación que existe entre ellos; si es menor la complejidad, la comprensión es más sencilla. También podemos tener niveles de acoplamiento que presentamos a continuación del más fuerte (el menos deseable) al más débil (el más deseable):

Acoplamiento por contenido. Se da cuando un módulo modifica los valores o las instrucciones de otro módulo.

Acoplamiento por zonas compartidas. Se presenta si una zona es utilizada por varios módulos, debido a que se encuentran ligados mediante zonas globales.

Acoplamiento por control. Se presenta cuando un módulo controla la secuencia de otro por contener el paso de banderas de control y de parámetros en forma global.

Acoplamiento por zona de datos. De manera similar a la de zonas compartidas, pero en este caso los elementos globales son compartidos en forma selectiva entre las diversas rutinas que requieren los datos.

Acoplamiento por datos. Ocurre cuando se tiene un módulo con la lista de parámetros, los cuales se pasan a los elementos de las rutinas. Este tipo de acoplamiento es el más deseado ya que si sufre variación sólo se modifican los detalles internos de los módulos, sin tener que modificar las rutinas que usan los módulos modificados.

2.9 La calidad en los sistemas

La situación actual en el desarrollo de software, exige la creación de nuevos productos o servicios con un tiempo de desarrollo mínimo, pero con requisitos estrictos de calidad y seguridad. Las técnicas de desarrollo de software utilizando prototipos, ayudan a reducir el tiempo de desarrollo, a realizar los sistemas de forma concurrente, y a validarlos desde un momento más temprano del ciclo de producción. De esta forma, se pueden cumplir los requisitos de calidad de los clientes y mostrarles una versión del sistema en un tiempo inmediato.

El aseguramiento de la calidad en el desarrollo y en la implantación de aplicaciones de software, amerita un análisis fino y un estudio detallado con el fin de prevenir o anticipar las posibles dificultades o problemas, que se puedan presentar en la realización del mismo.

En un proyecto de sistemas se comienza por la definición, se continúa luego por el diseño o un plan que puede ser general o detallado, pasa a un desarrollo o bien a una adaptación de lo existente, se hace una prueba o bien una instalación y, finalmente, se hace una serie de revisiones después de la instalación. Una desviación durante el proceso del desarrollo del sistema significa puntos menos en calidad total del proyecto y obviamente en el nivel de satisfacción del usuario que disfrutará del desarrollo y de la aplicación que se le esté implantando.

2.9.1 Normas del aseguramiento de la calidad de los sistemas

Para garantizar que un desarrollo de software tenga calidad es necesario revisar todo el entorno del proceso del sistema para su realización, como son: la gente, los productos o herramientas a utilizar, el dinero con el que se cuenta y por supuesto el tiempo disponible. Es necesario partir de unos integrantes sanos y de alta calidad que permitan obtener el más excelente desarrollo informático, como resultado de una buena preparación y de una excelente presentación del producto desarrollado o instalado. Si los desarrolladores no son los adecuados, o si los productos o herramientas de trabajo son limitados, poco confiables o bien no poseemos el suficiente presupuesto o el tiempo apremia y no es posible terminar todo lo que se necesita en el límite fijado, muy probablemente, cualquiera que sea el esfuerzo que hagamos por realizarlo, no obtendremos software de calidad.

Adicionalmente, para lograr una mejor calidad del producto de software se contemplan las siguientes actividades.

- Aplicar métodos y técnicas de control de calidad.
- Realizar revisiones técnicas formales.
- Apegarse a los estándares establecidos.
- Elaborar formas de control de cambios y modificaciones al sistema.
- Realizar pruebas al sistema.
- Realizar mediciones y estimaciones de calidad.
- Elaborar registros e informes concisos.

Elementos para lograr una calidad aceptable en el desarrollo de software.

- Métodos y herramientas de análisis, diseño, codificación y pruebas.
- Revisiones técnicas formales que se aplican en las etapas de desarrollo del proyecto.

- Elaboración de estrategias para realizar pruebas escalonadas.
- Control de la documentación de los cambios realizados al sistema.
- Procedimientos que fijen los estándares utilizados en el sistema.
- Mecanismos para medir la información.

2.9.2 Factores que afectan a la calidad del software

Corrección. El grado en que un programa realiza satisfactoriamente las especificaciones establecidas por los clientes.

Fiabilidad. El grado en que se espera que los resultados del programa realicen sus funciones con la precisión requerida por el cliente.

Eficiencia. La cantidad de recursos de computadora y de código requeridos por el programa para llevar a cabo las funciones.

Integridad. El grado en que se controla el acceso al sistema o a los datos por personal no autorizado.

Facilidad de uso. El esfuerzo requerido para aprender, trabajar, preparar la entrada e interpretar la salida de un programa.

Facilidad de mantenimiento. El esfuerzo requerido para localizar y arreglar un error en un programa.

Facilidad de prueba. El esfuerzo requerido para probar un programa de forma que se asegure que realiza las funciones requeridas.

Portabilidad. El esfuerzo necesario para transferir el programa desde un hardware y/o entorno de sistemas a otro.

Reusabilidad. El grado en que un programa (o partes de un programa) se pueden reutilizar en otras aplicaciones.

2.9.3 Características métricas en la calidad de software

"Facilidad de auditoría. La facilidad con que se puede comparar con los estándares establecidos.

Exactitud. La precisión de los cálculos y el control del programa.

Completitud. El grado en que se ha conseguido la total implementación de las funciones requeridas.

Concisión. Lo compacto que es el programa en términos de líneas de código.

Consistencia. El uso de un diseño uniforme y de técnicas de documentación a lo largo del desarrollo del proyecto.

Estandarización en los datos. El uso de estructuras de datos y de tipos estándar a lo largo de todo el programa.

Tolerancia de errores. El daño que se produce cuando el programa encuentra un error.

Eficiencia en la ejecución. El rendimiento en tiempo de ejecución de un programa.

Modularidad. La independencia funcional de los componentes del programa.

Facilidad de operación. La facilidad para operar el programa.

Seguridad. La disponibilidad de mecanismos para controlar o proteger los programas y la información.

Simplicidad. El grado en que un programa puede ser entendido sin dificultad.

Facilidad de seguimiento. *La posibilidad de seguir paso a paso las instrucciones del programa hacia atrás, hasta llegar a los requerimientos."*[12]

2.10 Pruebas del software

La prueba es uno de los pasos de la ingeniería de software que define una serie de estrategias posibles para garantizar que, al usarse por primera vez el sistema, se encuentre libre de problemas. Algunas veces las pruebas ocasionan que nos demos cuenta que no se tiene el producto esperado; sin embargo, son necesarias para obtener un software de calidad.

2.10.1 Características de las pruebas

En el proceso de aplicación de pruebas se pueden establecer algunas normas características para que el objetivo de la prueba se cumpla.

- La prueba es un proceso que se inicia a nivel de módulo y trabaja "hacia fuera", es decir, hacia la integración del sistema.
- Se aplican diferentes tipos de prueba apropiados en cada caso para diferentes momentos.
- La prueba la lleva a cabo el desarrollador de software y en el caso de grandes proyectos la realiza un grupo de prueba independiente.
- Se realiza la prueba y depuración, aunque son actividades diferentes, es importante incluir la depuración en cualquier estrategia de prueba.
- Decimos que una prueba es exitosa cuando se tiene una alta probabilidad de mostrar algún error, que no se había descubierto hasta el momento.

[12] Zaldivar Zamorategui, Orlando; Apuntes de Ingeniería de Programación; UNAM, Facultad de Ingeniería; Edición 2000; México, Págs. 173

2.10.2 Pasos en la prueba de los sistemas

Las pruebas constan de una serie de tres pasos, que incluyen la dirección a la cual se enfocan las pruebas y se ilustra en la Fig. 16.

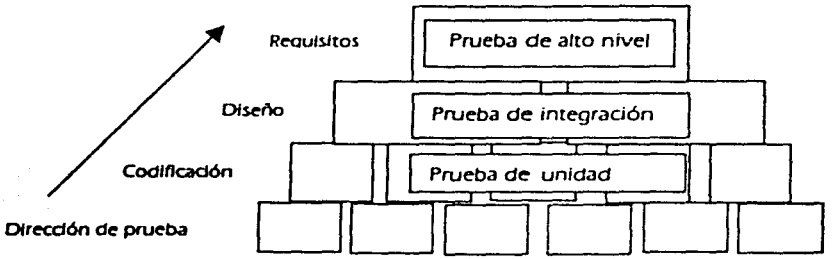


Fig. 16 Pasos en la prueba de los sistemas.

"Un sistema de programación sigue los siguientes tres pasos:

1. Prueba de unidad. Se centra en cada módulo individual, asegurándose que cada uno de ellos funcione correctamente como unidad. Se utilizan caminos específicos de la estructura de control del módulo para asegurar un alcance completo y una detección máxima de errores

2. Prueba de integración. Se deben integrar los módulos para formar el sistema completo. Aquí la prueba se dirige a todos los aspectos asociados con el problema de verificar y construir el programa. Después de que el sistema se ha integrado, se realizan un conjunto de pruebas de alto nivel donde se verifican los criterios de validación. La prueba de validación debe proporcionar la seguridad de que el sistema satisface todos los requisitos funcionales, de comportamiento y de rendimiento.

3. En el último paso de la prueba una vez validado el software, se debe combinar con otros elementos del sistema, como pueden ser el hardware, usuarios, bases de datos. La prueba del sistema verifica que cada elemento encaja de forma adecuada y que se alcanza la funcionalidad y el rendimiento del sistema en su totalidad"[13]

2.10.3 Pruebas de validación

Un software debe contemplar una prueba de validación al procesar datos y ejecutar funciones, de manera que podamos visualizar que el sistema arroja los resultados esperados y si no es así, revisar qué sucede.

La validación del software se consigue mediante una serie de pruebas que demuestran la conformidad con los requisitos. Se determina un plan de prueba, el cual consiste en trazar las pruebas que se han de llevar a cabo. Tanto el plan como el procedimiento estarán diseñados para asegurar que se satisfacen los requisitos funcionales, que se alcanzan todos los requisitos de rendimiento, que la documentación es correcta.

Una vez que se lleva a cabo cada prueba de validación, pueden ocurrir dos condiciones: las características de funcionamiento o de rendimiento están de acuerdo con las especificaciones y son aceptables, otra que se dan cuenta que las especificaciones no se cubren y se encuentran deficiencias.

Un elemento importante en el proceso de validación es el repaso de la configuración del sistema, en donde se intenta asegurar que los elementos que componen el software se han desarrollado de forma correcta, y que están bien detallados para facilitar la fase de mantenimiento que más adelante será necesaria.

También se realizan pruebas de aceptación, para permitir que el usuario final valide todos los requisitos.

[13] Zaldivar Zamorategui ,Orlando; Apuntes de Ingeniería de Programación; UNAM F.I.;Edición 2000; México Pág. 152.

Existen también pruebas llamadas alfa y beta, en los casos de que el software sea empleado por un grupo de usuarios; el objetivo de esta prueba es detectar errores que sólo el usuario final puede determinar.

La prueba alfa es realizada por un cliente en el lugar de desarrollo y un entorno controlado y la prueba beta se lleva a cabo en uno o más lugares en donde se encuentren los usuarios finales.

2.11 Multimedia

La multimedia es una plataforma que emplea los medios (la palabra hablada y escrita, los recursos de audio, las imágenes fijas y en movimiento) para tener una mayor interacción con el usuario. La necesidad de tener una mayor manipulación de los recursos que la computadora nos ofrece a través de estos medios, ha impulsado en gran medida el desarrollo de aplicaciones multimedia, que van desde sistemas operativos gráficos hasta navegadores de internet.

No existe una definición formal del término multimedia, sin embargo, se define como "cualquier combinación de texto, arte gráfico, sonido, animación y video que llega a nosotros por computadora u otros medios electrónicos" [14]

Multimedia estimula los ojos, oídos, el tacto y lo más importante, el desarrollo mental, se integra de combinaciones entrelazadas de elementos de texto, arte gráfico, sonido, animación y video.

En el ámbito educativo, multimedia permite hacer uso de un sin fin de manifestaciones al involucrar información a través de los elementos que hemos mencionado como el texto, la imagen, el video y animaciones, que nos proporcionan un mayor acercamiento a la realidad, especialmente en situaciones en las que resulta difícil presenciar el fenómeno, entender conceptos abstractos o complejidad para mostrar un proceso o un funcionamiento.

[14] Vaughan, Tay; Todo el poder de la Multimedia, Mc Graw Hill, 2ª Edición, 1994, México, pág. 15.

Los profesores son los primeros en darse cuenta de las ventajas que el software de tipo multimedia ofrece a la educación, ya que propicia una enseñanza activa, basada en el descubrimiento, la interacción y la experimentación.

2.11.1 Herramientas de pintura y dibujo

Estas herramientas constituyen uno de los componentes más importantes dentro de la multimedia, porque gracias a éstos se logran crear imágenes y dibujos de gran calidad gráfica, hasta llegar a reproducir escenarios en tercera dimensión que más tarde causan un gran impacto en el usuario.

El software de pintura se utiliza para producir excelentes imágenes de mapas de bits; el de dibujo para trazar con mayor facilidad en papel utilizando formatos post script. Los paquetes de dibujo incluyen poderosas y costosas tecnologías de diseño asistido por computadora, el cual se utiliza cada vez más para proporcionar gráficos en tercera dimensión.

En algunas aplicaciones se combinan tanto capacidades de dibujo como de pintura, pero algunos sistemas sólo pueden importar imágenes de mapas de bits. En general, este tipo de imágenes son las que nos proporcionan los efectos y los detalles más finos, por lo tanto son las más utilizadas en multimedia y no tanto las imágenes dibujadas.

Las características que debe tener un software de dibujo o pintura son:

- Una interfaz intuitiva con menús desplegables, barras de estado, control de paleta y cuadros de diálogo para una selección rápida y lógica.
- Dimensiones escalables para redimensionar, estirar y distorsionar tanto los mapas de bits pequeños como los grandes.
- Una herramienta de pintura para crear formas geométricas, desde cuadrados hasta círculos y desde curvas hasta polígonos complejos.

- Habilidad para regar un color, patrón o gradiente en cualquier área.
- Habilidad para pintar con patrones y arte de recortes.
- Tamaño y forma de pluma ajustable.
- Soporte para fuente de texto escalable y sombreada.
- Acercamiento para edición de pixeles.
- Buena administración de paleta con el modo de 8 bits.

Software disponible para pintura y dibujo:

Canvas, Charisma, ColorStudio, CorelDraw, Superpaint, Designer, Deskdraw, Fractal, Cricket Draw, MacPaint, MacDraw Pro, Professional Draw, Desing Painter, Harvard Graphics, Image Studio.

2.11.2 Herramientas de edición de imagen

La aplicación de edición de imagen son herramientas especializadas y poderosas para realzar y retocar las imágenes de mapas de bits, usualmente designadas como separaciones de color para impresiones. Estos programas son también indispensables para presentar las imágenes utilizadas en las presentaciones de multimedia. Las versiones más recientes de estos programas brindan algunas características y herramientas de los programas de pintura y dibujo y pueden utilizarse para crear imágenes desde cero, digitalizarlas, tomar cuadros de video, cámaras digitales, archivos de reportes de arte o archivos de gráficos creados.

Algunas de las características de las aplicaciones de edición de imágenes son:

- Ventanas múltiples que proporcionan vistas de más de una imagen al mismo tiempo.
- Conversión de los principales tipos de datos, de imagen y formato de archivos.
- Introducción directa de imágenes del scanner y fuente de video.

- Herramientas de selección como rectángulos, lasos y varitas mágicas para seleccionar porciones de un mapa de bits.
- Característica de deshacer y restablecer.
- Capacidad de alisado y controles de rugosidad y suavidad.
- Buenas características de enmascarado.
- Transformaciones geométricas como girar, sesgar, rotar, distorsionar, y cambiar las perspectivas.

Formatos de imágenes

FORMATOS	DESCRIPCIÓN
GIF	Formato gráfico desarrollado por Compuserve y destinado en un principio a imágenes de 8 bits (256 colores). Lleva la extensión GIF.
JPG JPEG	JPEG (Joint Photographics Expert Group File Interchange Format) Es el formato más apropiado para comprimir imágenes fotográficas con gran detalle, permite utilizar hasta 16,777,216 colores a 24 bits.
TIFF	Tagged Information File Format de imagen de carácter universal. Admite información adicional como canales de Photoshop o compresión LZW. Lleva la extensión TIF.
Targa	Desarrollado por Truevision para la representación de imágenes en color real (24). Lleva la extensión TGA.
BMP	Formato Bitmap de Microsoft Windows, tanto para 8 como para 24 bits. Admite compresión propia (RLE). Lleva la extensión BMP.
PCX	Formato desarrollado en un inicio para Paintbrush y extendido luego a otros soportes. Lleva la extensión PCX.

Software disponible para la edición de imágenes:

Color It, Dfoto, Digital, Gallery Efects, Composer, Picture Publisher, ColorStudio, PhotoShop, Photo Styler.

2.11.3 Programas de edición de sonido

Las herramientas para edición de sonidos digitalizados y MIDI permiten ver la música mientras se escucha, al dibujar la representación de un sonido en pequeños incrementos, ya sea en partitura o en forma de onda, puede cortar, pegar o editar segmentos con gran precisión, algo imposible de hacer en tiempo real.

Para incorporar un archivo de sonido MIDI a un proyecto de multimedia, se debe conocer la manera en que la música se secuencia, representa o publica y se requiere de un sintetizador MIDI o dispositivo conectado a la computadora. Para lograr que las aplicaciones multimedia funcionen correctamente debe existir una compatibilidad de tecnología entre el software y el hardware.

Dentro de los formatos de sonido, más utilizados en la multimedia están:

FORMATOS	DESCRIPCIÓN
MIDI	(Musical Instruments Digital Interface) ocupan menos espacio que los ficheros WAVE.
WAV	Formato de forma de onda de Microsoft Windows. Admite 8 y 16 bits, y frecuencias de muestreo de 11, 22 y 44 KHz. Lleva la extensión WAV.
MP3	Se deriva de MPEG (Grupo de Expertos de Imágenes en Movimiento) 1 Layer 3, lo que comprende el 3er nivel de compresión del MPEG 1. Una compresión de audio de la mejor calidad en bajas tasas de bits. Trabaja con 64 kbps por canal de audio.

Software disponible para la edición de sonido:

Alchemy, Encore, WaveEdit, Midisoft Studio, AudioTrax, TurboTrax, AudioShop, SoundEdit Pro, Forge XP.

2.11.4 Programas de video y animaciones

Las características que definen la calidad de un video o animación digital, es por el número de fotogramas que puede mostrar por segundo, la resolución de color, la aplicación de algún algoritmo de compresión y el tamaño de la ventana donde será visualizado.

Los formatos más usuales para video y animación son los siguientes:

FORMATOS	DESCRIPCIÓN
Video For Windows	Sistema de video de Microsoft para Windows. Admite de 8 a 24 bits de imagen y de 8 16 bits de sonido. Lleva la extensión AVI.
QuickTime for Windows	Sistema de video de Apple Computer para Windows. Admite las mismas capacidades que AVI. Lleva la extensión MOV.
MPEG	Motion Picture Expert Group. Desarrollado por un grupo de expertos para la representación de video de alta calidad. Necesita software de descompresión propio. Lleva la extensión MPG.
Flic	Formato de animación desarrollado por Auotdesk para su programa Animator y que se ha universalizado dentro del PC. Extensión FLI o FLC.

2.11.5 Editores de texto

El texto es el apartado al que menor atención se le presta, pero suele ser indispensable en la mayoría de las producciones. Normalmente se presenta en formato ANSI (estándar de Windows), cuyo uso está muy extendido, por lo que el intercambio de la información suele ser de lo más fácil (comúnmente se usa el portapapeles).

Formatos de texto

FORMATOS	DESCRIPCIÓN
TXT	Texto universal en formatos ANSI o ASCII.
DOC	Formato más comúnmente utilizado de Microsoft Word, permite insertar fotografías, sonidos y películas dentro de un documento de texto.
RTF	Rich Text Format (Formato de Texto Enriquecido). Permite características de color, negritas, etc.

2.12 Otras herramientas

2.12.1 Corel draw

Entre las herramientas de dibujo más populares se encuentra Corel draw. Este software permite crear y editar líneas, formas o caracteres con facilidad y precisión; ajustar el texto tomando como referencia una curva y la vectorización automática de gráficos. También permite conseguir efectos especiales como situar una línea de texto o algún objeto, en perspectiva, entre muchas otras capacidades. Corel draw es un conjunto de aplicaciones y utilidades con distintos propósitos, pero generalmente se utiliza para

elaborar dibujos e ilustraciones con manejo de texto como se ilustra en la Fig. 17, que describe el ambiente de trabajo.

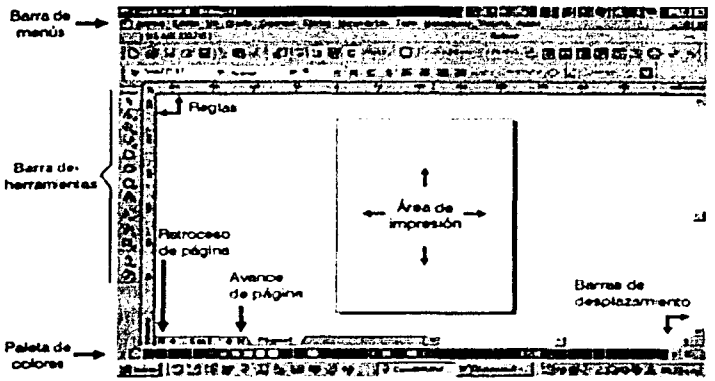


Fig. 17 Ambiente de trabajo de Corel draw.

