UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA TUTORIAL SOBRE LAS ASIGNATURAS OPTATIVAS QUE INTEGRAN EL PLAN DE ESTUDIOS DE LA CARRERA DE INGENIERO EN COMPUTACION

T E S I S

PARA OBTENER EL TITULO PROFESIONAL DE

INGENIERO EN COMPUTACION

P R E S E N T A N :

MARIA EUGENIA PEREZ APARICIO

LEOPOLDO NIEVES JAVIER

DIRECTOR DE TESIS:

ING. ORLANDO ZALDIVAR ZAMORATEGUI

1999

Dedicatorias

A Dios que siempre estuvo conmigo.

A mis padres y hermanos por todo el apoyo que me han brindado para alcanzar todas mis metas.

Agradecimientos

A Dios por darme la gracia de la vida y el don de la sabiduría.

A mis padres por haberme apoyado en todo momento.

A mis compañeros y maestros que me hicieron la vida difícil porque gracias a ellos aprendí a superar los obstáculos y luchar para alcanzar mis metas.

A todas aquellas personas que de una u otra manera estuvieron involucradas en la realización de este trabajo y que sin su apoyo nada se hubiera logrado.

Especialmente a mi familia, a la memoria de mi tía Meche, a Josefina Rosales que nos ayudó durante la realización del seminario, a mis compañeras de danza: Mariana Cuevas y Bibiana Ornelas, a mis compañeros y amigos: Leopoldo Nieves, Miguel Angel Torres y Andrés Santoyo.

Maru

Dedicatorias

A mi Dios, a quien le debo todo, como una pequeña retribución por lo que me ha dado a largo de mi existencia.

A mi familia como un homenaje por ser no sólo mi apoyo en todos los aspectos de mi vida, sino por ser parte de mi fuerza y muchas veces mi inspiración para ver siempre hacia delante.

Agradecimientos

A Dios, por brindarme la oportunidad de haber concluido una etapa importante en mi vida.

A mis padres, por su esfuerzo y a toda mi familia por su apoyo infinito, que me ha permitido salir siempre adelante.

Un agradecimiento especial a Josefina, por su ayuda durante la realización del seminario.

A *Maru, por la oportunidad de trabajar a su lado y por haber compartido no sólo este* trabajo, sino mucho más.

A todos mis amigos y compañeros de la Facultad, que me han apoyado no sólo para la elaboración de esta tesis, sino en toda mi carrera profesional.

Polo

ÍNDICE TÉMATICO

INTRODUCCIÓN

Durante el desarrollo de la carrera cursada, Ingeniería en Computación, llamó poderosamente nuestra atención el hecho de que en la Universidad Nacional Autónoma de México, y especialmente dentro de la Facultad de Ingeniería, donde se encuentran aquellas carreras cuyo campo de acción se enfoca a la solución de problemas y la innovación con proyectos que faciliten la vida del hombre, no existiera un sistema tutorial sobre las asignaturas optativas que integran el plan de estudios de la carrera de Ingeniero en Computación. Por tal motivo, hemos optado por la investigación y realización de este trabajo el cual tiene como objetivo principal ofrecer a los alumnos que cursan la carrera un instrumento que les sirva de orientación, cuando llega el momento de elegir las asignaturas optativas y a la vez, que se cuente con un elemento que facilite la comprensión y entendimiento de las inquietudes y dudas que puedan surgir al cursar una u otra asignatura optativa, al mismo tiempo que funciona como guía para los profesores y académicos que imparten estas asignaturas.

Dado que a los estudiantes nos surgen muchas dudas de cómo poder desarrollarnos en alguna área de la rama de la computación de manera adecuada y convincente, es necesario contar con una ayuda al momento de seleccionar las asignaturas optativas, ya que del aprendizaje y experiencia que obtengamos al cursar cualquiera de ellas, dependerá el éxito y satisfacción como profesionales de la computación.

Es bajo las hipótesis anteriores, que exponemos el presente trabajo, el cual pretende explicar el desarrollo del sistema elaborado para dar a conocer el contenido de las asignaturas optativas que contiene el plan de estudios de la carrera de Ingeniería en Computación de la Facultad de Ingeniería de la U.N.A.M.

Este trabajo se divide en tres partes fundamentales: en la primera parte se analiza y explica la teoría que involucra la ingeniería de software para la realización de cualquier producto de software. La segunda parte contiene la justificación de cada una de las actividades desarrolladas para la creación de este sistema tutorial. Finalmente, la tercera parte expone un recorrido por el sistema elaborado para la mejor comprensión de su funcionamiento.

Detallando cada una de estas partes encontramos el siguiente desglose:

PARTE I
- Marco Teórico
- Metodologías Adaptadas para el Desarrollo de Software Educativo
- Metodología Propuesta para el Desarrollo de Sistemas Tutoriales

PARTE II
- Aplicación de la Metodología

PARTE III
- Desarrollo de cada Módulo del Sistema

A continuación definiremos brevemente el contenido de cada capítulo.

El primer capítulo, *Marco Teórico*, contiene una breve introducción sobre los tipos de software que existen y su clasificación. Se analizan algunos de los factores que influyen en la calidad y productividad de un proyecto de software, así como las generalidades al realizar software interactivo. Posteriormente, dentro del mismo capítulo, se hace una recopilación de las principales metodologías que existen para el desarrollo de software, la cual incluye las características más importantes, ventajas y desventajas de cada método.

El propósito de incluir este capítulo consiste en proporcionar un panorama general de lo que existe en el área de ingeniería de software para el desarrollo de sistemas.

El segundo capítulo, *Herramientas para el Desarrollo de Software,* establece una descripción de las herramientas CASE utilizadas para el desarrollo de software. Con la inclusión de este capítulo se pretende complementar el marco teórico sobre el desarrollo de sistemas. Debido a la importancia de las herramientas CASE en la construcción de programas, este tema resulta de gran interés para entender la teoría del desarrollo de sistemas.

El tercer capítulo, *Metodologías Adaptadas para el Desarrollo de Software Educativo,* incluye la descripción de los métodos particulares para el desarrollo de software educativo, esto con la finalidad de tener un panorama más objetivo del diseño de estos programas. Se explica también, la diferencia entre la programación estructurada y la programación orientada a los objetos, así como la influencia que tiene el usar uno u otro tipo de programación como método para desarrollo de software. Para concluir este capítulo, se hace un análisis de cada metodología investigada evaluando las ventajas y desventajas que presenta cada una de ellas.

Finaliza esta primera parte con el cuarto capítulo, *Metodología Propuesta para el Desarrollo de Sistemas Tutoriales,* el cual define las etapas de la metodología que proponemos para el desarrollo de sistemas tutoriales y las actividades a desarrollar en cada una de ellas.

La segunda parte contiene un único capítulo, capítulo cinco, *Aplicación de la Metodología,* que comprende la explicación minuciosa de cada actividad realizada durante la aplicación de la metodología propuesta para el desarrollo de este sistema. Se describe todo el proceso utilizado en cada iteración y se escriben las conclusiones obtenidas al finalizar esas iteraciones. La idea de este capítulo es mostrar cómo se aplicó la metodología, ilustrando además, la importancia de seguir un método en el desarrollo de software estructurado.

La forma de mostrar la aplicación de la teoría con la práctica es conjuntando ambas partes, por ello, en la tercera parte se tiene un capítulo exclusivo para esta tarea, el capítulo sexto, *Desarrollo de los Módulos del Sistema* donde se presentan ejemplos de las pantallas del sistema así como, la estructura interna del mismo. Se incluye también, una explicación de los módulos principales que componen el sistema. Puesto que el sistema fue desarrollado por módulos, al ejemplificar alguno de los más importantes, se muestra el

funcionamiento integral. Esta explicación de módulos abarca la creación y diseño de diagramas, ejemplos del funcionamiento, pantallas que ilustran el contenido del sistema, sugerencias para facilitar la navegación, etc. En otras palabras, se incluye un pequeño manual de operación.

Para finalizar este trabajo, se expresan las conclusiones que obtuvimos después del desarrollo del sistema. En ellas se reflejan los resultados que se lograron, así como la experiencia personal de cada unos de los integrantes del equipo de desarrollo. Al terminar las conclusiones, el lector encontrará la bibliografía consultada.

Es así, como damos a conocer nuestro trabajo.

PARTE I.

CAPITULO I. MARCO TEÓRICO

I.1 Categorías de acuerdo con el tamaño

I.2 El ciclo de vida clásico

I.3 Construcción de prototipos

I.4 El modelo en espiral

I.5 Técnicas de cuarta generación

El desarrollo de programas presenta una serie de retos y problemas en su diseño debido a que existen procesos de producción de software de muy diversa complejidad.

I.1 Categorías de acuerdo con el tamaño

El tamaño de un proyecto es un factor importante que determina el nivel de control administrativo (crecimiento y requerimientos físicos, como computadoras, impresoras, redes, etc., el número de usuarios) y el tipo de herramientas y técnicas necesarias para la programación del proyecto.

Categoría	Número de programadores	Duración	Tamaño del producto
Trivial	1	1-4 semanas	500 líneas de código
Pequeño	1	1-6 meses	1K - 2K
Mediano	2-5	1-2 años	5K - 50 K
Grande	5-20	2-3 años	50k - 100K
Muy grande	100-1000	4-5 años	1M
Extremadamente grande	2000-5000	5-10 años	1M - 10M

Tabla 1 Comparativa tamaño-recursos de un proyecto de software

Analizando las categorías aquí presentadas, podemos decir que el sistema tutorial que se pretende realizar se encuentra dentro del tamaño mediano. De acuerdo con Richard Farley[1], las características de estos sistemas son las siguientes:

"Estos programas tienen poca interacción con otros programas. Entre estos proyectos se pueden considerar ensambladores, compiladores, sistemas pequeños de manejo de información, sistemas de inventarios y aplicaciones de control de procesos.

El desarrollo de proyectos medianos exige la interacción entre programadores y la comunicación con los usuarios; de ahí que se necesite cierta formalidad en la planeación, documentación y revisión del proyecto".

El grado de formalidad y la cantidad de tiempo asignada, varía de acuerdo con el tamaño y complejidad del producto que se desarrollará.

La calidad del producto y la productividad del programador puede elevarse al mejorar los procesos necesarios para el desarrollo y mantenimiento de los productos. Algunos factores que influyen sobre la calidad y productividad se muestran en la Tabla 1.

Con base en la Tabla 1, podemos deducir que, si existe una buena *comunicación* entre los elementos del grupo de trabajo se obtendrá, sin duda, un producto de excelente calidad, puesto que cada persona podrá expresar libremente sus requerimientos y el

[1] Fairley, Richard: "Ingeniería de Software."; Editorial McGraw Hill; México; 1987

programador y/o diseñador tendrán una perspectiva más amplia del problema con la cual evaluar y proponer las mejores alternativas.

Una característica importante que permite mejorar considerablemente la productividad es la correcta selección del enfoque sistemático. Pero, ¿qué significa esto? Quiere decir que si nosotros elegimos un buen método para desarrollar el software, obtendremos casi automáticamente un producto de calidad que contendrá todos los elementos para su buen funcionamiento.

Capacidad individual	Entendimiento del problema
Comunicación del grupo	Estabilidad de los requerimientos
Complejidad del producto	Habilidades necesarias
Notación adecuada	Facilidades y recursos
Enfoque sistemático	Entrenamiento adecuado
Cambio de control	Habilidades administrativas
Nivel tecnológico	Metas adecuadas
Confiabilidad requerida	Expectativas crecientes
Tiempo disponible	

Tabla 2. Características y habilidades necesarias para un proyecto de software

Dentro del mismo cuadro vemos que es necesario un *entrenamiento adecuado*, esto es complementar el desarrollo del proyecto mediante una etapa de capacitación para las personas que utilizarán el software una vez que éste sea terminado.

En el aspecto de productividad del software y en el desarrollo del mismo es importante establecer *metas adecuadas*. Del lado del usuario, al mencionar los requerimientos se debe procurar que éstos sean acordes a su necesidad y que verdaderamente solucionen su problema o mejoren su línea de trabajo, de esta manera, sin duda, nuestra meta (el producto de software) será apto para su actividad.

Por parte del analista/programador, debe tenerse cuidado de analizar bien la información que proporciona el cliente y establecer metas reales en cuanto a la estimación de tiempos, programación del software (¿qué se puede y debe hacerse y qué no?), entrega de documentos (manuales técnico y para el usuario) y mantenimiento del software, soporte técnico así como las *expectativas a futuro* sobre lo que pueda suceder, como crecimiento de la empresa, actualización de software (bases de datos, sistema operativo, etc.) y/o de hardware (instalación de red, cambio de equipo, etc.), pérdida de información en caso de siniestros, entre otros.

El objetivo principal de este proyecto consiste en crear un sistema tutorial que sirva de apoyo a los alumnos de la carrera de Ingeniería en Computación para la elección de sus asignaturas optativas. Para lo cual primero debemos definir qué es un sistema tutorial:

Un sistema tutorial es un software de información interactiva con cualquier tipo de usuario que desee aprender de manera autodidacta.

En este tipo de sistemas, la computadora es utilizada de forma tutorial con el objetivo de que sea éste el que proporcione la transmisión de información a cada uno de los alumnos en una determinada área de conocimiento.

La mayoría de los programas tutoriales se basan en modelos de diálogos cerrados en los cuales la computadora actúa presentando una determinada información, a partir de la cual realiza una serie de preguntas, cada una de ellas con posibles opciones de respuesta. En función de la respuesta obtenida, la computadora da más información o realiza más preguntas sobre el mismo tema hasta conseguir que el educando responda de la forma idónea.

La base de los sistemas tutoriales radica en el diálogo entre el alumno y la computadora. La respuesta que el alumno puede realizar es, en algunos programas, una respuesta abierta. En este caso, el programador tiene que anticiparse a la mayor parte de las respuestas posible del alumno para poder llevar a cabo un diálogo eficaz y fluido. Ello resulta muy difícil de realizar y requiere más espacio de memoria del que poseen la mayor parte de los microordenadores domésticos. Por esta razón, la mayoría de los diálogos adoptan el modelo cerrado a través de un formato de elección múltiple.

Los programas tutoriales suelen estar confeccionados previamente, pero existe también la posibilidad de que el profesor genere su propio material a través de la utilización de lenguajes de autor.

Según algunos artículos sobre educación enfocados al aprendizaje por medio de las computadoras, existen algunas estrategias propuestas por programadores desarrolladores de software; en las cuales afirman que "El *software interactivo* es un nuevo medio que necesita tiempo para madurar (Sherwood). Llevó mucho tiempo al cine madurar, alejarse de la realización de simples producciones en escenarios de filmación explotando cada uno de los diferentes ángulos de esos escenarios. El mismo recorrido se espera en el software educativo que difiere de todos los medios conocidos actualmente, ya que éste presenta dos caminos; es dinámico, modificable y requiere de entradas activas por parte del usuario. El único medio que hasta ahora conocemos como medio interactivo es la conversación humana. El modelo de un software interactivo tiene la limitante de que una computadora no es capaz de operaciones tan complejas como la interacción verbal o el mantener una conversación dinámica utilizando únicamente representaciones visuales. La computadora requiere entradas por parte del usuario. Explotar este carácter interactivo es una buena razón para utilizar una computadora como instructor. Pero la interacción crea un conjunto de problemas de diseño que difieren significativamente de aquellos medios no interactivos. Por ejemplo, si bien los diseñadores gráficos conocen mucho acerca de cómo proyectar un anuncio para una revista, ellos no necesitan preocuparse de cómo hacer la exhibición dinámica o interactiva. En contraste, una exhibición por computadora es frecuentemente construida cuidando el tiempo de respuesta a las entradas, en lugar de aparecer todo de una sola mirada en una página de revista. Quizás la analogía más cercana esté dada por los videos informativos, los cuales son construidos para ser exhibidos de manera dinámica. Sin embargo, esta secuencia es estrictamente lineal y no hay oportunidad para que el usuario maneje las entradas o tenga el control. En contraparte, una demostración por

computadora no sólo puede verse más atractiva, también debe explicar o mejorar, insinuar, cómo el lector debe interactuar (por ejemplo, tocar una porción específica de la pantalla, o escoger una opción del menú). De hecho, desde el contenido actual de la demostración, son importantes las entradas del usuario; el que desarrolla el programa puede no conocer totalmente que aparece en la exhibición"[2].

Se ha definido lo que es un sistema tutorial (un software de información interactiva con un usuario que desea aprender utilizando la computadora), así como algunas de sus características (principalmente que un tutorial debe ser autodidacta y establecer una relación hombre - máquina de enseñanza - aprendizaje), el siguiente paso es elegir una metodología adecuada para su diseño.

Investigando en material de desarrollo de software, se logró comprobar que existen *algunas metodologías propias para desarrollar sistemas tutoriales. Sin embargo, éstas* son muy variantes puesto que no es igual construir un sistema tutorial para un análisis clínico que un sistema que ayude a los niños a aprender geografía, por lo cual, consideramos necesario proponer una metodología basada en las que ya existen para desarrollar el software con las características que se requieren para aprender y decidir qué asignaturas optativas puede elegir un alumno de los últimos semestres de la carrera de Ingeniería en Computación.

Para contar con un apoyo en la construcción de este sistema tutorial y en la elección de una metodología para desarrollarlo, se llevó a cabo la tarea de buscar información sobre los métodos utilizados para desarrollar software. Se conoció así que, los cuatro métodos más populares en el desarrollo de programas estructurados, según Roger S. Pressman[3] son los siguientes:

1. El Ciclo de Vida Clásico
2. Construcción de Prototipos
3. El Modelo en Espiral
4. Técnicas de Cuarta Generación

I.2 El Ciclo de Vida Clásico

Este método es el más antiguo y el más ampliamente usado en ingeniería de software. Se le conoce también como "modelo en cascada" dada la similitud con el mundo real (el flujo de agua de una cascada es de arriba hacia abajo y no se ha visto lo contrario). El modelo del ciclo de vida clásico exige un enfoque sistemático es decir, que cada una de sus partes se relacionan entre sí para lograr un fin común -un buen software-, y secuencial (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).

---

[2] Gros, Begoña; "Diseños y programas educativos. Pautas pedagógicas para la elaboración de software"; Editorial Ariel; España; 1997.

[3] Pressman, Roger; "Ingeniería de Software"; Editorial: McGraw-Hill ; México; 1993.

Se consideran como etapas para este modelo, según Roger S. Pressman, las siguientes :

- Ingeniería del sistema
- Análisis
- Diseño
- Codificación
- Prueba
- Mantenimiento

Ingeniería. En esta etapa 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 (señales de antenas, cámaras de vídeo, robots, etc.).

Análisis. Los requisitos, tanto del sistema como del software, se documentan y se revisan con el cliente durante esta etapa.

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 comience la codificación.
Codificación. Se traduce el diseño en forma legible para la máquina.

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 que se requieren.

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 del Ciclo de vida 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:

- *No siempre se sigue el modelo.*

Los proyectos reales raramente siguen la secuencia que el modelo propone. Siempre ocurre la iteración y crea problemas en la aplicación de esta metodología.

- *El cliente no siempre establece explícitamente los requisitos.*

Es frecuentemente complicado en un principio para el cliente establecer todos sus requerimientos explícitamente. El método del ciclo de vida requiere esta definición de requisitos y tiene dificultad de acomodar la natural incertidumbre que existe en el inicio de muchos proyectos.

- El cliente debe tener paciencia

Una versión desarrollada del programa no estará disponible hasta mucho tiempo después de haberse iniciado el proyecto. Una falla o error no detectado, sino hasta que se revise este programa desarrollado, puede ser desastroso.

I.3 Construcción de prototipos

La construcción de prototipos presenta un diseño rápido sobre la representación de los aspectos del *software visibles al usuario. Puede realizarse en una* de las siguientes tres formas :

1. Un prototipo en papel o un modelo basado en computadora que describa la interacción hombre-máquina.

2. Un prototipo que implemente algunos subconjuntos de funciones requeridas en el programa.

3. 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.

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 iterativo* en el que el prototipo es *mejorado una* y otra vez *hasta que satisfaga las necesidades del cliente.*

Las ventajas que presenta este método son:

- *El cliente puede ver y en ocasiones trabajar* (utilizar el prototipo para una tarea sencilla que se ha implementado en el mismo para mostrar un ejemplo de sistema) 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 producto de software pueden detectarse con mayor facilidad las características faltantes, proponerse nuevas ideas, mejorar el diseño; en general, ampliar la perspectiva del proyecto que se quiere desarrollar.

Las desventajas de este método son:

- El cliente ve una versión del sistema en la cual no se considera calidad ni mantenimiento de software a largo plazo. Cuando se le informa que el producto debe ser reconstruido, el cliente cree que presenta fallas y demanda que unas pocas modificaciones sean aplicadas para hacer del prototipo, un producto trabajando.

- El desarrollo frecuentemente implica comprometerse para obtener un prototipo trabajando rápidamente. Un sistema operativo o lenguaje de programación inapropiado puede ser usado simplemente porque se conoce y está disponible; un algoritmo ineficiente puede ser implementado simplemente para demostrar capacidad. Después de un tiempo, el desarrollo puede empezar a familiarizarse con estas elecciones y se olvidan las razones por las que eran inapropiados. La opción menos ideal ha empezado a ser una parte integral del sistema.

I.4 El modelo en espiral

El modelo en espiral reúne las mejores características tanto del ciclo de vida clásico como las de creación de prototipos, añadiendo un nuevo elemento: el análisis de riesgo.

De esta manera, en el modelo se definen cuatro actividades principales:

1. *Planificación* en la cual se determinan objetivos, alternativas y restricciones. Al igual que en los métodos anteriores, esta actividad involucra la recolección de requisitos. El cliente debe exponer sus requerimientos de acuerdo con sus necesidades y con base en la información recopilada se hace el análisis y diseño del software a realizar.

2. *Análisis de riesgo* en el cual 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 desarrollar el software; así, se pueden detectar los inconvenientes y/o limitantes 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.

3. *Ingeniería* en la cual se hace un desarrollo del producto de "siguiente nivel". ¿A qué se refiere Pressman cuando nos dice "siguiente nivel"? . Si observamos la Figura 1, en relación con la espiral, tenemos una curva que se va cerrando o abriendo (según la espiral) a partir de un centro predefinido. 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 que se considera en

un nivel superior puesto que se ha avanzado y seguramente se ha rodeado en su totalidad la primera curva de la espiral.

4. *Evaluación del cliente* en la cual se hace una valoración 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 es lo que no funciona 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.

La figura 1 nos resume y a la vez nos ayuda a comprender mejor el funcionamiento del modelo en espiral.

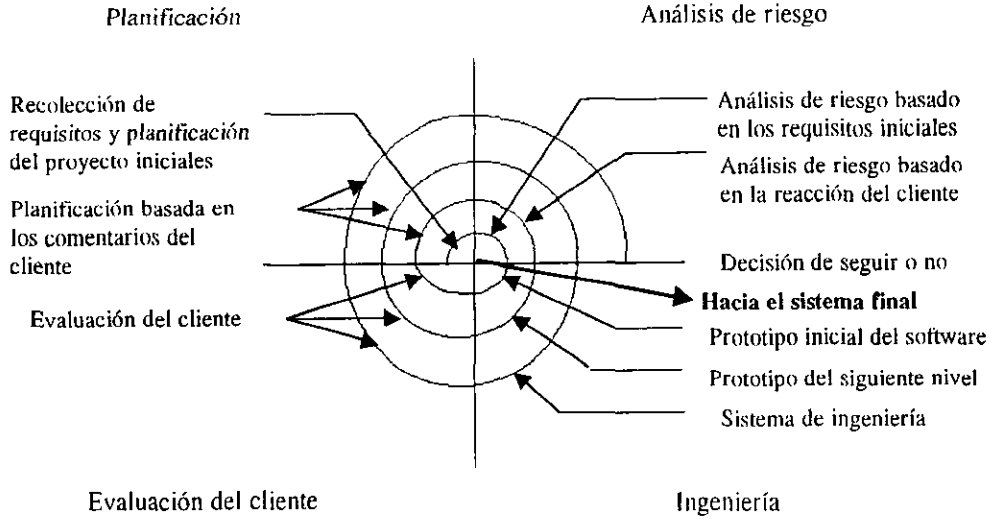

Figura 1. El Modelo en Espiral

Al conjuntar las mejores características de la metodología del ciclo de vida y construcción de prototipos, el modelo en espiral resulta ser el método más completo de los tres que hasta aquí hemos mencionado.

Las ventajas del modelo 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.

Desventajas que presenta este método:

- 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.

I.5 Técnicas de cuarta generación

Algunas personas dentro del mundo de la computación confieren designaciones de "generación" a los lenguajes de máquina, ensambladores y de alto nivel. Apuntan que el cambio de lenguajes de máquina de la *primera generación* y los lenguajes ensambladores de la *segunda generación* produjeron una mejora de la productividad de los programadores de aproximadamente de siete a uno. Ocurrió una mejoría similar cuando se introdujeron los lenguajes de alto nivel (*tercera generación*). Y ahora varios proveedores de programas han producido varias herramientas de desarrollo de aplicaciones que podrían mejorar más todavía la productividad. A estas herramientas se les llama a menudo y en forma colectiva *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.

Un lenguaje de alto nivel es un lenguaje de procedimientos, es decir el programador debe detallar los pasos de los procedimientos de proceso que se requieren para lograr un resultado deseado. En cambio, un lenguaje de cuarta generación es un lenguaje sin procedimientos es decir, permite a los usuarios especificar cuál debe ser la salida sin describir todos los detalles acerca de cómo deben manipularse los datos para producir ese resultado.

Las técnicas de cuarta generación facilitan el desarrollo de software puesto que incluyen todas o algunas de las siguientes herramientas: lenguajes no procedimentales 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. Todas estas herramientas están disponibles, pero sólo para ámbitos de aplicación muy específicos.

Esta técnica al igual que las anteriores comienza con la recolección de requisitos para proponer a continuación un prototipo (hay que considerar que el cliente quizá no tenga aún totalmente definidos sus requerimientos y el prototipo podría confundirlo). Posterior a

este paso se realiza una estrategia de diseño para después implementar el sistema y realizar las pruebas que sean necesarias.

Este modelo se orienta hacia la posibilidad de especificar el software a un nivel más próximo al lenguaje natural.

Sus características son:

- Con muy pocas excepciones el dominio de aplicación actual para lenguajes de cuarta generación es limitado a aplicaciones de sistemas de información de negocios, específicamente, análisis y reportes de información a grandes bases de datos. Actualmente, las técnicas de cuarta generación han sido usadas en productos de ingeniería y aplicaciones del área de sistemas.

- Los datos preliminares recolectados de compañías que usan lenguajes de cuarta generación indican que el tiempo requerido para producir software disminuye notablemente para pequeñas y medianas aplicaciones y que la cantidad de diseño y análisis para pequeñas aplicaciones es también reducida.

- Aunque, el uso de técnicas de cuarta generación para el desarrollo de grandes aplicaciones demanda mucho más tiempo para análisis, diseño y evaluación (actividades de la ingeniería de software), se reduce substancialmente el tiempo al depurar los archivos mediante la eliminación de código.

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.

CAPITULO II. HERRAMIENTAS PARA EL DESARROLLO DE SOFTWARE

II.1 Metodología CASE

Existen también otras herramientas que se utilizan para el desarrollo de software. Estas nacen a partir de una combinación de paradigmas que toman las características más importantes de cada una de las metodologías anteriormente mencionadas. Lo que se busca al realizar esta combinación de ideas es encontrar una forma sencilla, eficiente e ingenieril que permita la creación de software, rápida y apta a las necesidades de los diferentes clientes. Una de estas herramientas es la Metodología CASE (Computer Aided Software Engineering) cuyas técnicas han alcanzado un lugar importante en el desarrollo de sistemas en la década de los 90's y de la cual se hablará a continuación.

II.1 Metodología CASE

¿ Qué es Ingeniería de Software Asistida por Computadora?[4]

Existen diferentes definiciones de CASE; la siguiente, es una definición muy amplia pero quizá una de las más acertadas:

CASE es el uso de la computadora como apoyo para un desarrollo de software.

Esta definición incluye todo tipo de ayuda de la computadora: dirección, administración y componentes técnicos de un proyecto de software.

¿Qué es una herramienta CASE?

En los primeros días cuando comenzó a escribirse software para computadora, se requirieron algunas herramientas como traductores, interpretadores, compiladores, ensambladores, etc., para realizar las tareas básicas durante el desarrollo del software. Como la demanda de software se incrementaba se requirieron herramientas CASE que fueran más complejas y poderosas. En particular, la interactividad y respuesta en tiempo real, obligó a tener mejores herramientas CASE que posteriormente se convirtieron en editores, debuggers (correctores de errores), analizadores de código, etc.

Dos avances importantes permitieron dar el salto a nuevas y sofisticadas herramientas CASE. Estos adelantos fueron:

1. El hardware para computadora llegó a ser más poderoso con la introducción de las estaciones de trabajo y las computadoras personales a principios de 1980. Estas máquinas tenían una gran capacidad de memoria, procesadores de mayor velocidad y la habilidad de realizar un mejor mapeo de los bits hacia el monitor (producir mejores gráficas). Estas tecnologías permitieron un mejor despliegue de mapas, diagramas y modelos gráficos a la vez que inducieron la introducción de las Interfaces Gráficas de Usuario (GUI).

2. Las investigaciones en el área de desarrollo de software dieron como resultado herramientas CASE que soportaban un gran numero de metodologías; esto, debido a que se adherían paso a paso a los métodos de desarrollo de software.

---

[4] Moore, Aubrey: "Software engineering"; Sheffield Hallam University; EEUU; 1996

Las dos metodologías principales para desarrollo de software donde se emplean herramientas CASE son la programación estructurada de Jackson y el método de Yourdon.

Definición de herramienta CASE:

Una herramienta CASE es un producto basado en la computadora encaminado a apoyar una o más actividades de la ingeniería de software sin llegar a realizar todo el desarrollo del software.

¿ Qué opinan los expertos sobre las herramientas CASE ? [4]

Ed Yourdon:

Las mejores herramientas CASE permiten el análisis de los requerimientos de un modelo o arquitectura de software mediante diagramas (cuentan con una herramienta para generar el código).
Algunos de sus elementos esenciales son:

- Flexibilidad para realizar los diagramas. El usuario puede ver todo lo que desee, lo cual le permite adecuar el diagrama a sus necesidades.
- Corrector de errores que refuerza la metodología de análisis y diseño orientado a objetos, OOA/OOD (por sus siglas en inglés).
- Precio bajo en cuanto a dependencia, pues se puede experimentar con la metodología sin temor de llegar a sujetarse de ella.
- La animación es algo deseable. La metodología CASE permite crear diagramas y simular la conducta del sistema con su ejecución.

Larry Constantine:

Una buena herramienta CASE es algo más que un paquete para dibujar diagramas. Actualmente, una buena herramienta debe ser capaz de trabajar con desarrollos reales que incluyan metodologías particulares. El promotor de esta metodología necesita un conocimiento profundo de las técnicas y procesos de modelado que consigue mejorar utilizando la herramienta CASE en vez de las otras metodologías. El diseño de interfaces con el usuario es un ejemplo de las mejoras con que cuenta la herramienta CASE.

Una herramienta CASE óptima permite al analista/programador realizar modificaciones discretamente entre vistas y modelos diferentes de un modelo preestablecido. El usuario puede diseñar un diagrama de flujo de datos con una notación y cambiarla en cualquier momento por otra totalmente diferente; también puede relacionar modelos mediante jerarquías de clase, comunicación de objetos, descomposición funcional de métodos y código en una forma tan sencilla como si navegará en hipertexto. Las mejores herramientas CASE soportan la interactividad pensando que los desarrollos y la evolución de los sistemas reales son no lineales e incluyen procesos experimentales (al contrario de los modelos de ciclo de vida que son lineales y limitados).

Peter Coad:

La herramienta CASE óptima, debe permitir analizar un objeto en una ventana y su código en otra; de esta manera, el usuario puede realizar modificaciones al código y actualizar simultáneamente al objeto como ocurre en las ventanas de C++.
Peter Coad dice: "He construido modelos con propósitos específicos de diseñar objetos y establecer dinámicas en sus escenarios. Mediante este experimento comprendo por qué el analista observa y filtra todos los componentes necesarios del modelo-objeto específico (dominio del problema, interacción hombre - máquina, administración de datos, manejador de tareas, interacción del sistema, etc.); los sujetos (agrupaciones de objetos que significan algo juntos), y cada escenario con sus objetos y responsabilidades específicas que cada uno de ellos implica.

Grady Booch:

La herramienta CASE ideal debe ayudar en la creación y visualización de arquitecturas (no sólo producir imágenes bonitas), debe vincular ambas, ingeniería de software e ingeniería inversa o reingeniería, así también, debe evitar un manejo tedioso en sistemas complejos a través de un amplio conocimiento semántico para revisar concientemente y corregir la arquitectura del sistema.

Estas características "obligatorias" incluyen un amplio conocimiento semántico de la notación y acoplamiento de otras herramientas vía APIs (Application Programmers Interface - Interfaz programada para aplicaciones), mecanismos como OLE (Object Linking and Embedding - Relación y acoplamiento de objetos), scripts o significados programables; debe soportar varios usuarios a la vez y tener la opción de expandirse para tareas complejas.

James Rumbaugh:

La transparencia es un elemento clave en una herramienta CASE. El mecanismo que utilice la herramienta permanece oculto ante el usuario; esto significa que el manejo directo de la notación debe ser tan sencillo como en una hoja de papel. Algunos programas de Windows, UNIX y Mac cuentan con esta cualidad dentro de sus programas y aplicaciones. La falta de transparencia ha sido motivo de que muchas herramientas CASE hayan quedado en el pasado.

Las características obligatorias en una herramienta CASE incluyen un buen balance entre simplicidad y poder. Esto es fácil de decir, quizá fácil de reconocer, pero difícil de lograr. Cuando una herramienta trata de hacer algo, por simple que sea, llega a ser complicado porque el usuario tiene que hacer muchas elecciones. Una buena herramienta facilita las cosas, no trata de automatizar cada caso, sólo busca automatizar las tareas más simples y representativas del proceso y ofrecer alguna alternativa para los trabajos difíciles y complicados.

Una buena herramienta sabe tomar decisiones correctas (defaults). Así, el usuario puede comenzar su trabajo omitiendo las elecciones triviales obteniendo los mismos resultados

que si hubiera seleccionado cada respuesta. Por ejemplo, un generador de código C++ sugiere un código eficaz sin necesidad de que el usuario indique todas las características del programa.

Algunas de las ventajas de las herramientas CASE son:

- Reducen la complejidad de la lógica de sistemas.
- Revisan opciones de diseño.
- Aceleran el proceso de desarrollo.
- Desarrollan opiniones.
- Integran soporte para la empresa.
- Aplican disciplina en la construcción del sistema.
- Proveen de una base para integración.
- Usan librerías modulares.

Desventajas que presentan las herramientas CASE

- La notación que emplean en ocasiones resulta confusa para el usuario.
- Se debe ser cuidadoso al elegir la herramienta CASE que se utilizará, pues algunas de ellas son diseñadas para software con enfoque estructurado y otras para software orientado a objetos.
- Algunas herramientas, especialmente las que toman decisiones, a veces limitan las aplicaciones.

CAPITULO III. METODOLOGÍAS ADAPTADAS PARA EL DESARROLLO DE SOFTWARE EDUCATIVO

III.1 Programación estructurada
III.2 Programación orientada a objetos
III.3 ¿Qué se dedujo del análisis realizado a cada método?

Otra de las metodologías existentes para la creación de software es el modelo sistemático de diseño instructivo, el cual tiene su origen en la ingeniería del software y ha sido adaptado a la producción de software educativo por autores como Gagné (1987) y Dick & Carey (1978). Este modelo, Figura 2, se basa en un proceso lineal constituido por cinco fases independientes: análisis, diseño, desarrollo, evaluación e implementación. Es muy similar al modelo del ciclo de vida clásico.