Área de impresión. Al crear un dibujo nuevo en Corel draw, la gran porción blanca de la pantalla es la ventana de dibujo. El rectángulo del centro con la sombra proyectada, representa la página Dibujo. Normalmente, sólo se imprime la parte del dibujo que queda incluida en la página de dibujo.

Imaginemos que el espacio restante en la ventana Dibujo, es el área de trabajo en la cual podemos tener a la mano las herramientas y los elementos que emplearemos en nuestra la ilustración.

Barra de herramientas. La caja de herramientas puede desplazarse a cualquier parte de la ventana de dibujo, haciendo clic y arrastrándola al área que rodea a las herramientas. Los comandos de aplicación disponibles en las barras de menú también están disponibles en las barras de herramientas y menús laterales. Las barras de propiedades y persianas permiten acceder rápidamente a funciones utilizadas con frecuencia.

Las barras de propiedades, están accesibles mientras se trabaja en un documento, nos proporcionarán acceso a comandos relacionados con la herramienta activa o a la tarea que se esté llevando a cabo en ese momento. Contiene herramientas que realizan la mayoría de las funciones de dibujo y edición de Corel draw. Para seleccionarlas, basta con pulsar sobre el botón de la herramienta.

Paleta de colores. Nos permite aplicar colores de relleno y de contorno. Para cambiar los colores de la paleta de color, se hace clic en uno de los botones de desplazamiento para ver los colores, uno por uno o la paleta entera.

Regletas. Estas reglas se pueden activar o desactivar seleccionando el menú Ver- Reglas. Las reglas permiten juzgar con rapidez y precisión sobre el tamaño y situación relativa de los objetos.

Menú principal. Contiene diez menús que son Archivo, Edición, Ver, Diseño, Organizar, Efectos, Mapas de bits, Texto, Herramientas, Ventana y Ayuda; pueden abrirse o desplegarse pulsando sobre sus nombres.

Algunas ventajas:

La ventana acoplable es una nueva función de Corel draw que es similar a una persiana, es posible acoplarla al lado de la ventana de aplicación.

Otra función en Corel draw es la capacidad de crear varias áreas de trabajo a la vez. El Área de trabajo es una determinada configuración de valores en el cuadro de diálogo con opciones para poder guardar y volver a aplicar.

2.12.2 Photoshop

Adobe Photoshop es un editor que permite crear imágenes con excelente calidad, las cuales pueden ser publicadas en impresión, en internet u otro medio.

También es factible crear gráficos basados en datos, por lo que si cambian estos últimos el gráfico se actualiza, se puede anexar audio, y proteger los archivos con contraseña.

El ambiente de trabajo es una analogía con el ambiente de un pintor de arte, tales como pinceles, pluma, herramientas, licuar y pincel turbulencia que distorsionan la imagen con precisión; además, cuenta con un creador de motivos para texturas de fondo.

Con archivos photoshop se crean animaciones GIF con el uso de Adobe Ilustrator. Contiene varias herramientas para manejo de texto, similares a las aplicaciones de office, como Word o Power point, con las que se puede distorsionar el texto, formato de texto y caracteres, corrector ortográfico entre otros.

Como la mayoría de las aplicaciones de edición, sus barras de herramientas se ubican en las orillas de la pantalla de trabajo, ya sea en forma horizontal en la parte superior o inferior o en forma vertical en la parte izquierda o derecha. Su paleta de accesorios aparece en la parte izquierda de la pantalla, verticalmente, como se detalla en la Fig. 18.

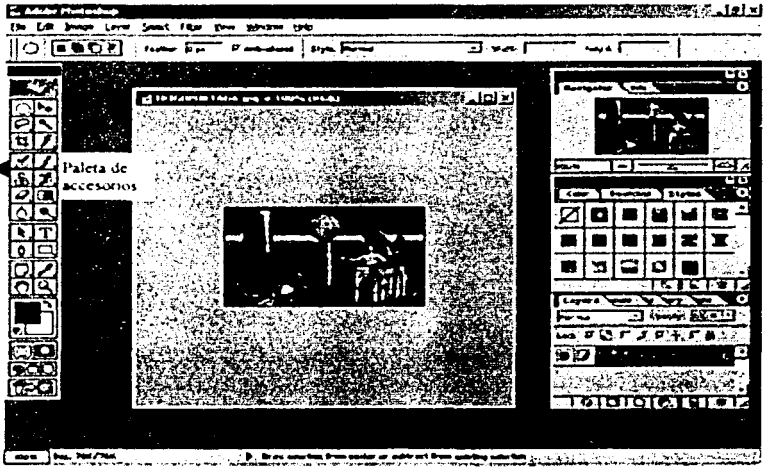


Fig. 18 Ventana de trabajo en Photoshop.

2.12.3 Xara 3D

Xara 3D es un programa de animación que, a diferencia de otros programas muy grandes y complejos, ayuda a los desarrolladores de software a realizar animaciones de texto y botones de una manera rápida y con una buena calidad; además, se pueden convertir las animaciones a formatos para ser transmitido en cualquier archivo para Internet.

En Xara 3D se pueden realizar imágenes tridimensionales totalmente animadas con texturas, iluminación, sombreados, efectos de profundidad, movimiento, de una calidad alta que pueden utilizarse en páginas web o en cualquier otra parte donde se necesite. Estas imágenes se pueden transportar a un formato de tipo película AVI, GIF o JPG.

La ventana principal al abrir Xara 3D es la que aparece en la Fig. 19.

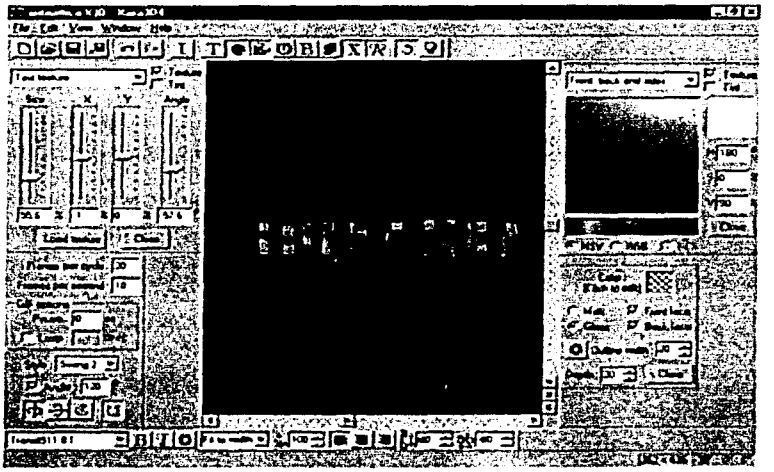


Fig. 19 Ventana Principal de Xara 3D.

Especificaciones mínimas de hardware para utilizar Xara 3D

Xara 3D opera con equipos Pentium que sean compatibles con Windos 95 o superior; requiere de una memoria RAM de 32 MB, resolución de monitor alta SVGA; como es muy pequeño requiere tan sólo de 5 MB de espacio disponible en disco duro para su instalación.

Descripción del ambiente de trabajo

Menú principal. Contiene cinco menús que son Archivo, Edición, Ver, Ventana y Ayuda; pueden abrirse o desplegarse pulsando sobre sus nombres.

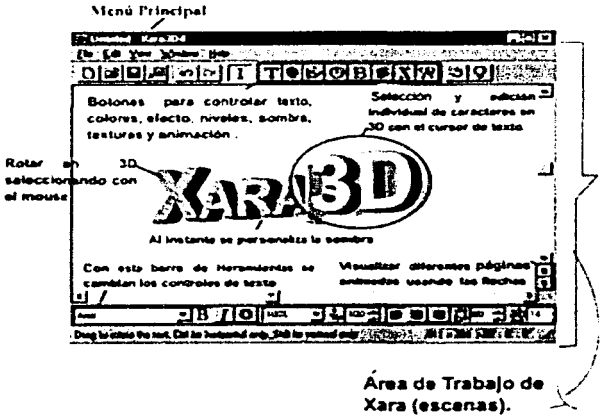


Fig. 20 Ambiente de trabajo de Xara 3D.

Barra de herramientas. La caja de herramientas se encuentra ubicada debajo del menú principal y haciendo clic en cada botón de ésta se puede obtener una función para ser aplicada a nuestro texto, como: color, efecto, luz, textura, animación. En esta barra también se puede guardar el archivo,

abrir uno nuevo o regresar a la acción anterior. Todas las funciones de la barra de herramientas también están disponibles en el menú principal.

Forma de trabajo. Una vez instalado Xara 3D, se abre una ventana llamada escena, en la cual aparece el texto XARA 3D; debemos dar doble clic para anotar la frase deseada. Posteriormente, se procede a escoger la fuente, ajustar el borde, agregar un fondo de tipo bitmap, sombra, luminosidad, diferentes colores y, finalmente, arrastrar el texto con el ratón hasta conseguir el ángulo y tamaño que se necesita; todas estas funciones se usan con la barra de herramientas que se detallan en la Fig. 20.

Una vez terminada la escena, procedemos simplemente a seleccionar la opción de menú, file y exportar el trabajo al formato que necesitemos (GIF, AVI, JPG).

2.12.4 Macromedia Flash 5

Las películas de Flash son imágenes y animaciones que están compuestas principalmente por imágenes vectoriales, aunque pueden incluirse imágenes de mapa de bits y sonidos importados. Las películas Flash pueden incorporar interacción para permitir la introducción de datos de los espectadores, creando películas no lineales que pueden interactuar con otras aplicaciones. Los diseñadores de la Web utilizan Flash para crear controles de navegación, logotipos animados, animaciones de gran formato con sonido sincronizado e incluso sitios Web con capacidad sensorial. Las películas Flash son gráficos vectoriales compactos que se descargan y se adaptan de inmediato al tamaño de la pantalla del usuario.

Flujo de trabajo de Flash. El trabajo en Flash para la creación de una película incluye el dibujo o la importación de una ilustración, su organización en el Escenario y su animación con la Línea de tiempo. La película puede hacerse interactiva utilizando acciones que hagan que la película responda a determinados eventos de cierta manera.

Una vez terminada la película, es posible exportarla para verla en Flash Player o bien como un proyector de Flash independiente, lo cual permite verla con un reproductor que se incluye con la película misma.

Las películas de Flash pueden reproducirse de varias formas:

- En navegadores Internet, tales como Netscape Navigator y Microsoft Internet Explorer, que estén equipados con Flash Player.
- En Flash Player, una aplicación independiente de manejo similar al complemento Flash Player.
- Con el control ActiveX de Flash en Microsoft Office, Microsoft Internet Explorer para Windows y otros entornos anfitrión de ActiveX.
- Como un proyector independiente, un archivo de película que se puede reproducir sin disponer de Flash Player.

Flash ofrece varios métodos, tanto para crear ilustraciones originales como para importarlas desde otras aplicaciones. Puede crear objetos con las herramientas de dibujo y pintura, así como modificar los atributos de los objetos existentes. También puede importar gráficos vectoriales y de mapa de bits desde otras aplicaciones y modificarlos en Flash. Además, tiene capacidad para importar archivos de sonidos y agregarlos en la película.

Animación en Flash. Flash permite animar objetos para dar la impresión de que se mueven por el Escenario, así como cambiar su forma, tamaño, color, opacidad, rotación y otras propiedades. También puede crear animación, fotograma a fotograma, creando una imagen diferente para cada fotograma. Otra posibilidad consiste en crear animación interpolada, es decir, crear los fotogramas primero y último de una animación y dejar que Flash cree los fotogramas intermedios.

Flash permite crear películas interactivas, en las que los espectadores pueden utilizar el teclado o el ratón para pasar a diferentes partes de la película, mover objetos, introducir información en formularios y realizar muchas otras acciones.

Entorno de trabajo de Flash. Para crear y editar películas, es necesario conocer el entorno de trabajo y hacer uso correcto de las propiedades que cada herramienta nos ofrece. En la Fig.21 se señalan los elementos de este entorno de trabajo.

PantallaPrincipal

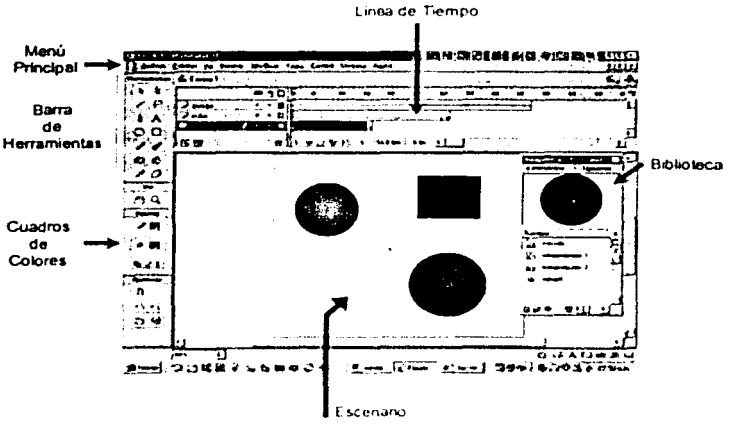
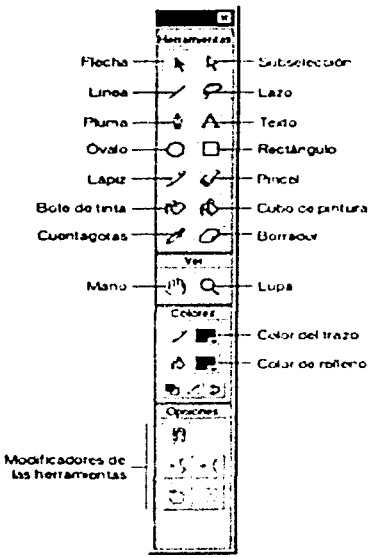


Fig. 21 Entorno de trabajo de Flash.

Herramientas de pintura y dibujo. Flash incorpora varias herramientas para dibujar formas libres o líneas precisas, formas y trazados, así como para pintar objetos rellenos. Estas herramientas se describen en la Fig. 22.



La herramienta Lápiz nos sirve para dibujar líneas y formas libres de manera muy similar a un lápiz real.

Si se necesita dibujar trazos precisos, como líneas rectas o curvas, se utiliza la herramienta Pluma.

Para dibujar formas geométricas básicas, se utilizan las herramientas Línea, Óvalo y Rectángulo.

La herramienta Pincel nos ayuda para crear trazos similares a los obtenidos al pintar en un lienzo.

Fig. 22 Herramientas de pintura y dibujo de Flash.

Al utilizar una herramienta de dibujo o pintura para crear un objeto, la herramienta aplica los atributos actuales de relleno y trazo al objeto. Para cambiar los atributos de relleno y trazo de los objetos existentes, se pueden utilizar las herramientas Cubo de pintura y Bote de tinta.

Los rellenos y los trazos son tratados como objetos independientes. Podemos seleccionar por separado los rellenos y los trazos para moverlos o modificarlos.

También se puede utilizar el ajuste para alinear automáticamente varios elementos entre sí, o bien con las guías o la cuadrícula de dibujo.

Escenario y Línea de tiempo. Al igual que en un largometraje, las películas de Flash dividen el tiempo en fotogramas. En el Escenario se compone el contenido de los fotogramas individuales de la película, dibujándolos directamente o bien organizando ilustraciones importadas.

En la Línea de tiempo se coordina el tiempo de la animación y se ensambla la ilustración en distintas capas. La Línea de tiempo muestra todos los fotogramas de la película, como se muestra en la Fig. 23.

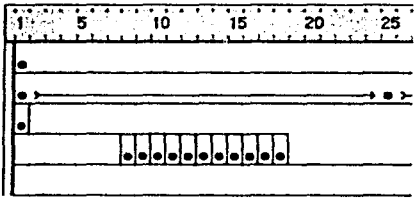


Fig. 23 Línea de tiempo de Flash.

En la Línea de tiempo se coordina el tiempo de la animación y se ensamblan las distintas capas.

Las capas actúan como una serie de hojas de acetato transparente superpuestas, manteniendo las diferentes ilustraciones por separado, de forma que puedan combinarse distintos elementos en una imagen visual cohesionada.

Las capas en la animación. Las escenas de las películas de Flash pueden constar de varias capas. Se utilizan capas para organizar los componentes de las secuencias de animación y para separar objetos animados de forma que no se borren, conecten ni segmenten entre sí. Si se desea que Flash interpole el movimiento de varios grupos de símbolos a un tiempo, cada uno debe estar en una capa distinta. En general, la capa de fondo contiene imágenes estáticas. Las capas adicionales contienen un objeto animado independiente cada una, Fig. 24.

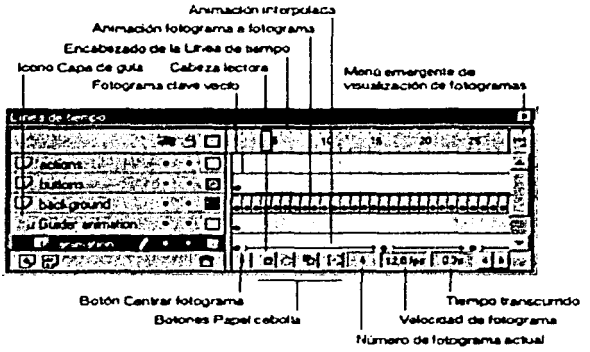


Fig. 24 Visualización de las capas de animación en Flash.

Las capas aparecen como filas en la Línea de tiempo.

Si una película tiene varias capas, puede ser difícil el seguimiento y la edición de los objetos en una o dos de ellas. Es más sencillo si se trabaja con el contenido de cada capa por separado.

Si la velocidad del fotograma es muy baja, la animación parece detenerse y volver a empezar y si es muy alta los detalles se ven borrosos. Es recomendable la velocidad de 12 fotogramas por segundo (fps) para obtener un óptimo resultado en la Web. En general, la velocidad estándar de movimiento de la imagen es de 24 fps, pero en las películas QuickTime y AVI es habitualmente de 12 fps.

La complejidad de la animación y la velocidad del sistema donde se reproduce afectan a la suavidad de la reproducción. La velocidad del fotograma es única para toda la película Flash, por lo que es preferible establecerla antes de comenzar a crear la animación.

Flash puede crear dos tipos de animación interpolada. En el primero, denominado interpolación de movimiento, se definen propiedades como la posición, el tamaño y la rotación de una instancia, un grupo o un bloque de texto en un punto en el tiempo, y estas propiedades se cambian en otro punto. En el segundo, denominado interpolación de forma, se dibuja una

forma en un punto del tiempo y se cambia o se dibuja una nueva en otro punto. Flash interpola los valores o formas de los fotogramas intermedios para crear la animación.

Ventana Biblioteca. La ventana Biblioteca es donde se guardan y organizan los símbolos creados en Flash, además de archivos importados, tales como archivos de sonido, imágenes de mapa de bits o películas de QuickTime. En la ventana Biblioteca podemos organizar en carpetas los elementos de biblioteca, ver con qué frecuencia se utilizan en una película y ordenarlos por tipo. Fig. 25.

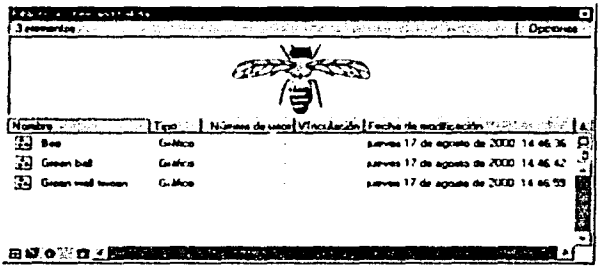


Fig. 25 Ventana de Biblioteca.

2.12.5 3D Canvas

Es un herramienta de animación y modelaje para un ambiente en tres dimensiones (3D). Se puede usar para crear modelos simples o largas escenas de animación. 3D Canvas contiene herramientas para crear, deformar, esculpir y pintar objetos en 3D. Se pueden situar objetos 3D en una escena y animar su posición usando animación basada en key-frame (cuadros clave).

Cuando se inicia el software, se crea automáticamente una escena semejante a la que se muestra en la Fig. 26, la cual contiene diferentes herramientas, como el panel de componentes, la barra de herramientas

primarias, la barra de estado, la de edición, con las cuales es posible modificar y crear objetos. 3D Canvas usa un sistema de coordenadas en tres dimensiones, donde el eje X es la ubicación en la pantalla de izquierda a derecha, el eje Y es de arriba a abajo y el eje Z lejos y cerca.

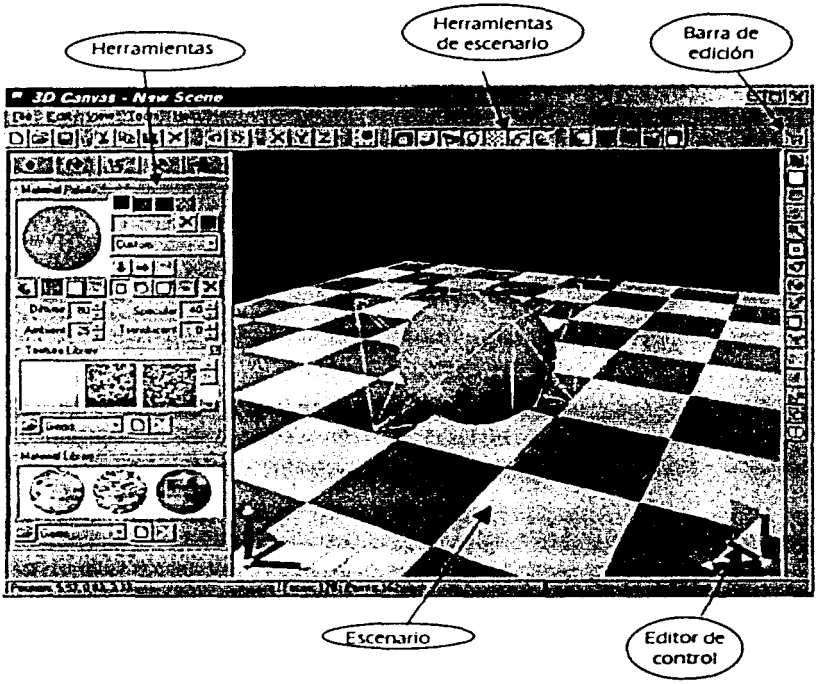


Fig. 26 Escenario 3D Canvas.

A continuación se explican las características de las barras de herramientas que utiliza 3D de Canvas.

Módulo de herramientas. Este módulo está formado a su vez por el panel de componentes, la paleta de materiales, el panel de operación y el panel de capas.

El panel de componentes se usa para crear y manejar escenas, contiene objetos básicos como un cubo, una esfera, un cilindro y un cono, los que se pueden agregar con tan sólo seleccionarlos y arrastrarlos, colocándolos en la escena; también contiene luces con las que se puede iluminar el escenario desde distintos ángulos.

El panel de operación se utiliza para aplicar modificaciones de los objetos y el panel de capas se usa para almacenar los cambios que se hagan en los objetos.

Para asignar una textura al objeto, se selecciona el panel de materiales, localizado en la derecha de la presentación, como se muestra en la Fig. 26, el que se divide a su vez en la paleta de materiales, la librería de texturas y librería de materiales. En la paleta de materiales se puede establecer que la textura sea de un color fijo, con una cierta transparencia, o se puede seleccionar de la librería de texturas.

Editor de control. Para modificar un objeto que se haya colocado en la escena se utiliza el editor de control, que se encuentra localizado en la esquina inferior derecha del escenario. Con este control se puede escalar el tamaño, mover la posición o girar el objeto en pantalla, de acuerdo a los ejes x, y, z. Para girar un objeto se selecciona con el ratón el eje sobre el cual se quiera girar el objeto.

Herramientas de escenario. Estas herramientas se utilizan para controlar el escenario; puede ocultar o mostrar los ejes de coordenadas, las luces, el plano guía, así como controlar la atmósfera del escenario.

Barra de edición. Por medio de las herramientas de esta barra, se hacen modificaciones en los objetos, teniendo la libertad de seleccionar ciertas áreas y puntos, para cambiar la forma y diseño del objeto.

Animación. Los objetos en 3D Canvas se pueden animar haciendo uso de funciones de animación basadas en key-frame (cuadro principal), que viene siendo una instantánea de cómo se ve una escena en cierto punto del tiempo.

La barra de herramientas de animación se encuentra localizada en la parte inferior de la aplicación, ver Fig. 27, la cual consta de los controles para grabar, reproducir en pantalla o bien para grabar como un archivo de formato de película "avi".

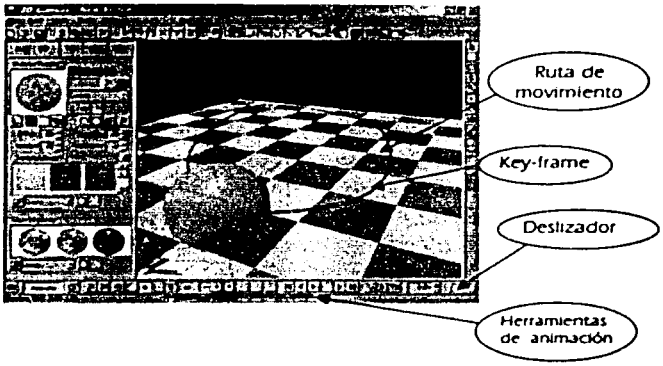


Fig. 27 Proceso de animación en 3D Canvas.

Las escenas pueden presentarse o grabarse desde un punto de vista de la cámara de animación.

Cada posición en el deslizador de la animación es un tiempo de key-frame; se puede poner la posición y orientación para cada objeto en cada momento de key-frame en el deslizador. Cuando la animación se ha reproducido, 3D Canvas interpolará las posiciones entre los key-frames.

2.13 Software de programación

2.13.1 Visual Basic

2.13.1.1 Introducción

Visual Basic es un lenguaje de programación visual, también llamado lenguaje de cuarta generación. Esto quiere decir que un gran número de tareas se realizan sin escribir código, simplemente con operaciones gráficas realizadas con el ratón sobre la pantalla.

Visual Basic es también un programa basado en objetos, aunque no orientado a objetos como Visual C++. La diferencia está en que Visual Basic utiliza objetos con propiedades y métodos, pero carece de los mecanismos de herencia y polimorfismo, propios de los verdaderos lenguajes orientados a objetos como Java y C++.

Como el nombre lo indica, una gran parte de la programación con Visual Basic se realiza visualmente. Esto significa que durante el tiempo de diseño se tiene la capacidad de ver la forma en que el programa se verá al ejecutarse. Ésta es una gran ventaja sobre otros lenguajes de programación, debido a que se tiene la capacidad de cambiar y experimentar con el diseño hasta que se esté satisfecho con los colores, proporciones e imágenes que incluyan en un programa.

Visual Basic está orientado a la realización de programas para Windows, pudiendo incorporar diferentes tipos de elementos de este ambiente como: ventanas, botones, cajas de diálogo y de texto, botones de opción y de selección, barras de desplazamiento, gráficos, menús, etc.

2.13.1.2 Conceptos básicos de Visual Basic

Formularios. El primer paso para iniciar una aplicación con Visual Basic es crear la interfaz, aplicación visual con la que va a interactuar el usuario. Se utilizan formularios y controles, que son los elementos de desarrollo básicos para crear aplicaciones.

Controles. Los controles son objetos que están contenidos en los objetos de formularios. Cada tipo de control tiene su propio conjunto de propiedades, métodos y eventos, que lo hacen adecuado para una finalidad determinada. Algunos de los controles que puede usar en las aplicaciones son más adecuados para escribir o mostrar texto, mientras que otros controles permiten tener acceso a otras aplicaciones y procesan los datos como si la aplicación remota formara parte del código.

Los formularios y controles de Visual Basic son objetos que exponen sus propios métodos, propiedades y eventos. Las propiedades se pueden considerar como atributos de un objeto, los métodos como sus acciones y los eventos como sus respuestas.

En la Fig. 28 se muestra el ambiente de diseño para elaborar aplicaciones en Visual Basic, en el cual observamos algunos ejemplos de un formulario y controles típicos que se utilizan generalmente.

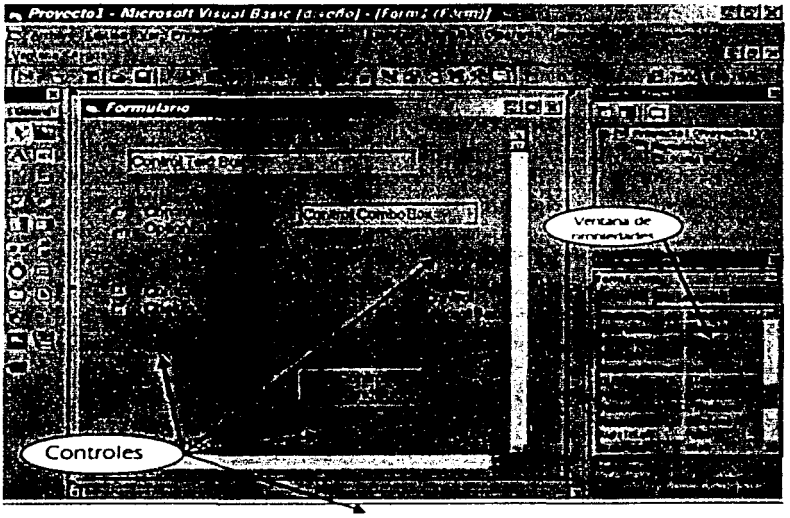


Fig. 28 Ambiente de diseño para Visual Basic.

Propiedades. Las propiedades proporcionan una información detallada y descriptiva de un objeto o control. Mediante el establecimiento de las propiedades del formulario y la escritura de código de Visual Basic para responder a sus eventos, se personaliza el objeto para cubrir las necesidades de la aplicación.

Métodos. Los métodos son funciones que también son llamadas desde programa, pero a diferencia de los procedimientos no son programadas por el usuario, sino que vienen ya preprogramadas con el lenguaje. Los métodos realizan tareas comunes y previsibles para todas las aplicaciones, de ahí que vengan con el lenguaje y que se le quite al usuario la tarea de programarlos. Cada tipo de objeto o de control tiene sus propios métodos. En general, sólo pueden ser ejecutados en tiempos de ejecución, no en tiempo de diseño.

Los métodos son invocados dando nombres al objeto y cuyo método se está llamando, listando el operador punto (.), y después listando el nombre del método. Como cualquier rutina, los métodos pueden incorporar argumentos.

Evento. Un evento es algo que sucede durante la ejecución de una aplicación, generalmente provocado por el usuario. Un evento puede ser la acción de oprimir una tecla, una selección de menú, un clic o un movimiento del ratón, o cualquier otro evento posible.

Procedimiento de evento. Una aplicación de Visual Basic es una colección de pequeñas rutinas llamadas procedimientos de eventos. Estos procedimientos se ejecutan sólo en el caso de que ocurra un evento. Los procedimientos son nombrados automáticamente bajo la siguiente convención:

```
NombreDelControl_nombreDelEVento( )
```

2.13.1.3 Instrucciones de Visual Basic

a) Tipos de datos

Los datos se agrupan en tres categorías: numéricos, cadenas y especiales. Los que soporta Visual Basic son los siguientes:

- Bolean. Sólo tienen dos valores, verdadero o falso.
- Byte. Valores numéricos positivos entre 0 y 255.
- Currency. Valores monetarios.
- Date. Contiene valores de fecha y hora.
- Double. Valores numéricos de doble precisión.
- Integer. Valores enteros, entre –32,768 y 32,767.
- Long. Valores enteros entre –2,147,483,648 y 2,147,483,647.
- Object. Contiene referencias a objetos o controles.
- Single. Valores numéricos no enteros de simple precisión.
- String. Contiene valores alfanuméricos, de longitud fija o variable.
- Variant. Tipo de datos predeterminado por VB. Se usa cuando se desconoce el tipo de datos que se almacenará en una variable.

b) Declaración de variables

Para declarar una variables se usan las siguientes instrucciones:

Dim nombreVariable [As tipo de datos]

Esta instrucción sirve para declarar variables en forma local para un procedimiento. Las variables sólo existirán mientras se ejecuta el procedimiento; cuando termina éste, desaparece el valor de la variable; por lo cual se pueden usar las mismas variables en diferentes procedimientos.

Public nombreVariable [As tipo de datos]

Mediante esta instrucción se declaran variables de manera pública para que estén accesibles a todos los procedimientos de una aplicación.

Static nombreVariable [As tipo de datos]

Declarar una variable local mediante la palabra clave Static preserva su valor aunque termine el procedimiento.

c) Procedimientos

Se pueden simplificar las tareas de programación, si se dividen los programas en componentes lógicos más pequeños. Estos componentes, llamados procedimientos, pueden convertirse en bloques básicos que permiten mejorar y ampliar Visual Basic.

En Visual Basic se utilizan los siguientes tipos de procedimientos:

Procedimientos Sub. Un procedimiento Sub es un bloque de código que se ejecuta como respuesta a un evento; puede tomar argumentos, realizar una serie de instrucciones y cambiar el valor de los argumentos. No devuelven valor. La sintaxis de un procedimiento Sub es la siguiente:

[Private | Public][Static]Sub nombreProcedimiento (argumentos)

instrucciones
End Sub

Procedimientos Function. Trabajan igual que un procedimiento Sub, la diferencia con los procedimientos Function es que sí pueden devolver un valor al procedimiento que realiza la llamada. Así mismo, se pueden crear procedimientos propios.

[Private | Public][Static]Function nombreProcedimiento (argumentos)[As tipo]
instrucciones
End Function

d) Sentencias condicionales

Estas estructuras de control determinan cuál será el flujo de la información que deberá seguir un programa dependiendo de la condiciones que se

cumplan. A continuación, se muestran las sentencias condicionales que utiliza Visual Basic.

Sentencia If ... then ... else. Un bloque If...then... else se utiliza para definir varios bloques de instrucciones, uno de los cuales se ejecutará:

```
If condición1 then
    bloque de instrucciones 1
Else
    bloque de instrucciones 2
End If
```

Select ... Case. Visual Basic proporciona la estructura Select Case como alternativa a If...then...else para ejecutar selectivamente un bloque de instrucciones entre varios bloques de instrucciones.

```
Select Case expresión
Case valor1
    bloque de instrucciones 1
Case valor2
    bloque de instrucciones 2
    ..............
Case Else
    bloque de instrucciones n
End Select
```

e) Bucles

Un bucle es un conjunto de instrucciones que se ejecutan repetidamente. Las estructuras de bucle que acepta Visual Basic son:

For...Next. Este tipo de bucles se utiliza cuando se sabe que se van a ejecutar las instrucciones un número determinado de veces. El bucle For utiliza una variable llamada contador que incrementa o reduce su valor en cada repetición del bucle. La sintaxis es la siguiente:

```
For contador = iniciar To finalizar [Step incremento]
    instrucciones
```

Next [contador]

Los argumentos contador, iniciar, finalizar e incremento son numéricos.

For Each...Next. El bucle For Each...Next es parecido al bucle For...Next, pero repite un grupo de instrucciones por cada elemento de una colección de objetos o de una matriz, en vez de repetir las instrucciones un número especificado de veces. Esto resulta especialmente útil si no se sabe cuántos elementos hay en la colección. Se muestra a continuacion su sintaxis:

```
For Each elemento In grupo
    instrucciones
Next elemento
```

Do ... Loop. Se utiliza este bucle para ejecutar un bloque de instrucciones un número indefinido de veces. Las instrucciones se ejecutarán siempre y cuando la condición sea True (Verdadera) o False (Falsa), según la modalidad que se escriba.

Si la Condición =	Hace el bucle cero o más veces	Hace el bucle al menos una vez
False	Do While condición Instrucciones Loop	Do Instrucciones Loop While condición
True	Do Until condición Instrucciones Loop	Do Instrucciones Loop Until condición

Compilación. Visual Basic es un lenguaje de cuarta generación, esto quiere decir que un gran número de tareas se realizan sin escribir código, simplemente con operaciones gráficas realizadas con el ratón sobre la pantalla. También está considerado como lenguaje de alto nivel. Los lenguajes de alto nivel son más o menos comprensibles para el usuario, pero no para el procesador. Para que éste pueda ejecutarlos es necesario traducirlos a su propio lenguaje de máquina. Al paso del lenguaje de alto nivel al lenguaje de máquina se le denomina compilación. En Visual Basic

esta etapa no se aprecia tanto como en otros lenguajes, donde el programador tiene que indicar a la computadora explícitamente que realice dicha compilación. Los programas de Visual Basic se dice que son interpretados y no compilados, ya que el código no se convierte a código máquina, sino que hay otro programa que durante la ejecución "interpreta" las líneas de código que ha escrito el programador.

El proceso de desarrollo de las aplicaciones tradicionales se puede dividir en tres etapas diferentes: escritura, compilación y comprobación del código. A diferencia de los lenguajes tradicionales, Visual Basic utiliza una aproximación interactiva para el desarrollo, difuminando la distinción entre los tres pasos.

En la mayoría de los lenguajes, si comete un error al escribir el código, el compilador intercepta este error cuando comience a compilar la aplicación. Debe encontrar y corregir el error y comenzar de nuevo con el ciclo de compilación, repitiendo el proceso para cada error encontrado. Visual Basic interpreta el código a medida que lo escribe, interceptando y resaltando la mayoría de los errores de sintaxis en el momento. Es como tener un experto vigilando cómo escribe el código.

Además, para interceptar errores sobre la marcha, Visual Basic también compila parcialmente el código según se escribe. Cuando esté preparado para ejecutar y probar la aplicación, tardará poco tiempo en terminar la compilación. Si el compilador encuentra un error, quedará resaltado en el código. Puede corregir el error y seguir compilando, sin tener que comenzar de nuevo.

A causa de la naturaleza interactiva de Visual Basic, se encontrará ejecutando la aplicación frecuentemente a medida que se desarrolle. De esta forma se pueden probar los efectos del código según se escriba, en lugar de esperar a compilarlo más tarde.

El lenguaje de programación Visual Basic proporciona una facilidad para aprenderlo y desarrollar cualquier tipo de aplicaciones en ambiente windows en comparación con otros lenguajes, como C++, que implican una mayor complejidad de aprendizaje y desarrollo de aplicaciones, aunque

el inconveniente podría ser un menor rendimiento y velocidad. Debido a que su ambiente y desarrollo visual es sencillo se utiliza en una gran variedad de aplicaciones.

2.14 Base de datos

La base de datos es un archivo lógico que puede contener uno o varios archivos físicos, en los cuales se almacena la información que genera un sistema.

Los archivos físicos se pueden visualizar como tablas (renglones y columnas) y se nombran de acuerdo a su contenido. Por ejemplo, si tenemos una tabla llamada CLIENTES, ésta contendrá información referente a los clientes como: razón social, nombre corto, dirección y teléfono, entre otros.

Un buen diseño de la base de datos garantiza la obtención de información significativa que permita al usuario hacer uso de la misma.

2.14.1 Access

Es una aplicación de Microsoft, la cual forma parte del grupo de aplicaciones de Office. Esta es una herramienta para crear bases de datos, no sin antes tener claro los elementos importantes que integran el sistema, de aquí se obtendrán las tablas y los campos que contendrán.

El tipo de información para cada campo puede ser diferente; los más utilizados son texto, número y fecha, pero no son los únicos.

Se diseña la estructura definiendo los campos y su nombre, así como el orden en el que deben aparecer estos campos en la tabla e identificando los campos que serán llaves primarias.

La relación entre las tablas es una parte importante, ya que permite obtener información coherente y significativa; aquí es donde podemos aplicar la normalización, la cual es la aplicación de reglas sobre las relaciones entre tablas para evitar inconsistencias.

En Access existen tres formas para crear una tabla: usando el Asistente para tablas, crear una tabla introduciendo datos y crear en Vista Diseño. Las tres son sencillas; por ejemplo, si usamos la de creación en Vista Diseño se siguen los pasos descritos a continuación:

1) En la ventana de Base de Datos se pulsa "Tablas" y después se pulsa "Crear una tabla en vista Diseño".
2) Se abre una ventana como la de la Fig. 29.

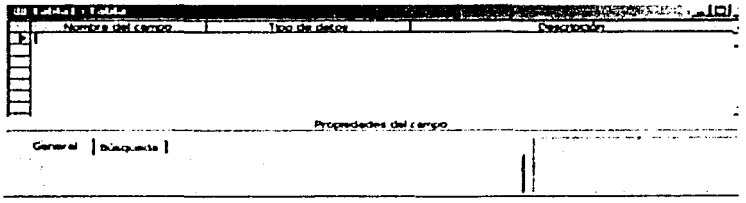


Fig. 29 Creación de tablas en vista de diseño.

Y se puede comenzar a agregar los campos. Ingresando primero "nombre del campo" en la columna "Nombre del Campo ". Pulsando la tecla "enter" se ve que en el "Tipo de Datos" Access puso automáticamente "Texto" y más abajo se abre una lista de las propiedades: Tamaño del campo, Formato, etc.

Y ahora, si se pulsa en la flecha (se llama "Selector de fila") de la primera columna del campo seleccionado con el botón secundario, aquí se encuentran unas cuantas opciones: Cortar, Copiar, Insertar filas, Generar y Propiedades que permiten realizar toda clase de cambios en el diseño de la tabla. Y si se pulsa cualquier celda en la columna de "Tipo de datos" se ve una flecha que permite cambiar el tipo de datos.

Se puede también mover los campos arrastrándolos con el ratón hacia la nueva posición. Para esto es necesario mantener pulsado el Selector de Fila.

Finalmente se guarda la Tabla, pulsando el icono de "Guardar" en la barra de Herramientas o la combinación de teclas "Ctrl+G". Access preguntará por el nombre de la tabla (por defecto pone "Tabla1" pero se puede cambiar por el que se quiera y hará la misma pregunta sobre la clave principal.

Una vez guardada la tabla, se puede abrir en vista de Hoja de Datos y comenzar a introducir datos.

Los campos tienen propiedades como:

Formato. Determina cómo se muestran los datos (por ejemplo moneda o fecha). Cuando un valor puede tener formato se selecciona desde la lista desplegable.

Lugares decimales. Aquí se selecciona el número de decimales que Access muestra en los campos de tipo Moneda o Numérico.

Máscara de entrada. Esta propiedad sirve para introducir datos válidos en un campo. Por defecto no hay ninguna máscara de entrada, pero en ocasiones puede ser útil (sobre todo para las fechas).

Título. Es una propiedad opcional, sirve cuando se requiere que el nombre de un campo en vista de Hoja de datos sea distinto del nombre del campo en la Vista Diseño.

Valor predeterminado. Puede ser útil si se tiene siempre el mismo valor en el campo (o casi siempre). Si por ejemplo, en una tabla de autores el apellido que más se repita es "García" se puede definir como el valor predeterminado. Para eso, simplemente se escribe el valor predeterminado en la propiedad del campo o se pulsa el botón con "..." para generar expresiones complejas.