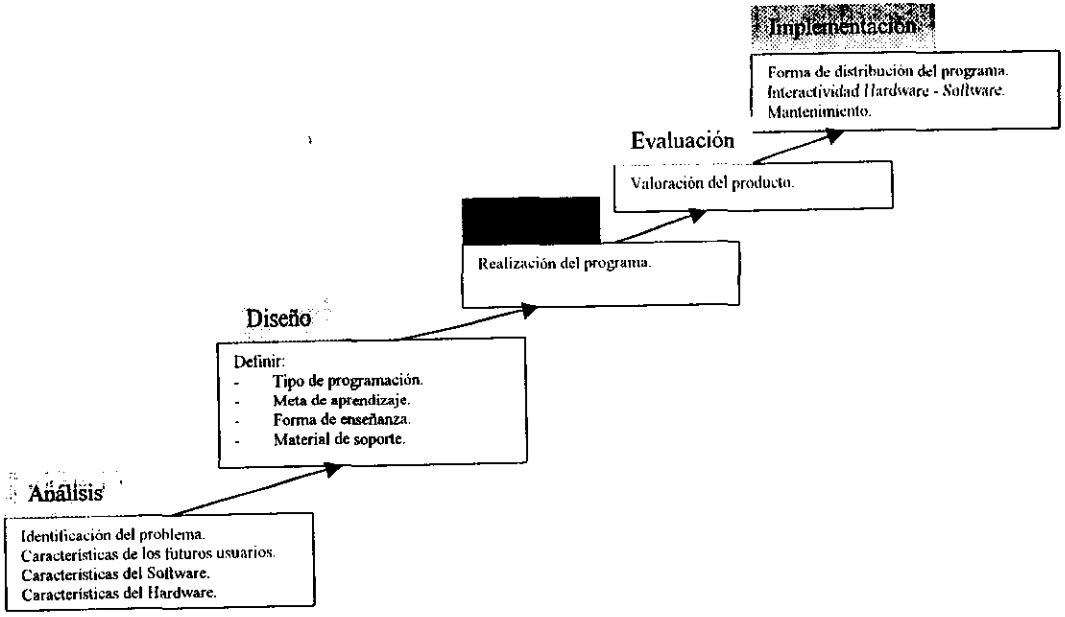

Figura 2. Modelo Lineal propuesto por Gagné Y Dick & Carey

La fase de *análisis* tiene por objeto el estudio de los resultados esperados y las condiciones de utilización. De este modo, algunas de las subtareas de esta primera fase de análisis son: la identificación de los problemas instructivos que se quieren solucionar, las características de los futuros usuarios del programa, el tipo de software que se desarrollará, en qué tipo de hardware y *con qué tipo de lenguaje* de programación o herramienta informática.

La segunda fase, el *diseño*, es un borrador de lo que será el producto final. Esta fase implica cinco subtareas:

1. La *elección del tipo de programa* a desarrollar (práctica y ejercitación, tutorial, etc.)
2. Los tipos de aprendizajes que se desea desarrollar.
3. El tipo de diseño instructivo que se va a adoptar en el programa.
4. La elaboración del guión del programa.
5. El diseño de los materiales de soporte (manuales, orientaciones para el profesor, el alumno, etc.).

La fase de *desarrollo* supone materializar en el programa informático el borrador efectuado en la fase de diseño. Se trata pues, de la fase técnica de informatización del contenido de los guiones.

Una vez desarrollado el producto, durante la fase de *evaluación* se efectúa una valoración del producto en sí. Esta evaluación tiene como objetivo comparar el *análisis efectuado* y el diseño con el producto final ya elaborado.

Finalmente, el producto debe ser implementado en un contexto real. Durante esta fase de *implementación* es necesario tener en cuenta la forma de distribución del programa, su *mantenimiento* y las posibles evaluaciones en contextos reales de utilización.

Las fases enunciadas son claras, sirven para diferenciar tareas, pero cuando se aplican se observa *cómo la separación* de las tareas y la linealidad es difícil de mantener. La práctica muestra cómo en un proceso de producción de software es difícil cerrar fases hasta que el producto está totalmente elaborado. Hay una revisión continua en función de los resultados que se van obteniendo.

III.1 Programación estructurada

Los fundamentos del diseño procedimental se forjaron a principios de los años sesenta y se consolidaron con el trabajo de Edgar Dijkstra y sus colegas. A finales de los años sesenta, Dijkstra y otros propusieron el uso de un conjunto de construcciones lógicas con las que podría formarse cualquier programa. Las construcciones reforzaban el "mantenimiento del dominio funcional". Esto es, cada construcción tenía una estructura lógica predecible; se entraba a ella por el principio y se salía por el fin.

Las construcciones son la secuencia, la condición y la repetición. La secuencia implementa los pasos de procesamiento esenciales de la especificación de cualquier algoritmo, la condición da *la posibilidad de seleccionar* un procedimiento basado en alguna ocurrencia lógica y la repetición proporciona iteración. Estas tres construcciones son fundamentales para la programación estructurada.

Las construcciones estructuradas se propusieron para limitar el diseño procedimental del software a un pequeño número de operaciones predecibles. Las métricas de complejidad indican que el uso de construcciones estructuradas reduce la complejidad de los programas y, por tanto, facilitan la legibilidad, la prueba y el mantenimiento. El uso de un número limitado de construcciones lógicas también contribuye a un proceso de comprensión humana que los psicólogos llaman troceado. Para comprender este proceso, considere la forma en que está leyendo este escrito. Usted no lee las letras individuales, *sino que reconoce* patrones o trozos de letras que forman palabras o frases. Las construcciones estructuradas son trozos lógicos que permiten a un lector reconocer los elementos procedimentales de un módulo, en vez de leer el diseño o el código línea a *línea. La comprensión aumenta cuando* se encuentran formas lógicas fácilmente reconocibles.

Cualquier programa, independientemente del área de aplicación y de la complejidad técnica, puede diseñarse e implantarse usando sólo las tres construcciones estructuradas. Sin embargo, debe tenerse en cuenta que el uso dogmático de tan sólo estas construcciones puede provocar a veces dificultades prácticas.

III.2 Programación orientada a objetos

A medida que se acercaban los años 80, el "paradigma orientado a los objetos" para la ingeniería del software comenzaba a madurar como un enfoque de desarrollo de software. Empezamos a crear diseños de aplicaciones de todo tipo usando la forma de pensar orientada a los objetos e implementamos programas usando lenguajes y técnicas orientados a los objetos. Sin embargo, el análisis de requisitos se quedó en el pasado.

En teoría, la creación de objetos y la construcción de software orientado a los objetos se puede llevar a cabo utilizando cualquier lenguaje de programación convencional (por ejemplo, C o PASCAL). Pero en la práctica, el soporte para métodos orientados a los objetos debe estar incorporado directamente en el lenguaje de programación que se vaya a utilizar para implementar un diseño orientado a los objetos. Un lenguaje de programación orientado a los objetos debe proporcionar un soporte directo para la definición de clases, la herencia, la encapsulación y el paso de mensajes. Además de estas construcciones básicas orientadas a los objetos, muchos de estos lenguajes implementan características adicionales, como herencia múltiple y polimorfismo (diferentes objetos que pueden recibir mensajes con el mismo nombre). La definición de clases es fundamental en un enfoque orientado a los objetos. Un lenguaje de programación orientado a los objetos define un nombre de clase y especifica los componentes privados y públicos de la clase.

Actualmente, el análisis orientado a los objetos (AOO) va progresando poco a poco como método de análisis de requisitos por derecho propio y como complemento de otros métodos de análisis. En lugar de examinar un problema mediante el modelo clásico de entrada - proceso - salida (flujo de información) o mediante un modelo derivado exclusivamente de estructuras jerárquicas de información, el AOO introduce nuevos conceptos tales como clases, objetos, atributos y operaciones como componentes principales de modelización. Coad y Yourdon escriben:

"El AOO (Análisis Orientado a los Objetos) se basa en conceptos que una vez aprendimos en la guardería: objetos y atributos, clases y miembros, todo y parte. Nadie sabe por qué hemos tardado tanto tiempo en aplicar estos conceptos en el análisis y la especificación de los sistemas de información - quizás hemos estado demasiado ocupados "siguiendo el flujo" durante nuestros análisis estructurados diarios, para ponernos a considerar otras alternativas."

Los objetos modelan casi cualquier aspecto identificable del ámbito del problema: entidades externas, cosas, papeles, sucesos, unidades organizativas, lugares y estructuras pueden ser representados como objetos. Como punto importante, los objetos delimitan datos y procesos. Las operaciones de procesamiento son parte del objeto y son

iniciadas pasando un mensaje al objeto. Una definición de una clase forma la base para la reusabilidad en los niveles de modelización, diseño e implementación. Se pueden crear nuevos objetos de una clase.

El método de análisis orientado a los objetos proporciona una notación y un conjunto de heurísticas para construir un modelo de AOO. Para hacer una especificación gráfica de un sistema basado en computadora, se usan estructuras, temas, conexiones de instancia y caminos de mensaje. El objetivo principal del AOO es identificar las clases en las que se integrarán los objetos.

El modelado de los datos se puede ver como un subconjunto del AOO. Usando el diagrama de entidad-relación como notación principal, este modelado se centra en la definición de objetos de datos (objetos que no encierran el procesamiento) y en la manera en que están relacionados entre sí. El modelado de datos se usa en aplicaciones con procesamiento de datos masivos y se puede aplicar como notación complementaria para el análisis estructurado.

Dado el amplio uso del Análisis Estructurado, AE, y del Diseño Estructurado, DE, y el gran interés por el enfoque orientado a los objetos, surge una pregunta: ¿se pueden integrar estas dos estrategias de análisis y diseño para formar un método único? Algunos investigadores piensan que las estrategias son muy diferentes, mientras que otros creen que se pueden utilizar elementos de ambos métodos para desarrollar un modelo completo de un problema. Nosotros coincidimos con esta última opinión pues así como el modelo en espiral recopila lo mejor del método de ciclo de vida y de construcción de prototipos, puede darse el caso de realizar una nueva metodología para desarrollo de software que combine estrategias tanto de la programación estructurada como de la orientada a los objetos.

Los elementos clave de AE/DE son el modelo de flujo, el diccionario de datos, la *especificación de control*, la *especificación de procesos* y el *diagrama entidad - relación* (E/R) para el modelado de datos. Este diagrama E/R puede ayudar a aislar clases y subclases candidatas, así como las relaciones entre ellas. Ward describe una correspondencia entre el AE/DE para sistemas de tiempo real y el diseño orientado a los objetos. Lo que él sugiere es que se puede representar una clase o una subclase como una burbuja aparte. Los flujos hacia y desde esa burbuja siguen representando flujo de datos, pero implican operaciones asociadas a la clase. Para definir una instancia de un objeto, se incluye la burbuja independiente en el contexto del modelo de flujo. Por ejemplo, una clase *sensor* puede estar modelada como una abstracción de datos (utilizando un diagrama E/R) y representada por una burbuja aparte. Se podría instanciar un objeto de la clase *sensor* (por ej.: *sensor de humo*) en un modelo de flujo específico, incluyendo una burbuja *sensor de humo* en el diagrama de flujo de datos. Así, se establecería una relación entre el objeto sensor de humo y la clase sensor mediante un diagrama entidad - relación.

Algunos investigadores piensan que "no hay una oposición clara entre el AE/DE y el AOO/DOO" mientras que otros creen que los métodos orientados a los objetos y el enfoque de diseño estructurado se excluyen mutuamente (y el AOO/DOO es con mucho

superior). Sólo el tiempo dirá si los dos valiosos métodos de diseño siguen caminos diferentes o si llegan a un matrimonio duradero.

Análisis de la información.

Una vez recopilada la información sobre los diversos métodos encontrados para el desarrollo de software, se procedió a realizar un análisis comparativo entre ellos para determinar cuál se adecuaba mejor a la satisfacción de nuestras necesidades.

Analizando la metodología del ciclo de vida, y algunos otros métodos similares a él, observamos que un inconveniente que presenta es que el cliente no ve el resultado deseado sino hasta que la metodología es aplicada en su totalidad o casi por completo, por lo que el cliente debe tener paciencia para esperar hasta que el sistema esté terminado. Esta cuestión no se presenta en la construcción de prototipos, donde se realiza un primer diseño a partir de las características que se definieron en la recolección de requisitos. La ventaja de tener un prototipo es que el cliente puede ver un esbozo de lo que podrá obtener como resultado, y tiene la oportunidad de evaluarlo y hacer sugerencias de correcciones sobre el desarrollo que se está realizando o bien sobre las especificaciones de las características que desea que contenga el sistema. Es una buena oportunidad del cliente para saber si los objetivos que se plantea son los correctos o debe hacer una reestructuración de los mismos.

Otra característica que presenta el método del Ciclo de Vida es un fácil manejo de la información, ya que en cuanto termina una etapa se inicia otra que está relacionada con la anterior. Así, el desarrollo se realiza de manera ordenada porque se siguen uno tras otro los pasos que marca cada etapa, como si se tratará de un programa estructurado.

Existen metodologías que tienen varias características comunes como son:

- Un mecanismo para la traducción de la representación del campo de información en una representación de diseño; en otras palabras, existe una etapa de recolección y análisis de datos que posteriormente se representan a través de esquemas y diagramas de flujo listos para ser interpretados por el o los programadores.
- Una notación para representar los componentes funcionales y sus interfaces; como ya se mencionó, se define mediante diagramas de flujo los requerimientos indispensables para el correcto funcionamiento del software;
- heurísticas para el refinamiento y la partición, es decir, contar con un conocimiento previo de la existencia, número y calidad de los documentos e información que se desea procesar; y
- criterios para la valoración de la calidad.

Las metodologías que se han revisado, (el método del ciclo de vida, el modelo propuesto por Gagné y Dick & Carey, la construcción de prototipos y el modelo en espiral), cuentan con alguno de los incisos anteriores y se comportan de manera similar a cómo se desarrolla un programa estructurado, por lo cual, se puede afirmar que se trata de

"*metodologías estructuradas*" pues cuentan con un patrón o estructura definida para la realización del software.

Lo anterior se respalda con lo que Roger Pressman nos dice acerca de la evolución del diseño de software. "La evolución del diseño de software es un proceso continuo que se ha ido produciendo durante las últimas tres décadas. Los primeros trabajos sobre diseño se centraron sobre los criterios para el desarrollo de programas modulares y los métodos para mejorar la arquitectura del software de una manera descendente. Los aspectos procedimentales de la definición del diseño evolucionaron hacia una filosofía denominada *programación estructurada*. Posteriores trabajos propusieron métodos para la traducción del flujo de datos o de la estructura de los datos en una definición de diseño. Nuevos enfoques para el diseño proponen un *método orientado a los objetos* para la obtención del diseño"[5].

Por otra parte, dentro del método en espiral se encuentra una etapa que no hay en los demás métodos, ésta es la de análisis de riesgo. En esta etapa, se observan los posibles factores que podrían afectar el funcionamiento del sistema, por lo que su seguimiento es importante para un buen desarrollo del proyecto de software puesto que analizar y poder prevenir aquellos factores que pueden alterar la funcionalidad del sistema nos dará un producto de calidad y con la garantía de que no quedará obsoleto al primer cambio en la tecnología de hardware o software.

Del método de construcción de prototipos la ventaja que se encontró es el diseño rápido y la construcción del prototipo, pues se puede tener un esbozo del sistema que se quiere desarrollar para que el cliente lo evalúe y pueda sugerir, con tiempo, lo que hace falta para cubrir al máximo su proceso y que no suceda lo que con métodos como el de Gagné y Dick & Carey, donde no existe un prototipo y lo único que ve el cliente es su sistema final, tal y como lo solicitó en un principio, pero en el cual no tuvo oportunidad de sugerir por algo que haya olvidado mencionar o por la presentación, colores, tipo y tamaño de letra que le hubiera gustado que incluyera su sistema final. En la Tabla 3 se muestra un comparativo de las metodologías revisadas.

III.3 Resultados del análisis realizado a cada método.

Como se esperaba, ninguno de los métodos encontrados se ajusta de manera completa a nuestro propósito, por lo que decidimos proponer una metodología en la cual se tomarán aquellas partes de los métodos encontrados que son de utilidad a nuestro problema y se integrarán correctamente para satisfacer adecuadamente los requerimientos del software.

La primera conclusión que obtuvimos fue que todos los métodos parten de la actividad de recopilar información sobre los requisitos que debe reunir el sistema a desarrollar. Debido a que esta actividad es de suma importancia en el desarrollo de cualquier tipo de software, consideramos conveniente incluir una etapa de análisis o planeación, en la cual se recogerán todos los requisitos y características que debe reunir el sistema tutorial a desarrollar. En esta etapa se definen los objetivos y los fines para los cuales será creado

[5] Ibidem 3

el sistema, cuáles serán las características mínimas deseables que debe tener, qué funciones se quiere automatizar o simplificar, cuál es el propósito principal de sistematizar el trabajo, etc.

Método	Característica	Ventaja	Desventaja
Construcción de prototipos. Modelo en espiral.	Creación de un prototipo.	El cliente puede ver un esbozo de su sistema.	El cliente ve un sistema incompleto, sin calidad que puede confundir con su proyecto final.
Todos los analizados.	Forma de diseño secuencial.	El proyecto se desarrolla con orden y sin temor de que crezca incontroladamente.	El cliente no puede cambiar de opinión una vez que se ha finalizado una etapa.
Modelo en espiral.	Iteraciones de las etapas.	El proyecto final será óptimo y resolverá casi al 100% las necesidades del cliente.	Si no se delimita bien el número de iteraciones el proyecto puede crecer sin saber en qué momento finalizará.

Tabla 3 Comparativo de los métodos analizados.

Seguida de la etapa de planeación y una vez que se tiene el diseño del sistema se optó por incluir una etapa de análisis de riesgo en la metodología propuesta. Esta etapa, entre la planeación y el desarrollo del prototipo, nos servirá para determinar las posibles limitantes que se puedan presentar durante el desarrollo del tutorial, así como para estudiar una posible solución a los problemas que en un 95% de los desarrollos se presentan.

Así también, creemos conveniente incluir en nuestra metodología una etapa de construcción del prototipo pues como hemos visto, el hecho de desarrollar un prototipo previo al desarrollo y entrega del sistema final, ayuda al analista/programador a forjar una imagen del sistema final ante el cliente y esto sirve para fomentar retroalimentación de ambas partes. Al finalizar el desarrollo del prototipo se tiene contemplada una etapa de evaluación del cliente, para contar con una revisión del trabajo desarrollado, que nos permita seguir adelante con la elaboración del sistema.

Estas etapas de creación del prototipo y evaluación del cliente son complementarias, ya que no sólo creamos un esbozo de lo que será nuestro sistema, sino que además contamos con la oportunidad de tener una revisión de éste y por consiguiente, contaremos con argumentos válidos que nos permitan continuar con el esquema trazado en el diseño, hacer las mejoras que se requieran al mismo (el diseño) o bien reestructurarlo.

Una vez concluida la evaluación del cliente es necesario continuar con la elaboración del sistema, y sería recomendable volver a tener una nueva evaluación sobre los cambios y modificaciones que se hagan al sistema, por lo que la etapa de construcción de prototipos y evaluación del sistema se volverían a implantar. Para realizar la construcción del nuevo prototipo o mejora del anterior, es necesario rectificar o ratificar las características deseadas en el sistema, por lo que la etapa de planeación debería llevarse a cabo una vez más, pero ahora, tomando en cuenta los resultados obtenidos de la etapa de evaluación del cliente, así también consideramos conveniente volver a realizar un análisis de riego del sistema en desarrollo, para determinar los posibles factores que nos puedan causar problemas en la elaboración y funcionamiento del producto.

La característica de volver a realizar cada actividad tomando en consideración resultados anteriores, sólo se observa en el método de espiral en el cual se definen una serie de etapas, las cuales se van repitiendo de forma iterativa hasta que los objetivos para los cuales fue creado el sistema se han cubierto en su totalidad.

Uno de los inconvenientes que presenta el método de espiral, es que si no se tiene bien establecido el número de iteraciones a realizar, la espiral crecerá de manera desordenada y por lo mismo, sin saber en qué momento se detendrá. Así, es recomendable y necesario, fijar el número de iteraciones con las que se pretenden cubrir las necesidades del sistema, y delimitar bien los resultados que se quieren obtener al finalizar cada una de estas iteraciones. Esto se logrará definiendo las metas a alcanzar al finalizar cada vuelta del espiral. El alcance de estas metas se irá cubriendo de manera progresiva a través de los objetivos principales para los cuales es desarrollado el sistema.

Por todo lo anterior, se consideró seguir una metodología basada en el modelo de espiral, la cual constará de una serie de actividades que se realizarán recursivamente hasta llegar al resultado deseado. De esta manera, hemos definido en nuestra metodología, las siguientes etapas:

1. Planeación
2. Análisis de riesgo
3. Desarrollo
4. Evaluación del cliente

CAPITULO IV. METODOLOGÍA PROPUESTA PARA EL DESARROLLO DE SISTEMAS TUTORIALES

IV.1 Planeación
IV.2 Análisis de Riesgo
IV.3 Desarrollo
IV.4 Evaluación.

Las etapas que hemos definido en nuestra metodología, la cual denominaremos como "*Método Marpol*", para el desarrollo de sistemas tutoriales son las siguientes:

- Planeación
- Análisis de riesgo
- Desarrollo
- Evaluación del cliente

IV.1 Planeación

Consideramos que esta etapa es fundamental en el desarrollo de cualquier software por lo que decidimos incluirla en esta metodología. Durante la planeación, lo que se busca es preparar todos los elementos necesarios para obtener un producto de calidad. Pero, ¿cuáles son los elementos de qué hablamos?

- Nuestro primer elemento es la recolección de requisitos. Para esta actividad partimos de que se ha definido perfectamente el problema que se desea resolver y es el momento de reunir las características del sistema a desarrollar. Para esto, podemos valernos de algunas técnicas como son:
  - Entrevistas con el cliente, en las cuales podamos indagar la problemática esencial de su empresa e identificar los requerimientos indispensables que solucionen y perfeccionen sus procesos de trabajo.
  - Encuestas, donde las preguntas sean diseñadas y encaminadas para captar las necesidades reales del cliente.
  - Observación cuidadosa y relación con las actividades principales que realiza la empresa del cliente.
  - Cateo del o los procesos, que son parte esencial en el desempeño de labores cotidianas.
  - Minutas. Después de cada entrevista con el cliente deben elaborarse las minutas. Estos resúmenes de los puntos más importantes que se acordaron, servirán para complementar la lista de los requerimientos.

- El siguiente elemento que consideramos importante es saber con qué herramientas se cuenta para la realización del sistema. Se debe investigar cuál es la capacidad del hardware y software de la empresa del cliente para, de esta manera, elegir correctamente el lenguaje en qué se va a programar el sistema, el alcance y las limitantes que tiene el hardware, la viabilidad de realizar algunos cambios en la plataforma de software, instalar o actualizar la red y/o el cableado, los dispositivos en común (impresoras, scanners, etc.), cambiar el sistema operativo, entre otras cosas más.

- El tercer elemento es el tiempo. Es muy conveniente conocer el tiempo del cual se dispone para el desarrollo del proyecto.

- Algunas ocasiones al cliente le urge tener un sistema funcionando en un tiempo muy corto y esto provoca que se omitan puntos importantes durante el

desarrollo. Por ejemplo, en nuestra metodología, cuando el sistema se requiere en un lapso de tiempo pequeño, puede omitirse la elaboración de un prototipo y comenzar directamente la programación del sistema final sin pasar por el sistema de muestra (prototipo). Esto es algo aventurado y requiere de un mutuo acuerdo entre ambas partes (analista/programador y cliente).

- Un cuarto elemento es la solvencia económica de quien contrata. El analista puede sugerir la adquisión de equipo nuevo o de algún software en especial, pero quien tendrá la última palabra será el cliente.

Cuando existe un presupuesto determinado para el desarrollo del proyecto, debemos limitarnos a éste y buscar en la extensa gama de herramientas que existen, las que optimicen los recursos económicos y sean útiles para nuestros propósitos.

Al igual que en los métodos descritos para el desarrollo de software, en esta etapa se reunirán las características y los requisitos que se pretende que contemple el sistema. Pensamos que si tomamos en cuenta estos cuatro elementos obtendremos una buena planeación, indispensable si queremos un desarrollo perfecto.

IV.2 Análisis de Riesgo.

Esta fase que sólo la encontramos en el modelo en espiral, la consideramos tan necesaria que decidimos incluirla en nuestra metodología pero:

¿Qué procuramos con esta etapa?

Lo primero que buscamos es evitar tener problemas, ¿por qué? Porque si desde un principio se analizan y predicen los obstáculos que pudieran presentarse durante el desarrollo del software, es factible plantear soluciones para esos problemas y definitivamente evitarlos.

Durante esta etapa se deben revisar cuidadosamente las ventajas y desventajas de *utilizar tal o cual herramienta, ya sea para el hardware o para el software*. Nos detendremos principalmente en aquellas características que puedan generar algún tipo de problema en el futuro como por ejemplo, que se descontinue alguno de los productos que empleamos, que se produzca un cambio inesperado en el presupuesto destinado para el desarrollo, que se roben alguna parte del proyecto, que el personal encargado de la programación del sistema se enferme o accidente, etc. Algunas de las situaciones anteriores parecen imposibles; sin embargo, llegan a presentarse.

Como se ha mencionado, esta etapa es muy importante pues al superar cualquier obstáculo se podrán cumplir satisfactoriamente cada una de las metas establecidas en la etapa de planeación y esto permitirá tener un desarrollo más limpio, sin retrasos en los tiempos definidos o excederse en el presupuesto destinado.

En esta etapa detectaremos también, los factores que puedan afectar el desempeño del producto, tales como una falla del servidor, una interrupción en el suministro de energía eléctrica, etc. Estos factores no son predecibles, pero se pueden tomar medidas precautorias antes de que lleguen a presentarse.

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

Los fines son:

- agregar funcionalidad
- incorporar o actualizar contenidos
- mejorar, reemplazar o incluir medios
- mejorar rendimiento
- adaptar a nuevas plataformas

IV.3 Desarrollo.

La tercera etapa de la metodología es el desarrollo. Esta etapa contempla el diseño y la realización del sistema para lo cual se utilizan aquellas herramientas de programación que mejor se acoplen (por lo menos en un 85%) para desarrollar los requisitos solicitados por el cliente.

Consideramos que se cubra mínimo un 85% porque, en ocasiones, las peticiones de los clientes son ambiciosas y una sola herramienta no las satisface. Por lo cual, se suele hacer una combinación de herramientas de tal manera que se involucren todas las peticiones y se satisfagan en un principio el 85% de ellas, con lo que se tendrá un sistema aceptable. Probablemente, durante el desarrollo se encuentren herramientas complementarias a las utilizadas que logren cubrir el 100% de los requerimientos. Usualmente, en los desarrollos reales es lo que ocurre.

La primera actividad que se plantea en la etapa de desarrollo es el diseño del sistema, tomando en cuenta los requerimientos del cliente. A continuación se presenta una definición que nos parece adecuada para definir esta etapa de diseño.

Diseño.

El diseño de software es un proceso mediante el que se traducen los requisitos en una representación del software. Inicialmente, la representación describe una visión holística del software. Posteriores refinamientos conducen a una representación de diseño que se acerca mucho al código fuente.

En el diseño se realizan dos pasos. El *diseño preliminar* se centra en la transformación de los requisitos en los datos y arquitectura del software. El *diseño detallado* se ocupa del

refinamiento de la representación arquitectónica que lleva a una estructura de datos detallada y a las representaciones algorítmicas del software.

Dentro del contexto de los diseños preliminar y detallado, se llevan a cabo varias actividades de diseño diferentes. Además del diseño de datos, del diseño arquitectónico y del diseño procedimental, muchas aplicaciones requieren de un diseño de la interfaz. El diseño de la interfaz establece la disposición y los mecanismos para la interacción hombre máquina (no cubierto por las herramientas del diseño estructurado)[6].

Para el diseño de la interfaz, proponemos la construcción de un prototipo, el cual nos ayudará a presentar ante el cliente un esbozo del sistema que se pondrá a su disposición. Es recomendable que este prototipo contemple la mayor parte de las características requeridas, pues así el cliente verá hasta qué punto es posible automatizar una tarea y las limitantes y desventajas que podrían presentarse.

¿Qué se debe considerar para hacer un buen prototipo?

En primera instancia se debe ordenar y clasificar la información recolectada en la etapa de planeación, una vez hecho esto se puede comenzar el diseño y programación del prototipo de tal manera, que se incluyan las características más sobresalientes o enfatizadas por el cliente puesto que, así verá un modelo más real (pero no el producto final) de su sistema.

El prototipo debe evaluarse periódicamente, con la participación del equipo de trabajo y una muestra de los usuarios finales. La validación involucra una evaluación de procesos, desde el punto de vista técnico, de diseño gráfico y de educación. A partir de las evaluaciones, el prototipo es desechado y en el sistema real se corrigen los detalles u observaciones hasta satisfacer los requerimientos de diseño.

Durante la construcción del prototipo se realizan las siguientes tareas:

- producción de medios de alta calidad
- reprogramación de todos los componentes de software (modular, portable, reusable, robusto)
- depuración de los aspectos de diseño gráfico
- confección de manuales, guías explicativas y material de apoyo
- incorporación de todos los contenidos

Para la programación del software, se debe decidir el o los lenguajes de programación que se utilizarán, es importante definir un estilo de programación, es decir, el o los programadores deben establecer las reglas y estándares para nombrar las funciones, variables, clases, estructuras y objetos (dependiendo si es programación estructurada PE, o programación orientada a objetos POO); a su vez, se deben definir los criterios para

[6] Gane, Chris and Sarson, Trish; "Structured Systems Analysis: Tools and Techniques; Editorial Prentice Hall; EEUU; 1979.

establecer constantes, las normas para aprobar la creación de un nuevo módulo (sea en el programa o en la base de datos) que no haya sido contemplado en el diseño, decidir hasta qué punto se permitirá alterar o modificar el diseño de base de datos, etc.

Para obtener el producto final se debe desarrollar el software completo, a partir de los requerimientos de diseño, y dejarlo en condiciones de ser comercializado.

IV.4 Evaluación.

Para saber si nuestro diseño y desarrollo del sistema son los adecuados, es necesario someterlos a una evaluación, razón por la cual se ha implementado esta cuarta etapa.

El objetivo de la prueba del software es descubrir errores. De la evaluación, se desprenderán los criterios que nos permitirán conocer si la metodología seguida es la adecuada o es conveniente añadir alguna otra actividad. En nuestro caso, esta parte nos servirá para comprobar si la metodología propuesta en verdad resulta ser óptima para el desarrollo de sistemas tutoriales.

La evaluación también arrojará aquellos requerimientos que no hayan quedado cubiertos o que no estén contemplados dentro del prototipo. Es importante, tomar en cuenta, que un buen sistema no sólo debe estar bien diseñado, sino que además debe ser de utilidad para los usuarios, por eso, deben ser ellos, de manera *objetiva*, quienes realicen la evaluación; así se tendrá un mejor panorama del desarrollo del sistema.

Esta etapa complementa el diseño del sistema, porque permitirá corregir errores, añadir nuevas características, mejorar las partes que no fueron contempladas en el prototipo y, en general, optimizar cada vez más el sistema final.

De acuerdo con las características definidas para la metodología, se le puede incluir en el grupo de metodologías estructuradas, debido a que desde el análisis hasta el desarrollo del software, se realizan en forma muy similar a como se hace un programa estructurado; esto es, siguiendo una serie de pasos consecutivos previamente definidos en cada una de las etapas. El software se desarrolla de manera iterativa, elaborando las actividades en el orden establecido y procurando no comenzar una nueva etapa sin haber concluido la anterior. Esta forma de diseñar el proyecto nos da la ventaja de mantener una secuencia a seguir durante el desarrollo, y de esta manera tener el control de cada actividad que se ejecuta. Así mismo, se pueden establecer los tiempos adecuados para presentar al cliente los avances del sistema.

"Dentro del análisis estructurado se parte el sistema funcionalmente y, según los distintos comportamientos, se establece la esencia de lo que se debe construir". Esto forma parte de lo que se realizará en la etapa de análisis, pues se tomarán las partes principales de cada asignatura optativa y que sean de mayor utilidad para el alumno.

Después de haber definido la metodología a emplear, es oportuno delimitar el número de iteraciones por realizar para obtener el producto deseado.

De acuerdo a las características y objetivos definidos para nuestro sistema tutorial, se acordó fijar en tres el número de iteraciones a efectuar para concluir la realización del sistema. Consideramos que con estas tres vueltas a la espiral de la metodología que proponemos, será suficiente para cubrir los requisitos que debe contener el sistema.

Además de definir el número de iteraciones, es conveniente establecer el alcance que se pretende lograr en cada una de ellas; para esto, existirán metas específicas para cada vuelta de la espiral.

Durante la primera iteración se pretende reunir los requisitos que tendrá el sistema tutorial para satisfacer las finalidades para las cuales será creado. Con base en ello, realizar un análisis de los riesgos que implica desarrollar un sistema que contenga información sobre las asignaturas opcionales que se imparten en la facultad y buscar soluciones a estos problemas. Posteriormente, se piensa elaborar un prototipo en el cual se aprecien las características más importantes del sistema y una vez terminado, encontrar la forma que algún grupo de estudiantes de la facultad pueda observarlo y evaluarlo. Esta evaluación permitirá saber hasta dónde se han cumplido las metas iniciales y cuál es el avance que se tiene en cuánto al diseño y presentación del sistema.

En la segunda iteración, de los comentarios que tenga el usuario sobre el prototipo y su sistema, se realizará un nuevo planteamiento de metas que contemplen quizá otros requisitos adicionales a los que se tenían. Enseguida de esto, se efectuará nuevamente un análisis de riesgo para indagar los posibles problemas que se pudieran originar. Durante la etapa de desarrollo de esta segunda vuelta, se hará el sistema real que se entregará al cliente y posteriormente, este sistema se someterá a la evaluación de los usuarios.

De esta segunda evaluación se obtendrán los argumentos necesarios para elaborar el sistema tutorial definitivo. En esta tercera vuelta de la espiral se espera obtener el producto deseado que cubrirá las expectativas del(os) usuario(s). En la etapa de planeación de esta tercera iteración se presentarán las nuevas características (si es el caso) y los últimos detalles para la liberación del sistema. Lo anterior deberá realizarse en la etapa de desarrollo.

Por tercera ocasión, se hará una revisión de los riesgos que podrían presentar estas adecuaciones o ajustes; así mismo, se contemplará la posibilidad de un crecimiento del sistema, y la viabilidad de tener una etapa de mantenimiento una vez que el sistema se encuentre como sitio Web. Habrá una nueva etapa de desarrollo donde se realizarán los ajustes previstos en la planeación para dar por terminada la elaboración del sistema. Finalmente, sólo se deberá dar el visto bueno para liberar el sistema y hacerlo público en la red donde estará a disposición de los estudiantes de la Facultad, y principalmente de los alumnos de la carrera de Ingeniería en Computación pues, serán ellos, quienes obtendrán el mayor beneficio dado que la función principal de este sistema es orientar a los estudiantes de Ingeniería en Computación en su elección de las asignaturas optativas. En este trabajo, el sistema tutorial, se tratará de conjuntar el mejor material posible para dar a conocer al alumno las distintas opciones que tiene. Quizá, la principal preocupación

y a la vez, la parte más importante, estará en qué tanto se debe profundizar en los temas que se desarrollan en cada asignatura, de forma tal que el estudiante tenga un panorama claro sobre el contenido de cada una de ellas. A su vez, este proyecto puede ser el pionero en dar a conocer al estudiante el contenido de las asignaturas que cursa, sean optativas u obligatorias. Se inicia con las asignaturas optativas porque son precisamente éstas las últimas que se cursarán en la carrera y además son las que debemos escoger; pero quizá en un futuro es posible que las diferentes Divisiones que componen a la Facultad se interesen en dar a conocer el contenido de las demás asignaturas. Esto también, puede ser de gran utilidad para los profesores porque pueden dar a conocer su forma de impartir las clases y los criterios que emplean para la evaluación; aunque sin duda, el más beneficiado será el estudiante, ya que tendrá una idea general sobre todo lo que puede aprender de cada asignatura que va a cursar, provocándole un mayor interés y motivación.

El presente trabajo, traería muchas ventajas pues:

- si los alumnos eligen la asignatura que más le interese, pondrán más empeño en su aprendizaje y esto redundará en un mejor aprovechamiento, lográndose elevar el índice de aprobación.

- si se pone al alcance de todos los estudiantes esta información, como puede ser si se coloca en internet; por un lado, tendrán la oportunidad de interactuar y/o conocer las nuevas tecnologías y estar a la vanguardia, por el otro, podrán consultarlo desde cualquier lugar que se encuentren (trabajo, casa, servicio social, laboratorio, en otro país, etc.) siempre y cuando se tenga una computadora y un acceso a la red mundial.

- si se toman asignaturas optativas que se complementen, el alumno se podrá especializar en áreas específicas de ingeniería. Esto ya se lleva a cabo en la carrera de Ingeniería Eléctrica Electrónica, en la cual el estudiante elige una de cuatro áreas de especialización, llamadas módulos de salida.

- si los estudiantes se especializan en cierta área, estudiando las asignaturas de su interés, esto puede influir para que inicien un desarrollo profesional enfocado a esa área donde se fortalezca su espíritu de autosuperación.

- el interés sobre las asignaturas optativas puede desembocar en la posibilidad de buscar especialización en los campos de investigación que le rodean, motivando al alumno a cursar estudios de postgrado o diplomados para llevar a cabo esta especialización.

- una elección correcta de las asignaturas significará que el estudiante tenga una completa formación en sus estudios de ingeniería, debido a que eligió aquellas asignaturas que más le interesan y, que por consecuencia le dejarán mayores beneficios, como una mejor preparación, o bien la inquietud de buscar una especialización en el área de interés y confianza en sí mismo para la toma de decisiones.

- Si se eligen adecuadamente las asignaturas optativas, podría despertarse el interés para proponer algún proyecto de seminario y/o tesis sobre algún tema relacionado con la(s) asignatura(s) cursada(s).

Como se observa, los beneficios que se obtienen por el uso de este sistema son muy amplios.

PARTE II

CAPITULO V. APLICACIÓN DE LA METODOLOGÍA

V.1 PRIMERA VUELTA

V.1.1 PLANEACIÓN

Definición del problema

El objetivo principal al desarrollar este sistema tutorial, es que sirva de apoyo a los alumnos de la carrera de Ingeniería en Computación en la elección de las asignaturas optativas y a la vez, sirva de apoyo a dichas asignaturas.

Primeramente se definirá por qué se les llama *asignaturas optativas*.

El plan de estudios que marca la Universidad Nacional Autónoma de México, UNAM, para la carrera de Ingeniería en Computación consta de un total de 56 asignaturas que se cursan dentro de un sistema escolarizado con una duración de 10 semestres. Este plan de estudios marca a su vez, una precedencia obligatoria entre algunas de las asignaturas; esto significa que no se puede cursar una asignatura sin haber aprobado la asignatura antecedente.

El plan se encuentra dividido en tres niveles. El primero de éstos abarca los cuatro primeros semestres y agrupa principalmente las asignaturas de *Ciencias Básicas*, es decir, todas aquellas que son indispensables en la formación de un buen ingeniero; así, estas asignaturas básicas las cursa todo individuo que sigue una carrera de la rama ingenieril. El segundo nivel, al que corresponden el quinto, sexto y séptimo semestres, lo componen en su mayor parte, asignaturas de *Ciencias de la Ingeniería*, éstas son aquellas que se relacionan con el área de la ingeniería que se desea estudiar.

El tercer nivel, denominado de *Ingeniería Aplicada*, abarca los últimos tres semestres y lo comprenden las asignaturas más exclusivas del área de la ingeniería que se ha elegido. Así, por ejemplo, el plan de estudios de la carrera de ingeniería eléctrica y electrónica subdivide este segmento en cuatro módulos de especialización para encaminar a los estudiantes a un área particular de la amplia variedad que existe en el campo de la electrónica. Dentro del plan de Ingeniería en Computación no existe una subdivisión; cada estudiante tiene la oportunidad y la responsabilidad de elegir qué asignaturas optativas cursará para finalizar su preparación.

El primer nivel, se cursa dependiendo del resultado que obtenga el alumno en el examen diagnóstico que se aplica una vez que ha sido aceptado en la Facultad de Ingeniería de la UNAM. De acuerdo con la puntuación obtenida en dicho examen, el alumno ingresa a un grupo de alto rendimiento (PARA), a un grupo propedéutico o comienza a cursar el primer semestre de la carrera que ha elegido. En este último caso, la Facultad proporciona al alumno el horario y las asignaturas que cursará dentro del primer semestre. A partir del segundo semestre, el estudiante es libre de elegir sus horarios y asignaturas siempre y cuando se apegue a los reglamentos establecidos en su plan de estudios, es decir, que respete la seriación entre asignaturas y el orden que establece el plan. Así mismo, por ejemplo, para tomar alguna asignatura del segundo nivel es necesario haber cubierto un mínimo de 118 créditos del primer nivel y para elegir

asignaturas del tercer nivel, se tiene que cumplir con el 100% de créditos del primer nivel y un mínimo de 73 créditos del segundo nivel.

El plan de estudios de la carrera Ingeniería en Computación, establece 31 créditos como número total por cursar las asignaturas optativas y corresponde al alumno verificar que la suma de créditos de las asignaturas que elija cumpla con este requisito.

Toda la reglamentación con que cuenta este plan de estudios es con el objeto de formar a los mejores estudiantes, pues ellos se convertirán en profesionistas al terminar la carrera y contribuirán a definir el futuro de nuestro país. Por ello, cada uno de los estudiantes que llegan al tercer nivel que establece el plan de estudios son muy valiosos para la universidad y para la sociedad en general y es importante que desarrollen una buena capacidad para la toma de decisiones. Debido a esto, se les brinda la oportunidad de seleccionar sus últimas asignaturas con las que concluirán sus estudios.

Para realizar la elección de las asignaturas, el alumno es libre de escoger de acuerdo con sus preferencias, áreas de interés, horario o cualquier otro criterio que defina; lo primordial es escoger correctamente y para conseguirlo, tendrá a su disposición este sistema tutorial de asignaturas optativas, el cual le auxiliará antes, durante y después de haber elegido y/o cursado cualquiera de las asignaturas optativas. Con la ayuda de este tutorial, el alumno podrá:

- ser autodidacta para aprender cualquier tema que le interese de alguna asignatura en particular,
- preparar algún tema en caso de que requiera hacer una exposición o simplemente desee obtener un conocimiento más completo de la asignatura,
- investigar por su cuenta de qué trata cada asignatura optativa y de esta manera realizar la mejor elección de ellas y cerrar con éxito su formación a nivel licenciatura,
- aclarar sus dudas sobre temas que no haya entendido en la clase o alguna inquietud que se le presente y,
- lo más importante y que quizá involucra todo lo anterior, el alumno podrá estudiar libremente sin tener ningún límite como: no existe el libro en la biblioteca, la biblioteca está cerrada, el profesor se ausentó, el alumno no pudo asistir a clase, entre tantos otros sucesos que se pudieran presentar.

Hasta ahora, se ha hecho mayor énfasis en la importancia de que un estudiante del grado de licenciatura tenga la capacidad de elegir correctamente las asignaturas optativas con las cuales concluirá sus estudios y que debe escoger cuatro de las once alternativas pero, ¿cuáles son las asignaturas optativas que contiene nuestro plan de estudios?

Las asignaturas optativas son las siguientes:

- Bioingeniería
- Calidad
- Diseño asistido por computadora
- Graficación por computadora

- Organización y administración de centros de cómputo
- Procesamiento digital de imágenes
- Procesamiento digital de señales
- Reconocimiento de patrones
- Robótica
- Sistemas expertos
- Temas especiales de computación

En el plan de estudios de la carrera de Ingeniero en Computación se maneja la posibilidad de escoger cuatro asignaturas optativas de las once disponibles. Como estas asignaturas son "obligatorias" no se encuentran establecidas, el alumno tiene la opción de elegir las cuatro que más le convengan o le interesen sin ninguna restricción. Sin embargo, frecuentemente se encuentra ante la interrogante de ¿cuáles asignaturas elegir?

De acuerdo a lo expuesto por alumnos conocidos y experiencia propia, existen muchos criterios para elegir estas asignaturas. Uno es, optar por las que aparentan ser más sencillas. Otro suele ser por recomendación de estudiantes que ya cursaron tal o cual materia. Un criterio más es por conveniencia de horario y, quizá el más común, es por el interés que despierta en el estudiante el nombre de la asignatura; pero ¿qué implicaciones tiene cada uno de estos criterios?

- Se puede optar por escoger aquellas que aparenten ser más sencillas, pero ¿cuáles son esas?, ¿Se sabe en verdad, qué tal o cuál materia no es más complicada que las demás?

- Tomar en cuenta la opinión de los compañeros que ya cursaron alguna asignatura y que les pareció interesante. Esta puede ser una buena opción pero, se tiene el problema de que las opiniones de estos compañeros no sea del todo objetiva, es decir, lo que para algunos puede resultar interesante no lo puede ser para otros y cada uno opinará de acuerdo a su experiencia al cursar la asignatura.

- Elegir las asignaturas de acuerdo al horario que más nos convenga. Ésta suele ser la opción para aquellos compañeros que trabajan o que comenzarán a realizar su servicio social y no tienen flexibilidad de horario.

- Elegir las que parecen ser más interesantes, resulta la misma situación, ¿sabemos realmente, cuáles son de mayor interés?. Si lo sabemos, ¿bajo qué criterio se hace esta afirmación? Por el nombre de la asignatura o por el temario que tiene o por las opiniones de los compañeros que ya cursaron la asignatura o porque se conoce algo de la asignatura.

- Situaciones extraordinarias como: a) el profesor es excelente para exponer sus clases (de acuerdo con el punto de vista del alumno) o, b) el alumno y el profesor se llevan bien o, c) se está desarrollando un proyecto o, d) se piensa

trabajar un tema de tesis donde influye tal o cual asignatura optativa o, e) alguna otra situación que parece poco común o imposible, pero llega a ocurrir.

Podemos deducir de lo anterior que no existe un parámetro exclusivo que nos permita emitir un juicio sobre cuáles asignaturas optativas podemos cursar, sin embargo, sabemos que de la correcta elección dependerá terminar con una buena preparación la carrera y desenvolvernos abiertamente como profesionales en computación, con seguridad y confianza; entonces, para hacer una elección certera, es necesario conocer las distintas opciones que se nos presentan, pero la pregunta que surge ahora es: ¿cómo conocerlas de manera objetiva y veraz?

Pensamos que el punto de mayor interés para el alumno es conocer los temas que se estudian en dichas asignaturas, de qué trata cada una de ellas, qué beneficios (de aprendizaje) se obtendrán de cursar una o otra asignatura puesto que, por lo general, para la elección de estas asignaturas, los alumnos recurrimos a alumnos de generaciones anteriores para que nos recomienden determinada asignatura optativa que ellos cursaron, o bien optamos por alguna de ellas especulando sobre el contenido de la misma una vez que hemos acudido a la División correspondiente a cada asignatura y hemos conseguido el temario de la misma. No obstante, son pocos los alumnos que conocen a qué División pertenece cada asignatura, y más aún son menos los que acuden a conseguir un temario.

Por otra parte, aunque con el hecho de poseer estos documentos se puede saber el contenido de cada asignatura, los objetivos, antecedentes y otras características, es cierto que no se describen detalladamente cada uno de los temas, sólo se tiene una idea general, por lo que resulta necesario e indispensable para el estudiante contar con algún material de apoyo que le proporcione mayor información que la que obtiene del temario, y de esta forma tenga más elementos para realizar una buena discriminación.

Por las razones anteriores y la experiencia propia, es que consideramos de suma importancia contar con un sistema tutorial de asignaturas optativas de la carrera de Ingeniería en Computación ya que actualmente, en la facultad no se cuenta con algún material que describa el contenido de cada una de estas asignaturas optativas. Para dicho proyecto se tomarán en consideración los puntos que necesita saber el estudiante acerca de cada una de las optativas, para realizar una correcta elección independientemente del criterio que se utilice.

Un punto principal a resaltar, es el hecho de que como estas asignaturas tienen un carácter optativo, y tenemos la oportunidad de elegir las que queramos, es *importante* que nuestra elección sea satisfactoria. Resultaría decepcionante y quizá frustrante, que aquellas asignaturas que podemos elegir libremente no resulten ser lo que nosotros esperábamos. Además, el cursar una u otra asignatura nos ayudará a conocer la rama de especialización a la que podemos dedicarnos.

Como puede apreciarse, la elección de las asignaturas optativas es un asunto muy importante, por las circunstancias y repercusiones que traerá consigo la correcta elección, en la formación de los estudiantes.

De lo anterior, nace la idea de crear un sistema tutorial de asignaturas optativas, para dar a conocer a los alumnos de la carrera de Ingeniería en Computación un panorama general de cada asignatura, para que de esta forma las conozcan, entiendan los temas que tratan y puedan saber si cumplen con sus expectativas e intereses; a la vez que, podrán contar con un criterio más amplio para hacer una correcta elección que es el objetivo principal de este sistema tutorial.

Sin embargo, surgen como objetivos secundarios o dependientes del principal los siguientes:

- Si los alumnos ya eligieron sus asignaturas optativas, este sistema les puede servir de complemento a lo que ven en clase o como alternativa de repaso.

- En caso de que alguno haya decidido qué optativas cursar, ya sea con ayuda de este tutorial o por otro medio, el presente sistema le puede proporcionar información suficiente para conocer de manera general las demás asignaturas y, así tener un panorama general de todas las asignaturas que se pueden cursar de forma optativa dentro del plan de la carrera.

Finalmente, ¿por qué un sistema tutorial como solución a este problema?

Debido al carácter interactivo con el usuario y sus características informativas, pensamos que un sistema tutorial es la opción más adecuada de hacer llegar a los usuarios el material que les hace falta para conocer con mayor detalle una asignatura optativa y, para facilitar la difusión del sistema tutorial, una alternativa es colocarlo en Internet, ya que en la red mundial el acceso a la información es más sencillo y tiene un costo accesible. Lo único que el alumno necesita es: contar con acceso a Internet, el cual, aquellos que no lo tengan desde sus casas pueden contar con éste a través de los diferentes laboratorios de cómputo existentes dentro de la Facultad de Ingeniería

Metas

La principal meta en esta primera vuelta de la metodología es establecer el tipo de información que se introducirá en el primer prototipo para buscar acercarse al objetivo principal, que es el dar a conocer a los Ingenieros en Computación un panorama de lo que son las asignaturas optativas.

Para cumplir con esta meta, es necesario investigar qué tipo de información requieren los alumnos para conocer el contenido de una asignatura. Por lo tanto, una meta anterior a la principal es: saber la opinión de los usuarios acerca de lo que debe contener un sistema tutorial que trata sobre las asignaturas optativas.

Después de saber qué tipo de información se debe incluir en el sistema y elaborado el primer prototipo a partir del conocimiento de esta información, se debe poner a disposición de los alumnos la primera versión del sistema para que éstos lo puedan evaluar y puedan dar su opinión sobre si el prototipo cumple con su cometido. Por lo que una meta posterior a la realización del primer prototipo es buscar la forma de que un

grupo de alumnos pueda ver el primer prototipo, para que de este análisis surjan ideas y sugerencias para una mejora de esta primera versión.

Requisitos

Los requisitos que debe contener el primer prototipo del sistema son:

- Información

La información manejada en esta primera versión (prototipo) del sistema debe ser suficiente para llamar la atención del alumno acerca del contenido de las *asignaturas optativas que puede cursar, sin profundizar mucho en el detalle* de esta información incluida

- Presentación

La presentación manejada en esta primera versión del sistema debe ser agradable pero sin profundizar en el detalle. No manejar imágenes en este primer prototipo, simplemente buscar una adecuada combinación entre fondos y textos, que permitan al usuario leer adecuadamente la información desplegada sin desviar su atención.

- Organización

A diferencia de la presentación, en esta primera vuelta se debe poner mucho énfasis en la organización del sistema. La estructura de este primer prototipo debe ser de tal forma que permita al usuario navegar tranquilamente por donde quiera, evitando llevarlo a algún sitio del cual ya no pueda regresar o salir. De la elaboración de esta estructura depende que no hay muchas modificaciones al sistema en versiones sucesivas del mismo.

Opciones de solución

Una vez definidos los objetivos, metas y requisitos a alcanzar en esta primera vuelta de la metodología, es necesario plantear las opciones que se tienen que aplicar para alcanzarlos.

Primeramente se debe definir el tipo de información que se incluirá en el sistema tutorial. Dado que el sistema tutorial tiene la finalidad de servir de guía a los alumnos de la carrera de Ingeniería en Computación en su decisión sobre cuáles asignaturas optativas pueden elegir, es indispensable conocer qué tipo de información requieren o creen necesitar para llevar a cabo una buena elección. La mejor manera de saber su opinión es preguntándoles directamente, y una forma muy eficaz de realizar este sondeo es a través de una encuesta. Por esta razón, se decidió redactar una, en la cual se plantean preguntas que permiten conocer la opinión de los alumnos acerca de lo que les interesaría saber sobre las asignaturas optativas y una vez formulada y revisada esta encuesta, aplicarla a un grupo de compañeros de la carrera de los últimos semestres.

Las preguntas están encaminadas a conocer qué tipo de información requieren los compañeros para poder realizar una adecuada elección de las asignaturas optativas, así

como la forma en que les gustaría que estuviera presentada y organizada. Los resultados obtenidos dan una idea más clara sobre lo que los alumnos desean saber acerca de las asignaturas optativas, y permiten establecer el tipo de información que se incluirá en el sistema tutorial. Es importante hacer la observación de que la mayoría de las personas encuestadas han cursado asignaturas optativas, por lo que sus aportaciones y sugerencias resultan ser de gran utilidad debido a la experiencia que tuvieron al tener que elegir estas asignaturas.

A continuación se anexa la encuesta aplicada, junto con las respuestas obtenidas, así como las conclusiones que pudimos deducir de éstas

ENCUESTA APLICADA

1.- Has cursado asignaturas optativas:
Sí____ No____

2.- ¿Cuáles asignaturas optativas piensas tomar o has tomado?
_____
_____
_____
_____
_____

3.- ¿Por qué elegiste (o piensas elegir) esas asignaturas?
Horario_____ Recomendación_____ Nombre_____ Profesor____
Interés_____
Otro___ ¿Cuál?_____

4.- ¿Has encontrado en Internet algún tutorial sobre las asignaturas optativas de la carrera de ingeniería en computación?
Sí_____ No_____

Si respondiste que sí, ¿que te pareció?
Excelente_____ Bueno_____ Regular_____ Malo_____

¿Qué le agregarías? ¿Qué le quitarías? ¿Por qué?
_____
_____
_____
_____
_____
_____
_____

5.- Si hubiera un programa tutorial que te mostrara esta información, ¿qué te gustaría que tuviera integrado? Marca con una cruz (X).
Temario de la materia_____
Panorama general sobre la materia_____
Sugerencias de asignaturas (dependiendo del campo de interés)_____
Aplicaciones_____
Áreas de trabajo_____
Otros
_____
_____
_____
_____
_____

6.- Cuando ves en Internet un tutorial sobre algún tema de tu interés, ¿qué es lo que te llama la atención? Marca con una cruz (X).
Los colores_____
Las pantallas_____
La organización del tutorial_____
Las ligas a otras páginas_____
La información_____
Las imágenes_____

7.- ¿Qué es lo que te llamaría la atención de un tutorial acerca de las asignaturas optativas?
Marca con una cruz (X) las que consideres más importantes.
Muchas imágenes_____
Pocas imágenes_____
Textos largos_____
Textos cortos_____
Contenido_____
Fondos_____
Botones_____
Tamaño de letra_____

8.- Cuando lees un tutorial en Internet, ¿qué es lo que menos te gusta? ¿Por qué?
Información_____
_____
_____
_____
_____
Presentación_____
_____
_____
_____
_____

Organización_____

Después de aplicar esta encuesta a un grupo de alumnos, se obtuvieron las siguientes:

RESPUESTAS.

No. de encuestas aplicadas: **64**.
(alumnos *Generación 93 y 94*)

1. **Han cursado asignaturas optativas:**

Sí	53
No	11

NOTA. *Obsérvese que aproximadamente el 80% de los encuestados han cursado asignaturas optativas*

2. **Asignaturas optativas que han tomado o piensan tomar:**

Bioingeniería	36
Temas Especiales de Computación	34
Procesamiento digital de imágenes	24
Graficación por computadora	22
Org. y admón. de Centros de Cómputo	18
Robótica	17
Calidad	15
Reconocimiento de Patrones	10
Procesamiento digital de señales	9
Diseño Asistido por Computadora	7
Sistemas Expertos	6

NOTA. *Las asignaturas optativas fueron ordenadas de acuerdo al número de preferencia. Definitivamente esta información es de utilidad para la División, ya que se puede hacer un análisis de por qué unas asignaturas optativas son más demandadas que otras. Hay que tener en cuenta que esta encuesta es sólo una muestra de la población, pero sin duda arroja resultados interesantes.*

3 **Las asignaturas optativas las eligen por:**

Interés	54
horario	17
recomendación	16
nombre de la materia	5
profesor	3

Otros:

Utilidad
Seminario
Le llamó la atención

NOTA. *La mayoría de los alumnos elige un materia optativa porque les interesa o por recomendación; si cuentan los alumnos de la carrera de Ingeniería en Computación con un sistema que les permita tener un panorama general sobre cada una de las asignaturas optativas, éste podría ser de gran ayuda en su elección.*

4. **Han visto un tutorial en Internet de asignaturas optativas de ingeniería en computación:**

No	51
Sí	13

Les pareció:

excelente	1
bueno	6
regular	6

Le agregarían:

Más ejemplos y comentarios
Más Información sobre puntos importantes de la carrera.
Aplicaciones.
Información específica
Consulta rápida
Mejor acceso

NOTA. *Estos comentarios son de gran utilidad, ya que permiten tener una idea de lo que los usuarios requieren de un sistema tutorial. Como puede verse, la mayoría le da prioridad a la información, siempre y cuando ésta sea importante y útil.*

5. **Les gustaría que tuviera el tutorial de asignaturas optativas:**

Aplicaciones	55
Temario	53
Área de trabajo	50
Panorama general	48
Sugerencias de asignaturas	29

Otros:

- Profesor
- *Referencias*
- Enlaces con otras asignaturas
- Bibliografía
- Evaluación
- Horario
- Material y equipo a utilizar
- *Objetivos*
- Antecedentes
- Ejemplos y ejercicios
- Aplicaciones prácticas
- Investigación

NOTA. *Como se ve los usuarios le dan mayor importancia a la utilidad que puedan tener las asignaturas optativas sobre todo en aplicaciones profesionales; además de contar con un panorama general sobre las asignaturas optativas en cuanto a contenido, profesores, material didáctico y todo lo necesario para saber cómo aprovechar la asignatura.*

6. **De un tutorial les llama más la atención:**

Información	62
Organización	47
Imágenes	41
Ligas	26
Pantallas	20
Colores	11

NOTA. *Nuevamente se le da prioridad a la información, pero debidamente organizada, que sea substancial. El tutorial debe ser creado de forma tal que cuente con la información suficiente para cumplir con los objetivos para los que será creado permitiendo una buena navegación, siendo útil sin llegar a ser tedioso.*

7. **Les llamaría la atención de un tutorial de asignaturas optativas:**

Contenido	58
Textos cortos	34
Tamaño de letra	26
Muchas imágenes	18
Botones	16
Fondos	15
Pocas imágenes	10
Textos largos	5

NOTA. *El contenido de la información debe ser lo más importante, para lo cual se tiene que tomar en cuenta que se deben proporcionar los datos más interesantes de cada asignatura. En pocas palabras, no hay que perder de vista la buena redacción, manejar un tamaño de letra adecuado, las imágenes deben complementar la idea, no se debe abusar de ellas ni deben desviar la atención de los usuarios. Además, se deben usar fondos e imágenes adecuadas para obtener una buena presentación y utilizar las herramientas necesarias para permitir una buena navegación.*

8. **Lo que menos les gusta de un tutorial en relación a:**

INFORMACIÓN	PRESENTACIÓN	ORGANIZACIÓN
Imprecisa	*Sin tapiz*	*Sin secuencia*
No disponible	*Textos largos*	*Confusa*
Demasiado técnica	*Letra pequeña*	*Difusa*
Reducida	*Poco original*	*Sin control de ligas*
Rebuscada	*Demasiado movimiento*	*Demasiadas ligas*
Textos largos	*Muchos colores*	*Difícil acceso*
Repetitiva	*Muchas imágenes*	*No localización de*
Confusa	*Pocas ligas*	*información*
Incompleta	*Colores brillantes*	*Navegación excesiva*
Sin ejemplos	*Sin relación con el tema*	*Sin índice*
Redundante	*Sin colores*	*Desordenada*
Sin contenido	*Exagerada*	*Sin botones*
Sin fuentes	*Poco práctica*	*Sin referencias*
No actualizada	*Que no llame la atención*	*Muy general*
Incongruente	*Simple*	
Mal redactada		
Innecesaria		
Poco importante		
Letra pequeña		
Poco específica		

NOTA. *Como se puede ver, estos tres puntos son los que hay que tener más en cuenta. Una buena presentación llamará la atención de los usuarios del sistema. La información es el punto más importante, porque es la razón principal por la que este tutorial será consultado. El contenido debe ser tal que pueda contribuir realmente en la toma de decisiones para elegir una materia optativa adecuadamente. Para lograr esto, el tutorial debe estar debidamente organizado, permitiendo que el usuario pueda recorrerlo a su gusto, pero sin que su recorrido llegue a ser tedioso, además de resultar provechoso.*

De la información recopilada se pudieron obtener conclusiones muy interesantes.

En primer lugar, podemos observar que la mayoría de los encuestados han tomado asignaturas optativas, por lo que tienen experiencia en cuanto a su elección. El principal motivo que tienen para elegirlas es el interés que sienten de aprender determinados temas. Sin embargo, si cuentan con un apoyo para facilitar esta elección, como es el tutorial sobre asignaturas optativas y les resulta de utilidad, habremos cubierto el objetivo principal para el cual fue planeado este sistema.

En general, con las aportaciones de los encuestados podemos decir que el sistema tutorial debe:

- Tener una buena organización en cuanto a su estructura; esto es, no debe llevar al usuario a una página de la cual no pueda regresar, además de tener el número adecuado de ligas sin excederse.

- Lo primordial es la información, la cual debe ser concreta, concisa y precisa. Manejar aspectos importantes pero usando textos cortos, sin redundar. Emplear letra adecuada y ejemplos que complementen el contenido de la información desplegada.

- En cuanto a la presentación, no debe ser demasiado ostentosa, debe llamar la atención pero sobre todo respetar el contenido, que sea claro y objetivo. Evitar colores brillantes. Usar una buena combinación de colores, que además de discreta sea agradable.

- No excederse en el uso de imágenes, procurando que éstas no desvíen el interés de los usuarios. Además, las imágenes usadas deben corresponder al tema desarrollado. Debe existir armonía entre el texto y las imágenes.

- Las ligas deben ser suficientes para llevar de la mano al usuario a través del tutorial, pero sin caer en recorridos tediosos. El tutorial debe estar debidamente estructurado de tal forma que cada liga conduzca a alguna parte, facilitando la navegación a través del mismo.

- En cuanto a la información podemos manejar:
  - Temario de cada una de las asignaturas, desglosando el contenido por tema.
  - Objetivos y antecedentes de cada materia
  - Bibliografía
  - Tareas y proyectos
  - Información de los profesores que imparten la materia (cuando sea posible).

- Para hacerlo más interactivo podemos incluir también:
  - Comentarios y sugerencias por parte de los usuarios del sistema
  - Evaluación del sistema hacia los usuarios para conocer el grado de utilidad del mismo.

Planeación del proceso de desarrollo

Como se estableció en las etapas de la metodología a utilizar, la etapa de desarrollo corresponde, en su primera vuelta, a la realización de un prototipo para lo cual se empleará el método de creación de prototipos al que se añadirán algunas características del método de versiones sucesivas.

El desarrollo de productos mediante el método de versiones sucesivas es una extensión del método de construcción de prototipos. En ambos métodos, como primer paso se define un diseño patrón para la creación del prototipo. Mientras la construcción de *prototipos se enfoca en tener un sistema de prueba como muestra para el usuario*, el método de versiones sucesivas, elabora un prototipo que se refina una y otra vez; de tal manera que al concluir cada versión existe un sistema funcional y capaz de realizar un trabajo útil. Sin embargo, éste no será el sistema final que se entregara al usuario.

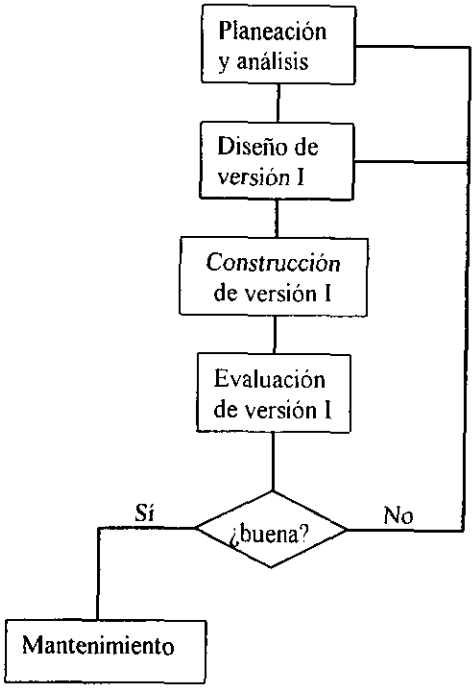

Figura 3. Método de Versiones Sucesivas

El desarrollo de esta primera vuelta se iniciará con el diseño y la construcción de un prototipo, primera versión del sistema tutorial puesto que como se utilizará la característica de iteración del método de versiones sucesivas (Figura 3), el prototipo será un sistema funcional aunque no el sistema final. Posteriormente, se evaluara dicha versión para ver si cumple con los requerimientos, si es así se depura y se libera, si no lo es se hace un nuevo diseño o se mejora el anterior. La finalidad de tener dicho prototipo

es que el usuario vea, critique y sugiera cualquier duda o idea que ayude a realizar un mejor y más completo sistema tutorial. Cabe señalar que el prototipo que resulte en esta primera vuelta será desechado, pues únicamente es una muestra del producto final.

Verificación y validación

Como en esta metodología utilizada se emplea la construcción de prototipos y algunas características de versiones sucesivas, es necesario ir evaluando en cada vuelta del espiral la nueva versión del sistema. Debido a esto, se incluye la etapa de evaluación, en la cual el prototipo será analizado por un grupo de usuarios, los cuales darán sus opiniones y críticas de lo que les parezca el tutorial. Con estas sugerencias, se puede observar en qué medida se van cumpliendo las metas planteadas en esta sección, y hasta qué punto se ha cubierto el objetivo principal.

Al concluir la etapa de planeación, se habrá definido perfectamente el problema; se tendrán determinados los objetivos principales y las metas, así como, las alternativas para alcanzarlos y, se conocerán los requisitos que debe tener el sistema tutorial. De esta manera, se procederá a la siguiente etapa, el análisis de riesgo.

V.1.2 ANÁLISIS DE RIESGO

Hasta este momento, se han establecido los requerimientos del sistema. Uno de ellos y quizá el más importante, es la idea de que el sistema pueda ser consultado por todos los alumnos de la carrera de Ingeniería en Computación principalmente, por lo cual una alternativa que se propone es poner este tutorial en Internet pero para ello, ¿qué métodos y herramientas se deben usar?.

Una herramienta muy práctica y sencilla es el desarrollo de páginas utilizando HTML pero, ¿qué ventajas y desventajas se tendrían?

Las ventajas son:

- Es un lenguaje fácil de aprender y sencillo de manipular ya que por medio de algunas etiquetas permite agregar imágenes, otras páginas Web, programas para un propósito específico, crear applets, manejar bases de datos, etc.

- A pesar de que los archivos suelen verse demasiado grandes, ocupan poco espacio en el disco.

- Existe una amplia bibliografía a consultar sobre cómo desarrollar tal o cual cosa; así también, existen grupos de trabajo en la red a los cuales se les pueden preguntar dudas específicas y recibir respuesta casi inmediata.

- Permite personalizar una página al poder escoger el fondo que se considere más adecuado a la página, el tipo, color y tamaño de la letra, el color de las ligas, la ubicación de las imágenes, etc.

- Un archivo HTML de ser leído por cualquier browser, siempre y cuando se realicen en el código las consideraciones pertinentes.

Como desventajas están:

- Resulta tedioso tener que colocar etiquetas en todo el texto.

- Es complicado agregar caracteres especiales como letras acentuadas, la letra ñ, signos de interrogación o admiración, símbolos matemáticos como pertenece o producto punto, etcétera.

Otro de los riesgos que se tienen al desarrollar este sistema tutorial es qué tan significativo pueda resultar el tener este tipo de material en la red, tanto para los alumnos como para los profesores que imparten estas asignaturas, debido a que, como se ve en los requisitos, se pretende incluir información relacionada con los conocimientos y experiencia de los profesores y las tareas y proyectos que se realizan en cada una de las asignaturas. Esto beneficiaría a los alumnos, porque podrían conocer un poco más que lo que les puedan comentar otros compañeros sobre el currículum del profesor y sobre las aplicaciones que se pueden alcanzar después de cursar una u otra de estas asignaturas. Para los profesores, sería muy útil que los alumnos conocieran los temarios y objetivos del curso para que puedan dar sugerencias sobre cómo prefieren aprender los conceptos, qué les gustaría profundizar y qué se les hace más complicado. Además, si los alumnos pudieran ver proyectos realizados por otros compañeros de semestres anteriores, esto serviría para elevar la calidad en el aprendizaje, puesto que si algunos compañeros pudieron realizar algún proyecto, ellos también pueden hacerlo e incluso superarlo y el profesor, podría tener una referencia para la creatividad y el desarrollo de los proyectos.

Otra consideración que se debe tomar en cuenta en la etapa de análisis de riesgo, según Robert Charette[7], es en relación con el futuro ¿cuáles son los riegos que pueden hacer que fracase el proyecto software? a lo que se puede responder simplemente: la ignorancia sobre la existencia de este sistema tutorial. ¿Por qué? Porque si se ha realizado todo este trabajo considerando las opiniones de los usuarios y los profesores, aceptando las críticas y corrigiendo los errores al final de cada etapa de evaluación en la espiral, el producto final será el óptimo para las necesidades de la Facultad pero, si los alumnos o los profesores no saben de su existencia todo este trabajo habrá sido en vano. Por lo que otro obstáculo a vencer consiste en la difusión de este sistema.

V.1.3 DESARROLLO

Una vez definido el tipo de información que requiere el sistema tutorial, se procedió a realizar un primer prototipo de lo que será el sistema, tomando en consideración las opiniones de los alumnos encuestados.

---

[7] Ibidem 3

Ahora bien, ¿qué es un prototipo? Un prototipo es una representación o modelo del producto de programación que, *a diferencia de un modelo de simulación, incorpora* componentes del producto real.

Hay varias razones para desarrollar un prototipo; una de ellas es ilustrar los formatos de datos de entrada, mensajes, informes y diálogos al cliente, éste es un mecanismo adecuado para explicar opciones de procesamiento y tener un mejor entendimiento de las necesidades de él.

La segunda razón consiste en explorar aspectos técnicos del producto propuesto. Con frecuencia, una decisión importante del diseño dependerá, por ejemplo, del tiempo de respuesta del controlador de un dispositivo o de la eficiencia de una algoritmo de clasificación; en tales casos, un prototipo puede ser la mejor o única manera de resolver el problema.

"El *desarrollo rápido de prototipos* se está convirtiendo en el modelo de diseño y desarrollo predominante. Propuesto por Tripp y Bichelmeyer en 1990, este modelo está basado en cinco fases: formulación de los objetivos, diseño del programa, soluciones, prototipos, revisión de las soluciones y revisión de los objetivos, tal como se muestra en la figura 4. A diferencia del modelo sistemático, añade el factor de revisión continua y actualización del producto.