Regla de Validación y Texto de Validación. Son propiedades avanzadas que permiten limitar los valores que se introducen en un campo (Regla) y definir el mensaje de error cuando se introduce un valor prohibido por la regla (Texto).

Requerido. Por defecto está puesto "No", pero si se selecciona Access no permite dejar un campo en blanco.

Permitir longitud cero. Permitir o no las cadenas de longitud cero. Por defecto es "No".

Indexado. El indexado permite acelerar los procesos de búsqueda y ordenación, pero hace aumentar el tamaño de la base de datos. Por defecto sólo la clave principal (si se tiene) aparece indexada, para el resto de los campos es opcional. Las opciones de esta propiedad son:

- Sí (con duplicados) el campo se indexará, pero permitiría tener valores repetidos (duplicados) en más de un registro.
- Sí (sin duplicados) el campo se indexará, pero no admitiría valores duplicados.
- No, el campo no se indexará.

La relación entre las tablas puede ser: uno a varios, varios a varios o uno a uno.

Para crear una relación se llevan a cabo los siguientes pasos:

- Se mantiene abierta la ventana de relaciones, pero cerradas todas las tablas.

- Para añadir tablas a la ventana se selecciona "Mostrar tabla" en el menú "Relaciones" o se pulsa el botón "Mostrar tabla" en la Barra de herramientas. Se selecciona la tabla que se quiere agregar y se pulsa "Agregar". Se pulsa "Cerrar" cuando se termina de agregar tablas.

- En la ventana de relaciones se selecciona el campo que se quiere relacionar y se arrastra soltando en el campo de la otra tabla, Fig. 30.

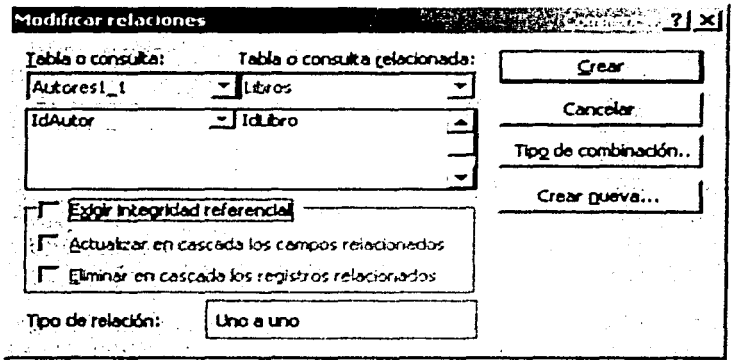


Fig. 30 Ventana para crear las relaciones entre las tablas.

En resumen se puede catalogar esta aplicación como sencilla, amigable y de bajo costo.

2.15 Selección y uso de herramientas para el desarrollo de SECUMATIC.

Una vez definida toda la teoría de ingeniería de software se procede a la selección de los métodos, procedimientos y herramientas que se utilizarán para el desarrollo del sistema que se pretende desarrollar, basado en la problemática establecida.

Analizando las ventajas de los paradigmas de la ingeniería de software, se decidió trabajar bajo el modelo en espiral, ya que contempla las características del ciclo de vida clásico (análisis, diseño, codificación, pruebas y mantenimiento), las de prototipo y añade la etapa de análisis de riesgo, que sirve para evaluar los problemas que pueden surgir e ir proponiendo algunas alternativas de solución. Este modelo es el que más se acopla a las características del sistema propuesto, porque nos permitirá hacer un análisis inicial para la primera vuelta y ofrecer al usuario un prototipo inmediato que se irá perfeccionando en base a nuevos requerimientos, además de ir evaluando los riesgos que pudieran surgir.

La herramienta de programación que se eligió utilizar para el desarrollo de SECUMATIC es Visual Basic 6, debido a que es un lenguaje gráfico que nos permite crear objetos, cambiar y establecer sus propiedades; además de utilizar un procedimiento de evento para estructurar el código. Con esto se crea un ambiente de desarrollo donde el trabajo con tales objetos y eventos llega a ser un proceso directo y bien estructurado. También interpreta el código a medida que lo escribe, interceptando y resaltando la mayoría de los errores de sintaxis en el momento y compilando parcialmente el código, según se va escribiendo; esto, reduce el tiempo de ejecución para probar y terminar la compilación.

La programación se llevará a cabo tomando como referencia los aspectos de la programación estructurada, para tener las ventajas que ésta nos ofrece y tener un mejor control sobre el desarrollo.

Como también pretendemos tener una base de datos de los alumnos, profesores y calificaciones, se selecciono Microsoft Access como herramienta de base de datos que nos permite crear nuestras tablas y campos de la base de datos de una forma rápida y sencilla, a bajo costo y que cubre la necesidad de una herramienta de base de datos.

Cuando se desarrolla un software de tipo multimedia, se requiere integrar medios que nos permitan implementar sonido, animación, video y edición de texto, por lo que es necesario seleccionar el software adecuado a nuestro proyecto, tomando en cuenta las ventajas que cada uno de éstos nos proporcionan.

El software que se eligió para animación fue Flash 5, porque tiene un alto grado de interactividad al permitir importar sonido de cualquier formato y darle efectos; así como mapas de bits o creación de imágenes vectoriales, con las cuales se pueden lograr animaciones de gran calidad.

Es necesario también crear textos animados. Para este caso, se seleccionó Xara 3D, por ser un software sencillo y que permite importar animaciones que manejan texto a cualquier otro formato.

Se requiere también utilizar un software sencillo que permita crear y animar imágenes vectoriales geométricas. Es por ello que se eligió el 3D Canvas, ya que nos permite hacer animaciones tridimensionales con imágenes predefinidas del mismo software, que nos ayudarán a crear las animaciones para las figuras geométricas del software a desarrollar.

Para la edición de textos y creación de las páginas html, se optó por usar el Microsoft Word, ya que en él se pueden integrar textos con diferentes fuentes, imágenes en cualquier formato, animaciones, videos y sonidos, haciendo el ambiente de trabajo más flexible.

CAPÍTULO 3

DESARROLLO DEL SOFTWARE EDUCATIVO SECUMATIC

3.1 Justificación inicial del sistema

Debido a los cambios y procesos de modernización que se viven en México, ha sido necesario establecer como obligatoria la educación secundaria básica, con el fin de formar nuevas generaciones con mejores niveles de preparación que se vean reflejados en el ámbito laboral. Sin embargo, es difícil alcanzar estos objetivos ya que existen diversos factores que impiden lograr una educación de calidad.

Las instituciones educativas se enfrentan cada día a diferentes problemas tanto de infraestructura como de organización educativa. Una de las materias en la cual los alumnos presentan mayor problema en el aprendizaje es en el área de matemáticas, ya que la enseñanza está basada principalmente en el libro de texto, por lo que los alumnos muestran poco interés por la materia, concibiéndola como aburrida y compleja por los conceptos tan abstractos que se estudian.

En las visitas realizadas a tres secundarias de la SEP dentro del Distrito Federal, y en varias entrevistas con profesores y alumnos en la materia de matemáticas de tercer grado, nos comentaron que sí cuentan con software de matemáticas, el cual fue proporcionado por el programa Red Escolar. Sin embargo, este software no cubre el contenido temático de la asignatura, por lo que nos sugirieron el desarrollo de un software interactivo, que cubra los temas más representativos de la materia y al mismo tiempo que sirva de apoyo didáctico, ya que es en este nivel donde los alumnos adquieren los conocimientos básicos que más adelante sustentarán los del nivel medio superior. Por lo tanto, es importante reforzar el aprendizaje y despertar el interés, mediante el uso de software educativo.

Bajo esta perspectiva se plantea el desarrollo de un software educativo que contemple los temas representativos del curso de matemáticas tres de la educación secundaria de la SEP, como son aritmética, álgebra, geometría y trigonometría, que estará dirigido principalmente a jóvenes de 12 a 15 años de edad.

Para el desarrollo de dicho software hemos considerado las cuestiones psicopedagógicas, apoyándonos principalmente en las teorías de Piaget, Vigotsky y Ausubel, por considerar las características evolutivas de la conducta en el adolescente, las condiciones que deben existir para la adquisición de sus propios conocimientos y la interacción con el medio.

Estos conceptos deben estar presentes en el proceso de enseñanza aprendizaje, además de considerar las opiniones directas de los profesores que imparten la materia, los cuales manifiestan que los alumnos tienen una mayor retención de conocimientos al asociar un concepto a una imagen o icono y más aún cuando es tridimensional; esto es de gran utilidad cuando los conceptos abstractos de las matemáticas no pueden ser representados en el pizarrón, pero sí mediante el uso de animaciones o juegos en un software.

3.2 Metodología para el desarrollo del software educativo SECUMATIC.

De acuerdo al planteamiento citado, que consiste en la necesidad de crear un software educativo que cumpla con el plan de estudios que tiene la SEP para el tercer año de secundaria "SECUMATIC", se establece una estrategia de solución basada en el modelo o paradigma de la Ingeniería de software, tomando en cuenta que dicho modelo se acople a las necesidades y naturaleza de nuestro sistema. Por lo tanto, el modelo o paradigma que se eligió para desarrollar el software SECUMATIC, es el modelo en espiral.

3.3 Modelo en espiral

"Este modelo es híbrido debido a que contempla las características del ciclo de vida clásico modificado y la creación de prototipos; tiene la forma de una espiral en la cual se puede establecer una relación entre cada giro y una fase del proceso de desarrollo"[15] Cada vuelta de la espiral define

[15] Pressman, Roger S. Ingeniería de Software Un enfoque práctico. McGraw-Hill;5ª Edición 2002; México; pág. 78

cuatro procesos principales representados en los cuatro cuadrantes de la espiral, como se muestra en la Fig. 31.

El primer cuadrante corresponde a la etapa de Planificación, es aquí donde se determinan los objetivos, se hace la recolección de requisitos y se plantean algunas restricciones.

En el segundo cuadrante se realiza el proceso de Análisis de Riesgo, donde se establecen alternativas de solución de los riesgos.

El tercer cuadrante abarca la etapa de Ingeniería donde se lleva a cabo el análisis del sistema, el diseño, el desarrollo o codificación y las pruebas.

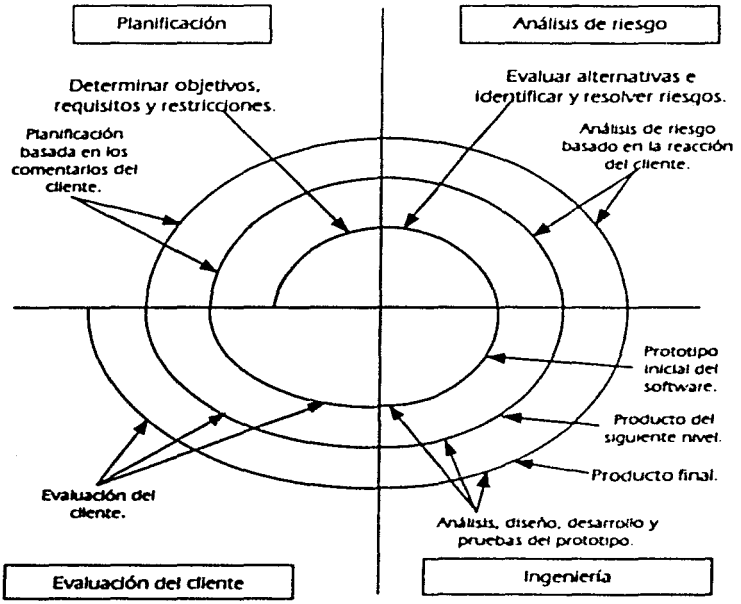


Fig. 31 Modelo en espiral de Pressman.

Dentro del análisis del sistema, se determinan las funciones que deben cumplir los elementos del sistema, tanto para el alumno como para los profesores.

En el diseño del sistema, se consideran la población objetivo, las expectativas del sistema, el área de contenido, las limitaciones y recursos, el diseño computacional, diseño de la interfaz gráfica, diseño de comunicación y el diseño educativo.

Durante el desarrollo del sistema, se realiza la codificación y se hace una descripción de los elementos que contiene cada pantalla.

En la etapa de pruebas se realizarán actividades para verificar que la codificación no presenta errores y los procesos funcionan correctamente, además de cubrir los requisitos solicitados por el cliente. Para las vueltas sucesivas en la espiral, se consideran las especificaciones iniciales, añadiéndole los nuevos requerimientos del cliente.

Finalmente, en el cuarto cuadrante se hace la evaluación del producto obtenido en cada vuelta, el cual es evaluado por el cliente para que dé su aprobación o solicite modificaciones, basándose en sus necesidades que, son tomadas en cuenta para la siguiente fase de planificación.

3.3.1 Primera vuelta

3.3.1.1 Planificacion

El objetivo de esta etapa es determinar el contexto en el cual se va a crear la aplicación de SECUMATIC y derivar de ahí los requerimientos que deberá atender la solución del sistema.

Requisitos del cliente

- El software educativo que sirva como apoyo didáctico para los alumnos.

- Que cubra los temas de matemáticas del tercer año de secundaria en base al plan de estudios de la SEP.
- Calidad en los contenidos temáticos y que la información se presente de manera interactiva.
- Que contenga muchos ejemplos resueltos, explicados en forma detallada.
- Utilizar juegos didácticos para la mejor comprensión de conceptos.
- Diseño atractivo en las pantallas, menús, iconos, botones, barras de navegación, fondos y texto legible.
- Integración de música, videos, animaciones e imágenes asociadas a cada tema en particular.
- Facilidad de uso e instalación.

Objetivos

- Desarrollar el Módulo correspondiente a SECUMATIC que comprenda el tema de aritmética, basado en el plan de estudios de la SEP del tercer año de secundaria.
- Diseñar el contenido del tema de aritmética, tomando en cuenta las estrategias didácticas, psicológicas y pedagógicas que sirvan para motivar y reforzar los conocimientos adquiridos de los estudiantes.
- El software desarrollado deberá ser amigable, atractivo y fácil de utilizar.

Restricciones

- Integrar información exclusivamente del tema de aritmética.
- Utilizar ejemplos representativos en cada subtema.
- Diseñar las pantallas de acuerdo a las preferencias visuales de los alumnos.
- Presentar la información con un vocabulario sencillo en los contenidos.

3.3.1.2 Análisis de riesgo

El objetivo del análisis de riesgo es llevar un seguimiento del desarrollo del producto, adaptar el sistema a nuevos requerimientos en el tiempo, corregir errores no detectados en las etapas anteriores y distribuir correctamente las horas por actividad.

Opciones

- Utilizar fondos, imágenes, animaciones, videos y sonidos que ayuden a crear un ambiente de diseño atractivo, que llame la atención de los alumnos en el tema de aritmética.
- Diseñar ejemplos interactivos en los cuales se le permita al alumno aplicar los conceptos adquiridos de una manera más divertida y fácil.
- Análisis y estructuración de los objetivos de acuerdo al temario vigente.
- Realizar visitas de campo a las secundarias de la SEP, para determinar las características de equipo en el que se implantará el sistema.

Riesgos

- Existe la posibilidad de que a los estudiantes no les parezca atractivo el diseño de las pantallas, botones, animaciones e imágenes.
- El sonido seleccionado puede no ser el de mayor preferencia.
- El grado de complejidad de los ejemplos propuestos no sea el adecuado.
- Que no se cumpla con el propósito de enseñanza-aprendizaje.
- Que el contenido temático no cubra los objetivos de la materia según los criterios de cada profesor.

- Dificultad en la utilización del tutorial para alumnos con poca o nula experiencia en la computadora.

3.3.1.3 Ingeniería en el desarrollo del sistema

La parte correspondiente a la primera vuelta de la espiral, para el desarrollo del sistema, se dividió en cuatro etapas.

- Análisis del sistema.
- Diseño del sistema.
- Desarrollo e implementación del sistema.
- Pruebas.

3.3.1.3.1 Análisis del sistema

En esta etapa analizamos las funciones que debe realizar cada elemento que formará parte del sistema.

Las funciones que el sistema debe cumplir con respecto al alumno son las siguientes:

- Ejercitación, se logra mediante la exposición de ejemplos aplicados a cada tema en particular.
- Transmisión, muestra temas completos y ejercicios como base para reforzar el aprendizaje.
- Registro, guarda las entradas y salidas del alumno por sesión, nombre, grupo y evaluación por cada examen.
- Interfaz, permite utilizar las opciones de selección por medio del ratón, navegar por todos los temas a través de botones de control y ligas.

Las funciones que el sistema debe ofrecer en apoyo a la labor del profesor son las siguientes:

Evaluación de Alumnos. El profesor tiene la facilidad de evaluar el trabajo de los temas expuestos mediante un módulo especial de exámenes, en el cual puede diseñar, modificar, actualizar, insertar o borrar su propio examen.

Control de Alumnos. Permite dar de alta a los alumnos de cada grupo, siempre y cuando tengan asignado a un profesor.

Control de Profesores. Permite dar de alta a los profesores y la asignación de sus respectivos grupos.

Reportes. Genera listas de alumnos, profesores, calificaciones, sesiones y relación de grupos por profesor.

Alcances del Sistema

El software abarca en la primera vuelta, el desarrollo del contenido del tema de aritmética, basándonos en los objetivos propuestos en el plan de estudios de la educación secundaria de la SEP.

Análisis y especificación estructurada.

Diagrama de flujo de datos de SECUMATIC

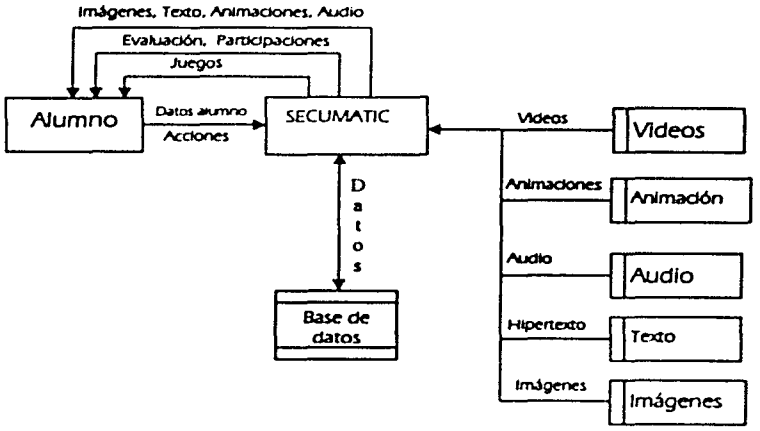


Tabla de decisión del menú principal de SECUMATIC

A continuación se muestra la tabla de decisión del menú principal de SECUMATIC, siguiendo el diagrama de control de flujo. En primer lugar se solicita el nombre de usuario y contraseña, información que se verifica en la base de datos. Se dan dos condiciones, C1 si existe usuario y C2 en caso de que no exista, bajo las siguientes reglas:

Regla R1: si C2 es verdadero, se incrementa contador (acción A1).
Regla R2: si C2 es verdadero y contador es menor a 3 (condición C3) se da otra oportunidad solicitando el registro del alumno (acción A2).

Regla R3: si la condición C2 se cumple y el contador es igual a 3 (condición C4) se termina la aplicación (acción A3).

Regla R4: si C1 es verdadero, muestra menú principal de temas (acción A4).
Regla R5: si C1 es verdadero y caso = 1, muestra tema Aritmética (A5)
Regla R6: si C1 es verdadero y caso = 2, muestra tema Álgebra (A6).
Regla R7: si C1 es verdadero y caso = 3, muestra tema Geometría (A7).
Regla R8: si C1 es verdadero y caso = 4, muestra tema Trigonometría (A8).
Regla R9: si C1 es verdadero y se cumple C9, se termina la aplicación (A3).

V = Verdadero, F = Falso

	Reglas								
Condiciones	R1	R2	R3	R4	R5	R6	R7	R8	R9
C1 Existe usuario	F	F	F	V	V	V	V	V	V
C2 no existe usuario	V	V	V	F	F	F	F	F	F
C3. Contador < 3	-	V	F	-	-	-	-	-	-
C4. Contador = 3	-	F	V	-	-	-	-	-	-
C5. Caso = 1	-	-	-	-	V	F	F	F	F
C6. Caso = 2	-	-	-	-	F	V	F	F	F
C7. Caso = 3	-	-	-	-	F	F	V	F	F
C8. Caso = 4	-	-	-	-	F	F	F	V	F
C9. Salir	-	-	-	-	F	F	F	F	V
Acciones									
A1. Incrementa contador	V								
A2. Registro alumno		V							
A3. Terminar aplicación			V						V
A4. Muestra Menú Principal				V					
A5. Muestra tema Aritmética					V				
A6. Muestra tema Álgebra						V			
A7. Muestra tema Geometría							V		
A8. Muestra tema Trigonometría								V	

3.3.1.3.2 Diseño del sistema

Entorno para el desarrollo del sistema

Una vez realizada la etapa de análisis, se tienen establecidas las características que determinan la naturaleza del sistema a desarrollar.

Para el desarrollo de SECUMATIC se considera en primera instancia, que va dirigido a una población objetivo, con un área de contenido específico, así como cubrir una necesidad educativa que fomente la motivación y el reforzamiento de conocimientos hacia esta población. También se consideran los recursos y las limitaciones que tendrán los usuarios y las características del equipo en donde será instalado el sistema.

Población objetivo

SECUMATIC es un material que está dirigido para jóvenes de ambos sexos, que cursan el tercer año de secundaria de la SEP, específicamente entre 12 y 15 años de edad.

Expectativas del sistema

Los alumnos del tercer grado de secundaria en la materia de matemáticas, según lo expuesto por los profesores que la imparten, tienen dificultades en el aprendizaje de los conceptos, por tratarse de temas tan abstractos y que muchas veces no pueden ser representados en el pizarrón, frenando de esta manera el interés de los alumnos.

Por tal motivo, SECUMATIC intenta despertar el interés por el estudio de las matemáticas tomando en cuenta los aspectos pedagógicos, psicológicos y de conocimiento del alumno población, además de hacer más atractivo el proceso de la enseñanza de las matemáticas con el uso de imágenes, videos y ejemplos representativos de cada tema, en un ambiente gráfico en el cual puedan visualizar y practicar.

Conocimientos previos que debe tener el usuario

Los alumnos de secundaria en los primeros dos años de formación conocen temas previos de matemáticas como son: La historia de los números, Sistemas de numeración, Fracciones, Áreas y Perímetros, El lenguaje algebraico, Introducción al Plano cartesiano. Por lo tanto, el alumno ya tiene los conocimientos básicos para entender el contenido de los nuevos temas.

Actualmente en México, la mayoría las secundarias de la SEP cuentan con un laboratorio de computadoras proporcionadas por el plan Red Escolar en las cuales, desde el primer año de secundaria los alumnos tienen contacto con ellas y conocen software multimedia que les sirve de apoyo en los temas de historia, civismo, matemáticas. Esto representa una ventaja para el alumno al momento de conocer SECUMATIC, al estar familiarizado tanto con las computadoras, como con el software de tipo multimedia.

Área de contenido

SECUMATIC cubre los cuatro temas más representativos de matemáticas del tercer grado de Secundaria del plan de estudios de la SEP.

Aritmética (la raíz cuadrada y errores de aproximación), álgebra (plano cartesiano, operaciones con expresiones algebraicas, ecuaciones y sistemas de ecuaciones lineales, productos notables y factorización, ecuaciones de segundo grado y cuadráticas), geometría (triángulos y cuadriláteros, círculo, semejanza, teorema de pitágoras, sólidos), trigonometría (razones trigonométricas, ángulos específicos, resolución de triángulos).

Dificultades en el área de contenido

La enseñanza de las matemáticas resulta ser muy abstracta, mostrar contenidos de este tipo a jóvenes de esta edad es muy complicado, los alumnos casi siempre conciben los temas como muy complejos y poco atractivos. Normalmente el profesor es el que proporciona toda la información y de esta manera el aprendizaje se torna un tanto aburrido, provocando el desinterés por la materia.

Necesidad educativa

Brindar la oportunidad para que el adolescente interactué con la computadora y al mismo tiempo ofrecerle una herramienta didáctica e interactiva y que le sirva para reforzar sus conocimientos en el área específicamente de matemáticas.

Limitaciones y recursos

El sistema está diseñado para que se trabaje en el aula dentro de cada secundaria, en grupo colectivo bajo la coordinación de un profesor. El alumno puede iniciar una sesión de manera individual cuando solamente requiera consultar los temas sin ser evaluado, sin embargo, debe existir una sesión simultánea, tanto del profesor como de los alumnos para poder enviar las participaciones.

Diseño del sistema

En el diseño de SECUMATIC, se consideran tres aspectos fundamentales, diseño educativo, comunicativo y computacional, que nos ayudarán a obtener un software de calidad que ofrezca la posibilidad de resolver una necesidad educativa.

Diseño educativo

Pretendemos que SECUMATIC sirva como una herramienta de apoyo didáctico, en el estudio de las matemáticas, para los alumnos del tercer grado de secundaria.

El ambiente del sistema se centra principalmente en el aprendizaje y en la enseñanza de las matemáticas, de una forma sencilla y atractiva para despertar el interés y la motivación de los estudiantes. La versatilidad e interactividad que representa el uso de la computadora y la posibilidad de jugar con ella, hace una experiencia fascinante.

Diseño de comunicación

Como medio de comunicación entre el alumno y el software, se utiliza el ratón, con el cual se pueden seleccionar los botones del menú principal, ligas a los subtemas y botones de avance y retroceso.

El menú principal esta diseñado con elementos propios de las matemáticas, como ecuaciones, figuras geométricas, funciones trigonométricas, todas

contenidas en una animación. Además, incluye botones de fácil acceso a los temas.

Los textos se componen con tipografía de tamaño adecuado, siendo fácil identificar cuando existe una liga a una imagen o a otro texto complementario, ya que se encuentra de un color diferente a todo el texto.

Los gráficos utilizados son de gran calidad, con una composición armónica de colores, que resultan muy llamativos y están asociadas con cada contenido particular.

Los sonidos se encuentra relacionados con los textos, videos o animaciones, en cada pantalla y pueden ser activados o desactivados por el usuario dando un clic sobre ellos.

Las animaciones se incluyen en el menú principal de cada subtema y representan visualmente, de manera general, el contenido de los temas.

Los video incluidos están relacionados al tema específico y pueden ser activados por el alumno, si así lo prefiere.

Diseño computacional del software

Nos hemos apoyado en el uso de herramientas multimedia para lograr en lo posible actividades interactivas.

Para navegar dentro del software se utilizó una arquitectura basada en menús y un motor controlado por datos.

El menú principal contiene cuatro botones de control, donde cada botón nos da el acceso a un tema respectivo (aritmética, álgebra, geometría y trigonometría).

Para mostrar la información se utilizó una plantilla base, la cual despliega páginas web, un botón de activación de sonido, de calculadora, de juegos y de salida. Para cada tema se modificó la posición de cada uno de estos elementos.

Los contenidos de cada tema se despliegan a través de páginas de hipertexto y en la página principal se muestra un subíndice al cual podemos acceder mediante ligas.

Diseño de la interfaz gráfica

La interfaz gráfica nos define la manera en que el usuario puede interactuar con el sistema, por lo tanto se busca que tenga simplicidad, ya que entre más simple es el diseño es más fácil usar la interfaz para el usuario.

La interfaz gráfica de SECUMATIC se compone de botones, ligas de texto e imágenes sensibles, por lo que resulta muy sencillo hacer uso del sistema.

Es importante el aspecto de la interacción del alumno con el sistema ya que le brinda la seguridad de controlar el inicio o fin, el avance o retroceso entre cada uno de los temas.

SECUMATIC es un sistema que utiliza una variedad de imágenes, sonidos, animaciones y no tiene problemas para ejecutar este tipo de aplicaciones, por lo cual el tiempo de respuesta de estos procedimientos es mínimo.

Diagrama de control de flujo
del menú principal de SECUMATIC
primera vuelta.

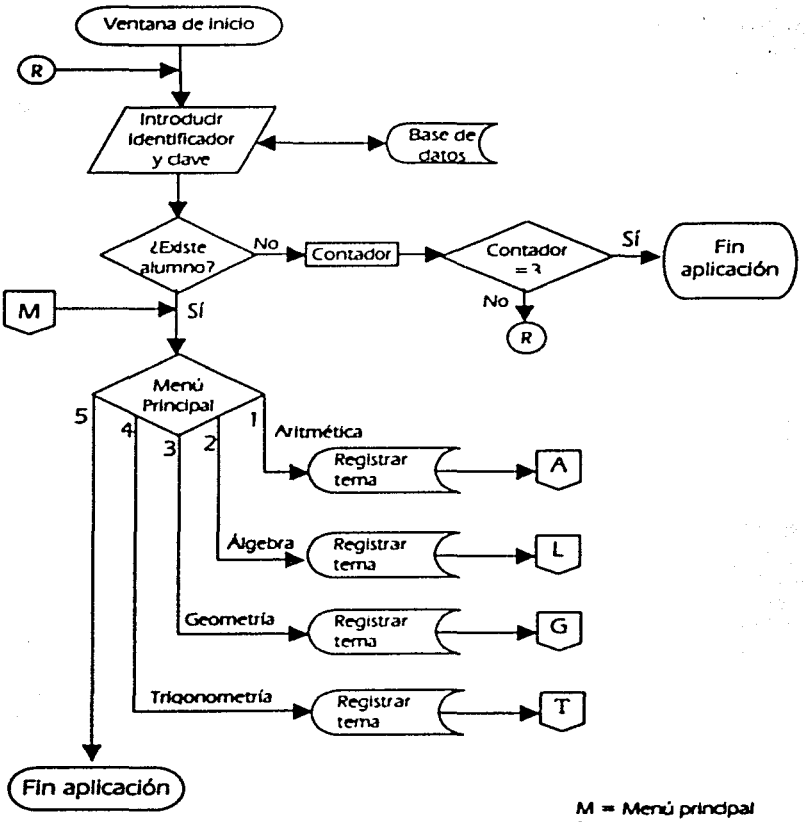
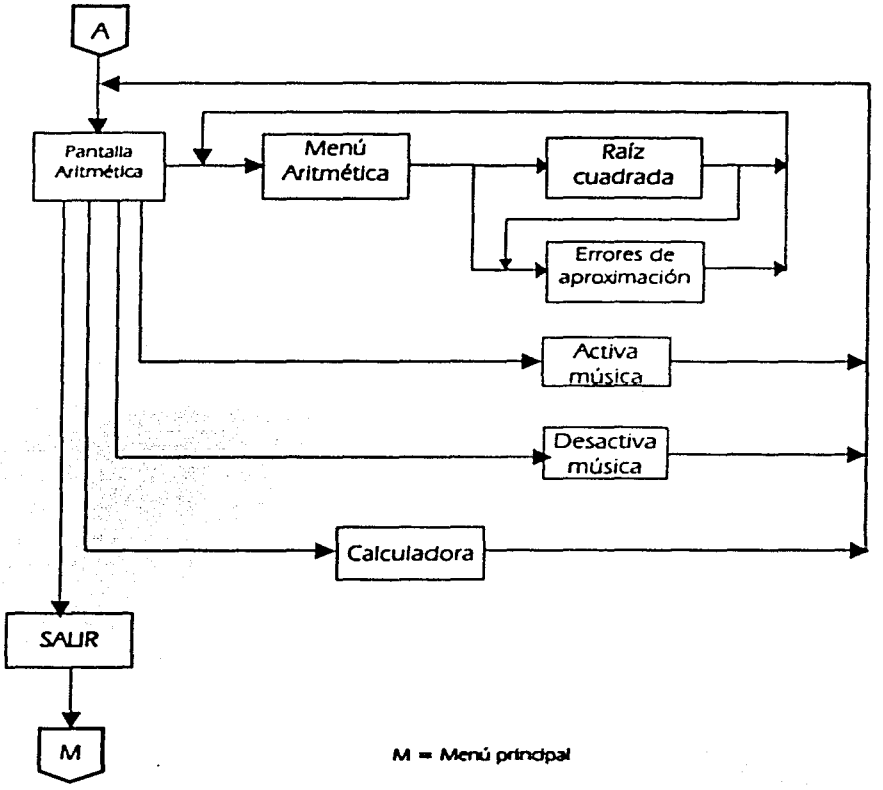


Diagrama de control de flujo del tema Aritmética.



3.3.1.3.3. Desarrollo del sistema

Para el desarrollo del sistema haremos uso principalmente de Visual Basic 6, como herramienta de programación y de Access para generar la base de datos.

Para la creación de imágenes, animaciones y videos, se hace uso de Flash 5, Xara 3D, Photoshop 5 y Corel draw.

Desarrollo del menú principal

a) Descripción

Es la pantalla principal donde se encuentran el menú de los temas a seleccionar: Aritmética, Álgebra, Geometría y Trigonometría. Muestra una animación significativa con el tema de las matemáticas, localizada en el centro de la pantalla, como se muestra en la Fig. 32.

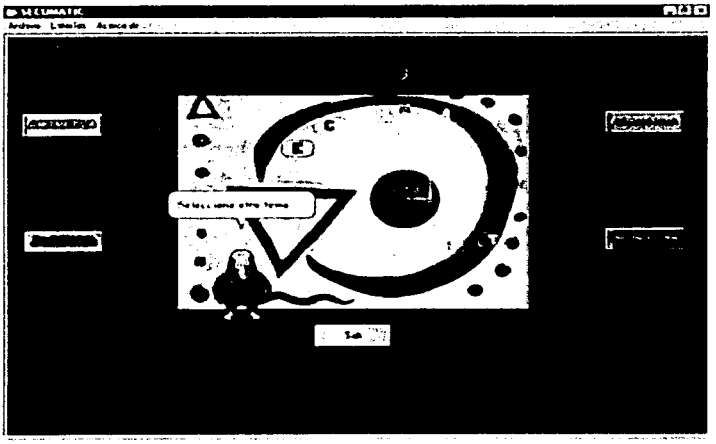


Fig. 32 Menú principal de SECUMATIC.

b) Pseudocódigo

```
Inicia el Menú Principal
   Mientras Función_tema exista
        Selecciona Función _ tema
        En caso de que Función_tema sea:
             1: Función Aritmética
             2: Función Álgebra
             3: Función Geometría
             4: Función Trigonometría
        Fin caso
   Fin Mientras
Terminar
```

c) Código

Al pulsar el botón del tema específico se ejecuta un código semejante al que sigue y mandará llamar la función obtieneTema; para este caso específico es número de tema, 1 = Aritmética.

```
Private Sub Command1_Click()
    ' Declaración de variables locales
      Dim sConsulta As String

    ' El agente toma nota del tema
      Mago.ActividadAgente "Write"
      Mago.ActividadAgente "WriteContinued"
      Mago.ActividadAgente "WriteReturn"
      Mago.OcultaAgente
      Unload Me
      obtieneTema 1
End Sub
```

Enseguida se muestra el código de la función obtiene tema.

Esta función registra en la base de datos el tema accesado y manda a llamar la forma del tema escogido.

```
Public Function obtieneTema(ByVal iTema As Integer)
' Declaración de variables locales
  Dim sInserta As String

' Inicializa la cadena para insertar el registro
        sInserta = "INSERT INTO sesion "
  sInserta = sInserta & "(alum_id, ses_fecha, ses_hent, ses_hsal, tem_id) "
  sInserta = sInserta & "VALUES ( "
  sInserta = sInserta & "'" & Usuario & "', "
  sInserta = sInserta & "'" & Format(Date, "dd/mm/yyyy") & "', "
  sInserta = sInserta & "'" & Format(Time, "hh:mm:ss") & "', "
  sInserta = sInserta & "'" & Format(Time, "hh:mm:ss") & "', "
  sInserta = sInserta & "'" & iTema & "')"

' Ejecuta la inserción
  oConexion.SoloEjecutar sInserta

' Muestra el tema seleccionado
  Select Case iTema
       Case Is = 1
          frmAritmetica.Show
       Case Is = 2
          frmAlgebra.Show
       Case Is = 3
          FrmGeometria.Show
       Case Is = 4
          frmTrigonometria.Show
  End Select
End Function
```

d) Elementos

El menú principal cuenta con los siguientes elementos:

1 Animación realizada en Flash
4 Imágenes con extensión jpg
5 botones de comando

Desarrollo del tema Aritmética

Este es el primer tema desarrollado, el cual se toma como prototipo en la primera vuelta del modelo en espiral.

a) Descripción

El módulo de Aritmética está compuesto por una página principal, que contiene el índice de los subtemas de Raíz Cuadrada y Errores de aproximación. Dicha pantalla contiene un botón de comando para mostrar la calculadora interna de Windows, dos botones para activar o desactivar la música de fondo, un botón para accesar a un juego y otro para salir del tema, como se muestra en la Fig. 33.

Fig. No. 33 Pantalla inicial del tema de Aritmética.

b) Pseudocódigo

Inicia Menú del tema
Muestra Agente animado
Mientras Función exista
    Ejecuta Función
    En caso de que función sea
        1: Función Indice tema
        2: Función Calculadora
    Fin caso
Fin Mientras
salida
Termina Menú del tema

c) Código

```
' Declaración de variables locales del procedimiento
Private oBandera As Integer

Private Sub Form_Activate()
' Inicializa la forma
  Me.Top = 0
  Me.Left = 0
' Validando la cadena de la ruta
  sCaracter = Mid(sRutaHTML, Len(sRutaHTML), Len(sRutaHTML))

' Muestra el documento html
  If sCaracter = "\" Then
     WebBrowser1.Navigate2 sRutaHTML &
"Aritmetica\Indice\INDICEARIT.htm"
  Else
     WebBrowser1.Navigate2 sRutaHTML &
"\Aritmetica\Indice\INDICEARIT.htm"
  End If

   ' Muestra el agente
  If oBandera = 0 Then
      ' Inicializa la bandera
        oBandera = oBandera + 1
      ' Asigna las propiedades al agente
        Mago.MuestraAgente
        Mago.HablaAgente "Estudiemos aritmética"
        Mago.ActividadAgente "Read"
        Mago.OcultaAgente
  End If
End Sub
```

```
Private Sub Command3_Click()
' Declaración de variables locales
  Dim lHandler As Long
' Activa la calculadora
  Shell "calc.exe", vbNormalFocus
' Obtener el handler de la calculadora
  lHandler = FindWindow(0&, "Calculadora")
' Establecer la calculadora en primer término
  SetWindowPos lHandler, conHwndTopmost, 100, 100, 271, 279,
conSwpShowWindow
End Sub

Private Sub Form_Unload(Cancel As Integer)
' Termina la forma
  Call Command1_Click
End Sub

Private Sub Command1_Click()
' Descarga la forma
  Unload Me
' Destruye la forma
  Set frmAritmetica = Nothing
' Muestra la forma padre
  frmMdiInicio.Show
End Sub
```

d) Elementos totales en el módulo de Aritmética

12 animaciones
28 imágenes
4 fondos de papel tapiz
12 botones
1 agente animado
1 archivo de sonido
1 juego crucigrama

3.3.1.3.4 Pruebas

"La prueba es un grupo de actividades que se pueden planear por adelantado y llevar a cabo sistemáticamente"[15].

Una estrategia para llevar a cabo la prueba es considerar pruebas de unidad que se refieren a la revisión de la codificación, pruebas de integración, en las cuales debe revisarse cómo va el diseño del sistema y pruebas de alto nivel para cubrir los requisitos.

Para el caso de SECUMATIC, al concluir el prototipo de la primera vuelta se realizaron las siguientes pruebas:

Se revisó el código programado en Visual Basic, ejecutando la aplicación para determinar si no existían errores de sintaxis, continuidad de procesos, ligas, validaciones y que las llamadas a función no fallaran.

En la parte de diseño, el sistema debía cubrir los aspectos multimedia planteados en la justificación del sistema para el tema seleccionado (Aritmética).

En cuanto a los requisitos, se verificaron las propuestas iniciales comentadas por alumnos y profesores para determinar si el sistema está avanzando de una manera apropiada o no.

3.3.1.4 Evaluación de la primera vuelta

Para realizar la evaluación del software desarrollado, se utilizó un disco compacto para cada equipo. El módulo de SECUMATIC Aritmética, en la primera vuelta, fue evaluado en tres secundarias de la SEP.

[15] Pressman, Roger S; Ingeniería de Software. Un enfoque Práctico; Mc Graw Hill;5a. Edición 2002,México; Pág. 662.

- Escuela Secundaria "Nochcalco", turno vespertino, que se ubica en la Calle Simón Bolivar No. 180, Delegación Milpa Alta, D.F.

- Escuela Secundaria "Juventino Rosas Cárdenas", turno vespertino, que se ubica en la Calle Tarascos s/n, Col. Tlalcoligia, Delegación Tlalpan, D.F.

- Escuela Secundaria No. 39, turno matutino, que se ubica en la Calle Moneda s/n, Delegación Tlalpan, D.F.

En las tres secundarias antes mencionadas, la presentación se realizó en el aula de computación, en el cual se encuentran instaladas las computadoras conectadas en red, proporcionadas por el sistema de Red Escolar, con las siguientes características:

- Equipos IBM Pentium 2 Celeron
- Sistema operativo Windows 98
- 64 MB de memoria RAM
- 8 GB en disco duro
- Equipo multimedia (CD, bocinas)

Acudieron para la evaluación del módulo en la secundaria Nochcalco; 3 profesores y 20 alumnos de los grupos tercero A y B; para la escuela Juventino Rosas Cárdenas; la directora, 5 profesores y 10 alumnos del grupo tercero B, C y D; en la escuela No. 39; la subdirectora, 3 profesores y 5 alumnos del grupo tercero A de la materia de matemáticas.

Después de una breve explicación y demostración de las principales funciones a los profesores que imparten la materia de matemáticas, la evaluación se realizó partiendo de la pantalla de registro de alumno, donde cada uno pudo ingresar su nombre y clave de acceso, quedando su sesión registrada en la base de datos.

Se procedió a consultar el tema de Aritmética, explicando los botones de control y la manera de accesar a los dos temas principales, Raíz cuadrada y Errores de aproximación.

En general, los profesores mostraron gran interés por los contenidos y a los alumnos les pareció muy atractivo el diseño de las pantallas, así como algunas animaciones.

Recibimos observaciones por parte de los profesores: es necesario agregar más ejemplos interactivos, con el fin de que los alumnos puedan ejercitar y reforzar los conceptos básicos de los dos subtemas presentados.

Los profesores comentaron también que les pareció muy atractivo el diseño del prototipo presentado y como nuevo requerimiento pidieron la integración de los demás temas del plan de estudios de tercer grado, al proyecto.

En algunos casos, SECUMATIC no ejecutó algunos procesos como se esperaba. Tal es el caso del uso simultáneo de videos e imágenes, ya que no se tomaron en cuenta algunas especificaciones técnicas de la red en la que se implantó el sistema.