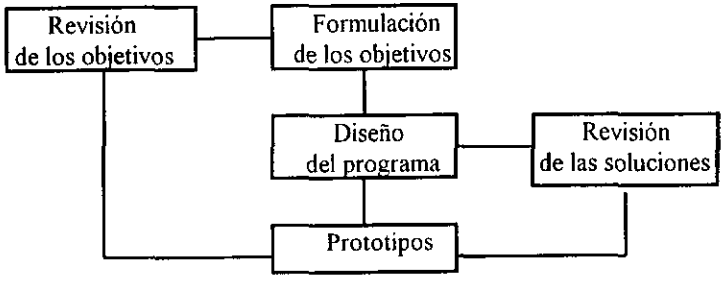

Figura 4. Modelo para el desarrollo rápido de prototipos

Una vez formulados los *objetivos* del programa, se debe buscar el mejor *diseño* educativo posible para conseguir los objetivos propuestos. Este diseño consiste en la formulación de una serie de soluciones pedagógicas que deben desarrollarse a través de un *prototipo* funcional. De este modo, es posible ir evaluando las soluciones adoptadas antes de acabar de elaborar todo el programa".

Elección de la metodología de diseño

Lo primero que hay que definir para el desarrollo del prototipo es el tipo de diseño a utilizar. Según Ian Somerville[8] existen tres tipos de metodologías de diseño.

[8] Sommerville, Ian; "Ingeniería de Software"; Editorial Addison Wesley Iberoamericana; Argentina, México; 1988

1. *Diseño funcional descendente.* El sistema se diseña desde un punto de vista funcional, empezando con una visión de alto nivel y refinándola de manera progresiva, hasta llegar a un diseño más detallado.

2. *Diseño orientado al objeto.* El sistema se ve más como una colección de objetos que como funciones que pasan mensajes de un objeto a otro. Cada objeto tiene su propio conjunto de operaciones asociadas.

3. *Diseño controlado por los datos.* Esta metodología, propuesta por Jackson y Warnier, plantea que la estructura de un sistema de software debe reflejar la estructura de los datos que éste procesa. Por tanto, el diseño del software se obtiene de un análisis de los datos del sistema de entrada y salida.

Debido a las características estructurales del sistema que se pretende construir se optó por seguir el método de diseño funcional descendiente, es decir partir de una idea general e ir descendiendo hasta un nivel más detallado de las partes que componen al sistema.

Para la construcción del prototipo, se tomó en consideración la delimitación del problema realizada en la etapa de planeación. En esta etapa se definieron el objetivo y los requisitos que debe contemplar el sistema, así como las posibles alternativas. De la misma manera, se definió el tipo de información a incluir dentro del tutorial.

Notaciones de diseño

Lo que sigue es crear una posible estructura del prototipo. Para esto se recurre a lo que son notaciones de diseño. Estas notaciones son las siguientes:

1. *Diagrama de flujo de datos.* Son diagramas que se utilizan para describir un diseño de sistemas de alto nivel; muestran cómo se transforman los datos al pasar de un componente del sistema a otro.

Estos diagramas documentan cómo los datos de entrada se transforman en datos de salida, donde cada etapa del diagrama representa una transformación diferente:

Los diagramas de flujo de datos constan de tres componentes:

- Flechas con anotaciones
- Módulos
- Terminadores
- Almacenes

Las flechas indican el flujo de la información. Los módulos indican los procesos de la información.
Los terminadores son los indicadores de entrada y salida que son los elementos que marcan el inicio y fin del flujo de datos. Los almacenes, son los registros, archivos y demás elementos donde se deposita la información.

2. Diagrama de estructura. Son gráficas de jerarquía que muestran la relación estructural de los componentes de un sistema de software.

Los diagramas de estructura describen el sistema de programación como una jerarquía de partes y lo muestran gráficamente como un árbol. Estos diagramas documentan cómo se pueden aplicar los elementos de un diagrama de flujo de datos para formar una jerarquía de unidades de programa.

Un diagrama de estructura muestra las relaciones entre las unidades de programa sin incluir ninguna información acerca del orden de activación de esas unidades. Se traza mediante tres símbolos.

- Un rectángulo con el nombre de la unidad
- Una flecha que conecta los rectángulos.
- Una flecha con un círculo, con el nombre de los datos que se pasan entre los elementos del diagrama de estructura. Las flechas con círculos suelen dibujarse paralelas a las flechas que conectan los rectángulos del diagrama.

Obtención de los diagramas de estructura.

Una etapa importante en el proceso de diseño es la transformación de un diagrama de flujo de datos en un diagrama de estructura. Esta etapa convierte las transformaciones abstractas en una jerarquía de unidades de programa, lo que representa un paso importante en la transición de una solución abstracta de un problema en una realización concreta de esa solución. La meta debe ser obtener un diseño donde las unidades de programa muestren un alto grado de cohesión y un bajo grado de acoplamiento.

La identificación de unidades poco acopladas y con alta cohesión se simplifica si se considera que las unidades son las principales encargadas de tratar con uno de cuatro tipos de flujos de datos.

1. Entrada. La unidad de programa es la encargada de aceptar los datos de una unidad de un nivel inferior en el diagrama de estructura y de pasar esos datos a una unidad de un nivel superior en alguna forma modificada.

2. Salida. La unidad de programa es la encargada de aceptar los datos de una unidad de mayor nivel y de pasarlos a otra de menor nivel.

3. Flujo de transformación. Una unidad de programa acepta datos de una unidad de nivel superior, los transforma y los devuelve a esa unidad.
4. Flujo coordinado. Una unidad es la encargada de controlar y administrar otras unidades.

El primer paso en la conversión de un diagrama de flujo de datos en uno de estructura, es identificar las unidades de entrada y salida de más alto nivel. Estas unidades son aquellas que aún están implicadas en el paso de datos hacia arriba y hacia abajo de la jerarquía, aunque lo más distante de la entrada y salida físicas. Este paso no suele incluir

todas las burbujas; las transformaciones restantes se denominan transformaciones centrales.

La identificación de las burbujas de entrada y salida de más alto nivel depende de la habilidad y experiencia del diseñador del sistema. Una manera posible de enfocar esta tarea es rastrear las entradas hasta encontrar una burbuja cuya salida sea tal que su entrada no se pueda deducir del examen de la salida. La burbuja anterior representa entonces la unidad de entrad de más alto nivel. Se usa un criterio similar para establecer la burbuja de salida de más alto nivel. El primer nivel del diagrama de estructura se produce mediante la representación de la unidad de entrada y de cada transformación central como una sola caja. La caja en la raíz del diagrama de estructura se designa como unidad de control. Este proceso de factorización puede entonces repetirse para las unidades de primer nivel en el diagrama de estructura hasta que estén representadas todas las burbujas del diagrama de flujo de datos.

El diagrama que representa cada una de las partes que componen al sistema en su primera versión está representado en la Figura 5.

En el diagrama cada rectángulo muestra una parte del prototipo, representando una página Web.

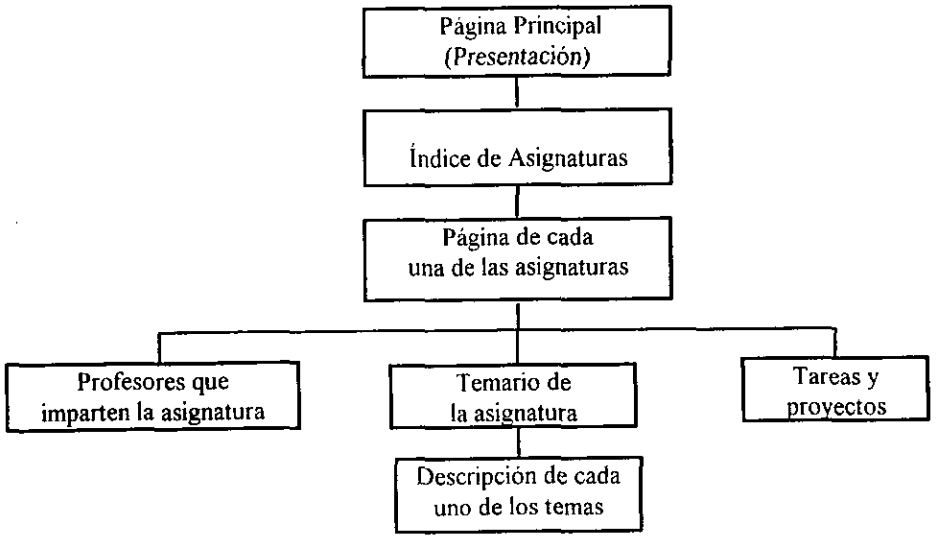

Figura 5. Diagrama de estructura del prototipo

El prototipo consta de:

- *Una página de presentación* o página principal, que contiene una liga al índice de las asignaturas optativas

- Una página índice de las asignaturas optativas, donde cada nombre es una liga a cada una de las páginas principales de las asignaturas optativas.

- Cada página principal de las asignaturas optativas contiene lo siguiente:
  - Título
  - Clave de la asignatura
  - Objetivo
  - Descripción
  - Antecedentes
  - Créditos
  - Bibliografía
  - Ligas a otras páginas: profesores, temario desglosado, tareas y proyectos
  - Botón de regreso a página principal
  - Botón de regreso a índice de asignaturas

- La página de profesores tiene la siguiente información:
  - Nombre del profesor
  - Grupo
  - Horario en que imparte la materia
  - Días en que se imparte la materia
  - Salón donde se imparte la asignatura.
  - Botón de regreso a página principal
  - Botón de regreso a índice de asignaturas
  - Botón de regreso a la página anterior

- La página de cada tema contiene
  - Nombre del tema
  - Antecedentes particulares
  - Contenido desglosado por tópicos
  - Botón de regreso a página principal
  - Botón de regreso a índice de asignaturas
  - Botón de regreso a la página anterior

- La página de Tareas y Proyectos incluye:
  - Ejemplos de las tareas y proyectos que suelen realizarse durante un semestre en cada asignatura.
  - Botón de regreso a página principal
  - Botón de regreso a índice de asignaturas

Verificación del diseño

Después de definir la estructura del prototipo, se prosiguió a revisar el diseño del mismo. Existen al menos tres tipos de revisión del diseño que pueden emplearse en el desarrollo de un sistema de software. Estos son:

1. *Revisiones informales del diseño* El trabajo de diseño de un individuo es revisado por sus compañeros.

Las revisiones informales del diseño deben ser efectuadas a intervalos regulares por un equipo de diseño. Su objetivo es detectar el máximo de errores posible en el diseño del *software*. Las especificaciones del diseño y los requisitos del software deben distribuirse antes de la revisión para su estudio por los miembros del grupo de revisión. El diseñador debe guiar al resto del equipo a través de su diseño paso por paso, explicando la función y la necesidad de cada proposición. La intención es que el grupo de revisión detecte errores e inconsistencias en el diseño y las indique al diseñador.

La tarea principal de una revisión informal del diseño es verificar la correspondencia entre la especificación del software y su diseño. Quizá sea inevitable, en esta etapa, que la revisión también detecte errores en la especificación y, tal vez, en la definición de requisitos.

2. *Revisiones técnicas formales.* El trabajo de diseño de un individuo o de un equipo es revisado por un grupo compuesto de miembros del proyecto y administradores técnicos. Al igual que en la confirmación del diseño, las revisiones técnicas formales también tienen que comprobar si el diseño se encuentra dentro del tiempo programado, el estado del desarrollo de las distintas partes del diseño, la correspondencia entre las especificaciones y el diseño, etc.

Una revisión técnica formal suele implicar un estudio menos detallado de los componentes individuales del diseño. Se relaciona más con la comprobación de las interacciones de los componentes y con la determinación de si el diseño cumple con los requisitos del usuario. Esta clase de revisión se realiza esencialmente de la misma manera que una revisión informal. El diseñador del software describe su diseño al equipo de revisión, se anotan los errores y se registran las acciones que deben tomar los distintos individuos. Además, el equipo de revisión técnica también tiene que ver la programación del diseño y como, el desplazamiento en una parte del diseño puede afectar a otros componentes.

Una tarea importante de la revisión técnica formal es decidir la forma de controlar los errores en las especificaciones del software y en la definición de requisitos que hayan sido detectados por los equipos de diseño. En algunos casos, estos errores están en la descripción de los servicios al usuario y debe informarse de ellos al contratante del software. En otros casos, se debe valorar el efecto de cambiar los requisitos o la especificación. Si el costo de un cambio es alto (esto es, si significa que se deben cambiar muchos componentes asociados), puede ser necesario conservar el error y dar instrucciones al equipo de diseño para que trabajen sobre él en vez de corregirlo.

3. *Administración de las revisiones del producto.* Este tipo de revisión tiene por objeto proporcionar información a la administración acerca del progreso del diseño de software.

Para verificar el diseño del prototipo se utilizó la revisión formal del diseño, esto es, entre los integrantes del equipo de desarrollo se hizo una revisión detallada del diseño del prototipo del sistema. Para esta revisión, se tomaron en cuenta los factores de la calidad

del software. De común acuerdo los integrantes del equipo aceptaron el diseño elaborado.

Metodología de la programación

Enseguida hay que definir el tipo de programación a seguir. El proceso de desarrollo de un programa a partir de un diseño de software puede enfocarse de dos formas: el desarrollo descendente y el ascendente. *Si el programa se considera como una jerarquía* de componentes, entonces el desarrollo descendente implica empezar en el tope de la jerarquía y trabajar hacia abajo, mientras que en el desarrollo ascendente se empieza en el nivel más bajo y se prosigue hacia arriba.

Para la construcción del prototipo de este sistema tutorial se escogió el desarrollo descendente, es decir se partió de niveles más altos a niveles más bajos.

Estilo de programación

Nombres de los programas

Para facilitar la programación de los módulos que componen al sistema hay que utilizar nombres que vayan acordes con las entidades del mundo real. De este forma se facilita el mantenimiento del software.

En este caso, como la programación se hizo en HTML para diseñar páginas Web, se utilizaron los siguientes nombres para facilitar la programación:

A la página principal del sistema tutorial se le denominó *princip*

A la página que contiene el índice de las asignaturas optativas se le llamó *asignaturas*.

A *la página de cada materia optativa se le denotó por tres letras que resumieran el* nombre de la asignatura optativa, generalmente sus iniciales, por ejemplo para llamar a la página de Graficación por Computadora utilizábamos las letras *gpc*

A la página que contiene información de los profesores se les denotó con las primeras cuatro letras de la palabra profesores más las tres letras que definen a la optativa, por ejemplo para el nombre de la página de profesores de Bioingeniería se utilizaba la palabra *profbio.*

A la página que contiene información del temario de la asignatura se le denotó con las primeras tres letras de la palabra temario más las tres letras que definen a la optativa, por ejemplo para el nombre del temario de Calidad se utilizaba la palabra *temcal.*

A la página que contiene la información de las tareas y proyectos desarrollados en la *materia se utilizan las iniciales de tareas y proyectos más las letras que caracterizan a la* materia, por ejemplo para la página de Diseño Asistido por Computadora se escribe *typdac*

A cada página correspondiente a cada tema comprendido en el temario de la asignatura se le denota con las letras que representan a la materia más la inicial de tema y el número de tema en cuestión. Por ejemplo, para definir el tema uno del temario de la asignatura Procesamiento Digital de Imágenes, utilizamos la palabra *pdit1*

Construcciones de control en los programas

Como el prototipo está desarrollado en un lenguaje que maneja hipertexto, cada página se crea de manera aislada, sin embargo a la hora de armar el sistema estas páginas deben estar relacionadas entre sí. Para lograr esto, se hizo uso de ligas en las páginas para permitir llamar a otras páginas y ponerlas en activo; de esta forma se logra la unión de las diferentes páginas construidas.

Instrumentos de Software

Instrumentos para la preparación de programas.

Como el sistema se construyó en un lenguaje para realizar páginas web, es necesario realizar modificaciones a los documentos creados para el sistema. Como este lenguaje se basa en la manipulación de hipertexto, para la elaboración y modificación de páginas Web sólo basta con tener un editor de texto.

Para cambiar el contenido del programa de desarrollo de cada página incluida en este prototipo se utilizó el bloc de notas.

Instrumentos para la traducción de programas

El segundo elemento para la elaboración del prototipo es un software donde se pueda visualizar el desplegado de las páginas construidas.

Para observar cómo se despliega cada página creada, se utilizó el navegador Netscape para visualizar los cambios que se realizaban dentro de la construcción de cada página. El navegador Netscape es un software utilizado para observar páginas creadas en el Web, mejor conocida como red mundial de información. Una de las ventajas de este programa es que no sólo se pueden ver páginas dadas de alta en algún servidor o sitio web, sino cualquier página creada en el lenguaje html, aún cuando no esté cargada en la red. Para la creación de páginas sólo se editan documentos que contengan etiquetas del lenguaje de manejo de hipertexto. Estos documentos son muy fáciles de crear, sólo se necesita un editor de texto, y conocer las etiquetas de este lenguaje para manipular el texto.

Elección del lenguaje de programación

Debido a que el sistema tutorial se diseñará para colocarlo en Internet, se decidió utilizar el lenguaje HTML (Lenguaje de Manipulación de HiperTexto) para la creación del prototipo. La idea en la que se basa el hipertexto es que en lugar de leer un texto siguiendo una estructura rígida y lineal (como es un libro), es posible avanzar de un punto

a otro fácilmente, obtener más información, regresar al primer punto, brincar hacia otros temas y desplazarse (navegar) por el texto según los intereses que se tengan.[9]

Otra de las razones por las cuales se optó por utilizar hipertexto es porque este lenguaje resulta fácil de manipular dentro de la red mundial Internet, y permite de una manera *sencilla y rápida realizar ligas, gráficas, texto utilizando varios tipos de formato, insertar* imágenes, etc.

HTML no es un lenguaje complicado, pero requiere que el programador siga las reglas específicas para las etiquetas en cada parte del documento. Utilizando HTML se puede definir la estructura de un documento para ser leído por cualquier browser presente o futuro; esto hace posible el desarrollar información sin importar la versión o el tipo de browser en que estemos trabajando, ya sea Lynx, Cello, Netscape o Mosaic, por mencionar algunos. Otra de las ventajas de HTML es que éste crea texto ASCII claro y *sencillo con caracteres de control y código binario incrustado; así el desarrollador puede* fácilmente interpretar o editar un archivo HTML en un simple editor de texto.[10]

Como el Web crece en la medida en que avanza la tecnología y se incrementa su difusión como medio de comunicación eficaz y económico, HTML ha desarrollado varias versiones desde sus inicios teniendo por ejemplo HTML 2.0 nacido en Noviembre de 1995 con un conjunto de instrucciones sencillo y muy limitado por lo cual se tuvo la necesidad de crear otra versión. HTML 3.0 comienza su difusión a principios de 1996, pero es desplazado por la versión 3.2 la cual es reconocida por el W3C (World Wide Web *Consortium) el 14 de enero de 1997. No obstante, el interés y motivación de los* programadores y usuarios de páginas de Internet continúa avanzando y los creadores de este lenguaje se ven en la necesidad de propagar una nueva versión, HTML 4.0 que cuenta con nuevas etiquetas que permiten aumentar la interactividad con el usuario conservándose la fácil interpretación del browser. El browser es el navegador Netscape o cualquier software utilizado para cargar páginas web. Esta versión anunciada el 8 de julio de 1997 también es aceptada por el W3C.

Por añadidura a este trabajo oficial sobre HTML, los browsers han creado sus propios *aditamentos para HTML. Algunos cambios son eventualmente adoptados por las* recomendaciones del HTML W3C, otros se convierten en aspectos exclusivos del browser que los interpreta. Las versiones de browsers interpretadores de Lenguaje de HiperTexto también han sufrido cambios en la medida que el mercado en Internet y los desarrolladores de páginas Web lo sugieren.

Para saber más sobre este lenguaje de programación recurrir al Anexo 1.

---

[9] Lemay, Laura; "Aprendiendo HTML 4 para WEB en una semana"; Prentice Hall Hispanoámericana; México; 1998.

[10] December, John and Ginsburg, Mark; *"HTML & CGI UNLEASHED"*; Editorial Sams; *EEUU; 1995.*

Pruebas del sistema.

Después de desarrollado el prototipo del sistema tutorial y antes de subirlo a la red para iniciar la etapa de evaluación de la primera vuelta de la metodología, se procedió a hacer una prueba del sistema para ver cómo funcionaba.

Para esto, después de programadas las páginas que componen al sistema, realizadas las ligas pertinentes entre estas páginas e incluida la información que se pretende proporcionar a los usuarios, se realizó una prueba de cómo se vería el tutorial una vez colocado en la red.

La prueba consistió en conjuntar todas las páginas creadas, ligarlas y observar su desplegado en algún navegador de internet para ver su apariencia, funcionalidad y otras características importantes.

Como se mencionó anteriormente, para visualizar la presentación del sistema se cuenta con un software para visualizar páginas web, en este caso el navegador Netscape. Así, dicha prueba del prototipo consistió en ir cargando en este navegador las páginas creadas en el prototipo e ir observando la presentación del mismo. Se verificó que la estructura fuera la adecuada, permitiendo recorrer todo el sistema sin mandar al usuario a algún lugar del que ya no pudiera retroceder, que estuvieran completas todas las páginas que conformarían el primer prototipo, que la sintaxis fuera la correcta, que se manejara una buena presentación, en fin que el prototipo estuviera bien detallado

Esta prueba permitió corregir errores principalmente de presentación, y uniformar el contenido de cada una de las páginas creadas.

Una vez terminada la fase de prueba y de depuración del prototipo, se consideró que estaba listo para la primer evaluación de los usuarios

Así, se concluyó la fase de desarrollo y se procedió con la última etapa de la primera vuelta, es decir, la etapa de evaluación por parte del cliente

V.1.4. EVALUACIÓN DEL CLIENTE

Los objetivos de las actividades de verificación y validación son valorar y mejorar la calidad de los productos del trabajo generados durante el desarrollo y modificación del software.

Hay dos tipos de verificación: formal y del ciclo de vida. Esta última consiste en el proceso de determinar el grado en que los productos de trabajo de una fase dada del ciclo de desarrollo cumplen con las especificaciones establecidas durante las fases previas. La verificación formal es una rigurosa demostración matemática de la concordancia del código fuente con sus especificaciones. La validación es la evaluación del software al final del proceso de desarrollo del software para determinar su conformidad con los requisitos. Boehm expresa estas definiciones de la siguiente manera:

Verificación: "¿Estamos construyendo correctamente el producto"?
Validación: "¿Estamos construyendo el producto correcto"?

Para el sistema que se está desarrollando, la evaluación del sistema fue realizada por parte del cliente, en este caso alumnos de la carrera de Ingeniería en Computación que son a quienes está dirigido el sistema.

De acuerdo con los resultados de esta evaluación realizada por los usuarios, se pudieron obtener conclusiones que permitieron realizar una valoración del sistema. Es decir, al saber si el usuario tiene el producto correcto, se sabrá que se está construyendo adecuadamente el producto. En otras palabras, de los usuarios del sistema se obtendrá información sobre el cumplimiento de los objetivos y requisitos planteados en la etapa de planeación y, de esta manera, se comprobará que se está desarrollando adecuadamente el sistema o en su defecto, si es necesario realizar modificaciones al diseño del sistema.

La alta calidad no se puede lograr sólo mediante la prueba del código fuente. Aunque un programa debe estar, en su totalidad, libre de errores, rara vez sucede en el caso de grandes productos de software. En el supuesto de que los errores del código fuente fueran la única medida de la calidad, las pruebas, por sí solas, no podrían garantizar la ausencia de errores de un programa. Una máxima muy conocida establece que el número de errores ocultos en un programa es proporcional al número de los ya descubiertos. Esto es porque se tiene más confianza en programas donde no se han detectado errores ocultos después de pruebas minuciosas, y menos confianza en programas que ya han sufrido correcciones. La mejor manera de minimizar el número de errores en un programa es encontrarlos y eliminarlo durante el análisis y diseño, de modo que se introduzcan pocos errores al código fuente

La verificación y validación implican la valoración de los productos de trabajo para determinar el apego a las especificaciones. Se deben examinar los requisitos para asegurarse que concuerden con las necesidades del usuario, así como con las restricciones del ambiente y los estándares de notación.

Los errores ocurren cuando cualquier aspecto de un producto de software es incompleto, inconsistente o incorrecto. Las tres grandes clases de errores del software son las de requisitos, de diseño y de implantación. Los errores de requisitos se provocan por una propuesta incorrecta de las necesidades del usuario, por falta de una especificación completa de los requisitos funcionales y de desempeño, por inconsistencia entre los requisitos y por requisitos no factibles.

Los errores de diseño se introducen por fallas al traducir los requisitos en estructuras de solución correctas y completas, por inconsistencias tanto dentro de las especificaciones del diseño como entre las especificaciones del diseño y los requisitos. Un error de requisitos o un error de diseño, que no se descubre sino hasta las pruebas del código fuente, puede ser muy costoso de corregir. De modo que es importante que la calidad de los requisitos y de los documentos del diseño se valoren pronto y con frecuencia.

Los errores de instrumentación son los cometidos al traducir las especificaciones del diseño en código fuente Estos errores pueden ocurrir en las declaraciones de datos, en las referencias a los datos, en la lógica del flujo de control, en expresiones computacionales, en interfaces entre subprogramas y en operaciones de entrada/salida.

Una vez realizada la etapa de desarrollo, con la creación del prototipo, se prosiguió con la evaluación de esta versión del sistema.

Como se mencionó la evaluación del sistema fue hecha por parte del cliente, en este caso alumnos de la carrera de Ingeniería en Computación.

Para realizar esto fue necesario colocar el prototipo desarrollado en Internet. Esto se logró gracias al apoyo del administrador del servidor Web que labora en el Laboratorio de Computo del Departamento de Ingeniería en Computación. El administrador Web del Departamento subió las páginas que comprenden el prototipo del sistema al servidor Odin, que es un servidor Web, con lo cual el sistema pudo ser visto desde cualquier navegador. Como este prototipo estaba a prueba, se incluyó un password para poder acceder al sistema. Esto como una medida de exclusividad; una vez que se libere el *sistema su consulta será libre.*

Después de lo anterior, se formó un grupo piloto que tuviera acceso al sistema tutorial vía Internet para navegar en él, y de esta manera lo observará, criticará y emitiera sus opiniones acerca del mismo.

El grupo piloto lo conformaron los alumnos de la asignatura Ingeniería de Programación del Ing. Orlando Zaldívar Zamorategui, los cuales durante una sesión que tuvo una duración aproximada de una hora y media, se conectaron a Internet y desde ahí visualizaron el prototipo realizado.

Después de haber navegado a través del sistema tutorial, se les pidió que proporcionarán sus comentarios sobre el sistema.

Entre las observaciones y sugerencias que se reunieron se encuentran las siguientes:

- Errores de sintaxis en algunas partes del temario.
- Términos que no se comprenden por la falta de familiaridad con los mismos.
- Mejorar el tapiz o fondo
- Agregar mayor información
- Describir la manera en que los profesores se desempeñan en esta *actividad.*
- Colocar gifs animados
- Dejar un espacio para que los alumnos hablaran acerca de sus experiencias.
- Acentuar mayúsculas
- Checar la liga para recibir correo
- Usar gráficos

- Uniformizar el color de los temas y subtemas
- Definir que es ISO
- Quitar bordes a las imágenes usadas como ligas
- Anotar ventajas y desventajas de cada una de las asignaturas optativas
- Colocar iconos
- Checar el color de las letras
- Utilizar un tamaño de letra mucho mayor
- Usar imágenes y colores que completen el contenido de la información
- Justificar los textos
- Buscar uniformidad en las presentaciones de los temas, subtemas y bibliografía
- Cambiar el icono de indice.gif
- Buscar un icono para cada asignatura
- Incluir frames
- Resaltar página principal
- Checar formatos de horarios

A partir de los comentarios de los alumnos vertidos sobre el sistema tutorial, se puede concluir que son dos los aspectos principales a considerar para mejorar la versión del sistema. El primero de ellos es la información manejada dentro de la primera versión. La información incluida en esta versión del sistema es útil, sin embargo no es suficiente para que el usuario conozca el contenido de cada una de las asignaturas optativas. En la medida de lo posible, el estudiante desea conocer más acerca de cada asignatura optativa, de la forma en que la imparten los profesores, además de la oportunidad de un espacio en donde expresar sus opiniones y experiencias. Así como tener contacto con los profesores y alumnos vía internet.

El segundo aspecto a considerar es la presentación del prototipo. En este sentido existen sugerencias muy interesantes a tomar en cuenta para mejorar esta versión del sistema, como por ejemplo uniformizar los títulos de los temas, subtemas, elegir fondos que le den mayor atractivo al tutorial, utilizar frames, incluir imágenes. Estos puntos pueden ser muy útiles para mejorar la presentación del primer prototipo. Los frames son elementos del lenguaje html, que permiten dividir la pantalla en dos o más partes, según la conveniencia para hacer más dinámica la presentación dé las páginas

Ahora bien, ya que fue evaluado el sistema se puede realizar una verificación del sistema, es decir, analizar si se está construyendo correctamente el sistema. Para realizar este análisis se tomarán en cuenta los objetivos, requisitos y metas definidos en la etapa de planeación.

Se sabe que el objetivo principal de este sistema tutorial consiste en proporcionar a los alumnos de la carrera de Ingeniería en Computación información suficiente sobre las asignaturas optativas para que las conozcan y puedan elegir cuáles cursar. Si bien este objetivo es difícil de cubrir en esta primera vuelta de la metodología, se puede hacer uso de los comentarios reunidos para ver si se está cumpliendo o es necesario replantear nuevamente los requisitos y metas del sistema.

Otro de los objetivos derivados del principal fue el hecho de que si los alumnos ya eligieron sus asignaturas optativas, este sistema les pueda servir como recurso didáctico básico o como refuerzo o complemento de lo que se ve en las clases normales, o como alternativa de repaso. Además, se busca que, en caso de que alguien haya decidido qué asignaturas optativas cursar, ya sea con ayuda de este tutorial o por otro medio, el presente sistema le pueda proporcionar información suficiente para conocer las demás asignaturas y, así tener un panorama general de todas las asignaturas que se pueden cursar de forma optativa dentro del plan de la carrera. Como estos objetivos son derivados del objetivo principal, al cubrir el objetivo principal se habrán cubierto también estos objetivos.

La principal meta en esta primera vuelta de la metodología era establecer el tipo de información que se introducirá en el sistema para cumplir con el objetivo principal. Esta meta se logra de manera adecuada con la aplicación de la encuesta a los alumnos de la carrera sobre qué tipo de información requieren ellos para conocer el contenido de cada asignatura optativa. Las encuestas arrojaron resultados interesantes que nos llevaron a tener suficiente información para desarrollar lo que fue el primer prototipo del sistema.

Otra meta era saber la opinión de los usuarios acerca de lo que debe contener un sistema tutorial que trata sobre las asignaturas optativas. Esta meta se cumplió al cubrir la meta principal, por medio del sondeo realizado entre los estudiantes de la facultad. En dichas encuestas, los alumnos nos hicieron llegar sus opiniones sobre lo que les gustaría conocer de las asignaturas optativas.

Finalmente, la última meta era buscar la forma de que un grupo de alumnos pudiera ver el primer prototipo, y diera su opinión del mismo. Esto se logró al colocar el primer prototipo del sistema en internet y conformar un grupo para que lo observara y diera su opinión.

Como se puede determinar, para alcanzar el objetivo principal es necesario incluir mayor información sobre las asignaturas optativas, principalmente relacionada con los temas de la materia y los trabajos desarrollados en la misma. Para la segunda vuelta, se deberá hacer una planeación adecuada para cumplir con este objetivo.

Ahora es conveniente hacer un análisis de los requisitos establecidos en la planeación, para averiguar si se están cumpliendo. Los requisitos que establecimos para el sistema tutorial son los siguientes:

- Información

Se buscaba que la información manejada en esta primera versión del sistema fuera suficiente para llamar la atención del alumno acerca del contenido de las asignaturas optativas que puede cursar, sin profundizar mucho en ella

De acuerdo con lo que se pudo observar en la evaluación del sistema este requisito fue cumplido, ya que aun cuando la información manejada en este primer prototipo no es suficiente para cumplir con el objetivo principal, si se pudo observar el interés que despertó la realización del trabajo entre los usuarios que pudieron ver el sistema. Entre otros comentarios, les pareció una idea adecuada el

desarrollar este tipo de proyectos, porque permite al alumno contar con una herramienta para hacer uso de una oportunidad que presenta el plan de estudios de la carrera, que es el cursar asignaturas optativas.

- Presentación

La presentación manejada en esta primera versión del sistema debe ser agradable pero sin profundizar en el detalle. No manejar imágenes en este primer prototipo, simplemente buscar una adecuada combinación entre fondos y textos, que le permitan al usuario leer adecuadamente la información desplegada sin desviar su atención.

Este requisito no fue del todo cumplido, ya que el prototipo presentaba problemas de una mala combinación entre el color del fondo y del texto, errores ortográficos, etc., como puede verse en los comentarios recopilados. Por esta razón, este requisito tendrá mayor atención en la segunda vuelta de la metodología

- Organización

A diferencia de la presentación, en esta primera vuelta se debe poner mucho énfasis en la organización del sistema. La estructura de este primer prototipo debe ser de tal forma que le permita al usuario navegar tranquilamente por donde quiera, evitando llevarlo a algún sitio del cual ya no pueda regresar o salir. De la elaboración de esta estructura depende que no haya muchas modificaciones al sistema en versiones sucesivas del mismo.

Este requisito fue cumplido adecuadamente. La estructura que maneja esta primera versión del sistema permite al usuario navegar tranquilamente adonde el desee, dejándole la oportunidad de regresar al punto de partida. Esta estructura permitirá hacer cambios en los sucesivo, pero de una manera más sencilla debido a la organización de las páginas que comprenden al tutorial. La estructura puede ser mejorada, aunque puede ser ya la idónea.

Así, se concluyó la primera vuelta del modelo en espiral, en la cual se alcanzó a cubrir de manera parcial el objetivo del sistema, mediante la construcción de un prototipo.

Pues bien, como se está manejando una metodología basada en el modelo en espiral, y según se analizó esta primera versión (prototipo), no cumple con todos los objetivos, es necesario decidir si se trabajará el sistema real con base en el prototipo, haciendo las mejoras pertinentes, o bien, se empezará a desarrollar el sistema desde un principio.

De acuerdo a lo que se observó, aunque este prototipo no cumple con las características deseadas, sí permitió tener un buen avance en el logro de éstas, por lo que se puede concluir que el prototipo realizado cumplió su objetivo. Ahora para la segunda vuelta, donde se comenzará el desarrollo del sistema real, habrá que tomar en cuenta las opiniones más importantes de los alumnos, así como las conclusiones que se obtuvieron a partir de ellas.

V.2 SEGUNDA VUELTA

Con esto daremos inicio a lo que será la segunda vuelta de la metodología a seguir. En esta segunda iteración, se pretende cubrir la mayor parte del objetivo principal, realizando las modificaciones pertinentes al sistema para que cumpla con las características requeridas.

V.2.1 PLANEACIÓN

Definición del problema.

El problema general sigue siendo el mismo, es decir, desarrollar un sistema tutorial sobre las asignaturas optativas del plan de estudios de la carrera de ingeniería en computación.

El problema particular en esta segunda vuelta, es comenzar con la programación del sistema real tomando en cuenta los comentarios obtenidos gracias a la realización y evaluación del prototipo.

Analizando las sugerencias obtenidas se concluyó que los principales defectos que presentaba el primer prototipo eran referentes a la presentación, sobre todo en el texto utilizado, es decir errores ortográficos, tamaño de letra, color de letra, poca uniformidad entre los títulos manejados, etc.

Otro punto a considerar, es el tipo de información que se manejará dentro del tutorial. De acuerdo a las opiniones obtenidas, la información manejada dentro del prototipo es útil para conocer de manera superficial el enfoque de cada materia, sin embargo es necesario incluir otro tipo de información, que nos permita describirle al alumno las características principales de cada asignatura, y sobre todo buscar un reflejo, lo más apegado, a lo que se maneja en la realidad, sobre todo en los temarios impartidos dentro de las asignaturas.

Es importante no olvidar los comentarios positivos (puesto que también los hubo y de manera significativa), en esta primera evaluación del sistema. Considerar estas observaciones servirá para ver qué aspectos del prototipo se deberán conservar al desarrollar el sistema tutorial.

En resumen, se espera que al término de esta segunda vuelta el sistema tutorial esté casi terminado, completamente revisado y evaluado para poder colocarlo con toda confianza en la red mundial, Internet.

Metas

Las metas que se persiguen con la realización de esta segunda vuelta de la metodología son principalmente:

- Desarrollar el sistema real tomando en consideración las conclusiones a las que se llegó en la etapa de evaluación del prototipo realizado para alcanzar el objetivo principal planteado.