3.3.2 Segunda vuelta

3.3.2.1 Planificación

En esta etapa durante la segunda vuelta, se determinó que el objetivo es cubrir el desarrollo de los siguientes tres temas: álgebra, geometría y trigonometría, correspondientes al plan de estudios de la SEP de matemáticas del tercer año de secundaria.

Requisitos del cliente

- Cubrir los temas correspondientes a álgebra, geometría y trigonometría de la materia de matemáticas del tercer año de secundaria en base al plan de estudios de la SEP.
- Calidad en los contenidos temáticos y que la información se presente de manera interactiva.
- Que contenga ejemplos resueltos, explicados en forma detallada.
- Diseñar gráficas que puedan ser analizadas para la mejor comprensión de conceptos en algunos temas.
- Diseño atractivo en las pantallas, menús, iconos, botones, barras de navegación, fondos y texto legible.
- Integración de música, videos, animaciones e imágenes asociadas a cada tema en particular.
- Facilidad de uso e instalación.

Objetivos

- Desarrollar los módulos correspondientes a SECUMATIC para los temas de: álgebra, geometría y trigonometría basados en el plan de estudios de la SEP del tercer año de secundaria.
- Diseñar el contenido de los temas de álgebra, geometría y trigonometría, tomando en cuenta las estrategias didácticas, psicológicas y pedagógicas que sirvan para motivar y reforzar los conocimientos adquiridos de los estudiantes.
- Anexar, en lo posible, los requerimientos obtenidos en la primera evaluación.

Restricciones

- Integrar la información correspondiente a los temas de álgebra geometría y trigonometría.

- Presentar los contenidos con un diseño adecuado al tema y considerando algunas características propias de los gustos en los adolescentes entre 14 y 15 años de edad.
- Presentar la información con un vocabulario sencillo en los diferentes contenidos.

3.3.2.2 Análisis de riesgo

Opciones

- Utilizar fondos, imágenes, animaciones, videos y sonidos que ayuden a crear un ambiente de diseño atractivo, que llame la atención de los alumnos en los temas de álgebra, geometría y trigonometría.
- Diseñar ejemplos interactivos en los cuales se le permita al alumno aplicar los conceptos adquiridos de una manera más divertida y fácil.
- Análisis y estructuración de los objetivos de acuerdo al temario vigente.
- Manejar la opción de activar y desactivar el sonido

Riesgos

- Existe la posibilidad de que a los estudiantes no les parezca atractivo el diseño de las pantallas, botones, animaciones e imágenes.
- El sonido seleccionado puede no ser del agrado de los alumnos.
- El grado de complejidad de los ejemplos propuestos no sea el adecuado para cada tema en particular.
- Que no se cumpla con el propósito de enseñanza-aprendizaje.

- Que el contenido temático no cubra los objetivos de la materia, según los criterios de cada profesor.
- Dificultad en la utilización del sistema para alumnos con poca o nula experiencia en la computadora.

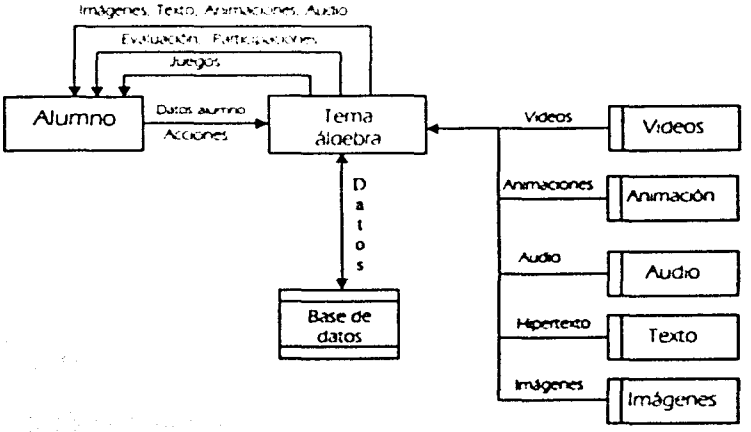
3.3.2.3 Ingeniería en el desarrollo del sistema

3.3.2.3.1. Análisis del sistema

Alcances del sistema

El software abarca en la segunda vuelta, el desarrollo del contenido de los temas de álgebra, geometría y trigonometría, basándonos en los objetivos propuestos en el plan de estudios de la educación secundaria de la SEP.

Diagrama de flujo de datos del tema Álgebra.



3.3.2.3.2 Diseño del sistema

En el diseño de SECUMATIC, para la segunda vuelta, se siguen considerando, el diseño educativo, comunicativo y computacional, para obtener el software propuesto inicialmente y así resolver la necesidad educativa expresada por los profesores.

Para lograr un impacto visual del sistema hacia los estudiantes, se diseñaron imágenes modernistas, que representan la idea general de los temas respectivos.

La pantalla principal de cada tema se presenta a través de un formulario como una página web e integra vídeo, animaciones, fondos de imágenes, textos animados y para navegar de una a otra. En la parte inferior se tiene una barra de botones y ligas de texto, con las cuales se puede avanzar o retroceder hacia los siguientes subtemas y regresar al índice.

Para cada subtema se crearon imágenes y animaciones asociadas al contexto, éstas se muestran en todas las pantallas variando únicamente la posición y el tamaño. Se incluyen ejemplos interactivos por medio del uso del applet de Descartes.

Este applet (programa en lenguaje Java) configurable, es una herramienta a disposición del público en general, creada para preparar páginas web sobre varios temas de las matemáticas, donde los gráficos y los cálculos cobran vida a través de escenas interactivas que permiten a los alumnos investigar propiedades, relacionar y adquirir conceptos, hacer deducciones, plantear y resolver algunos problemas y actividades que se requieren en las clases de matemáticas.

Aparecen incluidos algunos videos, que se activan dando clic a la imagen asociada a éstos, los cuales tratan conceptos y ejemplos relacionados a los temas de estudio.

Se ofrece el uso de la calculadora en todas las pantallas, con el fin de que los alumnos puedan realizar operaciones en el momento que así lo requieran.

Al momento de estudiar un tema, existe la opción de activar o desactivar el sonido a través de un botón, generando un ambiente agradable para el alumno.

Diagrama de control de flujo del tema Álgebra.

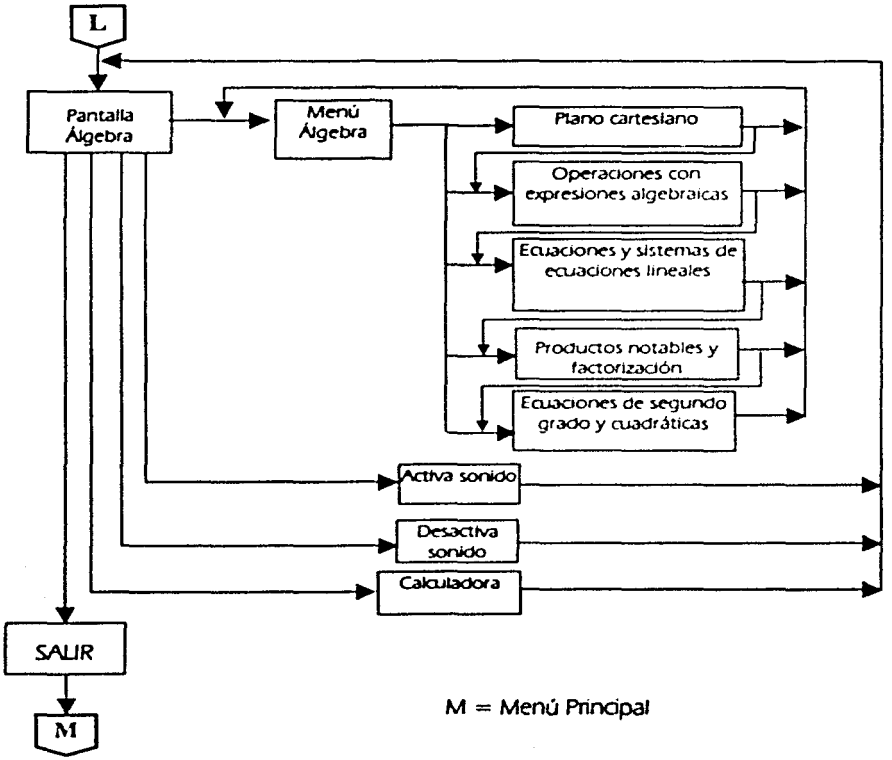
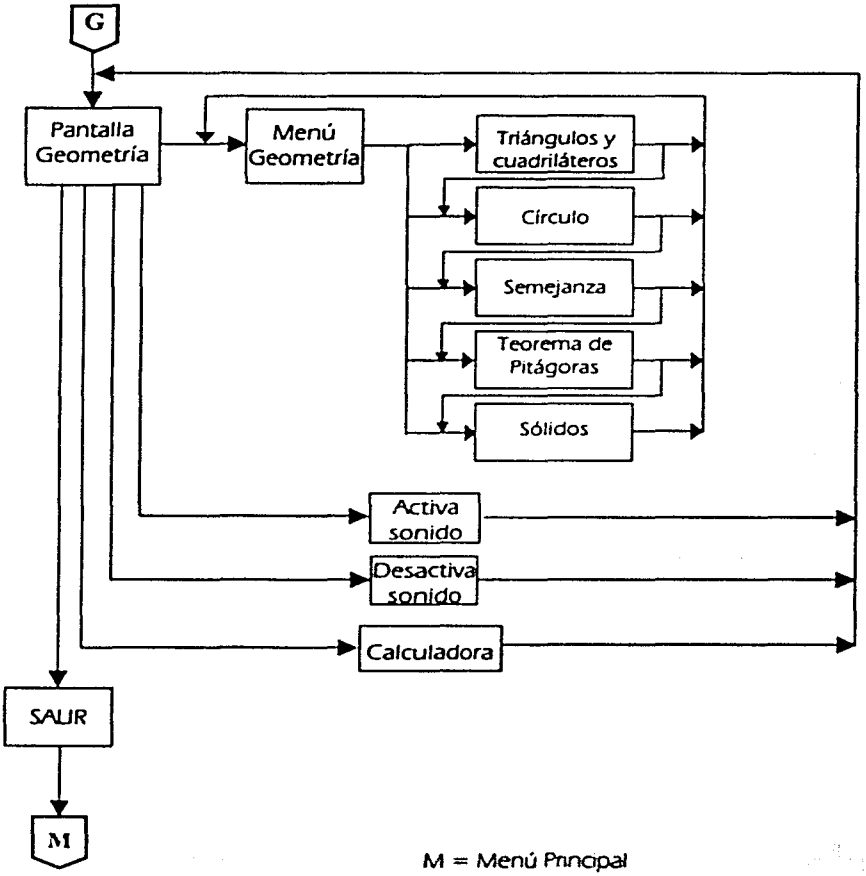
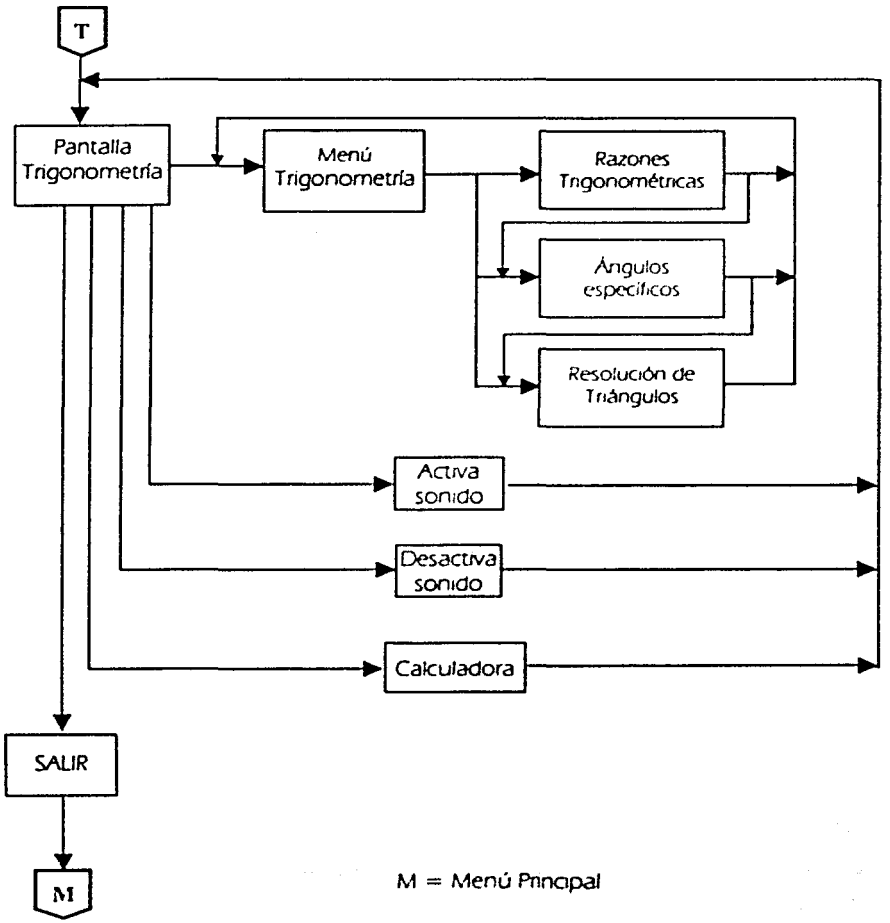


Diagrama de control de flujo del tema Geometría.



M = Menú Principal

Diagrama de control de flujo del tema Trigonometría.



3.3.2.3.3. Desarrollo del sistema

Desarrollo de la pantalla de entrada y registro de los alumnos.

a) Descripción

Al momento de iniciar la aplicación de SECUMATIC se muestra una pantalla de inicio con una animación con figuras geométricas haciendo alusión al área de matemáticas, el cual consta también de un sonido de fondo. Esta pantalla contiene un botón para pasar a la sesión donde se registra la entrada de los alumnos, como se puede visualizar en la Fig. 34.

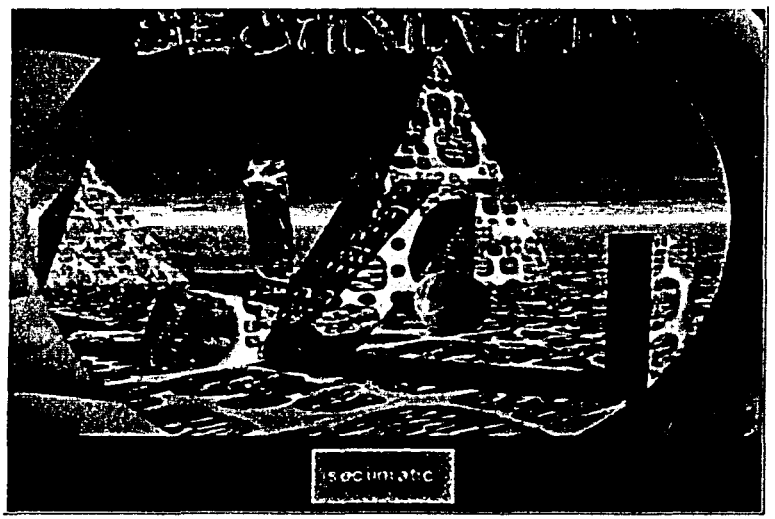


Fig. 34 Pantalla de entrada al iniciar SECUMATIC.

En la sesión del registro de los alumnos se muestra una ventana donde se captura el identificador y la contraseña del alumno para poder accesar al tutorial, aquí mismo aparece un agente animado que solicitará la introducción de estos datos, esta pantalla se detalla en la Fig. 35.

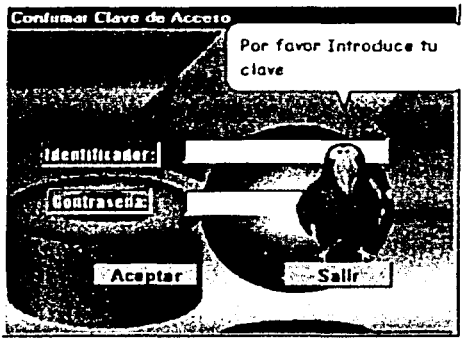


Fig. 35 Pantalla de registro de alumnos.

b) Pseudocódigo

```
Inicia Proceso
    Muestra pantalla inicial
    Muestra ventana de registro
    Muestra agente animado solicitando datos
    Si alumno no existe entonces
        Muestra error en registro
        Inicia contador
        Si contador = 3 entonces
            Muestra error no es usuario registrado
            Fin aplicación
```

```
        Sino
            Otra oportunidad
        Fin_Si
    Sino
        Entrar a Menú principal
    Fin_Si
Fin Proceso
```

c) Código

```
Public Contador As Integer  'Declaración de variable pública
Private Sub Form_Load()
' Muestra la pantalla de inicio de Secumatic e inicializa las variables
necesarias.
    ' Inicialización de objetos para la conexión
      Set oConexion = New SQLOle.SQLOleDB
    ' Inicia la conexión con la base de datos
      oConexion.Conectar C:\SECUMATIC\Reportes_BD\Registro.mdb

    ' Centra la forma en la pantalla
      Me.Top = (Screen.Height - Me.Height) / 2
      Me.Left = ((Screen.Width) - Me.Width) / 2

    ' Descarga la forma de entrada
      Unload frmEntrada
End Sub

Private Sub Form_Activate()
    ' Carga y muestra el Agente
    ' Liberar eventos
      DoEvents
    ' Oculta la etiqueta de espera
      frmFondo.Label1.Visible = False
    ' Inicializa el objeto del agente
      Set Mago = New oAgente
```

```
    ' Activa al agente
      Mago.CreaAgente App.Path & "\peedy.acs"
      Mago.MoverAgente Horizontal:=190, Vertical:=310, Velocidad:=1
      Mago.HablaAgente "Por favor Introduce tu clave"
End Sub

Private Sub cmdAceptar_Click()
    'Verifica en la base de datos si el usuario y la contraseña son correctos, se
    dan dos oportunidades de error, a la tercera se termina la aplicación.
    'Declaración de variables locales
     Dim sConsulta As String
     Dim igrupo As Integer

    ' Inicializa el contador de entradas
      Contador = Contador + 1
     If Contador = 3 Then
           Mago.ActividadAgente "Confused"
           Mago.HablaAgente "No eres usuario registrado, solicita al profesor
    tu login y contraseña. ¡Hasta Luego!"
           Mago.OcultaAgente
           End
     End If

    ' Inicializa la cadena de consulta
     sConsulta = "SELECT * "
     sConsulta = sConsulta & "FROM   alumno "
     sConsulta = sConsulta & "WHERE  alum_log = '" & TxtAlumId.Text & "' "
     sConsulta = sConsulta & "AND    alum_pwd = '" & TxtAlumPwd.Text & "'
    "
    ' Ejecuta la consulta en la base de datos
     oConexion.EjecutarSQL sConsulta

    ' Verifica si el alumno está registrado
    If oConexion.resultado.RecordCount < 1 Then
        Mago.ActividadAgente "Decline"
```

```
    Mago.HablaAgente "La clave o la contraseña son incorrectos, vuelve a
    teclearlos"

        TxtAlumId.Text = ""
        TxtAlumPwd.Text = ""
        TxtAlumId.SetFocus
        Exit Sub
    Else
      ' Asigna la matrícula del alumno
        Usuario = oConexion.resultado.Fields!alum_id
        Grupo = oConexion.resultado.Fields!gpo_id

      ' Muestra la etiqueta de espera
        frmFondo.Label1.Visible = True
        Mago.OcultaAgente
        Unload Me
        Unload frmEntrada
        Set frmEntrada = Nothing
        Set frmConocido = Nothing
        MDIMenu.Show
    End If
End Sub

Private Sub cmdSalir_Click()
' Este procedimiento termina la aplicación de Secumatic al pulsar el botón
salir, se destruyen todas las variables y objetos ocupados por la aplicación.
    ' Se despide el agente
      Mago.ActividadAgente "Surprised"
      Mago.HablaAgente "Haz decidido terminar tu sesión. Te espero pronto"
      Mago.ActividadAgente "Wave"

    ' Termina agente
      Mago.TerminaAgente
    ' Termina la aplicación
      oConexion.Desconectar
    ' Destrucción de los objetos
```

```
    Set oAgente = Nothing
    Set oConexion = Nothing
    Set frmFondo = Nothing

  ' Termina la aplicación
    End
End Sub
```

d) Elementos

Estas dos pantallas contienen los siguientes elementos:

2 animaciones en Flash, con sonido
1 agente animado
2 imágenes en formato jpg

Desarrollo de los temas Álgebra, Geometría y Trigonometría.

a) Descripción

Cada uno los tres temas, contienen una pantalla principal con el índice de los subtemas, correspondientes a cada uno de ellos. Estas pantallas contienen un botón de comando para mostrar la calculadora interna de Windows, 2 botones para activar o desactivar la música de fondo, 1 botón para accesar a un juego y otro para salir del tema. En la Fig. 36 se muestra la pantalla del tema de álgebra.