Requisitos

Entre los requisitos que debe contener el sistema que se construirá en esta vuelta, se encuentran.

- Manejo de suficiente información
  El tipo de información dentro del sistema debe ser la necesaria y suficiente para que el alumno conozca bien los contenidos de cada asignatura optativa. Se debe utilizar información que complemente a la que se ha manejado en la primera versión del sistema (prototipo), sobre todo, hay que desarrollar con más detalle los temas que se imparten en cada materia, y anexar en lo posible: tipos de tareas, trabajos, prácticas y proyectos que se realizan en cada asignatura optativa, con el fin de que el alumno tenga una mejor referencia acerca de la asignatura.

- Mejorar presentación
  En este sentido, de acuerdo con las conclusiones obtenidas, se debe buscar una mejor presentación en las páginas que comprenden el sistema. Para lograr esto, se pueden incluir fondos en las páginas, buscar un tipo de letra que sea legible pero sin llegar a ser llamativa, podemos manejar imágenes y gifs que le den mayor presentación al sistema, pero evitando que éstas desvíen la atención del usuario, así también, manejar diferentes colores en los textos, las ligas y datos importantes para facilitar el aprendizaje al alumno.

- Una buena organización
  Al respecto, el sistema debe contener una estructura que permita al usuario su fácil utilización. La estructura manejada en la versión prototipo del sistema logró en gran manera satisfacer este punto por lo cual, en cuanto a organización se realizarán únicamente las modificaciones pertinentes que hagan al sistema más interactivo.
  Se propone el uso de frames en las páginas del sistema puesto que así se mejorará significativamente la organización y se facilitará la navegación a través de él.

Opciones de solución

Para cumplir las metas y los requisitos planteados en esta etapa y con ello resolver el problema delimitado, es necesario enfocarse en las conclusiones obtenidas durante la etapa de evaluación de la primera vuelta, poniendo énfasis en aquellas opiniones que nos permitan cumplir con la finalidad planteada en esta etapa.

Es decir que, para facilitar la tarea, se analizarán tres puntos que se consideran los más esenciales y que engloban todos los aspectos relacionados con el tutorial. Estos puntos, contemplados anteriormente en los requisitos, son:

- La organización
- La presentación
- La información.

En la *organización* se cubre lo referente a la estructura del sistema. Aquí es importante considerar el orden en que se accede a las páginas, las ligas que existen en el tutorial para accesar a una u otra página, la manera en que se muestra la información, es decir, el orden que tiene la información dentro de la página, la estructura o el formato que tienen las páginas de acuerdo al tema que traten, etc.

La organización contempla muchas características importantes que es necesario planear correctamente para evitar enredos y/o confusiones de los usuarios al querer consultar el sistema tutorial.

En la *presentación* se cubre lo relacionado con la apariencia que se dará al sistema. Aquí se manejan aspectos relacionados con la vista que tendrá el usuario cuando visite una u otra asignatura que le interese.

La presentación es punto muy importante ya que debemos recordar que una imagen vale más que mil palabras; si el usuario que visita por primera vez el sistema lo encuentra atractivo, esto significará un punto a favor pues, aunque sea por la apariencia, (en primera instancia) se atreverá a navegar por cada una de las páginas del tutorial. Los fondos e imágenes que se utilicen, el tamaño y color de las letras en los títulos y los contenidos, el formato de distribución en las páginas, la facilidad en localizar mediante una vista rápida lo que se busca; son algunas de las características que se deben considerar para crear una mejor presentación.

En la *información* se cubre lo referente con el contenido del tutorial. Aquí se maneja todo aquello que se pretende informar al usuario del sistema tutorial. El material del cual se extraiga la información debe ser veraz y actual; pero de toda la información que se recopile debe hacerse una revisión con la finalidad de resumir lo más importante. Esto permitirá concretizar ideas sin tener redundancia en el texto.

La información es un punto que debe tratarse cuidadosamente, pues si bien no se trata de poner una Biblia de información en el sistema, si es necesario seleccionar correctamente aquellos datos que muestren un panorama general de la asignatura y por lo mismo den al usuario una idea más objetiva de lo que se aprenderá en tal o cual asignatura.

Cubriendo estos tres puntos, se logrará cumplir satisfactoriamente con las metas propuestas y al mismo tiempo se resolverá el problema planteado.

Después de analizar cada una de las opciones de solución y tomando en cuenta las opiniones de los usuarios, se decidió no cambiar nada en cuanto a la organización por el

momento, ya que se observó que la estructura manejada en el prototipo es hasta ahora la más conveniente y acorde a los fines que se persiguen y que el orden en que van apareciendo las páginas permite navegar con facilidad a través de ellas. Por otra parte, todas las páginas de las asignaturas optativas están uniformizadas para facilitar la creación y desarrollo del sistema tutorial; de esta manera si se realiza algún cambio en una página de una materia optativa, deberá hacerse la misma corrección en las demás páginas correspondientes a cada asignatura lo cual permite un trabajo organizado que hace uniforme la elaboración de las páginas por módulos.

El diagrama de la Figura 6 muestra los módulos en que se encuentran segmentadas cada una las páginas de las asignaturas optativas. Cada cuadro representa un módulo de la materia.

A partir del mismo análisis, se decidió realizar algunos cambios importantes referentes a la presentación del tutorial, como son:

1. Colocar fondos adecuados al tema en cada una de las páginas utilizadas en el tutorial
2. Uniformizar títulos, contenidos, fondos, organización y presentación de cada página
3. Utilizar frames en cada página principal de cada materia optativa
   El uso de frames repercute en la organización del tutorial, pero al implementar frames facilitamos la navegación sobre las asignaturas optativas, ya que desde la página que contenga los frames podemos acceder a cada una de las secciones que se manejan para las asignaturas optativas
4. Utilizar un color y un tamaño de letras que permitan visualizar mejor la información, además de buscar una buena combinación entre los fondos, letras e imágenes que se utilicen para fomentar en el usuario su gusto e interés al visitar este sistema.
5. Utilizar sólo las ligas necesarias para navegar a través de todo el tutorial.

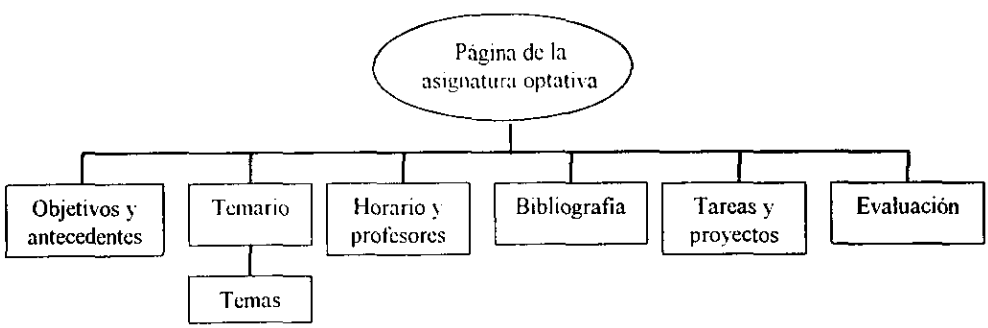

Figura 6. Diagrama de estructura del sistema tutorial

En lo referente a la información, se decidió incluir el contenido de los temas de que comprende una asignatura, así como las tareas y proyectos que se realizan durante el

curso. Esta información complementaria a la que existe en la versión prototipo del tutorial, es decir, los objetivos, los antecedentes, la bibliografía, la descripción de lo que trata cada asignatura, el temario y el horario de los profesores y grupos que las imparten. Con esta información se pretende describir un panorama más amplio sobre las asignaturas optativas.

Planeación del proceso de desarrollo

Como se definió en la etapa de planeación de la primera vuelta, el método de desarrollo a utilizar es una combinación de la creación de prototipos y el método de versiones sucesivas, es decir se creó un primer prototipo con los requisitos definidos por el cliente, el cual fue evaluado para ver si cumplía con las características necesarias; si se satisfacen los requerimientos del cliente, se utiliza como modelo para el desarrollo del sistema real, de lo contrario se vuelve a hacer un diseño para construir una nueva versión del sistema que contenga las mejoras marcadas en el prototipo y que intente cumplir con los fines para los cuales será creado.

De la evaluación se concluye que la versión prototipo se aproxima sobremanera a lo que se quiere en la parte de organización. Será necesario hacer mejoras en cuando a la presentación pero habrá que poner mayor atención, en la parte de información que deberá contener el tutorial.

Verificación y validación

Como se planteó en la metodología utilizada para el desarrollo del sistema tutorial, después de la etapa de desarrollo se incluye una de evaluación. Esta evaluación permite saber si la versión creada cumple con las características deseadas del sistema, si es necesario mejorarla o en un caso extremo construir una versión diferente. Esta etapa de verificación y validación, nos permite también ratificar los objetivos y requisitos planteados durante la etapa de planeación.
Una vez que se ha concluido la etapa de evaluación, en la que se rectificó si se cumplió o no con el objetivo principal, se definieron, durante esta segunda etapa de planeación, las opciones para alcanzar la meta propuesta y reunidos los nuevos requisitos que deben agregarse al sistema que se construirá en esta segunda vuelta, se procederá a la siguiente etapa de la metodología, el análisis de riesgo.

V.2.2 ANÁLISIS DE RIESGO

En lo que se refiere a la organización del sistema, hay que revisar que las páginas a las cuales se realice un enlace o liga, existan, para evitar que aparezcan errores al desarrollar el proyecto final. Con esto, surge un problema, si el sistema crece, el programador deberá conocer los estándares utilizados en la nomenclatura de cada una de las páginas para que, siguiendo esas reglas, sea más fácil añadir una nueva liga o incluir nuevas páginas al sistema. Para ello se continuará asignando nomenclatura similar a la empleada en el prototipo.

En cuanto a la presentación, como se tiene contemplado utilizar fondos e imágenes, es necesario, primeramente, conseguirlos ya sea en libros, revistas, Discos Compactos o en Internet; enseguida, se debe verificar que sean los adecuados y finalmente, buscar la combinación más agradable al usuario y más adecuada con el tema. En este caso, se presenta un problema: las imágenes (figuras y gráficas) deberán complementar la información y no distraer la atención del usuario. Los fondos de las páginas por su parte, deben contribuir a mejorar el aspecto, es decir, la visión general de la página.

Para conseguir los fondos e imágenes adecuados se contemplaron como alternativas:

- buscar en Internet en aquellas páginas que trataran temas que se incluyen en las asignaturas optativas (esto además de proporcionarnos imágenes de acuerdo al tema sirve para recopilar información),

- buscar en CDs que tuvieran demostraciones de mosaicos para computadoras y en aquellos que promocionan paquetes para manejo de gráficos o simplemente ofrecen varios modelos de texturas para establecer como fondos en la computadora,

- Conociendo de qué trata cada tema, diseñar una imagen que complemente la información y facilite la comprensión de la asignatura, es decir, crear nosotros mismos los gráficos.

Una vez encontrados los fondos idóneos, se tendrá que probar cada uno de ellos hasta encontrar el más adecuado para cada página. Esta actividad tomará algo de tiempo pero no altera el tiempo destinado para la elaboración del sistema dado que al realizar la planeación se han considerado estos detalles.

Después de haber definido el fondo y las imágenes para cada una de las páginas, se procederá a adecuar el color de la letra y de las ligas al fondo utilizado así como la distribución de los títulos, textos, ligas e imágenes en el contexto de la página. Para realizar esta actividad se cuenta con el apoyo del editor que contiene la versión de Netscape instalada en el Laboratorio de Cómputo. El uso de este editor facilita la tarea por realizar en la presentación de una página HTML. Éste permite manejar el tamaño, tipo y color de las letras, por lo que las pruebas para mejorar la presentación pueden llevarse a cabo de manera rápida y sencilla lo cual permitirá que no se sobrepasen los tiempos establecidos para esta actividad y se tenga la mejor presentación para el cliente.
Otro aspecto que debe cubrirse es la utilización de los frames ya que es necesario, en primera instancia, desglosar la información sobre cada materia optativa en secciones cada una de las cuales contendrá información concerniente a la asignatura optativa en turno. De esta manera, se desarrollarán páginas para cada tópico en particular, es decir, habrá una página con la información de los profesores, otra con el temario, otra con las tareas y proyectos, una más para evaluación y así sucesivamente, conforme se vayan necesitando. Con estos cambios, el sistema va a crecer -eso es inevitable- pero ese crecimiento será controlado, pues en primer lugar estará limitado por el número de asignaturas optativas que se imparten para la carrera de Ingeniería en Computación que son un total de once, así como por los temarios que establece la División de Ingeniería

Eléctrica (DIE) para cada materia. Para controlar más este crecimiento, desde un inicio se ha establecido una nomenclatura a seguir para nombrar los archivos que conforman el sistema así, cada nueva página debe tener un identificador alusivo a la información que contiene. Por ejemplo, la página donde esté la bibliografía relativa a la materia Procesamiento Digital de Señales se nombrará *bibpds*, bib porque son iniciales representativas de bibliografía y pds las iniciales de Procesamiento Digital de Señales.

Las nomenclaturas que se utilizan para este sistema se explican con más detalle en la Primera Vuelta en la sección Estilo de programación, Nombres de los programas.

En lo que se refiere a la información que contendrá el tutorial, se tienen algunos problemas por resolver.

Primero, es necesario realizar una correcta búsqueda de información, tarea quizá complicada, si se considera que se trata de un total de once asignaturas con un número significativo de temas en cada una. Si a esto se agregan los proyectos y tareas que se desarrollan normalmente por semestre en cada una de ellas, la búsqueda de la información se vuelve más compleja, ya que estos datos sólo se obtienen cursando la asignatura, preguntando a compañeros que la hayan cursado o pidiendo la información al profesor que imparte la materia. En este sentido, la primera solución (cursar la asignatura), implica invertir el tiempo que dura el semestre para poder seguir avanzando y sólo se resolvería el problema para una asignatura. La segunda solución, implica conocer o buscar compañeros que hayan cursado asignaturas optativas y que recuerden qué tareas y proyectos realizaron o bien, que tengan guardados sus apuntes y que éstos a su vez sean claros, veraces y concisos. La dificultad radica entonces en encontrar a dichos compañeros y que ellos estén dispuestos a colaborar proporcionando su información. Además, habría que comprobar que su información fuera confiable.

La tercera opción, es similar a la segunda sólo que ahora se sabría de antemano que la información es confiable pues es proporcionada por el profesor que imparte la asignatura. Este problema se vuelve más simple si se cuenta con el apoyo de los profesores, como sucedió en algunos casos para la realización de este sistema. La ventaja de la participación de los profesores en el desarrollo del sistema, es que dan a conocer su trabajo y su forma de impartir la materia, además de que pueden recibir sugerencias sobre la manera de hacerlo.

Por otra parte, se tiene que hacer frente a dos situaciones importantes. La primera de ellas consiste en que el temario que se maneja en el departamento al cual pertenece determinada asignatura optativa, y que es el que se incluye en el sistema tutorial, puede diferir un poco con respecto al que cada profesor sigue en el desarrollo de su clase, sobre todo en el orden de exponer los temas y sin embargo, el temario que utiliza el profesor es el que realmente interesa a los alumnos. No obstante, la diferencia es mínima, y ambos representan a una materia especifica, por lo cual no deben diferir mucho de los que se enseñan en las aulas.

La segunda situación, es el hecho de que algunas asignaturas optativas son impartidas por más de un profesor, lo cual trae consigo que cada profesor tenga diferencias en el

modo de impartir su clase, por lo que se cae en el caso de tener tareas, evaluaciones y proyectos diferentes para una sola asignatura. Esto es bueno porque se tiene mayor diversidad de ideas y criterios para aprender una materia, pero implica una serie de planteamientos que habrá que resolver antes de escribir la información en las páginas como serían los siguientes:

- ¿Se debe conseguir información basándose en el temario que proporciona el Departamento o en el temario que proporcionan los profesores?,
- ¿Qué tareas, proyectos y evaluaciones es conveniente incluir?,
- ¿Es mejor el método de enseñanza de uno u otro profesor?,
- ¿Cuál conviene adoptar en un sistema tutorial que pretende ser autodidacta?...

Analizando cada una de estas interrogativas, las cuales nos parecen las más significativas, se tiene:

Para la primera de ellas referente a los temarios, si se opta por el primer caso (elegir el temario del departamento), quizá el auxilio que puedan proporcionar los profesores no sea tan importante que si se utiliza el temario que ellos ocupan para impartir la asignatura. Así también, no se tendría un panorama real de lo que se enseña en los cursos. En disyuntiva, si se elige buscar información basándose en el temario que proporcionan los profesores, se corre el riesgo de encontrar algunas diferencias si la materia es impartida por más de un profesor; en este caso se tiene que decidir en cuál temario basarse, o buscar la mejor combinación de ambos temarios. En este sentido, nuestra idea consiste en adicionar al temario del departamento correspondiente a la asignatura, los temas que exponen los profesores en clase y a la vez, complementan el aprendizaje del alumno.

La misma problemática de toma de decisiones se presenta en las tareas y proyectos, pues siendo varios profesores para una sola materia, cada uno de ellos tiene sus propios criterios de enseñanza-aprendizaje y elegir uno u otro para adaptarlo al Web es una labor que debe hacerse cuidadosamente para alcanzar el objetivo que persigue este sistema tutorial: "Lograr un nivel de enseñanza-aprendizaje entre el alumno y la computadora".

No obstante, a pesar de las problemáticas que se presentan en la nueva planeación, se decide continuar con el proyecto, puesto que todos estos problemas tienen solución y desarrollar la mejor respuesta a cada uno de ellos llevará a obtener un mejor producto que satisfaga el objetivo inicial.

V.2.3 DESARROLLO

En la etapa de desarrollo de esta segunda vuelta se realizó la construcción del sistema. Ésta contará con mejoras en cuanto a facilidad de navegación y contenido de información; así mismo se pretende una calidad profesional en cuanto a la presentación de las páginas.

Como se mencionó en la etapa de planeación, se tomará como referencia la primera versión obtenida durante la primera vuelta lo cual facilitará y agilizará considerablemente

el desarrollo del sistema ya que se parte de la misma estructura que funcionó correctamente en el prototipo.

Elección de la metodología de diseño

Para la construcción del sistema se inicia con la delimitación del problema. Previamente en la etapa de planeación se definen el objetivo y los requisitos que debe contemplar el mismo, así como las posibles alternativas de solución. Durante el análisis de riesgo se evalúan los obstáculos que podrían presentarse y se decide si se continúa o se comienza nuevamente. De esta manera, al comenzar la etapa de desarrollo se empieza con la programación del sistema real.

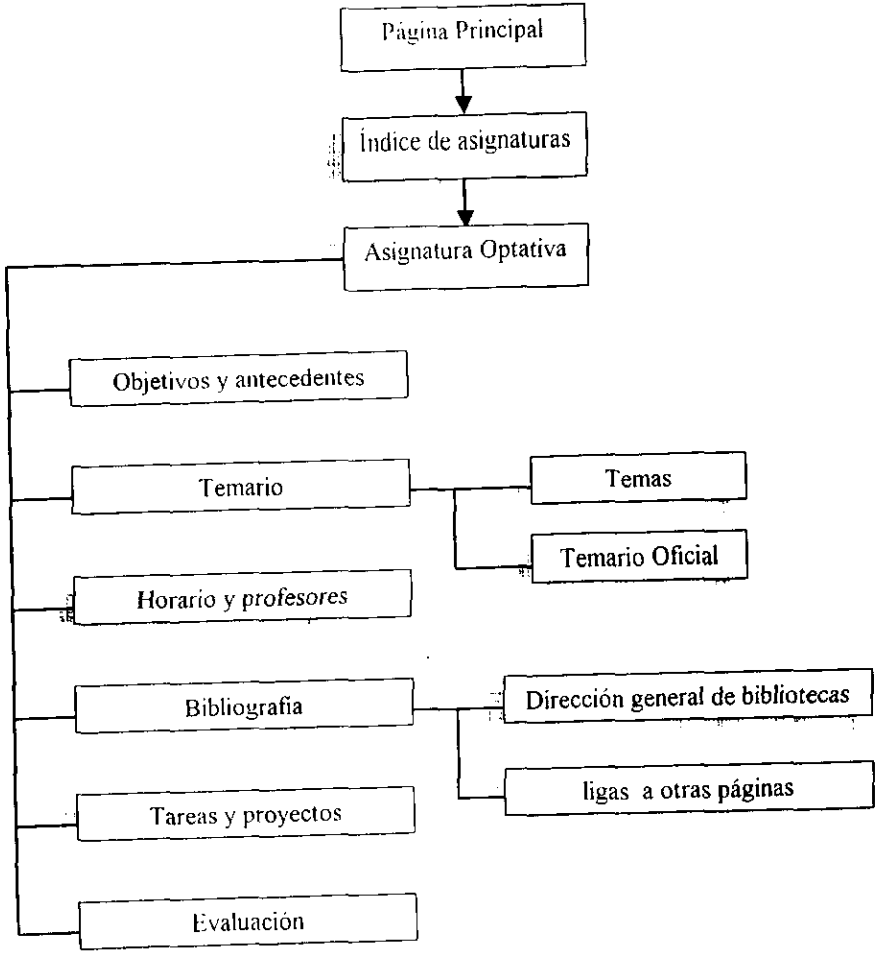

Figura 7. Diagrama de estructura general del sistema turorial

Como se va a partir del prototipo desarrollado durante la primera vuelta, *la metodología* de diseño a utilizar en esta etapa es igual a la que se empleó en la elaboración del *prototipo*, es decir, el método de diseño funcional descendente.

Las características del método de diseño funcional descendente quedaron definidas en la etapa de desarrollo de la primera vuelta de esta metodología. Como un breve recordatorio, en este método se parte del *desarrollo de los niveles* más altos del sistema hasta llegar a los niveles más bajos o detallados del mismo.

El diagrama de estructura que *representa cómo* estará organizado nuestro sistema tutorial se muestra en la Figura 7.

En cuanto a la *organización*, como se comentó en la etapa de planeación y de análisis de riesgo, las modificaciones a la estructura del prototipo serán mínimas. Se conservará el orden que siguen las páginas y la nomenclatura de los archivos.

En cuanto a la *presentación*, los cambios sugeridos se listan a continuación:

- Se *colocaron* fondos en cada una de las páginas utilizadas en el tutorial.
  Para tener uniformidad en cuanto a presentación se decidió que *las páginas en* común para cada materia optativa tuvieran el mismo fondo; así, el background de la *página* donde aparece el temario es el mismo para cada asignatura optativa. La página principal y la que contiene el índice de *asignaturas tienen* un fondo particular. Las páginas donde se define cada uno de los temas refleja un ambiente diferente.

- Uniformizar títulos y contenidos de cada página.
  Con la finalidad de enfatizar las palabras importantes se resaltaron los títulos de cada sección en cada página, *manejando* un tamaño de letra adecuado que sugiriera una distinción entre los títulos, subtítulos, contenidos, etc. Además, se buscó el color de letra más adecuado con el color del fondo utilizado en cada página, de forma *tal que la lectura* de la información no se dificulte. Así mismo, se buscó fijar estándares en tamaño, tipo, forma y color de letras utilizadas en títulos, fórmulas, pies de página, etc.

- Utilizar frames en cada página principal de cada optativa.
  Como se dijo en la etapa de planeación, se decidió fragmentar la información referente a cada materia optativa dividiéndola en secciones. Para acceder a cada sección de cada materia optativa se determinó el uso de *frames*, ya que con esto se facilita la búsqueda de información y hace que la navegación sea rápida y transparente.
  En este caso, para la página principal de cada materia optativa se incluyeron frames. El frame divide la pantalla en dos partes; en una de ellas aparece desplegado un índice de las secciones que conforman la información que se presenta de cada materia optativa, y en la otra, la información correspondiente a la sección seleccionada. El índice lo componen una serie de ligas a las secciones de cada materia.

- Se trató de utilizar sólo las ligas necesarias para navegar a través del sistema tutorial, pues como se ha mencionado anteriormente un exceso de ligas distraería la atención del usuario de la información que estuviera consultando. Además, con la utilización de frames se permitió reducir el número de ligas, optimizando la cantidad de éstas.

En resumen, la organización actual del sistema tutorial es la siguiente:

- Una página principal o de presentación, la cual incluye una liga a la página que contiene el índice de asignaturas tal y como estaba (una lista de las once asignaturas optativas).

- Una página que es el índice de asignaturas, la cual contiene una liga a la pagina principal de cada asignatura optativa. Además una liga hacia la página principal y una opción de salida del sistema.

- Una página principal de cada materia optativa, que es un frame dividido en dos subpantallas, la primera es un índice de las asignaturas optativas, que no son más que ligas a cada una de las secciones que componen la información de cada materia optativa, y la segunda subpantalla, que es en donde se visualiza la información de la sección seleccionada en el índice. Las secciones en las que se dividió la información de cada materia optativa se mencionan a continuación:
    - Antecedentes y objetivos
    - Temario. Esta sección contiene una liga para cada tema que comprenda la materia la cual lleva a la página correspondiente
    - Horario y profesores.
    - Tareas y proyectos.
    - Bibliografía.
    - Evaluación

- La sección de evaluación, se incluyó con la finalidad de que el usuario del tutorial, una vez que haya recorrido la asignatura de su interés, analice por si mismo qué tanto ha aprendido y comprendido sobre la misma al responder algunas preguntas representativas de cada tema y saber si su respuesta fue correcta o errónea.

- Además de ligas a estas secciones, el índice de cada optativa cuenta con una liga hacia el índice de asignaturas y otra hacia la página principal.

- Una página por cada sección de cada asignatura optativa.

- Una página por cada tema que comprende cada asignatura.

Revisión del diseño

Después de definir la estructura del prototipo, se prosiguió a revisar el diseño del mismo.

Nuevamente la revisión del diseño se hizo de manera formal. Esto es, entre los integrantes del equipo de desarrollo se realizó el análisis del diseño a implementar, concluyendo que era el adecuado para cumplir con los requisitos y metas planteados para esta vuelta de la metodología. Esta revisión informal se realizó tomando en cuenta los factores de calidad del sistema.

Es necesario indicar que también se llevó a cabo una revisión formal de sistema. Ésta la hicieron los estudiantes, que serán los usuarios del sistema. Ellos pudieron corroborar la funcionalidad del producto.

Metodología de la programación

Tal como se hizo en la primera vuelta de la metodología, y de acuerdo al método de desarrollo utilizado, funcional descendente, el proceso de desarrollo es también descendente, para el sistema.

Estilo de programación

Nombres de los programas

Para facilitar la programación de los módulos que componen al sistema hay que utilizar nombres que vayan de acuerdo con las entidades del mundo real. Con esto se facilita el mantenimiento del software.

Debido a que el lenguaje de programación utilizado es HTML, para diseñar páginas Web, se utilizan los siguientes nombres para facilitar la programación:

A la página de presentación se le cambió el nombre a *homepage*. Esto debido a que cuando un conjunto de páginas se suben a la red, aquella que carga por defecto cuando se accede al sitio es homepage.html

La página que contiene el índice de asignaturas se sigue denominando *asignaturas.*

El frame que define cada asignatura optativa se denota con las primeras dos letras de la palabra frame y las tres letras que describen a la materia. Por ejemplo, para definir el frame de Procesamiento Digital de Señales, se utiliza la palabra *frpds.*

La página que contiene el índice de las secciones en que se divide la información que se muestra de cada materia se denomina con las primeras tres letras de la palabra índice y las tres letras que representan a una optativa, por ejemplo el índice de Organización y Administración de Centros de Cómputo se llama *indoac.*

La página que contiene los objetivos y antecedentes de la materia se denomina con las tres letras que definen a la asignatura, por ejemplo para la página de Sistemas expertos utilizamos la palabra *sex.*

A la página que contiene información del temario de la asignatura se le denotó con las primeras tres letras de la palabra temario más las tres letras que definen a la optativa, por ejemplo para el nombre del temario de Calidad se utilizaba la palabra *temcal.*

A la página que contiene información de los profesores se les denotó con las primeras cuatro letras de la palabra profesores más las tres letras que definen a la optativa, por ejemplo para el nombre de la página de profesores de Bioingeniería se utilizaba la palabra *profbio.*

A la página que contiene la bibliografía de cada materia se le denomina con el prefijo bib más las tres letras que definen cada asignatura, por ejemplo para definir la página que contiene la bibliografía de Reconocimiento de Patrones se utiliza la palabra *bibrdp*

A la página que contiene la información de las tareas y proyectos desarrollados en la materia se utilizan las iniciales de tareas y proyectos más las letras que caracterizan a la materia, por ejemplo para la página de Diseño Asistido por Computadora se escribe *typdac*

A la página que contiene una pequeña autoevaluación sobre la consulta de alguna materia en particular la denominamos con el prefijo eva (de evaluación) más las tres letras que definen a cada asignatura optativas. Por ejemplo para definir la página de evaluación de Robótica utilizamos la palabra *evarob*

A cada página correspondiente a cada tema comprendido en el temario de la asignatura se le denota con las letras que representan a la materia más la inicial de tema y el número de tema en cuestión. Por ejemplo, para definir el tema uno de la asignatura Procesamiento Digital de Imágenes, utilizamos la palabra *pdit1*

Construcciones de control en los programas

Como el sistema está desarrollado en un lenguaje que maneja hipertexto, cada página se crea de manera aislada, sin embargo a la hora de armar el sistema estas páginas deben estar relacionadas entre sí; por esta razón se hace uso de ligas en las páginas que nos permitan llamar a otras páginas para ponerlas en activo. De este forma, se logra una unión de las diferentes páginas construidas.

Instrumentos de Software

Instrumentos para la preparación de programas.

Como se mencionó anteriormente, para cambiar el contenido del programa de desarrollo de cada página incluida en este sistema se utilizó un editor de texto, en este caso el bloc de notas. Para el manejo y creación de imágenes, se hizo uso del editor de imágenes Paint Shop Pro

Igualmente, para observar cómo se vería desplegada cada página creada, se utilizó el navegador Netscape para visualizar los cambios que se iban creando dentro de la construcción de cada página.

Elección del lenguaje de programación

Debido a que se partió del prototipo creado para el desarrollo del sistema, el lenguaje utilizado para desarrollar el sistema será HTML. Las características de este lenguaje de manejo de hipertexto ya se mencionaron anteriormente.

Además, para buscar una mejor presentación en el sistema y obtener una mayor interactividad se empleó el lenguaje Javascript. (Ver Anexo 2)

Pruebas del sistema.

Después de haber desarrollado el sistema del sistema tutorial y antes de subirlo a la red para la evaluación por parte de los usuarios, se procedió a hacer una prueba del mismo para ver cómo funcionaba.

Así, una vez creadas las páginas que componen al sistema, realizadas las ligas pertinentes entre estas páginas e incluida la información que se pretende proporcionar a los usuarios, se realizó una prueba de cómo se vería el tutorial una vez que estuviese colocado en la red.

Como se mencionó anteriormente, para visualizar la presentación del sistema se cuenta con el navegador Netscape (esto no significa que no pueda consultarse mediante otro browser como Internet Explorer). Así, para realizar la prueba del sistema se fueron cargando las páginas creadas en una cuenta de alumno de las que proporciona el Laboratorio de Computadoras de la DIE para cualquier estudiante de la Facultad y mediante el browser Netscape 3.1 se navegó a través de las diferentes páginas observando la presentación, la facilidad en el acceso, el contenido y la claridad de la información, entre otras cosas.

Esta prueba permitió corregir algunos errores principalmente errores ortográficos y sintácticos en la información. Así mismo se tuvo una mejor apreciación de la distribución en las páginas pudiendo uniformizar el contenido de cada una de las páginas creadas.

Es necesario comentar que cada modificación que se realizó al sistema se fue verificando a fin de no generar y, posteriormente, arrastrar nuevos errores. Se revisó cuidadosamente cada una de las ligas, sobre todo las que conducen a un tema específico de una asignatura. También se visualizó una y otra vez la presentación de cada página del sistema confirmando la uniformidad en los fondos dependiendo del tema que abordará la página Web.
Una vez terminada la fase de pruebas y depuración, se consideró que el sistema estaba listo para la evaluación de los usuarios. Así se concluyó la etapa de desarrollo y se

procedió con la última etapa de la segunda vuelta, o sea, la etapa de evaluación del cliente.

V.2.4 EVALUACIÓN

La evaluación del sistema fue hecha por parte del cliente, en este caso alumnos de los últimos semestres de la carrera de Ingeniería en Computación.

Antes de realizar esta actividad fue necesario colocar el sistema desarrollado en Internet. Esto se logró gracias al apoyo del administrador del servidor Web con que cuenta el Laboratorio de Computo del Departamento de Ingeniería en Computación. Esta persona (el administrador Web del Departamento) colocó las páginas que comprenden al sistema en el servidor *Odin, que es un servidor web, con lo cual* el sistema pudo ser visitado desde cualquier navegador dentro o fuera de la Universidad. Como para entonces el sistema aún no será liberado, se incluyeron algunas restricciones para que no cualquier persona que navega en Internet pudiera accesar al sistema. De esta manera se colocó un login y un password de seguridad, de tal manera que sólo aquellas personas que conocen las contraseñas tienen acceso a este sistema. Estas personas fueron compañeros nuestros que han cursado asignaturas optativas cuyas opiniones son de gran utilidad, ya que ellos mejor que nadie, saben si la información es suficiente o hace falta agregar algo más.

Así mismo, se pidió a un grupo de alumnos de la Facultad que tuviera acceso a un navegador de Internet no importando si habían cursado o no asignaturas optativas ni la carrera a la que pertenecieran ni tampoco si eran alumnos egresados o de nuevo ingreso que observarán el sistema tutorial y enviarán su opinión a través de la cuenta de correo electrónico que se incluyó en el sistema con la finalidad de recopilar sugerencias. También se pidió a un grupo aleatorio de la Facultad el cual conformaron alumnos de la asignatura de Ingeniería de Programación del Ing. *Orlando Zaldivar Zamorategui, que* revisarán y navegarán a través del sistema durante una sesión de laboratorio en la cual entregaron un análisis de acuerdo al nivel de conocimientos adquirido en su curso.

Entre las observaciones y sugerencias que se reunieron se encuentran las siguientes:

- *Poner en la pantalla* principal las opciones de las asignaturas optativas.
- Donde están las asignaturas optativas está la opción de salir. ¿Salir a dónde?
- Colocar algunas imágenes relacionadas con las asignaturas.
- *En algunas páginas*, las ligas (una vez ejecutadas) no se ven ni leen.
- No se pueden ver los proyectos de Graficación por Computadora.
- Aunque le falta información, y un poco de creatividad (incluir imágenes o más detalles sobre las asignaturas para hacer más comprensible lo que se esta exponiendo) cuenta con todos los elementos necesarios para solucionar el problema planteado, el cual es el informar todo lo referente a las asignaturas optativas.
- La creación de la página es una buena idea porque permite conocer mejor cada una de las asignaturas y saber cuáles nos pueden interesar, bien porque nos llame la atención o bien porque se ajuste a nuestras necesidades.

- La idea de crear este tutorial es muy buena, porque explica de manera clara los objetivos y contenidos de cada materia, lo cual puede sernos útil en la elección de las mismas. Uno de los inconvenientes es que la información no está del todo completa.
- La idea de la página es excelente, ya que brinda información detallada acerca de las asignaturas optativas. El contenido es muy completo porque incluye horarios, nombres de profesores, temarios, un breve resumen de los temas tratados, entre otros. El desarrollo de los temas me agrado, porque nos puede orientar en la sección de las asignaturas. Así mismo la evaluación me pareció adecuada. Sólo falta incluir animaciones, imágenes, fotografías o downloads de los programas más usados para dichas asignaturas.
- El contenido es de gran utilidad. Proporciona con detalle lo que se puede aprender en cada materia, contribuyendo a un mejor proceso de aprendizaje de acuerdo al perfil e intereses del estudiante. Tiene buena presentación y es fácil de acceder.
- Es de utilidad, la información es buena. Faltan algunas imágenes y profundizar un poco sobre los temas, aplicaciones e importancia de la materia con la carrera.
- Da una mayor visión de las asignaturas optativas, la información es interesante y es de utilidad. Es bueno que esté al alcance de todos. Está bien elaborado.
- La página es bastante buena, la información es muy completa.
- Tiene un diseño atractivo y la información es relativamente amplia.

Como puede observarse en los comentarios, la mayoría de las personas que evaluaron el sistema quedaron satisfechos con la presentación, los colores de los fondos, la combinación con las letras del contenido, los títulos fueron de su agrado y principalmente, la información incluida les pareció interesante. Hubo también críticas en contra, como que el sistema se ve muy serio.

Tomando en cuenta estas opiniones se mejoró la presentación del sistema, recurriendo para ello a la opinión de personas que han desarrollado páginas web, así como visitando páginas en internet para comparar, respetando en todo caso el estilo propio del sistema.

Otro comentario sobresaliente fue sobre el color de las ligas, se sugirió que cuando se utilizara una liga, ésta cambiara su color para saber que ya se había visitado esa página. Este cambio se consideró en el sistema sólo que el color indicador resultó muy similar al color del fondo de la página por lo cual se perdía la visibilidad de las ligas.

Otra de las críticas ha sido que no se encuentran algunas páginas, esto ocurre porque cuando se realizaron los archivos que contienen ligas a una u otra página se escribió el nombre del archivo de enlace con mayúsculas y minúsculas siendo que este nombre estaba completamente con letras minúsculas (nombre y extensión) y al subir las páginas a la red, el servidor trabaja con sistema operativo UNIX (HP-UX) el cual reconoce la diferencia entre las letras mayúsculas y las minúsculas por ello que hubo que revisar aquellas ligas cuyas páginas no aparecían y corregir el nombre. Con esta sencilla labor, el problema quedó solucionado.

De acuerdo con los comentarios obtenidos, podemos concluir que sólo son algunos detalles sin trascendencia los que hay que corregir. Estos detalles están relacionados con la presentación del sistema, sobre todo de texto. Esto puede considerarse comprensible debido al manejo de demasiados archivos con información la mayoría de ellos, lo que ocasiona que se cometan errores al transcribir la información.

Una vez que fue evaluado el sistema resulta conveniente realizar una verificación del mismo, analizando los objetivos, requisitos y metas planteados en la etapa de planeación.

El objetivo principal, desarrollar un sistema tutorial sobre las asignaturas optativas del plan de estudios de la carrera de Ingeniería en Computación se ha cubierto casi en su totalidad. De acuerdo con las opiniones recopiladas de aquellos alumnos que han visto el tutorial, quedan pendientes algunos detalles, relacionados principalmente con la presentación de la información. El sistema, ofrece un panorama general al alumno de lo que son las asignaturas optativas de la carrera de Ingeniería en Computación, conteniendo información suficiente para que todo estudiante de la carrera tenga una idea de lo que aprenderá en cada una de las asignaturas optativas. Se indica no sólo objetivos, antecedentes, créditos y bibliografía que contenía el prototipo; además, cada temario está detallado con información referente a lo que se estudia en cada tema, especificando referencias e incluyendo algunas ligas donde se puede ampliar su información. En lo que respecta a los profesores, cuando fue posible, se incluyeron ligas a las páginas personales de éstos, para que los alumnos los conozcan y sepan su trayectoria académica y área de desarrollo. Se anexan tareas y trabajos desarrollados en las asignaturas, y prácticas de laboratorio, cuando es el caso. Esto constituye una ayuda vital para los alumnos porque les permite reforzar lo aprendido en clase y para los profesores porque cuenta con más material para programar sus actividades del curso, teniendo la opción de dejar nuevas tareas o diferentes proyectos a sus alumnos además de consultar los trabajos que estudiantes de semestres anteriores han realizado para la asignatura.

También se incluyen ligas a otras páginas en donde el alumno puede conocer aplicaciones de la asignatura u obtener más información de la misma.

Se incluyó también una evaluación con el objetivo de que el usuario pueda saber cuánto ha aprendido en su visita por el tutorial. Ésta consta de una serie de preguntas las cuales están relacionadas con los temas descritos. Cada pregunta tiene tres opciones de respuesta, de acuerdo con la selección del usuario, aparecerá un mensaje que indicará si la contestación fue correcta o no. Dicha evaluación resulta un buen ejercicio para investigar qué tanto se sabe de alguna asignatura en particular.

Un problema específico que se planteó en este segunda vuelta, fue el corregir o modificar el prototipo de la primera vuelta para que cumpla en mayor medida con el objetivo principal. Como mencionamos, este problema fue resuelto satisfactoriamente y el objetivo principal ha sido cubierto en su totalidad. Como se esperaba, al término de esta segunda vuelta, el sistema tutorial está terminado y listo para colocarlo en la red para que todo alumno de la carrera de Ingeniería en Computación que tenga acceso a internet lo pueda consultar.

En lo que respecta a las metas planteadas en esta segunda vuelta de la metodología, se planteó mejorar en el sistema real las fallas que contenia el prototipo creado durante la primera vuelta, para lo cual se tomaron en consideración las conclusiones a las que se llegó en la etapa de evaluación de la primera vuelta, con el fin de alcanzar el objetivo principal que se planteó antes de iniciar la construcción del sistema tutorial. Como se mencionó en la etapa de planeación, esta meta se dividió en dos partes, mejorar por un lado la presentación y enriquecer la información manejada. Esta meta fue alcanzada con éxito en el sistema tutorial, al mejorar los aspectos de presentación e información de los que carecía el prototipo. En la presentación, se logró obtener una mejor visualización del sistema, con la implementación de fondos en las páginas, el hecho de incluir frames, la colocación de imágenes, gifs animados y un adecuado tipo de letra, de tal manera que el sistema resultara atractivo, pero sin que perdiera de vista su finalidad, la información. En lo que respecta a la información, se abundó en los temas manejados en cada asignatura, con lo que se logró dar un mejor panorama de cada materia, además se incluyeron ligas a otras páginas, en donde el alumno puede completar su formación, o bien estar mejor informado. Además se incluyó una evaluación, en donde el alumno puede saber cuánto aprendió sobre cada asignatura.

Finalmente, es necesario verificar si el sistema cumple ya con lo requisitos planteados en la etapa de planeación de la primera vuelta y verificados en la planeación de esta la segunda vuelta.

Entre los requisitos que debe contener el sistema se encuentran:

- Manejo de suficiente información

El tipo de información incluida dentro de esta versión debe ser la necesaria para que el alumno conozca bien el contenido de cada asignatura optativa. Se debe incluir información que complemente la ya manejada en la primera versión del sistema, sobre todo desarrollar más detalladamente los temas que se imparten en cada materia, y anexar en lo posible los tipos de tareas, trabajos, prácticas y proyectos que se realizan en cada asignatura optativa.
Este requisito fue cubierto en su totalidad, quedando por mejorar algunos detalles. Como se mencionó, los temas fueron desarrollados adecuadamente, de forma tal que el alumno puede saber qué se estudia en cada materia. Se incluyeron también tareas y proyectos, para que el alumno conozca lo que desarrollará en estas asignaturas optativas. Además se agregaron ligas, en las cuales el alumno puede terminar de documentarse.

- Buena presentación

El sistema tutorial debe tener una mejor presentación en las páginas que lo comprenden. Contener un ambiente agradable y atractivo para el usuario, que le llame la atención, pero sin desviar su atención del objetivo principal del sistema tutorial, la información.
Este requisito se cubrió casi en su totalidad. Para ello se emplearon frames, que permiten navegar fácilmente y mejoran la vista. Se utilizaron fondos para cada página que hacen más agradable a la vista el despliegue de las páginas del tutorial. Se incluyeron imágenes que complementaron el texto manejado. Además

de manejar un tipo de letra, con tamaño y color adecuados, que sea legible y entendible por parte del usuario

- Una buena Organización

En este respecto, el prototipo debe contener una estructura que permita al usuario su fácil utilización.
Como ya se había analizado en la primera vuelta de la metodología, este requisito ya lo contenia el prototipo desarrollado del sistema. Sin embargo con el uso de frames en esta nueva versión del sistema, se mejoró aun más este característica. El empleo de frames permite una mejor navegación a través de cualquier sitio Web, *lo cual es fácil* de verificar en esta nueva versión del sistema. Por lo que podemos concluir que este requisito quedó ampliamente cubierto

V.3 TERCERA VUELTA

V.3.1 PLANEACIÓN

Definición del problema.

El problema general, ya resuelto, es el desarrollar un sistema tutorial sobre las asignaturas optativas del plan de estudios de la carrera de Ingeniería en Computación.

El problema particular a resolver en esta tercera vuelta, es el realizar las últimas modificaciones al sistema, expresadas en los comentarios obtenidos en la etapa de evaluación anterior, para que sea liberado y puesto en la red para su libre consulta.

Los comentarios mencionados y la evaluación sobre el sistema indican que sólo son pequeños detalles que hay que corregir en el sistema para que éste pueda ser liberado. Por lo que no es necesario hacer un diseño elaborado para corregir estos errores. Sin embargo, *es necesario hacer una planeación y un análisis* de riesgo para afinar estos detalles y dar por concluido el sistema

Por consiguiente es necesario, desarrollar las *modificaciones necesarias al sistema para* que se corrijan este tipo de detalles. Los comentarios positivos que se encontraron en la evaluación del sistema, nos señalan que el sistema ya cumple con su objetivo principal.

Metas

La única meta que se persigue en esta tercera vuelta de la metodología es corregir los pequeños detalles que le faltan al sistema para estar listo y liberarlo. Para ello se tomarán en consideración los comentarios *más significativos obtenidos en* la etapa de evaluación anterior. Cabe aclarar que no se pueden tomar en cuenta todos los comentarios obtenidos, porque algunos son puntos de vista muy personales, es decir los detalles que observan, sobre todo de presentación, es porque no les gusta como está realizado o bien porque preferirían que se incluyeran más imágenes, que es lo que mas atrae en el Web . *El incluir* más imágenes, animaciones o downloads alentaría el despliegue de las páginas

que conforman el sistema. Sin duda, la versión del sistema que se libere cumple con el objetivo y requisitos planteados.

Requisitos

Los requisitos establecidos para el sistema a lo largo de su desarrollo han sido cumplidos:

- Información
  La información manejada en el sistema es la adecuada para que el alumno conozca bien el contenido de cada asignatura optativa. Los comentarios obtenidos así lo indican. Quizá sólo haya que profundizar en el desarrollo de los temas, pero como se mencionó el sistema es susceptible de mejorar. El desarrollo de los contenidos de los temas, así como su actualización es una actividad permanente.

- Presentación
  De acuerdo con los comentarios recopilados la presentación es agradable y atractiva. Se recomienda incluir más imágenes, pero esto alentaría la navegación a través del tutorial. Aunque si el sistema sigue creciendo es posible incluir en un futuro un mayor número de imágenes. De todas formas, se colocaron las necesarias

- Organización
  La estructura del sistema le permite al usuario recorrerlo por completo sin ninguna complicación. El acceso a las diferentes partes que componen al sistema es muy sencillo.

Opciones de solución

Con la finalidad de resolver los pequeños problemas en el diseño del sistema para su liberación se tomaron en consideración los puntos resaltados en los comentarios incluidos en la etapa de evaluación de la segunda vuelta. Aunque, como se mencionó sólo se tomarán en cuenta aquellos que ameriten una modificación importante en el sistema. Algunas consideraciones están incluso fuera de nuestro alcance, como el hecho de que la navegación sea lenta; ésta depende de la máquina utilizada y de la versión del navegador; de esta última depende también el que puedan ver algunas aplicaciones, como son los proyectos desarrollados en Graficación por Computadora.

Una alternativa para solucionar algunos detalles, consiste en observar el sistema tutorial en la red, revisarlo minuciosamente y tomar en cuenta los comentarios obtenidos de quien o quienes ya lo observaron, verificando aquellos detalles que aún falten por afinar.

Lo primero que se considera es hacer las correcciones en los nombres de los archivos y en el color de las ligas para indicar que una página ha sido visitada. Con esto hecho, se cubre una parte significativa de los comentarios recibidos.

Un siguiente paso consiste en revisar el texto, por posibles descuidos al escribir un trabajo con una extensión considerable.

Algunos ajustes sugeridos se pueden contemplar para una etapa de mantenimiento, es decir aquellos cambios que se le irán haciendo al sistema una vez que sea liberado. Como todo sistema, éste también es susceptible de mejorar. En una etapa de mantenimiento, se puede contemplar el crecimiento del sistema, y por lo tanto incluir imágenes, animaciones y profundizar en los temas. En otras palabras, el diseño del sistema está abierto para que pueda aumentar su contenido.

Planeación del proceso de desarrollo

En la etapa de planeación de la primera vuelta, se decidió utilizar el modelo de desarrollo de prototipos, esto es, se crea un primer prototipo con los requisitos definidos por el cliente, el cual es evaluado para ver si cumple con las características y requisitos definidos por el cliente, si es así se implementa y se pasa a la etapa de mantenimiento, de lo contrario se vuelve a hacer un diseño para construir una nueva versión del sistema que contenga mejoras del primer prototipo y que intente cumplir con los que lo generaron.

Se concluye que el sistema cubre los objetivos definidos, por lo que ya no es necesario desarrollar nuevos módulos del mismo, sino que sólo se ajustarán detalles para su liberación.

Verificación y validación

Como se planteó en la metodología utilizada para el desarrollo del sistema tutorial, después de la etapa de desarrollo se incluye una de evaluación, en la cual se hace una verificación de la versión del sistema construido en esa vuelta de la metodología, con la finalidad de observar si la versión desarrollada cumple con los objetivos planteados. De la última revisión realizada por compañeros nuestros que han cursado asignaturas optativas y los alumnos del grupo de Ingeniería de Programación, se concluyó que el sistema ya cumple con los requisitos definidos inicialmente, por lo que no es necesario tener una nueva evaluación, bastará con verificar que el sistema esté debidamente depurado y sin errores.

Una vez concluida esta etapa de planeación, en la que se definieron las alternativas para depurar el sistema y liberarlo para su libre consulta, se procedió a la siguiente etapa, la del análisis de riesgo.

V.3.2 ANÁLISIS DE RIESGO

Como se mencionó en la etapa de planeación este sistema es susceptible de mejorar. Ahora bien puede surgir una nueva interrogante ¿Qué pasa si el sistema crece y es necesario manejar más información, más imágenes, mejorar presentación, etc.? ¿Qué ocurrirá con la estructura y el contenido del sistema actual? Primero se debe considerar que para que esto ocurra es necesario, primeramente, hacerse de información que se pueda incluir en el sistema, encontrar imágenes y animaciones que ayuden a enriquecer

el contenido del tutorial. Para esto se considera una etapa de mantenimiento, en la cual se pueden asignar tareas para que, periódicamente, se realice el análisis al sistema para observar qué posibles mejoras se le pueden implementar.

Por otro lado, sabemos que la tecnología está cambiando continuamente, por lo que surgen nuevas herramientas de diseño. Internet es el más claro ejemplo de esto. Periódicamente surgen nuevas herramientas que explotar para mejorar el diseño de una página Web. De esta manera, en la etapa de mantenimiento se puede analizar cuáles herramientas pueden emplearse para mejorar el diseño del sistema.

Sin embargo, la estructura que contiene el sistema es flexible, por lo que permite el crecimiento del mismo, sobre todo en lo que respecta al contenido. De esta forma, la estructura actual seguirá siendo útil y si acaso serán necesarias sólo pequeñas modificaciones. De hecho, la estructura que maneja el sistema está pensada para que permita el crecimiento del sistema y sea susceptible de mejorar, conservando la esencia del tutorial.

V.3.3 DESARROLLO

En la etapa de desarrollo de la última vuelta de la metodología utilizada se realizaron las correcciones pertinentes al sistema, para que pudiera ser liberado.

Como se mencionó en la etapa de planeación, únicamente se hicieron pequeñas modificaciones a la versión obtenida durante la segunda vuelta.

Elección de la metodología de diseño

Como se hizo anteriormente en la elaboración de los prototipos del sistema, se utilizó el método de diseño funcional descendente.

Las características del método de diseño funcional descendente quedaron definidos en la etapa de desarrollo de la primera vuelta de esta metodología. Estas características son las siguientes, en este método se parte del desarrollo de los niveles más altos del sistema hasta llegar a los niveles más bajos del mismo de forma detallada

En la etapa de planeación se han definido los detalles que había que modificar en el sistema, así como las posibles alternativas para solucionarlos.

La estructura que presenta la versión definitiva del sistema es que se ilustró en la vuelta anterior.

Las siguientes modificaciones se realizaron al sistema tutorial:

- Se cambió el color de las ligas visitadas.
  Las ligas tienen una propiedad: la de poder cambiar de color una vez visitadas. Esto sirve de referencia al usuario para ver cuáles ligas ya visitó. El color que presentaban las ligas después de ser visitadas en el sistema tenía el problema

de confundirse con el color del fondo donde se encontraban. El problema fue resuelto cambiando el color de la liga visitada a uno que no se perdiera entre el color del fondo.

- Revisión de la ortografía.
  Debido a la gran cantidad de información que contiene el sistema, se cometieron errores ortográficos o de puntuación. Para corregirlos se revisaron nuevamente las páginas realizadas y con los comentarios de los compañeros que visitaron el sistema se modificaron los textos y palabras que lo requirieron.

- Revisión del texto.
  Esta tarea fue ardua pero muy fructuosa. Se reviso minuciosamente el tutorial, para checar problemas de contenido, de redacción. Aquí también se verificó que las imágenes se desplegaran adecuadamente, que se tuviera todas las ligas y estuviera perfectamente conectado el sistema

- Se modificó la página que contiene el índice de asignaturas, incluyendo una imagen relativa a cada asignatura. Además se incluyó una opción para salir del sistema. Esta opción, mediante una función de javascript, lleva al usuario después de recorrer el tutorial aleatoriamente a siete diferentes sitios Web de la Universidad, principalmente de la Facultad de Ingeniería.

- Se realizaron las páginas que faltaban de incluir, que no eran muchas, pero que resultaban necesarias para tener completo el sistema.

La organización del sistema tutorial permaneció sin cambio, tal como se mostraba en la vuelta anterior.

Revisión del diseño

Después de las modificaciones realizadas al sistema se prosiguió a revisar el diseño del mismo.

Entre los integrantes del equipo se realizó el análisis del diseño a implementar, concluyendo que el sistema está listo para ser liberado y dejarlo en la red para su consulta por parte de los alumnos de la carrera de Ingeniería en Computación.

Instrumentos de Software

Instrumentos para la preparación de programas.

Para cambiar el contenido de cada página incluida en el sistema se utilizó un editor de texto, el bloc de notas. Después de realizadas las correcciones se subían las páginas ya modificadas al servidor Web.

Para observar cada página modificada, se utilizó el navegador Netscape. De esta forma se verificaba que las correcciones eran las adecuadas, antes de subir las páginas a internet. Para trabajar con las imágenes incluidas en el tutorial, se hizo uso del editor de imágenes Paint Shop Pro

Pruebas del sistema.

Después de desarrolladas las modificaciones al sistema, se realizó una última prueba al sistema para aprobarlo y liberarlo.

Dicha prueba consistió en consultar el tutorial a través de la red, analizarlo minuciosamente y verificar si estaba listo para su consulta a través de internet.

Como el sistema pasó esta prueba, se dio por terminado su desarrollo y se decidió liberarlo y dejarlo en la red para que cualquiera pudiera consultarlo desde el Web. De igual manera, se eliminó la restricción de utilizar un password para acceder a él, permitiendo que cualquier estudiante que tenga acceso a internet, pueda usar también al sistema tutorial.

Debido a que se consideró al sistema listo para su liberación, no se considera hacer una evaluación más y aquí se da por terminada su elaboración

Con esto, se dio por concluido el desarrollo del "Sistema Tutorial de las asignaturas optativas de la carrera de Ingeniería en Computación".

De esta manera, los alumnos de la Carrera de Ingeniería en Computación cuentan ya con un sistema tutorial que describe las asignaturas optativas del plan de estudios. Su consulta les puede ayudar en la elección de éstas, o bien para complementar lo visto en clase, o simplemente para conocer de que tratan las diferentes optativas que los estudiantes de Ingeniería en Computación pueden cursar.

Finalmente escribimos la dirección en la cual se puede consultar este sistema tutorial.

http://odin.fi-b.unam.mx/die/departamentos/computación/optativas

En esta sección se mostrarán ejemplos de cómo se visualizan algunas de las pantallas del sistema, ya que en lo expuesto en las secciones anteriores (Primera, Segunda y Tercera vuelta) sólo se analiza y resume la parte teórica del trabajo realizado.

Intentaremos dar un breve resumen de las conclusiones obtenidas al final de cada iteración en la espiral propuesta, pues ello, ayudará a comprender mejor la forma y estructura de las páginas diseñadas para el sistema tutorial.

Primera Vuelta

Objetivo: Desarrollar un sistema tutorial que sirva de apoyo a los alumnos de la carrera de Ingeniería en Computación en la elección de sus asignaturas optativas y a la vez, sea complemento a dichas asignaturas.

Actividades Realizadas
- Encuesta aplicada a los alumnos de Ingeniería en Computación (preferentemente que hubieran cursado asignaturas optativas).
- Análisis de la información obtenida de las encuestas.
- Planeación, diseño y desarrollo de un prototipo del sistema tutorial.
- Evaluación del prototipo.

Conclusiones
El prototipo desarrollado fue práctico en cuanto a su estructura pero mostró deficiencias en la presentación, así como, carencia de información que resulta necesaria para elegir adecuadamente una asignatura optativa.

Segunda Vuelta

Objetivo: Tomando en cuenta las opiniones de la evaluación del prototipo, comenzar el desarrollo del sistema tutorial con miras a alcanzar el objetivo principal.

Actividades Realizadas
- Planeación del sistema a realizar.
- Búsqueda de información referente a cada una de las asignaturas optativas.
- Análisis de las diferentes herramientas de programación para desarrollar el sistema tutorial (comparación entre lenguajes de programación, alternativas para lograr una mayor difusión y fácil entendimiento del sistema tutorial).
- Desarrollo y evaluación del sistema tutorial.

Conclusiones
El sistema desarrollado fue adecuado para los fines creados (un apoyo para los alumnos de la carrera de Ingeniería en Computación en la elección de sus asignaturas optativas). Además, se encontró que el sistema también serviría de apoyo a los alumnos que estén cursando una asignatura optativa y a los profesores que la imparten.

De los comentarios obtenidos de la evaluación al sistema surgieron algunos detalles, principalmente en la ortografía.

Tercera Vuelta

Objetivo: Optimizar el funcionamiento del sistema tutorial.

Actividades Realizadas

- Corrección de errores ortográficos y/o de *síntaxis en las páginas del sistema.*
- Revisión de las ligas en las páginas que hacen referencia a otras páginas.
- Mejoramiento en la presentación de las páginas (agregando ilustraciones, cambiando colores en los textos para resaltar la información importate, incluyendo pantallas de información al usuario de los requerimientos de hardware necesarios para acceder alguna página específica, etc.)

Conclusiones

Se logró un sistema completo y al gusto de la mayoría de los usuarios. Los alumnos de la carrera de Ingeniería en Computación, se mostraron complacidos al tener un sistema tutorial que les ayudará a elegir sus asignaturas optativas.

Un detalle importante que no se expuso en la parte teórica es el diagrama de tiempos definido para la realización de este sistema tutorial por lo que, consideramos oportuno explicarlo ahora.

En base a la clasificación propuesta por Richard Fairley, este sistema se considera como un proyecto de tamaño mediano cuya duración fue de un año, aproximadamente. Para cada actividad realizada y descrita en los párrafos anteriores se especifica el tiempo de *realización.*

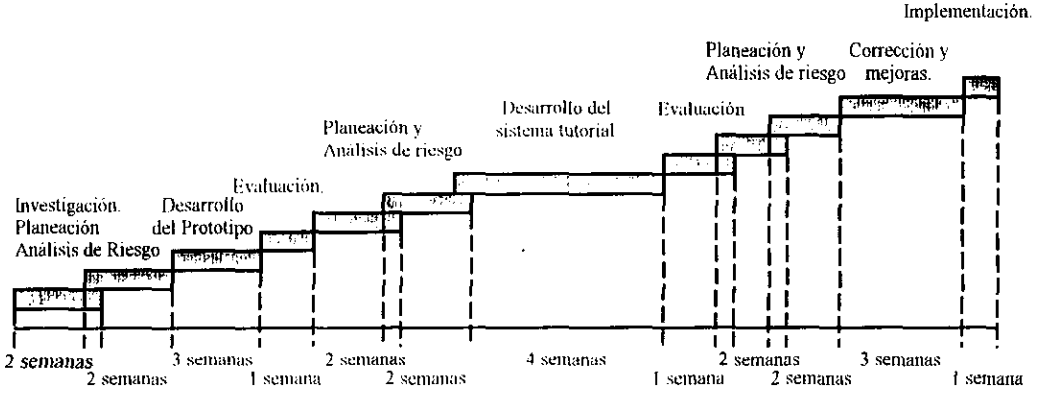

Figura 8. Diagrama de tiempos para el desarrollo del sistema tutorial.

En la Figura 8 se observan claramente los tiempos establecidos para el desarrollo del sistema. *El proyecto comienza formalmente en el mes de julio de 1998,* fecha en que, basados en la información recopilada de las investigaciones realizadas, inicia la planeación y recopilación de datos necesarios. En un periodo de cuatro semanas se realizaron las encuestas a los alumnos de la carrera de Ingeniería en Computación, a la vez, se buscaron y evaluaron los lenguajes de programación existentes para el desarrollo

de páginas Web, se analizó el equipo con que cuenta el Laboratorio de Cómputo de la DIE y la plataforma en qué trabaja, entre muchas otras actividades que complementaron la planeación y análisis de riesgo de la primera vuelta de la metodología.

Durante las tres semanas siguientes al trabajo de planeación y análisis de riesgo, se desarrolla un prototipo que contiene básicamente los módulos que se piensa contendrá el sistema tutorial y que serán útiles a los alumnos en la elección de las asignaturas optativas. De esta manera, una vez terminado el prototipo, se destinó una semana para la evaluación del mismo.

En las siguientes dos semanas, se estudiaron los resultados obtenidos de la evaluación del prototipo y se hizo una nueva planeación para el desarrollo del sistema tutorial real (ya no es prototipo). Se comenzó el análisis de los riesgos para esta nueva etapa del proyecto que finalizó dos semanas después.

De acuerdo con las características establecidas en la etapa de planeación y considerando los detalles obtenidos del análisis de riesgo, se destinaron cuatro semanas para el desarrollo del sistema tutorial,. Al terminar el sistema, se realizó una etapa de evaluación (con duración de una semana), para que, con los comentarios recopilados se procediera a una nueva planeación y análisis de riesgo (en un periodo de cuatro semanas). Finalmente, para realizar las correcciones y mejoras al sistema se estableció un periodo de tres semanas.

Dadas las características del Laboratorio de Cómputo de la DIE y del sistema tutorial se pensó en una semana para la implementación del sistema. Así, ésta quedaría programada para el mes de agosto de 1999, fecha en que los alumnos comenzarían el semestre escolar y podrían basarse de este sistema para hacer una correcta elección de las asignaturas optativas que desearían cursar.

Cabe aclarar que, el diagrama de tiempos fue planeado con base en que durante el primer semestre de 1998 se llevó a cabo la investigación acerca de las metodologías existentes para el desarrollo de software, los tipos de software y toda la información que respalda el marco teórico de este proyecto.

¿Qué comprende este sistema tutorial?

La organización del sistema se resume en el siguiente diagrama de estructura:

En la Figura 9 podemos observar que para cada módulo del sistema, encontraremos una característica particular. Por ejemplo, el Módulo 1 que contiene la pantalla o página principal, da una breve introducción al usuario de qué trata lo que va a visitar, esto es, quién o quienes diseñaron el sistema, quién respalda la veracidad de la información, orgullosamente, qué institución promueve este proyecto de Web, así como un contador para saber cuántas personas han visitado el tutorial. Esta página es importante, puesto que los datos que contiene son necesarios para conocer las bases (quién apoya, motiva y orienta) y orígenes (el problema que surge al tener la responsabilidad de seleccionar cuatro de once asignaturas optativas) de este proyecto.

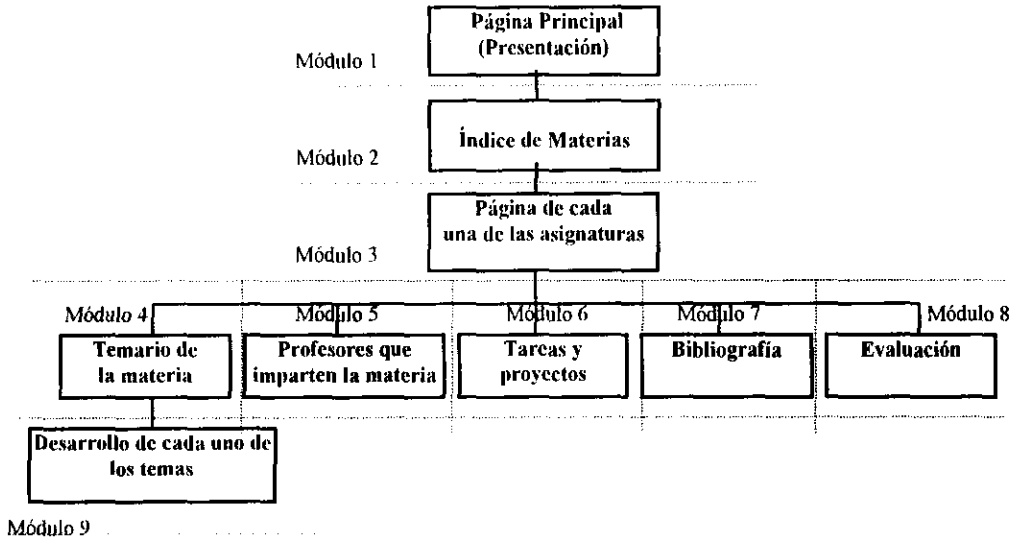

Figura 9. Estructura del sistema tutorial.

El segundo módulo, Módulo 2, contiene la página del índice de materias y muestra las once opciones de asignaturas que se tienen para elegir. La Figura 10 muestra esas opciones y, a su vez, propone una clasificación en grupos de acuerdo al área de especialización que se podría obtener al cursar determinadas asignaturas.

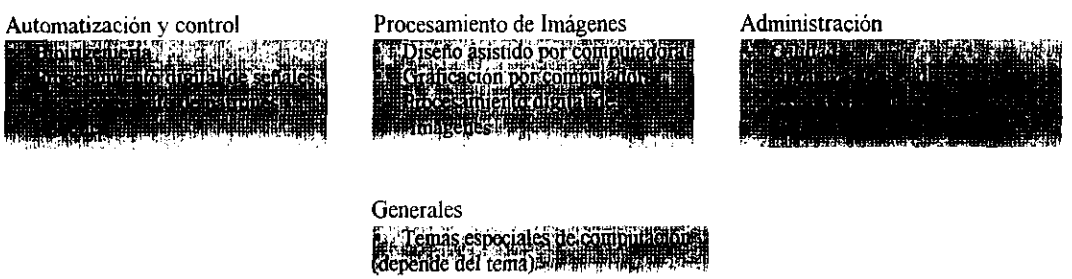

Figura 10. Organización propuesta para las asignaturas optativas.

El Módulo 4, más complejo que los anteriores por todo lo que abarca, es quizá el más importante, el corazón del sistema, ya que es donde se desglosa cada asignatura optativa, la derrama del Módulo 3 que se observa en el diagrama estructural (Figura 9), muestra claramente todo lo que abarca el conocimiento de una asignatura desde el temario, profesores y horarios hasta la bibliografía y una evaluación complementaria para saber qué tanto se aprendió en la navegación por el tutorial. De esta división general encontramos una subclasificación más en el Módulo 4, el del temario, ya que separamos su contenido para estudiar con más detalle cada una de sus partes. Esta subdivisión se resume en la Figura 11.

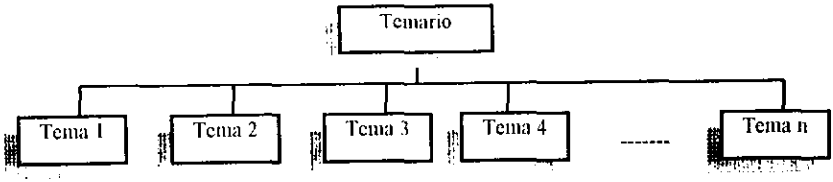

Figura 11. Subdivisión de un modulo del sistema.

Al tener esta subdivisión en los temas, comienzan a diferenciarse los distintos niveles que tiene el sistema puesto que algunas páginas que tratan los tópicos del temario se puede profundizar más, pasando a un siguiente plano. El diagrama de flujo de datos para el primer nivel se muestra en la Figura 12.

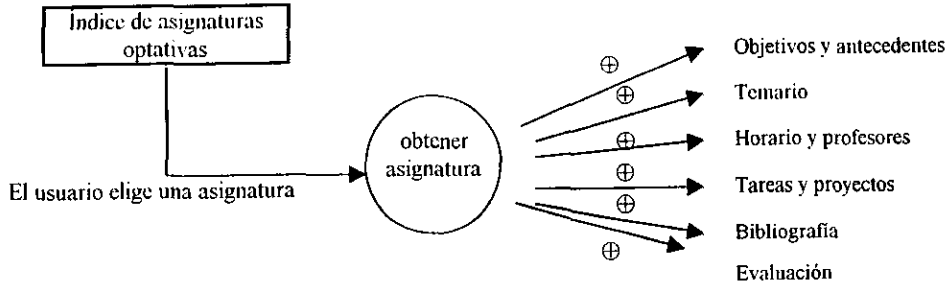

Figura 12. Diagrama de flujo de datos para el primer nivel.

Se considera primer nivel porque es la información que se puede observar en forma inmediata una vez que se ha seleccionado una asignatura.

El segundo nivel, el del temario, se muestra en la Figura 13.

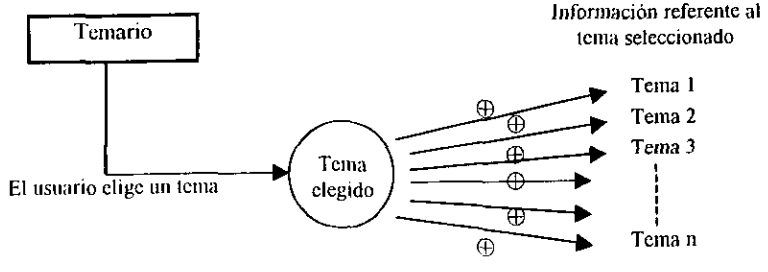

Figura 13. Diagrama de flujo de datos para el segundo nivel.

Para este segundo nivel se debió haber pasado por el primer nivel. El segundo nivel permite consultar la información referente a un tema particular de la asignatura seleccionada.

En el tercer nivel, se profundiza en un tema específico seleccionado previamente en el segundo nivel. El diagrama sería el siguiente:

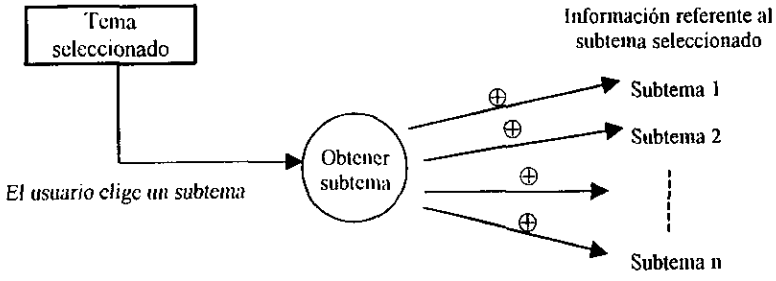

Figura 14. Diagrama de flujo de datos de tercer nivel.

Si agrupamos las Figuras 12, 13 y 14, el diagrama de flujo de datos completo quedaría representado en la Figura 15.