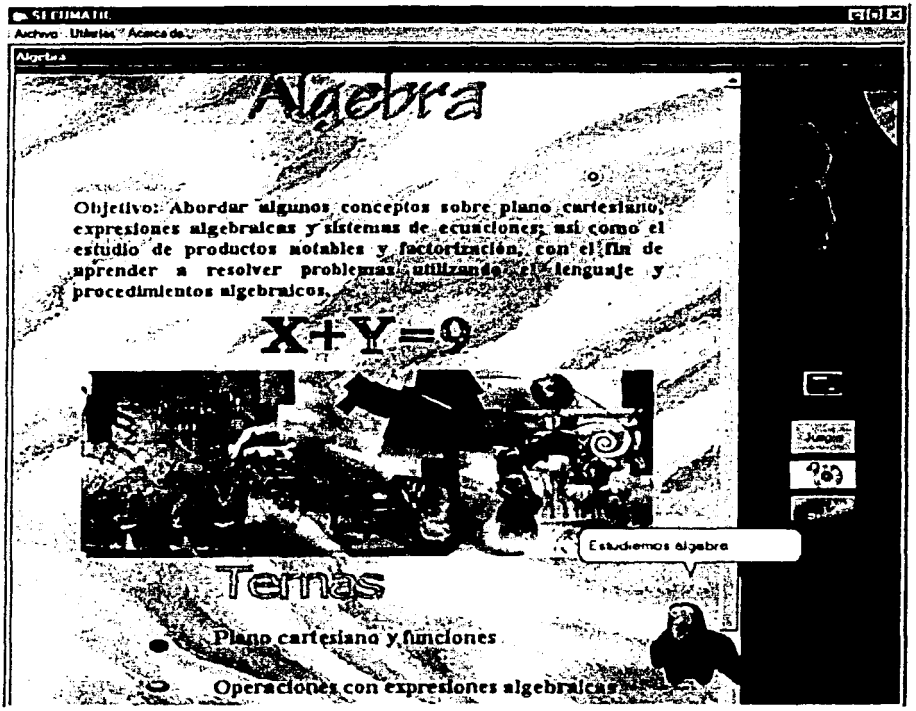


Fig. 36 Pantalla principal del tema de Álgebra.

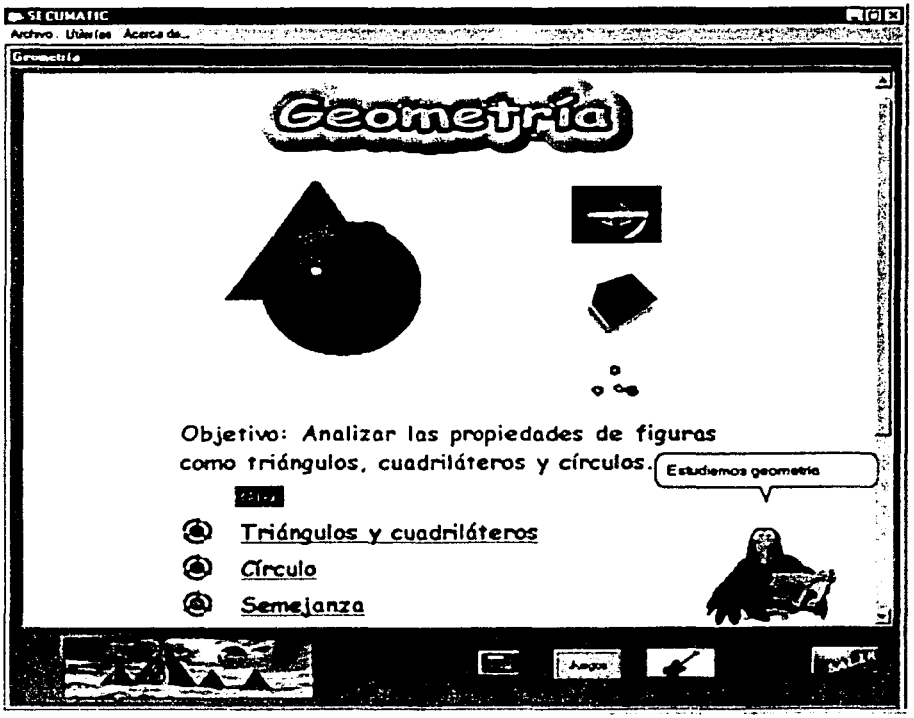


Fig. 37 Pantalla pricipal del tema de Geometría.

Fig. 38 Pantalla pricipal del tema de Trigonometría.

b) Pseudocódigo

Inicia Menú del tema

Muestra Agente animado

Mientras Función exista
    Ejecuta Función
    En caso de que función sea
        1: Función Indice tema
        2: Función Calculadora
        3: Función Sonido
    Fin caso

Fin Mientras

Termina Menú del tema

c) Código

El siguiente código se aplica para los temas de álgebra, geometría y trigonometría, cambiando solamente el tema que se haya escogido desde el menú.

```
Private Sub Form_Load()
  ' Inicializa la forma
    Me.Top = 0
    Me.Left = 0
' Carga pista de audio
  Music.FileName = App.Path & "\Música\Pista2.mp3"
End Sub

Private Sub Form_Activate()
```

En este procedimiento se obtiene la ruta donde se encuentra el tema almacenado, en este caso Algebra y carga el Agente animado.

```
  ' Obtiene la ruta donde se encuentran los temas
    sRutaHTML  = "C:\SECUMATIC\"
  ' Muestra el documento html
    If sCaracter = "\" Then
      WebBrowser1.Navigate2 sRutaHTML & _
                    "Algebra\ALGEBRAPRINCI\ALGEBRAINDICE.htm"
    Else
      WebBrowser1.Navigate2 sRutaHTML &_
                    "\Algebra\ALGEBRAPRINCI\ALGEBRAINDICE.htm"
    End If
  ' Muestra el agente
    If oBandera = 0 Then
     ' Inicializa la bandera
      oBandera = oBandera + 1
     ' Asigna las propiedades al agente
      Mago.MuestraAgente
      Mago.HablaAgente "Estudiemos álgebra"
      Mago.ActividadAgente "Read"
      Mago.OcultaAgente
     ' Activa el sonido
      Call cmdIniSonido_Click
    End If
End Sub

Private Sub cmdcalcalg_Click()
Abre la calculadora de windows.
    ' Declaración de variables locales
      Dim lHandler As Long

    ' Activa la calculadora
      Shell "calc.exe", vbNormalFocus

    ' Obtener el handler o identificador de la calculadora
```

```
    lHandler = FindWindow(0&, "Calculadora")

    ' Establecer la calculadora en primer término al frente de todas la
    ventanas
     SetWindowPos lHandler, conHwndTopmost, 100, 100, 271,279,
    conSwpShowWindow
End Sub

Private Sub cmdIniSonido_Click()
Dim ruta As String
    Music.play
    cmdIniSonido.Visible = False
    cmdFinSonido.Visible = True
End Sub

Private Sub cmdFinSonido_Click()
    Music.stop
    Music.CurrentPosition = 0
    cmdFinSonido.Visible = False
    cmdIniSonido.Visible = True
End Sub

Private Sub Form_Unload(Cancel As Integer)
    ' Termina la forma
     Call cmdfin_Click
End Sub
Private Sub cmdfin_Click()
    ' Descarga la forma
     Unload Me
    ' Destruye la forma
     Set frmAlgebra = Nothing

    ' Muestra la forma padre
     frmMdiInicio.Show
End Sub
```

c) Elementos

	Álgebra	Geometría	Trigonometría
Animaciones	51	72	15
Imágenes	29	98	20
Fondos de papel tapíz	22	13	4
Botones	47	43	13
Agente animado	1	1	1
Applets de Descartes	33	2	1
Archivo de Sonido	1	43	1
Videos		3	
Juegos	1	1	1

3.3.2.3.4 Pruebas

Al concluir la segunda vuelta se tienen teminados los siguientes tres módulos correspondientes a los temas de álgebra, geometría y trigonometía, para lo cual se realizaron las siguientes pruebas:

Se revisó el código programado en Visual Basic, ejecutando la aplicación primero a nivel de cada módulo, realizando las pruebas de unidad y determinando que no existieran errores de sintaxis, ejecución, proceso, llamadas a funciones, ligas y validaciones.

En lo que se refiere al diseño, se revisaron los aspectos multimedia que debe cubrir el sistema en cada uno de los temas descritos, de acuerdo al planteamiento de la justificación del sistema.

Una vez terminados los tres módulos antes descritos, se hicieron pruebas de integridad entre éstos.

3.3.2.4 Evaluación de la segunda vuelta

Para realizar la evaluación al módulo de SECUMATIC en la segunda vuelta, que abarca en su totalidad los cuatro temas Áritmética, Álgebra, Geometría y Trigonometría, se utilizó un disco compacto para cada equipo.

La evaluación se realizó en las tres secundarias seleccionadas inicialmente:

- Escuela Secundaria "Nochcalco", turno vespertino, que se ubica en la Calle Simón Bolivar No. 180, Delegación Milpa Alta, D.F.
- Escuela Secundaria "Juventino Rosas Cárdenas", turno vespertino, que se ubica en la Calle Tarascos s/n, Col. Tlalcoligia, Delegación Tlalpan, D.F.
- Escuela Secundaria No. 39, turno matutino, que se ubica en la Calle Moneda s/n, Delegación Tlalpan, D.F.

La presentación de los tres módulos (álgebra, geometría, trigonometría) se realizó en el aula de computadoras descrita en la primera evaluación.

En este caso, asistieron en la escuela "Nochcalco"; 4 profesores y 35 alumnos de los grupos tercero C y F; para la escuela "Juventino Rosas Cárdenas"; la directora, 3 profesores y 20 alumnos de los grupos tercero A, B y C; en la escuela "No. 39"; el director, 2 profesores y 15 alumnos de los grupos tercero B, D, y E de la materia de matemáticas.

Inicialmente se procedió a la explicación y demostración de las principales funciones de control del sistema, la evaluación se realizó partiendo de la pantalla de registro de alumno, donde cada uno ingresó su nombre y clave de acceso, quedando su sesión registrada en la base de datos.

En las tres escuelas se mostraron los temas de álgebra, geometría y trigonometría, explicando con mayor detalle los botones de control y la manera de accesar a los subtemas respectivos.

En la escuela "Nochcalco", los profesores resaltaron que el tema de álgebra debe contener más ejercicios interactivos, ya que es el de mayor complejidad para los alumnos y éstos comentaron que al asociar las animaciones y videos con los conceptos involucrados les permite tener mayor retención de los temas.

La directora de la escuela "Juventino Rosas Cárdenas", expresó que es de suma importancia poder hacer uso de sistemas interactivos, ya que de esa manera los alumnos pueden reforzar y comprender mejor los temas de matemáticas. Recibimos todo el apoyo de los profesores para mejorar el contenido de algunos temas y ejercicios muy particulares.

En la secundaria "No. 39", los profesores indicaron que los alumnos aprenden mejor a través del uso de juegos, porque sienten más confianza y al mismo tiempo adquieren mayor destreza para la solución de los problemas.

En general, el sistema les pareció muy completo en los contenidos y la forma en que se presenta la información está bien estructurada, además de un diseño muy atractivo.

Por su parte, los alumnos de las tres escuelas, se mostraron muy participativos y motivados al momento de explorar los temas, indicaron que es muy fácil navegar en las pantallas y que el hecho de incluir animaciones para algunos ejemplos, como es el caso del teorema de Pitágoras, les ayudó a comprender mejor el concepto, ya que se presenta la explicación y simultáneamente pueden visualizarlo. Sin embargo, el tamaño de la letra les pareció muy pequeño e indicaron que es más agradable leer en la computadora cuando se tienen textos con un tamaño de letra mayor y de diferente color.

En conclusión de las observaciones a la segunda evaluación, los profesores señalaron que es necesario agregar una parte donde se pueda evaluar el aprendizaje de los alumnos, con el fin de detectar en qué temas presentan mayor dificultad y trabajar más en éstos, además de que les permita generar reportes de cada profesor con sus respectivos grupos.

3.3.3 Tercera vuelta

3.3.3.1 Planificación

En la tercera vuelta, planteamos como objetivo desarrollar un módulo asignado a la evaluación de los cuatro temas implementados en la primera y segunda vuelta que conforman el sistema SECUMATIC, desde el cual se tendrá el control de alumnos, grupos y profesores.

Requisitos del cliente

- Anexar al sistema un módulo para la evaluación, que contenga preguntas para exámenes correspondientes a cada tema (aritmética, álgebra, geometría y trigonometría).
- Generación de reportes, por grupo y profesor, sesiones y calificaciones.
- Altas, bajas y cambios al catálogo de preguntas, para la elaboración de los exámenes.

Objetivos

- Desarrollar un módulo para el control de alumnos, profesores y reportes de calificaciones.
- Tomar en cuenta las observaciones hechas por el cliente en la segunda evaluación.
- Obtener el producto final.

Restricciones

- Que el módulo de evaluación resulte sencillo y fácil de utilizar para los profesores.
- Diseño amigable en las pantallas, para facilitar el ingreso de los datos.
- Que la plantilla de preguntas para la elaboración de exámenes, corresponda única y exclusivamente a los temas desarrollados en el módulo de SECUMATIC.

3.3.3.2 Análisis de riesgo

Opciones

- Evaluar al alumno mediante la aplicación de exámenes.
- Diseñar el catálogo de preguntas para la elaboración de los exámenes.
- Diseño de plantillas para la generación de reportes y exámenes.
- Realizar las correcciones finales al módulo de SECUMATIC.

Riesgos

- Que el módulo de administración de alumnos y profesores para la evaluación, resulte poco funcional para el profesor.
- El grado de complejidad de las preguntas para la evaluación, no sea el requerido.
- Planteamiento ambiguo de preguntas para los exámenes.
- Dificultad en la utilización del sistema para profesores con poca o nula experiencia en la computadora.

3.3.3.3 Ingeniería en el desarrollo del sistema

3.3.3.3.1 Análisis del sistema

Alcances del Sistema

El software abarca en la tercera vuelta, el refinamiento del contenido de los temas de la materia de matemáticas tres, basándonos en los objetivos propuestos en el plan de estudios de la educación secundaria de la SEP. Además de integrar un módulo adicional para la evaluación de los alumnos, donde el profesor pueda diseñar sus exámenes o modificar el catálogo de preguntas.

3.3.3.3.2 Diseño del sistema

El módulo para control de alumnos y profesores, está diseñado con la finalidad de que los profesores lleven un registro de las calificaciones de los exámenes aplicados a los alumnos, ya sea por tema, subtema o sesión y que pueda generar reportes con dichos datos, además de diseñar o modificar las preguntas de su examen.

La pantalla de inicio del módulo para control de alumnos y profesores, tiene un menú a través del cual podemos seleccionar la opción de administración, aula, exámenes.

- La opción de administración permite dar de alta a profesores y alumnos, ingresando los datos generales del profesor y los grupos a los que va a dar clase. También se ingresan los datos generales de los alumnos y se les asigna su grupo.
- Si seleccionamos la opción de exámenes, el profesor puede elaborar un examen, seleccionar uno ya existente, crearlo con nuevas preguntas o modificar solamente el catálogo existente.

3.3.3.3.3 Desarrollo del sistema

Desarrollo del módulo de evaluación

a) Descripción

Este módulo sirve para aplicar exámenes a los alumnos, éstos se obtienen de la base de datos y se despliegan mediante fichas donde los alumnos tienen tres opciones para elegir, como se muestra en la Fig. 39; cuando hayan terminado de contestar todas las preguntas de su examen, los alumnos tienen que pulsar el botón "terminar examen". Inmediatamente se evaluarán las respuestas del alumno y se mostrará el número de aciertos, así como la calificación del alumno, la que será almacenada en la base de datos; se dan unos segundos y se cierra el examen.

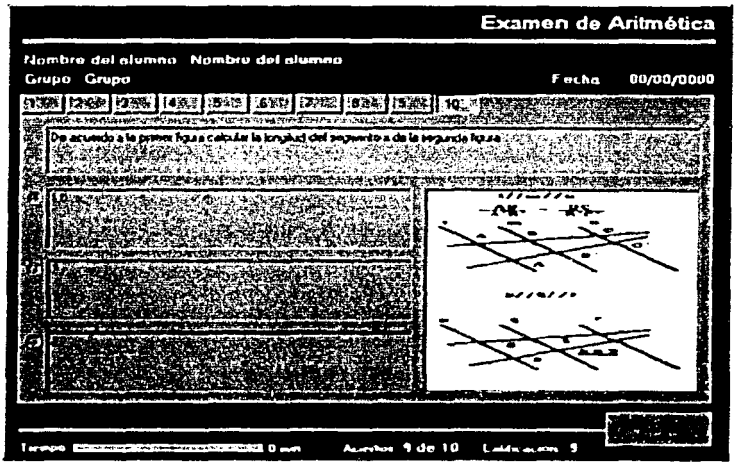


Fig. 39 Pantalla de un exámen enviado a los alumnos.

b) Pseudocódigo

Inicio
  Carga de examen
  Se obtiene preguntas del examen
  Obtener resultados del alumno
  Se comparan resultados
  Se obtienen el número de aciertos
  Se obtiene calificación
  Se almacena calificación y número de aciertos
  Se muestra aciertos y calificación
Fin

c) Código

```
Private Sub Form_Load()
Se carga la forma del examen para el alumno y se obtienen las preguntas.
    ' Declaración de variables locales
      Dim sConsulta As String

    ' Inicialización de la forma
      Me.Top = (Screen.Height - Me.ScaleHeight) / 2
      Me.Left = (Screen.Width - Me.ScaleWidth) / 2

    ' Obtiene las preguntas
              sConsulta = "SELECT exam_consec, exam_consec "
      sConsulta = sConsulta & "FROM   profesor_examen "
      sConsulta = sConsulta & "WHERE  exam_id = '" & pExamenId & "' "
      sConsulta = sConsulta & "AND    prof_id = '" & pProfesorId & "' "
      sConsulta = sConsulta & "AND    tem_id = '" & pTemaId & "' "

    ' Ejecuta la consulta
      zConexion.Ejecutar sConsulta
      zConexion.Llenado Tab_Strip, TabStrip1
```

```
    'Muestra la primer pregunta
      Call TabStrip1_Click

    ' Inicializa el progressbar
      ProgressBar1.Min = 0
      ProgressBar1.Max = pTiempoExam
      ProgressBar1.Value = pTiempoExam
    ' Inicializa el timer
      Timer1.Interval = 1000
End Sub

Private Sub TabStrip1_Click()
Se obtiene la clave de la pregunta seleccionada
    ' Declaración de variables locales
      Dim sLLave As String
      Dim iPosicion As Integer
      Dim sClave As String
      Dim sConsulta As String

    ' Obtiene la pregunta y respuesta
      sLLave = TabStrip1.SelectedItem.Key

    ' Obtiene la posición del caracter "_" para cortar la cadena K1_189
      iPosicion = InStr(1, sLLave, "_", vbTextCompare) + 1

    ' Obtiene la clave
    '                 K1_189 ,   4      ,   6
      sClave = Mid(sLLave, iPosicion, Len(sLLave))

    ' Manda a llamar a la función Obtienepregunta
      If ObtienePregunta(sClave) = False Then
        Option1.Value = False
        Option2.Value = False
        Option3.Value = False
      End If
```

```
End Sub

Public Function ObtienePregunta(ByRef opClave As String) As Boolean
Obtiene y muestra el número de la pregunta seleccionada.
  ' Declaración de variables locales
    Dim sConsulta As String

  ' Controlador de errores generales
    On Local Error Resume Next

  ' Obtiene la pregunta
            sConsulta = "SELECT * "
    sConsulta = sConsulta & "FROM   profesor_examen "
    sConsulta = sConsulta & "WHERE  exam_id = '" & pExamenId & "' "
    sConsulta = sConsulta & "AND    prof_id = '" & pProfesorId & "' "
    sConsulta = sConsulta & "AND    tem_id = '" & pTemaId & "' "
    sConsulta = sConsulta & "AND    exam_consec = " & CInt(opClave) & " "
  ' Ejecuta la consulta
    zConexion.Ejecutar sConsulta

  ' Valida el resultado
    If zConexion.resultado.RecordCount < 1 Then
      ObtienePregunta = False
    Else
      rtfPregunta.Text = zConexion.resultado.Fields!exam_pregunta
      txtResp1.Text = zConexion.resultado.Fields!exam_respuesta_a
      txtResp2.Text = zConexion.resultado.Fields!exam_respuesta_b
      txtResp3.Text = zConexion.resultado.Fields!exam_respuesta_c
      Image1.Picture =
  LoadPicture(zConexion.resultado.Fields!exam_imagen)
      Image1.Tag = zConexion.resultado.Fields!exam_imagen
      Valida_Pregunta pAlumnoId, sExamen,
        zConexion.resultado.Fields!exam_consec, _
              zConexion.resultado.Fields!exam_resultado

      ObtienePregunta = True
```

```
    End If
End Function

Public Function Valida_Pregunta(ByVal sAlumId As String, ByVal sExamId
As String, ByVal iPregId As Integer, ByVal iRespId As Integer) As Boolean
```

Esta función es llamada cada vez que se selecciona una pregunta. Valida si el alumno ya había contestado cierta pregunta y muestra la respuesta seleccionada.

```
    ' Declaración de variables locales
      Dim sConsulta As String

    ' Inicializa la cadena de consulta

            sConsulta = "SELECT * FROM evaluacion "
      sConsulta = sConsulta & "WHERE alum_id ='" & pAlumnoId & "' "
      sConsulta = sConsulta & "AND   exam_id ='" & pExamenId & "' "
      sConsulta = sConsulta & "AND   eval_pregunta =" & iPregId

    ' Ejecuta consulta
      zConexion.EjecutarSQL sConsulta

      If zConexion.resultado.RecordCount < 1 Then
              sConsulta = "Insert into Evaluacion "
        sConsulta = sConsulta & "(alum_id,exam_id,eval_pregunta, _
                                  eval_resp_tutor,eval_resp_alumno) "
        sConsulta = sConsulta & "VALUES ("
        sConsulta = sConsulta & "'" & pAlumnoId & "', "
        sConsulta = sConsulta & "'" & pExamenId & "', "
        sConsulta = sConsulta & " " & iPregId & ", "
        sConsulta = sConsulta & " " & iRespId & ", "
        sConsulta = sConsulta & "0)"
        'Ejecuta consulta
           zConexion.SoloEjecutar sConsulta
           Valida_Pregunta = False
```

```
    Else
      If zConexion.resultado.Fields!eval_resp_alumno = 1 Then
        Option1.Value = True
      Else
        If zConexion.resultado.Fields!eval_resp_alumno = 2 Then
          Option2.Value = True
        Else
          If zConexion.resultado.Fields!eval_resp_alumno = 3 Then
            Option3.Value = True
          Else
            Option1.Value = False
            Option2.Value = False
            Option3.Value = False
          End If
        End If
      End If
      Valida_Pregunta = True
    End If
End Function

Private Sub Option1_Click()
Determina el número de la pregunta seleccionada y guarda  la respuesta
contestada como la opción uno.
' Declaración de variables locales
  Dim sLLave As String
  Dim iPosicion As Integer
  Dim sClave As String
  Dim sConsulta As String

' Obtiene la pregunta y respuesta
  sLLave = TabStrip1.SelectedItem.Key

' Obtiene la posición del caracter "_" para cortar la cadena K1_4
  iPosicion = InStr(1, sLLave, "_", vbTextCompare) + 1

' Obtiene la clave
```

```
  sClave = Mid(sLLave, iPosicion, Len(sLLave))

' Actualiza la opción
        sConsulta = "UPDATE evaluacion SET eval_resp_alumno = 1 "
  sConsulta = sConsulta & "WHERE  alum_id = '" & pAlumnoId & "' "
  sConsulta = sConsulta & "AND    exam_id = '" & pExamenId & "' "
  sConsulta = sConsulta & "AND    eval_pregunta = " & sClave & " "

' Ejecuta la consulta
  zConexion.SoloEjecutar sConsulta
End Sub

Private Sub Option2_Click()
Determina el número de la pregunta seleccionada y guarda  la respuesta
contestada como la opción dos.
' Declaración de variables locales
  Dim sLLave As String
  Dim iPosicion As Integer
  Dim sClave As String
  Dim sConsulta As String

' Obtiene la pregunta y respuesta
  sLLave = TabStrip1.SelectedItem.Key

' Obtiene la posición del caracter "_" para cortar la cadena K1_4
  iPosicion = InStr(1, sLLave, "_", vbTextCompare) + 1

' Obtiene la clave
  sClave = Mid(sLLave, iPosicion, Len(sLLave))
' Actualiza la opción
        sConsulta = "UPDATE evaluacion SET eval_resp_alumno = 2 "
  sConsulta = sConsulta & "WHERE  alum_id = '" & pAlumnoId & "' "
  sConsulta = sConsulta & "AND    exam_id = '" & pExamenId & "' "
  sConsulta = sConsulta & "AND    eval_pregunta = " & sClave & " "
```

```
' Ejecuta la consulta
  zConexion.SoloEjecutar sConsulta
End Sub
Private Sub Option3_Click()
Determina el número de la pregunta seleccionada y guarda la respuesta
contestada como la opción tres.
' Declaración de variables locales
  Dim sLLave As String
  Dim iPosicion As Integer
  Dim sClave As String
  Dim sConsulta As String

' Obtiene la pregunta y respuesta
  sLLave = TabStrip1.SelectedItem.Key


' Obtiene la posición del caracter "_" para cortar la cadena K1_4
  iPosicion = InStr(1, sLLave, "_", vbTextCompare) + 1

' Obtiene la clave
  sClave = Mid(sLLave, iPosicion, Len(sLLave))

' Actualiza la opción
          sConsulta = "UPDATE evaluacion SET eval_resp_alumno = 3 "
  sConsulta = sConsulta & "WHERE  alum_id = '" & pAlumnoId & "' "
  sConsulta = sConsulta & "AND    exam_id = '" & pExamenId & "' "
  sConsulta = sConsulta & "AND    eval_pregunta = " & sClave & " "

' Ejecuta la consulta
  zConexion.SoloEjecutar sConsulta
End Sub
Private Sub BtnEnh1_Click()
Termina el examen.
   ' Mensaje de salida
     If MsgBox("¿Está seguro de finalizar el examen?", vbQuestion _
                              + vbYesNo, APPname) = vbNo Then
```

```
        Exit Sub
      End If


    ' Obtiene la calificación del alumno
      ProgressBar1.Value = 0
      iTiempo = 60
      Call Timer1_Timer

    ' Inicializa timer de salida
      Timer2.Interval = 10000
End Sub

Private Sub Timer1_Timer()
Controla el tiempo del examen, determina el número de aciertos y la
calificación del alumno y los muestra en la forma del examen.
Dim numPregunta As Integer
Dim numAcierto As Integer
Dim Calificacion As Double
Dim sConsulta As String
Dim lVeces As Integer

If iTiempo = 60 Then
 ' Inicializa la variable
   iTiempo = 0
 ' Descontar el minuto del progressbar
   ProgressBar1.Value = IIf(ProgressBar1.Value - 1 = -1, 0,
ProgressBar1.Value)
 ' Muestra el tiempo pendiente
   lblMinutos.Caption = ProgressBar1.Value & " min"
 ' Valida el tiempo restante
   If ProgressBar1.Value = 0 Then
    ' Bloquea la forma
      TabStrip1.Enabled = False
      Option1.Enabled = False
      Option2.Enabled = False
```

```
    Option3.Enabled = False
  ' Termina el timer
    Timer1.Enabled = False

  ' Obtiene el número de preguntas del examen
    sConsulta = "SELECT COUNT(exam_id) "
    sConsulta = sConsulta & "FROM profesor_examen "
    sConsulta = sConsulta & "WHERE exam_id = '" & pExamenId & "' "
    sConsulta = sConsulta & "AND prof_id = '" & pProfesorId & "' "
    sConsulta = sConsulta & "AND tem_id = '" & pTemaId & "'"
  ' Ejecutar consulta
    zConexion.EjecutarSQL sConsulta
  ' Obtiene el número de preguntas
    numPregunta = zConexion.resultado.Fields(0)


  ' Obtiene los aciertos
            sConsulta = "SELECT * "
    sConsulta = sConsulta & "FROM evaluacion "
    sConsulta = sConsulta & "WHERE exam_id = '" & pExamenId & "' "
    sConsulta = sConsulta & "AND alum_id = '" & pAlumnoId & "' "
  ' Ejecuta la consulta
    zConexion.EjecutarSQL sConsulta
  ' Validar consulta
    If zConexion.resultado.RecordCount < 1 Then
      MsgBox "No se encontró el examen del alumno o el alumno no lo
contestó", vbCritical, APPname
       Exit Sub
    Else
    ' Inicia ciclo para obtener el número de aciertos
      For lVeces = 1 To zConexion.resultado.RecordCount
        ' Determina si es acierto o error
          If zConexion.resultado.Fields!eval_resp_tutor = _
                        zConexion.resultado.Fields!eval_resp_alumno Then
            numAcierto = numAcierto + 1
          End If
```

```
    ' Se mueve el siguiente registro
      zConexion.resultado.MoveNext
    ' Valida si es fin de consulta
      If zConexion.resultado.EOF = True Then
        Exit For
      End If
  Next lVeces
' obtiene calificación
  Calificacion = (numAcierto * 10) / numPregunta
' Valida calificación
  If Calificacion < 6 Then
    Calificacion = Int(Calificacion)
  Else
    Calificacion = Round(Calificacion, 1)
  End If
' Asigna resultado
  lblAcierto.Caption = numAcierto & " de " & numPregunta
  lblCalificacion.Caption = Calificacion
' inicializa cadena para insertar calificacion
          sConsulta = "INSERT INTO calificacion "
  sConsulta = sConsulta & "VALUES ("
  sConsulta = sConsulta & "'" & pAlumnoId & "', "
  sConsulta = sConsulta & "'" & pTemaId & "', "
  sConsulta = sConsulta & "'" & pExamenId & "', "
  sConsulta = sConsulta & "'" & pProfesorId & "', "
  sConsulta = sConsulta & "#" & lblFecha.Caption & "#, "
  sConsulta = sConsulta & " " & Calificacion & ")"
' Ejecutar consulta
  zConexion.SoloEjecutar sConsulta
' Se inicializa cadena para eliminar instrucción temporal
  sConsulta = "DELETE FROM examen_asigna "
  sConsulta = sConsulta & "WHERE exa_alumno = '" & pAlumnoId & "' "
  sConsulta = sConsulta & "AND exa_examen = '" & pExamenId & "' "
  sConsulta = sConsulta & "AND exa_profesor = '" & pProfesorId & "' "
  sConsulta = sConsulta & "AND exa_tema = '" & pTemaId & "'"
' Ejecuta consulta
```

```
      zConexion.SoloEjecutar sConsulta
    ' Inicializa el timer para salida
      Timer2.Interval = 10000
    ' Inhibir el botón de salida
      BtnEnh1.Enabled = False
    End If
  End If
Else
 ' Incrementar variable
  iTiempo = iTiempo + 1
End If
End Sub

Private Sub Timer2_Timer()
' Cuenta diez segundos y descarga la forma del examen
   Unload Me
End Sub

Private Sub Form_Unload(Cancel As Integer)
    'zConexion.Desconectar
End Sub
```

3.3.3.3.4 Pruebas

Una vez terminados los módulos en la tercera vuelta (sistema completamente implementado), se probó cada módulo por separado y en conjunto, revisando que no existieran errores de sintaxis, ejecución, procesos, llamadas a funciones, ligas y validaciones, entre cada uno de ellos.

También se revisó que todos los componentes multimedia utilizados por el sistema funcionaran adecuadamente.

3.3.3.4 Evaluación de la tercera vuelta

Para llevar a cabo la evaluación del sistema en la tercera vuelta, se probaron simultáneamente los módulos de SECUMATIC y evaluación. Se hicieron pruebas de unidad en donde se aseguró que cada módulo funcionara correctamente de manera individual.

Para validar la integridad entre los dos módulos, se probaron conjuntamente, dando de alta a los alumnos y profesores en el módulo de evaluación y validando su acceso en SECUMATIC. También se probó que los exámenes se visualizaran en la pantalla correctamente y que los resultados fueran almacenados en la base de datos y de esta manera, generar reportes de calificaciones o listas de alumnos de las sesiones y temas consultados.

El sistema SECUMATIC incluyendo el módulo de control de alumnos y profesores, para la tercera evaluación se presentó nuevamente en las tres secundarias seleccionadas:

- Escuela Secundaria "Nochcalco", turno vespertino, que se ubica en la Calle Simón Bolivar No. 180, Delegación Milpa Alta, D.F.
- Escuela Secundaria "Juventino Rosas Cárdenas", turno vespertino, que se ubica en la Calle Tarascos s/n, Col. Tlalcoligia, Delegación Tlalpan, D.F.
- Escuela Secundaria No. 39 turno matutino, que se ubica en la Calle Moneda s/n, Delegación Tlalpan, D.F.

La presentación del proyecto final se realizó en el aula de computadoras de cada escuela, descrita en la primera evaluación.

En está última evaluación asistieron, en la escuela Nochcalco: cinco profesores y 18 alumnos de diferentes grupos de tercero; para la escuela

Juventino Rosas Cárdenas: la directora, cuatro profesores y 15 alumnos de los grupos tercero D y E; en la escuela No. 39: cinco profesores y 19 alumnos de todos los grupos de tercero de la materia de matemáticas.

En cada escuela se inició con la explicación y demostración de los principales botones de control en el módulo de evaluación, al director y profesores, en donde pudieron darse de alta, elaborar un examen, modificar el catálogo de preguntas, generar reportes y consultar la lista de alumnos asignados a sus grupos.

Posteriormente, se iniciaron las sesiones en el módulo de SECUMATIC con los alumnos, donde pudieron consultar todos los temas, practicar ejercicios interactivos y resolver un examen.

Los alumnos en general expresaron que la idea de estudiar matemáticas a través de un software resulta más divertida, interesante y que pueden retener más fácilmente la información cuando ésta se presenta asociada a imágenes, animaciones, video y ejercicios interactivos. Manifestaron que el sistema tienen un diseño adecuado en los contenidos y que resulta sencillo localizar cada tema y sus respectivos subtemas, debido a la forma en que se van presentando a través de ligas de texto o botones.

Por su parte, los profesores argumentaron que el sistema integra muy bien el aspecto psicopedagógico y que tiene calidad informativa en los contenidos desarrollados, además de que resulta muy atractivo el hecho de tener a su disposición un catálogo de preguntas para poder elaborar sus exámenes o modificarlos, permitiéndoles evaluarlos de una forma más rápida y sencilla.

CAPÍTULO 4

CONCLUSIONES

Conclusiones

La tecnología computacional avanza día a día, por lo que ha sido necesario adaptarnos a los cambios que ésta implica, particularmente en el ámbito educativo. Por tal motivo, los profesores utilizan software educativo para impartir sus clases, logrando así una participación activa del alumno, ya que se busca dotar a los estudiantes de una educación de calidad.

Con el desarrollo del sistema SECUMATIC cumplimos nuestro objetivo, ya que sirve principalmente como apoyo didáctico para complementar el aprendizaje de las matemáticas de manera interactiva, a través del uso de la computadora, además de contemplar los cuatro temas más representativos de la materia.

La integración de multimedia fue determinante en la implementación del sistema, ya que nos permitió transmitir conocimientos, conceptos y muchos atributos más a través de imágenes, videos, animaciones, textos y audio, que vistos de manera conjunta, crean un ambiente agradable para los alumnos, los cuales mostraron curiosidad y mayor seguridad al momento de utilizar el software.

Cuando el sistema se probó, los estudiantes lograron mayor comprensión de los temas al asociar conceptos a una imagen o sonido. Además, la constante interacción con la computadora los mantuvo motivados y al mismo tiempo, los invitó a vivir una experiencia más participativa en la construcción de sus procesos de enseñanza aprendizaje, navegando a través de la aplicación, donde pudieron observar y analizar ejemplos que muchas veces no pueden ser representados en el pizarrón. Por lo tanto, SECUMATIC ofrece gráficos tridimensionales para representar aquellos conceptos y figuras de índole abstracto, facilitando así la comprensión y mayor retención de éstos.

En este sentido, los profesores también se ven beneficiados porque SECUMATIC sirve como una guía para impartir los temas y al mismo tiempo pueden explicar los ejemplos ya diseñados en el sistema y no perder tiempo dibujándolos en el pizarrón.

Como apoyo adicional para los profesores, se diseñaron dos módulos, uno destinado a la evaluación, donde pueden diseñar o generar sus exámenes y otro, donde se lleva el control de alumnos, profesores y grupos.

Cuando se requiere desarrollar un sistema, siempre es necesario hacer uso de alguna metodología que la ingeniería de software nos proporciona, ya que todo el desarrollo se realiza mediante etapas. De esta manera se garantiza que nuestro software es flexible, de bajo costo y confiable.

El resultado de haber utilizado el modelo en espiral fue exitoso, pues se eligió por ser uno de los más completos al combinar los paradigmas del ciclo de vida clásico y construcción de prototipos. Por lo tanto, durante el diseño y desarrollo de SECUMATIC, cubrimos las etapas genéricas del modelo y de manera simultánea, fuimos evaluando los riesgos para tener mayor control en el crecimiento de la espiral.

Este modelo nos facilitó el desarrollo del software, ya que el prototipo inicial fue la base para continuar a través de las vueltas sucesivas, modificando o eliminando elementos, de acuerdo a los requerimientos del cliente en cada evaluación. De esta manera, en cada vuelta de la espiral se obtuvieron versiones cada vez más depuradas y completas hasta llegar al producto final, el cual cumple con todos los requisitos del cliente.

Consideramos que una de las etapas más importantes al utilizar el modelo en espiral, es la de análisis de riesgo, porque aquí es donde establecimos las limitaciones y alcances de nuestro sistema. Además, requerimos de gran habilidad para valorar los posibles riesgos existentes, con esto evitamos un crecimiento incontrolado de los giros a la espiral.

Tambien, definimos lo que realmente era factible desarrollar, sin perder de vista los requisitos del cliente.

Otro aspecto importante fue sin duda, la selección de la herramienta de programación. Visual Basic 6.0, nos permitió integrar elementos multimedia, que fueron necesarios para lograr un software interactivo y amigable, ya que gran parte de la programación se realizó visualmente. Es

decir, durante el tiempo de diseño tuvimos la opción de ver la forma en que el programa se vería al ejecutarse. Esta fue una gran ventaja sobre otros lenguajes de programación, porque pudimos cambiar y experimentar el diseño de las formas hasta quedar satisfechos con los colores, proporciones e imágenes, seguiendo los requerimientos establecidos por el cliente.

La parte inicial, para desarrollar el sistema fue conocer y entender el pensamiento de los jóvenes, para lo cual recurrimos a las teorías psicopedagógicas de Piaget, Vigotsky y Ausubel en las que pudimos encontrar aspectos que deteminan la construcción de las estructuras mentales en el individuo y su medio que le rodea, así como las condiciones que deben existir para que se lleve a cabo el proceso de enseñanza-aprendizaje. El profesor juega un papel muy importante, ya que les sirve de guía a los alumnos, en la construcción de sus estructuras mentales.

Los profesores deben conocer a fondo las ideas e inquietudes de los alumnos, para emplear estrategias adecuadas de aprendizaje. También organizar los contenidos que se van a enseñar, partiendo de los conocimientos previos y finalmente, despertar la motivación.

Dichas teorías nos mostraron una característica de nuestros clientes que no es palpable a simple vista y que sin embargo, es determinante para la aceptación del software. Porque el diseño está basado en los gustos y preferencias de éstos.

Los contenidos temáticos se presentaron de manera organizada, partiendo de lo general a lo particular, con un lenguaje matemático muy sencillo. Todo esto con la finalidad de que la enseñanza de las matemáticas a través de SECUMATIC ofrezca una experiencia agradable.

Se logró que existiera compatibilidad entre las herramientas seleccionadas para desarrollar este sistema, tanto en la parte de diseño como en la de programación, ya que explotando al máximo las características de cada uno de éstas, se logró obtener un código con menos líneas de programación, lo que nos facilitó en gran medida el mantenimiento.

A lo largo del desarrollo de este proyecto, conocimos lo que es aplicar la ingeniería en la vida práctica con más profundidad, ya que ese es el objetivo principal de nuestra profesión.

Además, comprendimos la importancia que tiene en el ámbito educativo, el poder diseñar, crear e implementar herramientas de este tipo, que sirven principalmente como soporte de comunicación y permiten complementar los conocimientos adquiridos en el aula.

El software SECUMATIC cubre los objetivos propuestos inicialmente, ya que durante las pruebas realizadas, los alumnos realmente muestran gran interés por el estudio de las matemáticas, teniendo mayor participación al momento de ser ellos mismos los que controlan el ritmo de aprendizaje y el hecho de interactuar con el sistema les crea una experiencia más interesante y agradable. Así, con el uso de este software educativo, las matemáticas lejos de ser complicadas y aburridas, se presentan de una forma más sencilla, ofreciendo una forma diferente de aprendizaje al tradicional.

La capacidad de poder conjuntar varias disciplinas que son indispensables al momento de desarrollar un software interactivo, nos dio la oportunidad de ampliar nuestro conocimiento y experiencia como profesionistas, porque tuvimos que involucrarnos en cada una de ellas y hacer un análisis exhaustivo para poder cubrirlas lo mejor posible e integrarlas con la ingeniería.

Es aquí donde la computación cobra mayor importancia, ya que puede ser aplicada a cualquier área, teniendo siempre presente, que se busca resolver un problema o una necesidad. Pero no debemos olvidar que las bases teóricas son fundamentales para desarrollar cualquier proyecto.

En nuestro caso, es muy grato ayudar a resolver una necesidad educativa mediante SECUMATIC, ya que muchos alumnos se verán beneficiados al ampliar sus conocimientos en matemáticas a través de este software y los profesores tendrán mejor control sobre sus grupos, detectando de manera directa las deficiencias de cada uno de ellos y a nosotros como ingenieros, nos da la satisfacción de ofrecer un servicio en beneficio de la sociedad.

Bibliografía

Guzmán Jesús, Carlos
Implicaciones educativas de seis teorías psicológicas.
El cognoscitivismo
UNAM, CONALTE., 1994
México.

Secretaría de Educación Pública
Libro para el maestro, 1994.

Esquivel Granados, Leticia
Software Identidad Nacional.Letysoft
UPN-SEP, 1996
México.

Pressman, Roger S
Ingeniería de Software. Un enfoque práctico
Mc.Graw. Hill
5ª Edición, 2002
México.

Bauer, Friedrich L
Software Engineering,
North Holland Publishing Co., 1993,
Amsterdam.

Shooman, M. L
Software Engineering
Mc Graw Hill, 1990
New York.

Fairley, Richard
Ingeniería de Software
Mc Graw-Hill, 1991
México.

Cruz, Alejandro
Evolución de las Culturas Mesoamericanas, 2000
México.

Zaldivar Zamorategui, Orlando
Apuntes de Ingeniería de Programación
UNAM F.I.;Edición 2000 México.

Vaughan, Tay
Todo el poder de la Multimedia
Mc Graw Hill; 2ª Edición; 1994
México.

Feldman, Roberts
Psicología con aplicaciones de habla hispana
3ra. Edición, Mc Graw Hill, 1999
México.

Ferrer, Eulalio.
Los Lenguajes del color
1ra. Edición, Fondo de cultura económica, 1999
México.

Perry Greg;
Aprendiendo Visual Basic 6 en 24 horas
Prentice Hall; 2000
México.

http://redescolar.ilce.edu.mx/ .