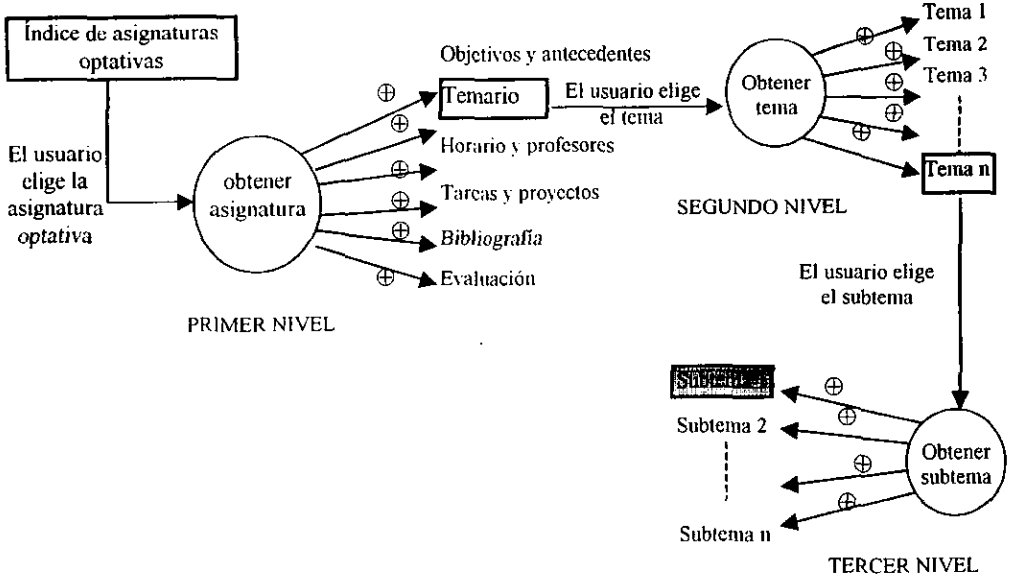

Figura 15. Diagrama de flujo de datos completo.

Para saber más sobre el funcionamiento de este sistema describiremos a continuación algunas de las pantallas que lo conforman.

El sistema consta primeramente de una *pantalla principal o de bienvenida,* la cual saluda al usuario y le informa de manera general, qué es el sistema y quién o quiénes lo respaldan. Además, permite contabilizar el número de personas que visitan el tutorial, así como, qué institución ampara la información.

Posteriormente se encuentra el *índice de materias* o *pantalla llave*, denominada así por la similitud que existe con el mundo real (quien tiene unas llaves puede abrir un auto o *entrar a su casa*). Esta pantalla muestra las once alternativas de asignaturas optativas que tiene la carrera de ingeniería en computación y permite la entrada a cualquiera de ellas como también el regreso a la pantalla de bienvenida o la salida del sistema. Este índice de materias muestra además, una breve introducción de lo que es el sistema en una ventana sobrepuesta, que aparece cada vez que se visita la página y desaparece cuando el usuario lo indica. A un costado de la lista de las materias se cuenta con un gráfico alusivo o relacionado con la asignatura que se pretenda visitar; así hace más amena y atractiva la consulta de una u otra materia.

Esta página a su vez, tiene una liga de comentarios y sugerencias. Aquí se despliega una encuesta en la cual el usuario reflejará sus impresiones y críticas sobre el sistema. También podrá expresar con libertad sus inquietudes y colaborar en el mantenimiento del tutorial.

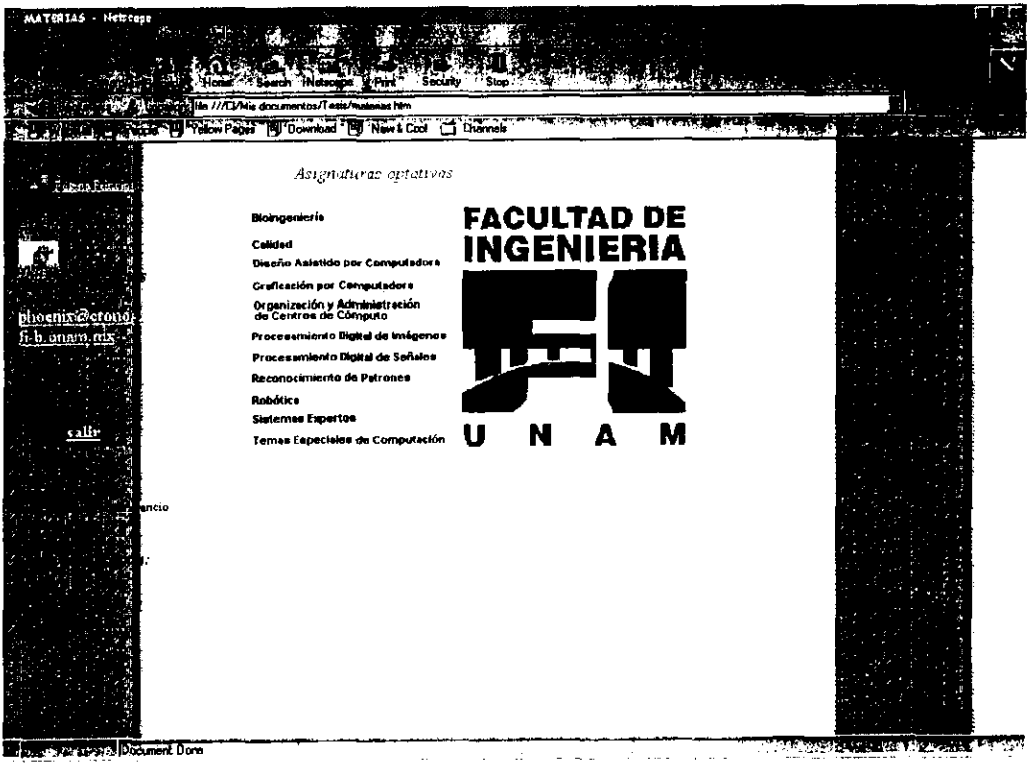

Figura 16. Página del índice de materias.

La Figura 16 muestra la pantalla índice de las asignaturas optativas pero, ¿qué hay detrás de esta imagen? Por supuesto que un código instrucciones y sentencias de HTML y Javascript que es interpretado por el navegador Netscape 2.0 y versiones superiores, o bien, por versiones superiores a la 3.0 de Intenet Explorer.

Para entender un poco el código fuente de esta página se muestra parte del mismo.

El fragmento de código que se encuentra en la Figura 17, se utiliza para que, cuando el usuario decida salir del sistema, al seleccionar la liga de salir, el sistema lo llevará, automáticamente y de manera aleatoria, a alguna de las direcciones de Internet indicadas en la línea linktext.

Como se observa, las siete opciones de salida son páginas relacionadas con la UNAM (página de la UNAM, *estudios de posgrado e información y actividades que se realizan* dentro de la Facultad de Ingeniería).

```
<SCRIPT >
<!—Oculta el código de javascript cuando los navegadores no son capaces de interpretarlo.
    function picklink() {
    var linknumber = 7 ;
    var linktext = "nolink.html" ;
    var randomnumber = Math.random() ;
    var linkselect = Math.round( (linknumber-1) * randomnumber) + 1 ;
    if ( linkselect == 1)
    {linktext="http://www.unam.mx/" }
    if ( linkselect == 2)
    {linktext="http://odin.fi-b.unam.mx/" }
    if ( linkselect == 3)
    {linktext="http://cosmeg.fi-a.unam.mx/" }
    if ( linkselect == 4)
    {linktext=="http://verona.fi-p.unam.mx/" }
    if ( linkselect == 5)
    {linktext="http://odin.fi-b.unam.mx/ptc/"}
    if ( linkselect == 6)
    {linktext="http://www.unam.mx/gaceta/" }
    if ( linkselect == 7)
    {linktext="http://www.cecafi.unam.mx/" }
    return linktext;
    }
    // Fin del script -->
    </SCRIPT >
<script language="JavaScript">
```

Figura 17. Fragmento de código de la página del índice de materias.

El fragmento de código siguiente es para que el usuario vea en una ventana sobrepuesta en la página del índice de materias una breve explicación sobre el *sistema tutorial*. En el código se define la posición, ancho y largo de la ventana así como, la información que mostrará dicha ventana, (archivo opta.htm)

```
<!--
 var gt = unescape('%3e');
 var popup = null;
 var over = "Launch Pop-up Navigator";
 popup = window.open('', 'popupnav', _
 'width=500,height=250,resizable=1,scrollbars=auto');
 if (popup != null) {
   if (popup.opener == null) {
         popup.opener = self;
   }
   popup.location.href = 'opta.htm';
 }
 // -->
```

El fragmento de código que muestra la Figura 18 se utiliza para visualizar los nombres de las once asignaturas optativas. Cada función activa, *muestra el título de la asignatura* que puede ser visitado, este título tiene su texto en color negro. La función inactiva muestra el mismo título de la asignatura sólo que para esta función, el texto es de color azul marino que significa que la materia ya ha sido visitada por el usuario. De esta forma, se pretende indicar al usuario que asignaturas ha seleccionado y cuáles le faltan por revisar.

```
<script>
    function activa() {
       document.imagen.src = "bio2.jpg"
    }
    function inactiva() {
       document.imagen.src = "bio.jpg"
    }
     function activa4() {
       document.imagen4.src = "gpc2.jpg"
    }
    function inactiva4() {
       document.imagen4.src = "gpc.jpg"
    }
     function activa11() {
       document.imagen11.src = "tec2.jpg"
    }
    function inactiva11() {
       document.imagen11.src = "tec.jpg"
    }
</script>
```

Figura 19. Fragmento de código de la página del índice de materias.

Los nombres de los archivos a los cuáles hace referencia la línea "document.imagen.src" son las iniciales de la asignatura que aparecerá. En la Figura 18, el archivo bio2.jpg refiere a la asignatura Bioingeniería cuando aún no es visitada por el usuario mientras que, bio.jpg, contiene el mismo título pero en color azul para indicar que la liga ha sido revisada.

El código que aparece en la Figura 19 son sentencias de HTML que se utilizan para definir las demás características de la pantalla como son, el color del fondo, la organización de la página, los gráficos que contiene (por ejemplo, la liga a la página principal y el buzón para comentarios y sugerencias), el tipo, tamaño y color de la letra, así como, los todos los textos que no se definen en funciones y que contiene la página como son: quién elaboró la página y cuando se realizó la última modificación.

La Figura 19 también incluye el código para llamar a la o las funciones de javascript definidas anteriormente a través de HTML así como, las sentencias para definir las direcciones de Internet a donde llevará cada una de las ligas.

Para un mejor entendimiento del código de HTML consultar la bibliografía que se incluye sobre este tema en el Anexo 1.

Teniendo un mayor conocimiento sobre el contenido de esta pantalla llave, supongamos que el usuario desea visitar la asignatura "Graficación por computadora" ¿qué información encontrará? :

Haciendo clic en la línea que dice Graficación por computadora, la liga lo llevará a una página donde, en su parte central aparecen los objetivos generales, el número de créditos que se logrará obtener después de cursar esa asignatura y los antecedentes requeridos para la misma y en el costado izquierdo encontrará un menú que le facilitará la navegación, pues resume las características más importantes de la asignatura en una lista como la siguiente:

- Objetivos y antecedentes,
- Horario y profesores,
- Temario,
- Tareas y proyectos,
- Bibliografía,
- Evaluación,
- Índice de materias.

Esta página, además, contiene un gráfico relacionado con la asignatura y si el usuario, posiciona el ratón sobre éste aparecerá una ventana en donde se explica qué es la figura, y porqué razón se emplea en la materia (Figura 22). Generalmente, estas imágenes que aparecen con los objetivos son lo más representativo de la asignatura. En el caso que estamos analizando, la imagen que encontramos con los objetivos de Graficación por computadora, es una de tantas versiones que existen del fractal de Mandelbrot, el cual es uno de los modelos que emplean los que cursan la materia para realizar sus experimentos y aprender nuevos conceptos y comportamientos de la graficación por computadora.

```
<HTML>
<HEAD>
<TITLE>MATERIAS </TITLE>  //Titulo de la página
</HEAD>
<BODY TEXT="#000000" BGCOLOR="#FFF0F0" LINK="#FFFFFF" VLINK="#0000FF" ALINK="#FF0000"
BACKGROUND="fond1'.jpg">
<table>
<tr><td>
<FONT COLOR="#FF6666" SIZE=-1><IMG SRC="home.gif"><A HREF="homepage.html"
onMouseOver="window.status='¿De regreso?';return true;" onMouseOut="window.status='Haz click sobre la asignatura
optativa de la cual quieras obtener información';return true;">
P&aacute;gina Principal</A>
<P>
</FONT>          //Termina definición de características del texto de la liga Página Principal
</td><td>
<center><I>
<BLINK><FONT COLOR="#FF0000"><FONT SIZE=+1>
                  
 Asignaturas optativas
</FONT></FONT></BLINK></I></center>
</td></tr>
</table>
<table>
<tr><td>
<br>
<IMG SRC="mail.gif" NOSAVE HEIGHT=40 WIDTH=40>
<FONT COLOR="#000000"><FONT SIZE=+1>
Sugerencias                            <br>y comentarios a <br>
<A HREF="mailto:phoenix@cronos.fi-b.unam.mx" onMouseOver="window.status='Nos interesa tu opini&oacute;n. Manda
tus comentarios al correo que se indica';return true;" onMouseOut="window.status='';return true;">
phoenix@cronos.<br>fi-b.unam.mx</A></FONT></FONT>
<br><br><br><br><br>
<font size=+1><a HREF="" onClick="this.href=picklink()"><b>salir</b></a></font>
<br>
</td>
<td><center><table border=0>
<tr><td>
<A HREF="frbio.htm" onMouseOver="document.images[25].src='neurona.jpg';activa();window.status='¿Quieres
saber sobre la materia de Bioingenieria?';return true;" onMouseOut="inactiva();window.status='Haz click sobre la
asignatura optativa de la cual quieras obtener información';return true;"><img src="bio.jpg" name="imagen">
</A><br>
```

// El código en negritas se repite para las once asignaturas. Lo que cambia, son los nombres de los archivos de referencia (fr*.htm, *.jpg) y los mensajes que aparecen en la sentencia windows.status.

```
</TD><TD valign=top>
<img src="ingenier.gif">
</TD>
</tr></table></center>
</td></tr></table>
<br>
```

```
<FONT COLOR="#000000" SIZE=-1>Elaboraron: <BR>
Mar&iacute;a Eugenia P&eacute;rez Aparicio, <BR>Leopoldo Nieves Javier.
</FONT>
<P><FONT COLOR="#000000"><I><SCRIPT LANGUAGE="JavaScript">
document.write( "Última modificación:<br>" + document.lastModified );
</SCRIPT></FONT></I> </P>
</BODY>
</HTML>
```

Figura 20. Código HTML de la página del índice de asignaturas optativas.

La pantalla que se muestra en la Figura 21 confirma lo dicho en los párrafos anteriores.

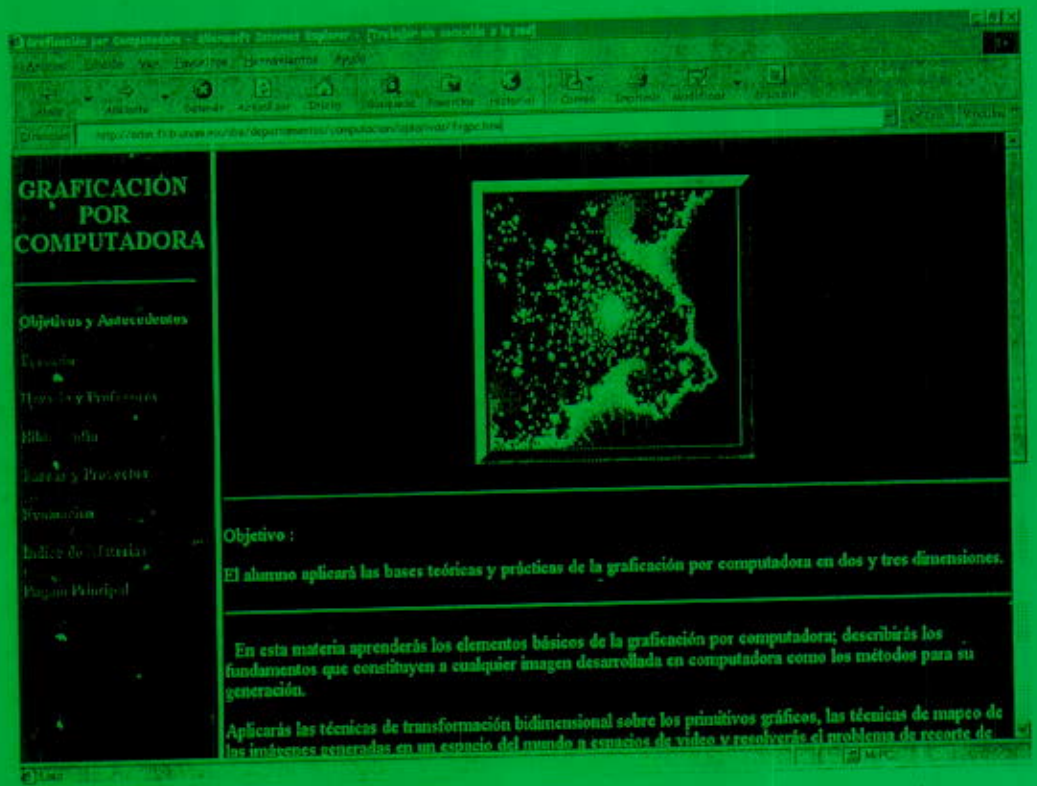

Figura 21. Ejemplo de la página para la asignatura de Graficación por Computadora.

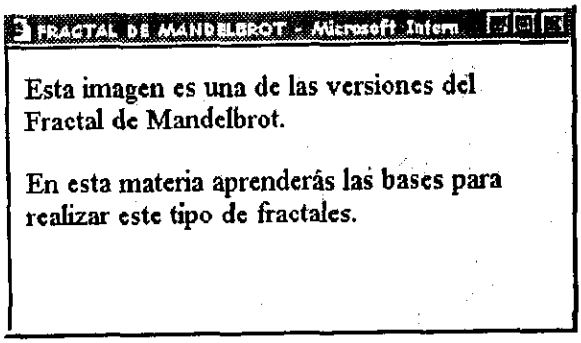

Figura 22. Ejemplo de la ventana explicativa que aparece cuando el usuario se posiciona sobre la imagen de la figura 21.

Siguiendo con nuestro recorrido, probablemente al usuario le interesará conocer el temario de la asignatura para saber qué puede aprender en esa materia o cuál es su contenido. Pues bien, basta con hacer clic en la palabra "Temario"; si elige esta liga automáticamente se encontrará en una pantalla similar a la mostrada anteriormente, sólo que en la parte central hallará una tabla con los nueve temas que comprende la asignatura. En la pantalla verá que para cada tema existe una liga que lleva a la información más importante y específica del tema seleccionado teniendo en algunas páginas la opción de profundizar en un tópico de interés o encontrar ligas a otras páginas existentes en Internet donde se trata el mismo tópico desde otro punto de vista.

La información contenida en cada una de estas páginas está basada en la bibliografía que contiene el temario que proporciona la coordinación de la asignatura, así como en los apuntes que dan algunos profesores que imparten la asignatura, con lo cual aseguramos que esta información es confiable y veraz.

En la pantalla que se muestra en la Figura 21, vemos en el menú que las ligas que han sido visitadas tienen un color anaranjado mientras que las que faltan por visitar, conservan el color verde. Esta característica del cambio de color en las ligas ya visitadas se mantiene en todo el sistema; de esta forma el usuario sabe cuáles páginas ya visitó y no se encierra en un ciclo sin fin navegando entre una y otra página que ha revisado con anterioridad o viceversa, si encontró información de su interés y no recuerda en qué página fue, basta con visitar nuevamente las ligas de color rojo y seguramente encontrará la pantalla que busca.

En las páginas donde se desglosa cada tema en particular, se continúa teniendo el menú de datos importantes de la asignatura al lado izquierdo de la pantalla; así se facilita y se hace más interactiva la búsqueda de algún dato en especial. Por ejemplo, para la materia en cuestión, se tiene la siguiente presentación en el tema I, Introducción (Figura 23).

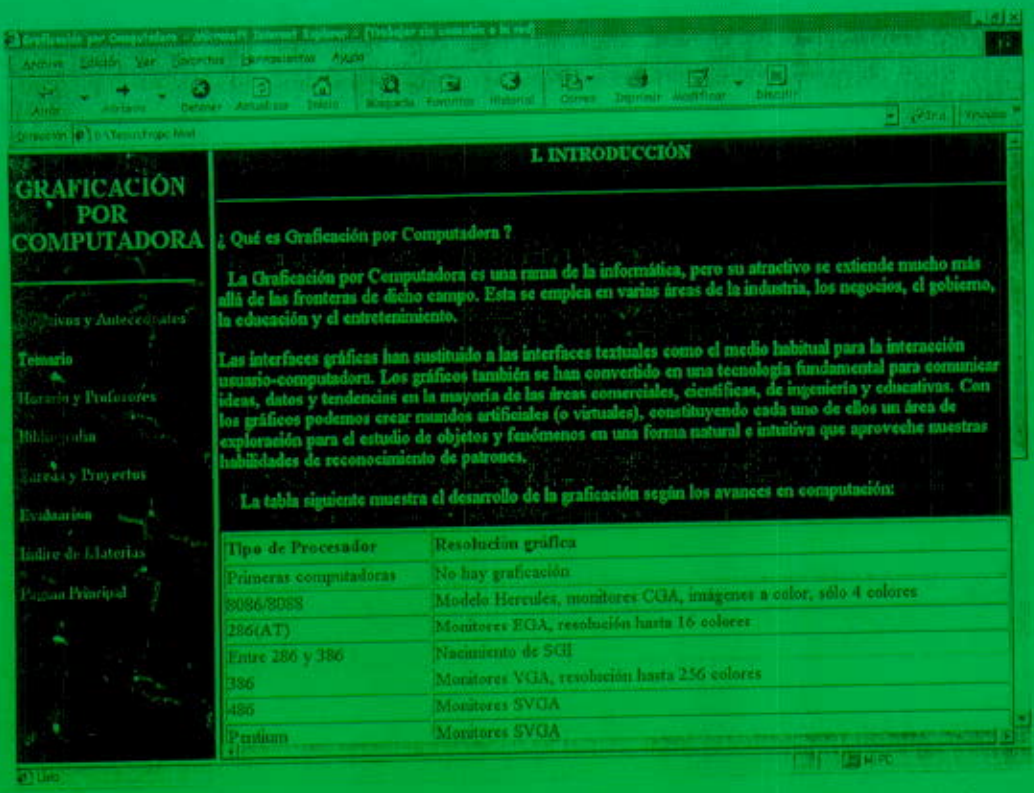

Figura 23. Ejemplo de la parte introductoria en la asignatura Graficación por Computadora.

En esta pantalla podemos ver las características que se han comentado con anterioridad, se tiene una página con frames para facilitar la navegación. Las páginas tienen su propio color de fondo que las identifica según el área y el nivel donde se encuentra; es decir, las pantallas que son contenido del temario tienen un background azul mezclilla, mientras que las pantallas de primer nivel -aquellas donde se encuentra el temario, la bibliografía, la evaluación, etc.- tienen un fondo más obscuro, casi negro. Así también, la parte donde está el menú de datos importantes tiene fondo con tonos de azul que se van aclarando en forma descendente.

También vemos en esta página de Introducción, además de la definición de la asignatura, una tabla donde se concentra el avance de la graficación por computadora a través de los años, se hace la comparación en relación al surgimiento de los procesadores. Así mismo, si desplazáramos la barra hacia abajo, encontraremos un breve glosario de términos que se utilizan en la materia y hasta el final de la página está escrita la bibliografía empleada para la realización de dicha página.

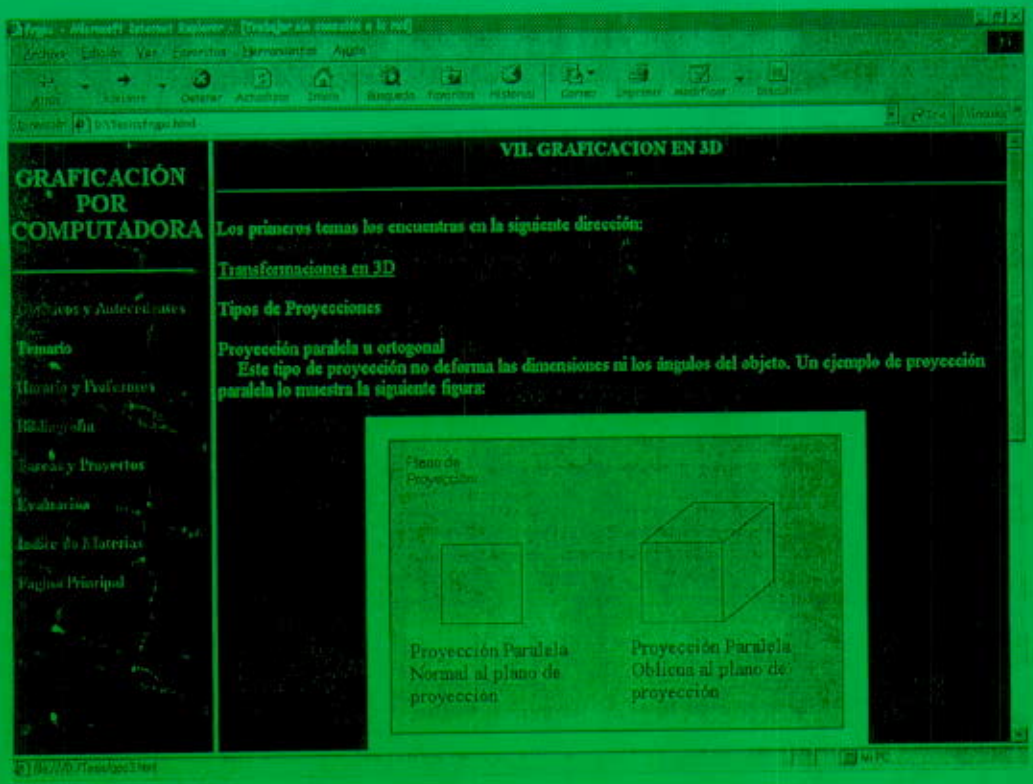

Figura 24. Ejemplo del tema VII de la asignatura Graficación por Computadora.

Así como esta página cuenta con una tabla que resume en cierta forma un capítulo del tema, otras páginas muestran gráficos que nos ayudan a entender mejor el tema como es el caso de la pantalla que se muestra en la Figura 24, donde se trata el tema VII de la misma materia (Graficación por computadora).

En la Figura 24 podemos apreciar que se cuenta con toda la información básica y más importante de la graficación en tres dimensiones. Se definen tipos de proyecciones, sus características básicas, clasificaciones en función del plano de proyección, etc. También encontramos un gráfico donde se muestra la diferencia entre una forma de proyección y otra según el plano de proyección. Además, se cuenta con una liga hacia otra página en la cual se describe con más detalle las características de la transformación en 3D.

De forma similar a como vemos esta pantalla, se encuentran los demás temas de esta materia. Citaremos un ejemplo más. En el tema V, podemos encontrar toda la información necesaria para entender qué es el *Diseño de curvas*. Se habla de tipos de curvas, sus características básicas, su clasificación en función de la forma de la curva (si es continua). Así mismo, se encuentran algunas fórmulas y matrices que se emplean con frecuencia para el diseño de curvas. También encontramos un gráfico donde se muestran cada uno de los parámetros que intervienen para poder identificar un tipo de curva.

Posteriormente se encuentran las descripciones simplificadas de los principales grupos de curvas teniendo la opción de profundizar en cada curva dependiendo del interés del usuario. En la parte más baja y separada por una línea, se encuentra la bibliografía particular para esta página pues como ya se ha dicho, toda la información que se incluye dentro de este sistema tutorial tiene una base confiable y si el usuario quisiera ahondar más en el tema puede consultar los textos o preguntar al profesor que imparte la asignatura. Inmediatamente después de la bibliografía viene un icono que permite regresar al temario de la asignatura, en caso de que se requiera visitar otro tema.

Aquí como en las demás páginas, se tienen nuevamente dos posibilidades, avanzar, profundizando más en el tema o retroceder y visitar otro tema. Supongamos que el usuario está interesado en conocer algo más sobre los principales tipos de curvas y decide oprimir el botón del ratón sobre el texto que se encuentra marcado como liga (Curvas de Hermite o hermitianas). Aparecerá entonces una pantalla similar a las dos que hemos visto sobre los temas de la asignatura. En esta pantalla encontrará una definición más amplia de este tipo de curva, cómo se puede generar una curva *hermitiana y las ecuaciones* que se requieren para ello, también verá un gráfico con algunos ejemplos de estas curvas. Al final de la información, de nueva cuenta se tiene una línea que indica el fin de la información e inmediatamente después la bibliografía de la cual se obtuvo dicha información. Al terminar la bibliografía se tiene un enlace hacia la página anterior, esto es, a la página del tema V que es el que se está explorando.

Bien, como hemos visto en este breve paseo por algunos temas de Graficación por computadora, el *alumno cuenta ahora con mayores* alternativas de consulta para informarse correctamente y elegir acertadamente las asignaturas optativas que cursará para concluir su carrera. Debemos recordar que es de vital importancia hacer una buena selección, porque de esto depende la rama de especialización que tendremos los ingenieros en computación una vez que finalicemos las materias que marca el plan de estudios. Habrá quienes se inclinen por la parte del hardware o quienes prefieran especializarse en software y dentro de cada uno habrá quienes gusten de las bases de datos y quienes se sumerjan en el *mundo de las imágenes y graficación*. Como vemos el campo de la computación es muy extenso y dependerá en gran parte de la selección de las asignaturas optativas el área en la que nos desenvolvamos como profesionales de la computación; por ello, es muy importante elegir bien, aprender las bases y conocer con veracidad las oportunidades de trabajo y desarrollo que ofrece cada materia optativa.

Pensando en la importancia de saber qué tanto aprendemos de una materia, nuestro sistema tutorial incluye un evaluación como parte *fundamental de la visita a una* asignatura. La evaluación consiste en una serie de preguntas relacionadas con los temas que incluye el tutorial y tres opciones de respuesta. El alumno lee la pregunta y con el ratón da un clic en la opción que considere, es la correcta. Si la respuesta es acertada, se muestra un mensaje de felicitaciones pues se ha aprendido el concepto, si no es correcta, aparece un mensaje diferente donde se indica que la respuesta fue errónea y se sugiere al alumno visitar de nuevo la página donde se trata el tema de la pregunta.

La Figura 25 muestra una parte de la pantalla de evaluación, si se recorre la barra de desplazamiento se podrá ver la continuación de la evaluación.

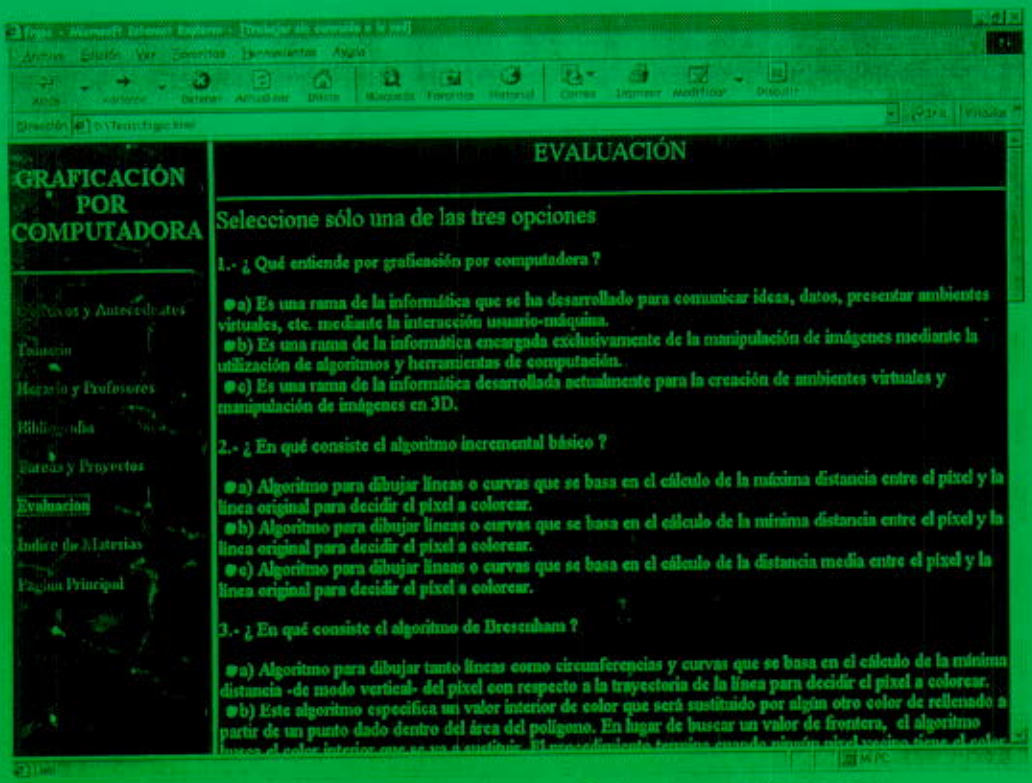

Figura 25. Evaluación para la asignatura Graficación por Computadora.

En el mismo nivel que existe la evaluación, se tiene una pantalla donde se ponen a disposición de los alumnos, las tareas y proyectos que se han desarrollado para la asignatura que están explorando. En nuestro caso, la asignatura es Graficación por computadora y en la página de tareas y proyectos se encuentra una lista de las tareas más frecuentes que suelen hacerse durante el semestre. Además, en la parte de proyectos, se muestran ejemplos que han hecho algunos compañeros en semestres anteriores para esta asignatura.

La Figura 26 muestra cómo aparece en nuestro sistema la parte de proyectos de la página "Tareas y Proyectos". En la pantalla podemos observar que existe una liga para cada nombre asignado a un proyecto, así como los autores del mismo y el semestre en que se fue realizado.

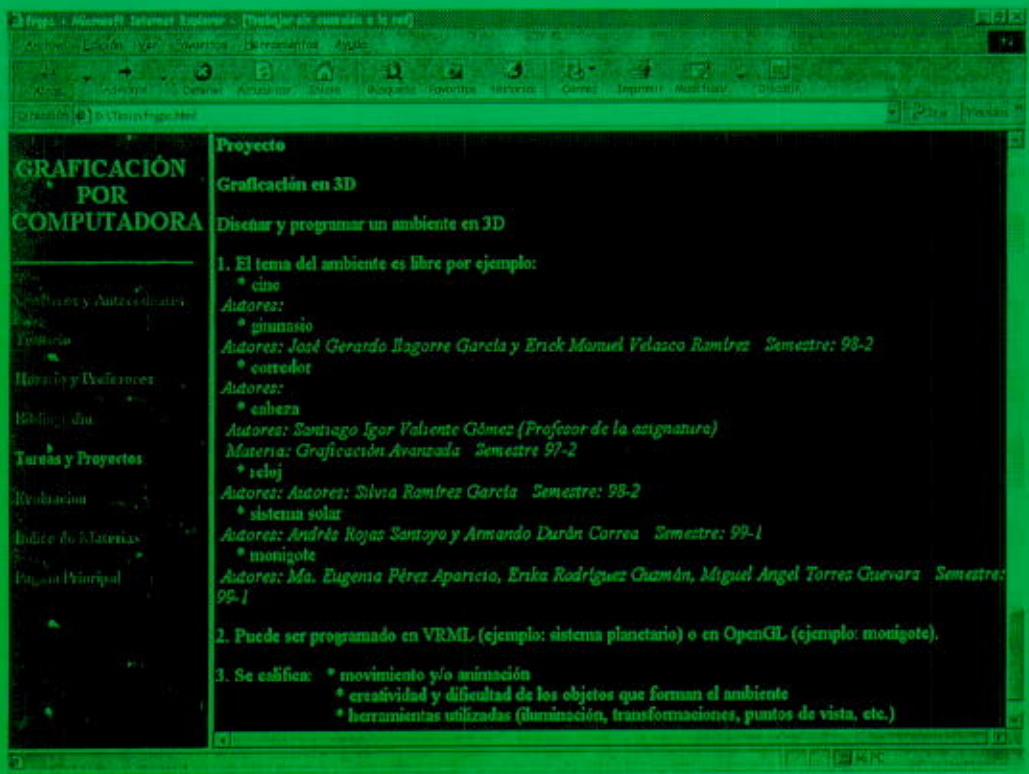

Figura 26. Página de tareas y proyectos para la asignatura Graficación por Computadora.

La finalidad de que los alumnos puedan acceder a estos proyectos es que tengan una idea más clara de lo que serán capaces de hacer al término del semestre y a su vez, superar la calidad en el aprendizaje de los temas de la asignatura puesto que el alumno pondrá mayor empeño en el diseño, originalidad e imaginación de su proyecto, a fin de mejorar lo que han realizado otros compañeros anteriormente. Del mismo modo, se inculcará en los alumnos el espíritu de motivación para elegir alguna asignatura, así como el deseo de superarse constantemente al desarrollar proyectos innovadores y de calidad pues tendrán como referencia proyectos de semestres anteriores.

Como se observa en la Figura 26, la última liga en el menú es hacia la página del índice de las asignaturas (Figura 16). Si el usuario acciona esta liga podrá seleccionar alguna otra asignatura que le interese y encontrará una clasificación similar a la que hemos descrito para la materia Graficación por Computadora.

CONCLUSIONES

Después de haber finalizado el desarrollo del sistema tutorial de las asignaturas optativas de la carrera de Ingeniería en Computación y de igual manera terminadas cada una de las etapas de evaluación de dicho sistema, podemos establecer las siguientes conclusiones:

1. El desarrollo de software no es una tarea sencilla, ni mucho menos. El problema se complica si no se tiene una metodología a seguir para su desarrollo. Ahora bien, si se cuenta con una metodología para desarrollar software se resuelve parte de su complejidad, aunque no en un cien por ciento. Se resuelve el conflicto de desarrollar sin ninguna dirección, pues se tiene una guía que nos indica por dónde atacar el problema y las posibles herramientas que se pueden emplear. Sin embargo, se encuentra una disyuntiva, ¿Qué metodología aplicar?. En efecto, debido a que se cuenta con diferentes metodologías para desarrollar software, y cada software es diferente, lo primero que hay que definir y delimitar el problema; una vez hecho esto, el siguiente paso es elegir una metodología que permita construir adecuadamente el sistema. El punto hasta aquí parece estar resuelto, no obstante, en la mayoría de los casos no es así, ya que algunas veces no es posible seguir la metodología al pie de la letra, y en casos, un poco más extremos, no se encuentra una metodología que satisfaga completamente los requerimientos del software que se planea realizar.

   En nuestro caso, la primera dificultad se presentó al momento de revisar las metodologías existentes y verificar que no había un método específico para el desarrollo de sistemas tutoriales; que este tipo de sistemas se han desarrollado a partir de una combinación de las metodologías existentes ajustándolas para cada caso.

   Para resolver este primer obstáculo se planteó la posibilidad de buscar entre los métodos que existen alguno que se adecuara a las necesidades requeridas. Sin embargo nos encontramos con la dificultad de que no había una metodología que se adaptara completamente a las necesidades de nuestro sistema por lo que tuvimos que implementar una metodología para este problema en particular.

   En base a las experiencias de otras personas e instituciones que han elaborado este tipo de programas, la nueva metodología estaría basada en las metodologías existentes para desarrollar software de tal forma que se cumpliera con cada uno de los requerimientos de nuestro sistema. Así, la metodología que propusimos para elaborar este proyecto está basada en la metodología de espiral con algunos cambios en la etapa de desarrollo con el fin de crear un sistema 100% útil a nuestros compañeros y demás personas que deseen consultarlo.

   Del modelo en espiral se incluyen partes fundamentales como son la planeación, el desarrollo y el mantenimiento, fases que a su vez se proponen en el método del ciclo de vida y en general en casi todos los métodos existentes para desarrollar software. Nuestro modelo contempla también, la etapa de análisis de riesgo que es una característica de la metodología de espiral y pensando en que nuestro sistema fuera

más completo, incluimos dentro de la parte de ingeniería, el modelo de construcción de prototipos el cual nos permite tener una versión de prueba de cómo quedará el sistema una vez que se haya terminado y de esta manera poder corregir a tiempo las imperfecciones que puedan surgir al momento de implementar una u otra herramienta de aplicación o simplemente tener una muestra de cómo se hará la presentación de la información al usuario final.

Como ya se ha terminado el sistema, podemos concluir que la metodología que propusimos fue la más adecuada para nuestra tarea pues nos permitió desarrollar el software dentro de los tiempos establecidos y una vez que fue presentado el producto ante los usuarios finales éste resultó ser una verdadera herramienta de ayuda para la elección de las asignaturas optativas que se imparten para nuestra carrera en la Facultad de Ingeniería de la UNAM.

2. La realización de este trabajo contribuyó de una manera significativa a conocer algo más acerca del contenido, los alcances, los materiales y las nuevas tecnologías que se emplean para cada una de las asignaturas optativas que comprende el plan de estudios de la carrera de Ingeniero en Computación. Este conocimiento nos brindó la oportunidad de entender y comprender aún más la importancia que representan cada una de estas materias en el desarrollo profesional del Ingeniero en Computación; razón que refuerza el hecho de saber elegir qué asignaturas cursar en los últimos semestres de acuerdo a nuestros intereses y habilidades. Podríamos decir que esta decisión contempla un aspecto fundamental tanto en nuestro desarrollo personal como profesional, equivalente a cuando elegimos una carrera o profesión.

   ¿Por qué consideramos de importancia el hecho de saber elegir una materia optativa?

   Primero, porque es la parte culminante de nuestra educación profesional a nivel licenciatura en donde se realiza esta elección y por lo tanto, es conveniente tener un buen final en otras palabras, "cerrar con broche de oro, dando nuestro máximo esfuerzo". Esto, porque al ser nuestra etapa final la decisión que tomemos representa, en cierto modo, nuestra especialización dentro del área de Ingeniería en Computación y tener al menos un breve conocimiento de lo que la asignatura de nuestro interés abarca, nos acerca a la realidad de la vida profesional.

   Otro aspecto sobresaliente lo conforma en toda su expresión, "saber elegir" debido a que una vez que concluimos nuestros estudios profesionales en el nivel de licenciatura es de esperarse en la sociedad y particularmente en nuestras responsabilidades, que tenemos la preparación y capacidad suficiente para poder tomar una decisión, acorde con los resultados esperados y que en verdad ayuden a solucionar el problema.

   Debido a lo expuesto en los párrafos anteriores de este punto, consideramos que el sistema tutorial de asignaturas optativas que hemos desarrollado cobra mayor importancia, pues se convierte en una herramienta útil para dar a conocer un panorama general de estas materias y servir a los alumnos como apoyo para la correcta elección de sus materias optativas.

3. El hecho de colocar este sistema en la red mundial de información, Internet, ejemplifica como puede emplearse una herramienta de la tecnología para resolver un problema real. En este caso utilizamos la herramienta tecnológica dentro del campo de la computación denominada Internet, la cual ha permitido la difusión e intercambio de todo tipo de información hacia diferentes lugares del mundo y actualmente se busca la manera de poder comercializar de manera confiable y segura a través de ella. La posibilidad de colocar información en el Web nos da la ventaja y la oportunidad de dar a conocer públicamente la información referente a las materias optativas que se imparten en nuestra carrera y así obtener un reconocimiento más, a nivel mundial, en cuanto a los servicios que nuestra Universidad ofrece a sus estudiantes y personal docente.

   El caso del sistema tutorial que se ha desarrollado, se convierte en un claro ejemplo de los servicios que la UNAM ofrece de forma gratuita a sus alumnos y en nuestro caso, a los compañeros de la Facultad de Ingeniería puesto que al habersenos permitido colocar este desarrollo en la red local del Laboratorio de Computación, todo aquel alumno que requiera consultarlo podrá hacerlo libremente con sólo acudir a este laboratorio o bien, lo podrá visualizar en cualquier computadora desde su casa u oficina con tan sólo tener una conexión para Internet y un navegador sobre cualquier plataforma (Windows, Novell, Linux, UNIX, etc.).

   Otra razón que motivó a desarrollar este sistema para que pudiera ponerse en el Web fue el objetivo principal de esta tesis, que los alumnos contáramos con un medio rápido y confiable para consultar información y/u obtener referencias sobre las asignaturas optativas que se imparten en nuestra escuela y de las cuales únicamente podemos elegir cuatro de un total de once. Si el sistema instalado en la red local del Laboratorio de Cómputo de la DIE no tuviera servicios de Internet, nuestro objetivo principal hubiera quedado limitado a aquellos alumnos que contarán con su credencial para accesar a este laboratorio. De esta manera, la mayor parte de este sistema fue desarrollada en el laboratorio de la DIE, y el sistema terminado fue colocado en el servidor de red que cuenta con los servicios de Internet con estas acciones quedo resuelto nuestro problema, ya que como hemos mencionado anteriormente, el sistema tutorial quedo accesible para una mayor cantidad de alumnos y además, de cualquier otra persona que esté interesada en alguno de los temas que cubre el tutorial.

4. La diversidad de herramientas que existen en la actualidad para desarrollo de software nos permitió crear un sistema tutorial dinámico y muy atractivo para todo aquel que desee aprender algo o tan sólo tener un panorama completo sobre las asignaturas optativas que ofrece la carrera de Ingeniería en Computación de la Facultad de Ingeniería de la UNAM. El comentario anterior sale a la vista porque cuando llegamos a la etapa de desarrollo se nos presentó la interrogante, ¿cuál herramienta sería la más conveniente para la programación de este sistema tutorial?. Después de hacer una investigación y posteriormente una revisión de las herramientas existentes, decidimos utilizar el lenguaje de manipulación de hipertexto, que fue un punto importante, pues ello nos permitió poder instalar este sistema en Internet y además manejar una mejor presentación al poder incluir colores en los

textos, fondos para diferenciar el contenido en cada una de las páginas, frames para facilitar la búsqueda y la navegación por el sistema, imágenes con animaciones que hicieran agradable la interfaz con el usuario así como una limitada cantidad de espacio en el disco duro de cualquier servidor.

Además del lenguaje de hipertexto, HTML, utilizamos el lenguaje de programación conocido como JavaScript para aumentar la interactividad del sistema, ya que se agregan pantallas de información cuando el usuario se posiciona sobre una imagen o cuando selecciona las respuestas en la evaluación. También dentro de las herramientas utilizadas se tuvo la facilidad de incluir audio en el tutorial para amenizar la estancia del visitante.

Así mismo, al usar HTML generamos código portable, fácil de entender y modificar.

Al utilizar Netscape como browser para revisar nuestras páginas aprovechamos también algunas de las herramientas que incluye el navegador como por ejemplo, el compositor o creador de páginas "Page Composer" que permite diseñar una página de Internet de manera sencilla y amigable.

Otra característica más de las herramientas que empleamos es que ellas son software libre que se puede utilizar sin tener una licencia y por lo mismo se pueden actualizar las versiones con sólo bajar el software del Web e instalarlo correctamente en la computadora. Por ello, nuestro sistema podrá usarse por mucho tiempo, pues fue construido sobre una base sólida y actual.

BIBLIOGRAFÍA

Textos de Consulta

1. Fairley,Richard; "Ingeniería de Software"; Editorial: McGrawHill; México; 1987.

2. Pressman, Roger S.; "Ingeniería de Software"; Editorial: McGraw-Hill.; México ; 1988.

3. Sommerville, Ian.; "Ingeniería de Software"; Editorial: Addison Wesley Iberoamericana.; México ; 1988.

4. Kenneth E. Kendall, Julie E. Kendall ; Tr. Hector Lopez Hernandez.; "Análisis y diseño de sistemas."; Editorial: Prentice-Hall.; México ; 1991.

5. Larson , James A.; "Interactive Software. Tools for Building Interactive User Interfases". ; Yourdon Press Comuting Series.; E.E.U.U. ; 1989.

6. Gros Salvat, Begoña; "*Aprender mediante el ordenador*. Posibilidades pedagógicas de la informática en la escuela."; Biblioteca Universitaria de Pedagogía.; España; 1987.

7. Gros Salvat, Begoña; "Diseños y programas educativos. Pautas pedagógicas para la elaboración de software"; Editorial Ariel; España; 1997.

8. Lemay, Laura.; "Aprendiendo HTML 4 para WEB en una semana." ; Editorial Prentice Hall Iberoamericana ; México ; 1998.

9. December , John and Ginsburg, Mark.; "HTML & CGI UNLEASHED." ; Editorial Sams ; E.E.U.U.; 1995.

10. Moore, Aubrey. ; "Software Engineering" ; Sheffild Hallam University ; E.E.U.U. ; 1996.

11. Gane, Chris and Sarson, Trish; "Structured Systems Analysis: Tools and Techniques" ; Editorial Prentice Hall. ; E.E.U.U. ; 1999.

12. Zelkowitz, Marvin V. & Shaw, Alan C. & Gannon, John D. "Principles of software engineering and design"; Editorial Prentice Hall ; México ; 1945.

13. Ince, D.C. version en español de Ernesto Morales Peake, con la colaboracion de Guillermo Levine.; "*Ingenieria de software*"; *Editorial Prentice Hall* Iberoamericana ; 1975.

14. McClure, Carma. traducción. José Manuel Ortega y Ortega "Case : la automatización del software".; Editorial Prentice Hall Iberoamericana; México; 1990.

ANEXO 1.

1.1 ¿ QUÉ ES "WORLD WIDE WEB"?

La World Wide Web *(Web)* es una red de recursos de información. Ésta requiere de tres mecanismos *para el manejo de los recursos*

El World Wide Web (Web) es una red de recursos de información. El Web confía en tres mecanismos para hacer estos recursos prontamente disponible al posible público más ancho:

- Un esquema uniforme de denominación para localizar los recursos en el Web (por ejemplo, URIs).
- Los protocolos, para el acceso a los recursos del Web (por ejemplo, HTTP).
- *El hipertexto, para la navegación fácil entre los recursos (por ejemplo,* HTML).

La relación entre estos tres mecanismos se describe a continuación.

INTRODUCCIÓN A URIS

Cada recurso del Web –documentos HTML, imágenes, video clips, programas, etc.- tienen una dirección que puede interpretarse por un identificador universal de recursos (URI, por sus siglas en inglés).

Los URIs constan de tres partes:

- Un esquema de denominación de los mecanismos utilizados para el acceso de los recursos.
- El nombre de la máquina donde se localiza el recurso.
- El nombre del recurso identificado por una ruta de acceso.

Por ejemplo, consideremos el URI designado por la página W3C
http://www.w3.org/TR

Este URI se inertpretara de la siguiente manera: Hay un documento available vía protocolo http residiendo en la máquina www.w3.org, al cual se puede acceder mediante la ruta "/TR".
Otros esquemas pueden verse en los documentos de HTML incluidos como son: "mailto" para el correo electrónico y "ftp" para transferencia de datos vía FTP.
Otro ejemplo de URI se muestra a continuación. Este se refiere a un usuario del buzón de correo (mailbox):

... texto ...
Para cualquier comentario por favor escriba a
<A href = "mailto: joe@someplace.com" > Joe Cool </A>.

Nota: Algunas personas pueden estar familiarizasdas con el término "URL" y no con el término "URI". Los URLs forman un subconjunto de un esquema más genereal llamado URI.

IDENTIFICADORES DE FRAGMENTO

Algunos URIs refieren a la localización de un recurso. Esta clase de URI terminan con "#" seguido por un identificador (llamado identificador de fragmento). Por ejemplo, el siguiente es un http://somesite.com/html/top.html#section_2

URIS RELATIVOS

Un URI relativo no contiene ningún nombramiento del esquema de información. Su ruta generalmente se refiere a un recurso contenido en la misma máquina como un documento actual. Los URIs relativos deben contener los componentes de una ruta *relativa (por ejemplo ".." que significa el nivel superior en jerarquía* definido para la ruta dada), así como, identificadores de fragmento.

Los URIs relativos son interpretados por completo usando la base URI. Para ejemplificar como se leen los URIs relativos imaginemos que tenemos como base URI http://www.acme.com/support/intro.html. El URI relativo será incluido en la liga de hipertexto como sigue:

<A href = "suppliers.html"> Suppliers </A>

de aquí, se puede construir el URI completo así:
http://wwww.acme.com/support/suppliers.html, mientras que el URI relativo para una imagen sería:

<IMG src = "../icons/logo.gif" alt ="logo">

podemos expander el URI completo *http://www.acme.com/icons/logo.gif*.

En HTML, los URIs se usan para:

- Relacionar otro documento o recurso con la página actual (elementos A y LINK)
- Relacionar una hoja de estilo externa o script (elementos LINK y SCRIPT)
- Incluir una imagen, un objeto o un applet en una página (elementos IMG, OBJECT, APPLET e INPUT)
- Crear el mapeo de una imagen (elementos MAP y AREA)
- Submit a form (FORM)
- Crear un marco (frame) en un documento (elementos FRAME e IFRAME)
- Citar una referencia externa (elementos Q, BLOCKQUOTE, INS y DEL)
- Referir a convenciones metadata descritas en un documento (elemento HEAD)

1.2 ¿ QUÉ ES HTML?

Para publicar la información para la distribución global, se necesita un idioma universalmente entendido, un tipo de publicación en lengua madre que todas las computadoras puedan entender potencialmente. El idioma de la publicación usado por el World Wide Web es HTML (Lenguaje de Manipulación de Hipertexto). HTML les da los medios a los autores *para:*

- Desplegar en línea los documentos con títulos, texto, las tablas, las listas, las fotografías, etc.,
- Relacionar la información en línea a través de las ligas de hipertexto, al hacer clic en un botón.
- Diseñar las formas para dirigir las transacciones con los servicios remotos, para su uso en buscadores de información, hacer reservaciones, pedir productos, etc.,
- Incluir el spread-sheets, video clips, sound clips, y otras aplicaciones directamente en sus documentos.

¿ De dónde nace HTML ?

HTML es un lenguaje utilizado para crear hipertexto en el Web. No es un lenguaje para etiquetar páginas, pero generalmente es utilizado para dar una estructura a las partes de un documento. Basándonos en la identificación de las partes de un documento podemos tener páginas fáciles de interpretar.

HTML se define a partir de SGML (Lenguaje de Manipulación Generalizada Estándar), un estándar internacional (ISO 8879 :1986, Information Processing - Text and Office Systems (SGML) para manejar información de textos. SGML por sí solo es un meta-lenguaje, *es decir, un lenguaje para* definir lenguajes. La finalidad de SGML es ayudar a establecer un formato eficiente para crear información "en línea" (fácil de consultar por cualquier usuario de Internet), realizar búsquedas y recuperar de una forma independiente los detalles en apariencia de un documento. Cuando un programa de presentación se une con un documento HTML sin un estilo de información, hace que la vista física y la apariencia del documento lleguen a ser claras.

Ya que SGML es un meta-lenguaje, los desarrolladores deben especificar reglas para la estructura de un documento mediante la dèfinición del tipo de documento (DTD). Un DTD especifica exactamente qué tipo de documento se tiene definido en particular.

SGML es usado para definir HTML y todos sus niveles. HTML sigue la misma filosofía de *datos, estructura* e *independencia de forma* que SGML. Los *datos* en un documento consisten en el contenido del documento, ya sea texto o multimedia así como cualquier información referente al documento mismo como información administrativa o técnica. Las etiquetas en un documento SGML o HTML identifican la *estructura*: los encabezados, subencabezados, párrafos, listas o cualquier otro componente. Finalmente, el *formato* o la *forma* de un documento es la apariencia final de éste, antes de haber definido los datos la estructura o de haber especificado el formato en que debiera hacerse el documento. Cabe señalar que todas estas partes son separables: los datos del documento pueden ser creados sin que el autor se preocupe por la estructura, la estructura puede ser

agregada sin preocuparse por el formato y las especificaciones de la forma pueden crease siguiendo un estilo propio o particular de la compañía.

BREVE HISTORIA DE HTML

HTML fue desarrollado originalmente por Tim Berners - Lee mientras a CERN, y popularizado por el navegador del Mosaico desarrollado en NCSA. En el curso de los años noventa ha florecido con el crecimiento explosivo del Web. Durante este tiempo, HTML se ha extendido de varias maneras. El Web depende de los autores de páginas Web y vendedores que comparten las mismas convenciones para HTML. Esto ha motivado el trabajo de la juntura en las especificaciones para HTML.

HTML 2.0 (noviembre, 1995) se desarrolló bajo el amparo del Internet Engineering la Fuerza de la Tarea (IETF) para codificar la práctica común de 1994. HTML + (1993) y HTML 3.0 (1995) propuso versiones mucho más ricas de HTML. A pesar de nunca haber *llegado a un acuerdo general en las discusiones de las normas*, estos proyectos llevaron a la adopción de un rango de nuevas características. Los esfuerzos del HTML del Consorcio de World Wide Web el Grupo Activo para codificar la práctica común en 1996 dió como resultado HTML 3.2 (enero, 1997).

La mayoría de las personas están de acuerdo que los documentos en HTML deben trabajar bien por los diferentes navegadores y plataformas. Logrando la interoperabilidad bajan los costos y los proveedores quedan satisfechos pues desarrollan sólo una versión de un documento. Sin este esfuerzo, habría un riesgo mucho mayor de que el Web se desarrollara en un mundo propietario de formatos incompatibles, mientras que reduciría su potencial comercial para todos los participantes.

Cada versión de HTML ha intentado reflejar un mayor acuerdo general entre jugadores de la industria para que la inversión logre proveedores satisfechos que no gastarán y que sus documentos no serán ilegibles en un periodo breve de tiempo.
HTML se ha desarrollado con la visión de que todos los dispositivos deben poder usar la información sobre el Web: PCs con los despliegues de los gráficos de resolución variante y profundidades de color, los teléfonos celulares, los dispositivos de mano, dispositivos para reportar el rendimiento en entradas y salidas, computadoras con alto o bajo ancho *de banda, y así sucesivamente.*
HTML 4

HTML 4 extiende HTML con los mecanismos para las hojas de estilo, el scripting, los marcos (frames), empotrando los objetos, apoyo mejorado para el manejo en la dirección del texto de derecha a izquierda o mixto, tablas más ricas, y perfeccionamientos a las formas, ofreciendo una mejor accesibilidad para las personas discapacitadas.

INTERNACIONALIZACION

Esta versión de HTML se ha diseñado con la ayuda de *expertos en el campo* de internacionalización, para que puedan escribirse los documentos en cada idioma y pueden transportarse fácilmente alrededor del mundo. Esto ha sido cumplido de acuerdo a los tratos con la internacionalización de HTML.

Un paso importante ha sido la adopción del estándar ISO / IEC: 10646 como el conjunto de carácter de documento para HTML. Ésta, es la norma más inclusiva del mundo que trata con los problemas de la representación de carácteres internacionales, dirección del texto, puntuación, y otros problemas de idioma en el mundo.

HTML ofrece ahora mayor apoyo para diversos idiomas humanos dentro de un documento. Esto permite posicionamiento o indexación más eficaz de documentos para los artefactos de búsqueda, tipografía de alta calidad, mejor conversión de texto a reporte, buena incorporación de guiones, etc.

ACCESIBILIDAD

Cuando la comunidad de Web crece y sus miembros diversifican en sus habilidades y habilidades, es crucial que las tecnologías subyacentes sean apropiadas a sus necesidades específicas. HTML se ha diseñado para hacer páginas Web más accesible para aquellos con limitaciones físicas. Los desarrollos HTML 4 inspirados por las preocupaciones para la accesibilidad incluyen:

- Buena distinción entre la estructura del documento y su presentación, fomentando el uso de hojas de estilo así en lugar de los HTML presentación elementos y atributos.
- Mejores formas, que incluyen la adición de llaves de acceso, la habilidad de agrupar formas controladas y las opciones SELECT semánticamente, y las etiquetas activas.
- La habilidad para interpretar un texto incluido dentro de un objeto (con el elemento OBJECT).
- Un nuevo mecanismo, cliente – lateral, para leer el mapa de imagen (el elemento MAP) el cual permite a los autores integrar ligas de imagen y texto.
- Los requerimientos para alternar texto acompañado de imágenes se incluyen con el elemento IMG y los mapas de la imagen, con el elemento AREA.
- Soporte para los atributos title y lang en todos los elementos.
- Soporte para los elementos ABBR (abreviaturas) y ACRONYM (siglas).
- Un rango más amplio de medios de comunicación designado (tty, braille, etc.) para el uso con las hojas de estilo.
- Mejoras en las tablas, al incluir subtítulos, agrupamiento de columnas, y mecanismos para facilitar el non - el dando visual.
- Largas descripciones de las tablas, las imágenes, los marcos (frames), etc.

Los autores que diseñan las páginas conociendo los problemas de accesibilidad no sólo recibirán las bendiciones de la comunidad de accesibilidad, sino también beneficiará de otras maneras: el buen diseño de los documentos de HTML los cuales al distinguir estructura y presentación se adaptarán más fácilmente a las nuevas tecnologías.

TABLAS

Los autores tienen ahora mayor control sobre la estructura y el esquema (por ejemplo, la columna se pueden agrupar). La habilidad de los diseñadores para sugerir el ancho de

las columnas permite desplegar al usuario los datos de la tabla en foma incremental (como se reciben) en lugar de esperar el llenado de la tabla entera para darle un formato.

Nota. En el momento de escribir, algunas herramientas HTML autorizadas confían en las tablas extensivamente por estructurar que puede causar problemas de accesibilidad fácilmente.

DOCUMENTOS COMPUESTOS

HTML contiene ahora un mecanismo estándar para empotrar objetos de los medios de comunicación genéricos y aplicaciones en los documentos HTML. El elemento OBJECT (junto con sus antecesores más específicos IMG y APPLET) proporciona un mecanismo para incluir imágenes, video, sonido, expresiones matemáticas, aplicaciones especializadas, y otros objetos en un documento. También permite a los autores especificar una jerarquía de renderings alternado para agentes del usuario que no apoyan un dando específico.

HOJAS DE ESTILO

Las hojas de estilo simplifican la interpretación del código HTML y facilitan el empleo del lenguaje. Esto permite que tanto los autores como los usuarios tengan un control de la presentación de sus documentos. Ellos deciden, tipo de letra de la información, la alineación, los colores, etc.,

El estilo de la información puede especificarse individualmente o en grupos de elementos. Así mismo, puede especificarse mediante un documento de HTML o en hojas de estilo externas.

Los mecanismos para asociar una hoja de estilo con un documento son independientes del idioma de la hoja de estilo.

Antes de que existieran las hojas de estilo, los autores estaban limitados a los atributos que existían en el lenguaje que utilizaran. HTML 3.2 incluye numerosos atributos y elementos que permiten un control sobre la alineación, el tamaño de la letra, y el color del texto. Los autores también han aprovechado tablas e imágenes como un medio para poner un toque a las páginas. El tiempo relativamente largo que toma a los usuarios actualizar sus navegadores significa que estas nuevas características continuarán siendo usadas durante algún tiempo. Sin embargo, desde que las hojas de estilo ofrecen mecanismos más poderosos para hacer presentaciones, el Consorcio de World Wide Web escalonará muchos de los elementos y atributos usados en las presentaciones de HTML.

SCRIPTING

Es imposible lograr que un programa de computadora sea idóneo para toda la gente. Los creadores de software hacen lo que pueden para asegurarse de que sus programas puedan satisfacer la mayor parte de las necesidades del usuario, pero no pueden prever todo. Para hacer sus programas más flexibles, muchos de ellos incorporan la capacidad

de extender o modificar el comportamiento de sus programas mediante un script (una secuencia de comandos).

Un script no es más que una secuencia de instrucciones de programa. El programa procesa dichas instrucciones una por una y realiza lo que el script le indica. Esto es exactamente lo mismo que "programar", sólo que los scripts suelen tener reglas más sencillas y requerir menos tiempo de aprendizaje. Como ejemplos de programas que permiten escribir scripts podemos citar a Dbase, Paradox y Microsoft Access (aunque hay muchos más). Perl y REXX son lenguajes independientes para scripts.

A través de scripts (escritos), los autores pueden crear páginas Web dinámicas (por ejemplo, "formas inteligentes" que sugieren alguna información mientras los usuarios proporcionan sus datos) y usar HTML como medio para construir aplicaciones interconectadas a una red de computadoras.

Los mecanismos proporcionados para incluir scripts en un documento HTML son independientes del idioma del scripting.

IMPRESIÓN

A veces, los autores querrán facilitar al usuario la impresión de más de un solo documento. Cuando los documentos forman parte de un trabajo más grande, pueden describirse las relaciones entre ellos usando el elemento LINK de HTML o usando el Esquema de Descripción de Recursos de W3C (RDF).

SEPARACIÓN ENTRE ESTRUCTURA Y PRESENTACIÓN

HTML tiene sus raíces en SGML que siempre ha sido un lenguage para código especificado en forma estructural. Cuando HTML madura, cada vez más de sus elementos y atributos de presentación son reemplazados por otros mecanismos, en particular las hojas de estilo. La experiencia ha mostrado que separando la estructura de un documento de sus aspectos del presentación se reduce el costo de servicio para una gama amplia de plataformas, medios de comunicación, etc., y facilita las revisiones del documento.

ACCESIBILIDAD UNIVERSAL AL WEB

Hacer el Web más accesible a todos, notablemente aquéllos con las invalideces, los autores deben considerar cómo sus documentos pueden darse en una variedad de plataformas: el discurso - los navegadores basado, braille - los lectores, etc., Nosotros no recomendamos que los autores limiten su creatividad, sólo que ellos consideran los renderings alternados en su plan. HTML ofrece varios mecanismos con este fin (por ejemplo, los alt atribuyen, los accesskey atribuyen, etc.)

Además, los autores deben tener presente un lejano - fuera del público con las configuraciones de la computadora diferentes. Para que los documentos ser interpretado correctamente, los autores deben incluir en su información de los documentos sobre el

idioma natural y dirección del texto, cómo el documento se pone en código, y otros problemas relacionaron a la internacionalización.

CONFORMANCE : REQUERIMIENTOS Y RECOMENDACIONES
En esta sección se conocerán las especificaciones de HTML 4, comenzaremos con algunas definiciones de algunos términos que son importantes para el lenguaje.

DEFINICIONES

- Documento HTML
  Un documento de HTML es un documento SGML que se encuentra los constreñimiento de esta especificación.

- Autor
  Un autor es una persona o programador que escribe o genera los documentos de HTML. Una herramienta de authoring (*authoring tool*) es un caso especial de un autor, a saber, el cual es un programa que genera HTML.
- Usuario
  Un usuario es una persona que actúa recíprocamente con un agente de usuario para ver, oir, o por otra parte usar un documento de HTML dado.

- Agente de usuario HTML
  Un agente de usuario HTML es cualquier dispositivo que interpreta los documentos de HTML. Los agentes de usuario incluyen los navegadores visuales (el texto - sólo o con gráficos), los navegadores no visuales (audio, Braille), robots de búsqueda, apoderados (proxies), etc.

Un agente de usuario conformado para HTML 4 es aquel que observa las condiciones obligatorias ("debe") de sus especificaciones, incluye los puntos siguientes:

- Un agente de usuario debe evitar imponer de forma arbitraria los límites de longitud del valor de un atributo literal.
- Un agente de usuario debe asegurar que el rendimiente no sea afectado por la presencia o ausencia de etiquetas de entrada y salida cuando el DTD de HTML indica que éstas son optativas.
- Por razones de compatibilidad hacia versiones anteriores, se recomienda utilizar las herramientas para la interpretación de HTML 4 con ello seguirán interpretándose los atributos de HTML 3.2 y HTML 2.0.

CONDICIONES DEL ERROR

Esta especificación no define cómo manejan los agentes de usuario las condiciones de error general, pero si incluye cómo los agentes de usuario se comportan cuando encuentran elementos, atributos, valores del atributo, o entidades no especificadas en el documento.
Sin embargo, para conocer cuando se comete un error y lo que éste ocasiona, se recomienda consultar notas sobre documentos inválido.

DESAPROBADO (DEPRECATED)

Un elemento o atributo desaprobado es aquel que ha sido anticuado para las más nuevas estructuras. Los elementos desaprobados son definidos en el manual de referencia en las situaciones apropiadas, pero son claramente marcados como desaprobados. Estos elementos podrían volverse obsoletos en futuras versiones de HTML.
Los agentes de usuario deben continuar apoyando a los elementos desaprobados por las razones de compatibilidad con versiones anteriores de HTML.
Las definiciones de elementos y atributos indican claramente cuáles son desaprobados.

Esta especificación incluye ejemplos que ilustran cómo evitar usar los elementos desaprobados. En la mayoría de los casos éstos dependen del agente de usuario que se ocupa para las hojas de estilo. En general, los autores acostumbran las hojas de estilo pues logran estilizar su página y estructurar los efectos en lugar de los atributos de presentación de HTML. Estos últimos (los atributos de presentación) han sido desaprobados cuando existen las hojas de estilo como altenativa. Un elemento o atributo obsoleto es aquel para el cual no existe ninguna garantía de apoyo por un agente de usuario.

SGML

HTML 4 es una aplicación de SGML conforme con a la norma internacional ISO 8879 para el lenguaje de manipulación generalizado estándar, SGML (*Standard Generalized Markup Language*).

Tipo satisfecho de text / html (The text/html content type)

Los documentos de HTML son enviados a tráves del Internet como una sucesión de bytes acompañados de información en código. La estructura de la transmisión, denominada entidad del mensaje. Una entidad del mensaje con un tipo satisfecho de " texto / html" representa un documento de HTML.

El tipo satisfecho para los documentos de HTML se define como:

Content type name:
text
Content subtype name:
html
Required parameters:
none
Optional parameters:
charset
Encoding considerations:
any encoding is allowed

El parámetro opcional "charset" se refiere a los caracteres de código que se utilizan para representar un documento HTML como una secuencia de bytes. Sin embargo, debe tenerse siempre presente que se trata de un parámetro opcional.

ANEXO 2.

2.1 INTRODUCCIÓN A JAVASCRIPT

Uno de los aspectos más interesantes de la Word Wide Web es su capacidad de ofrecer contenidos interactivos a muchas personas. La Web es el mayor conjunto de información asequible para un ser humano desde el origen de los tiempos

Por primera vez, las ubicaciones de la Web tienen capacidad de interacción con sus usuarios. Aplicaciones muy sofisticadas, como programas de dibujo, hojas de cálculo, juegos y complejas herramientas de cálculo matemático, pueden ejecutarse ahora en la ventana del navegador, entre las paginas HTML, con sólo disponer de un navegador preparado, sin necesidad de contar con un hardware o un software especializados.

Dos son las soluciones que han surgido en la Web para dotarla de contenidos interactivos: el sencillo y manejable lenguaje de hipertexto realzado (HTML) y el sofisticado y potente lenguaje de programación Java. Con estas dos herramientas, los usuarios pueden crear contenidos atrayentes en su aspecto visual y unirlos sin solución de continuidad con las aplicaciones interactivas que brinda Java. Sin embargo, lo que parece echarse de menos es un sistema que aproxime ambas tecnologías.

APARICIÓN DE JAVASCRIPT

Netscape vio la necesidad de tender un puente entre las dos tecnologías. Para ello se puso a trabajar en un nuevo lenguaje de órdenes que pudiera ocupar el vacío entre HTML y Java y que, a la vez, fuera lo suficientemente potente como para enlazar ambas tecnologías.

Cuando Netscape creó ese nuevo lenguaje lo llamó LiveScript y lo introdujo de inmediato en su navegador más popular, en un esfuerzo por conseguir que la comunidad de Internet lo adoptara cuanto antes. Poco después Netscape y Sun se unieron para ayudar a que LiveScript consiguiera mayor difusión y para establecerlo como el lenguaje estándar de Internet para la escritura de órdenes basada en la Web. Dado que la sintaxis de LiveScript era muy similar a la de Java, ambas compañías decidieron rebautizar su nuevo producto para que pudiera ser identificado mejor, y lo llamaron JavaScript.

JavaScript fue creado como un lenguaje de órdenes fácil de utilizar, abierto, de plataforma cruzada, capaz de enlazar objetos y recursos propios tanto de HTML como de Java. Si las miniaplicaciones de Java son fruto principalmente del trabajo de los programadores, JavaScript fue concebido para que pudieran usarlo los creadores de las páginas HTML para controlar dinámicamente la interacción y el comportamiento de sus páginas. JavaScript tiene la particularidad de haber sido diseñado para integrarse con HTML.

Una de las ventajas principales que ofrece es su capacidad para reducir el trafico de red, permitiendo la realización local de las tareas más simples. En otras palabras: en lugar de encomendar esas tareas al servidor y hacer que éste pase los resultados al navegador, el

navegador puede realizar algunas de esas tareas en el ámbito local, con lo que el usuario, además, obtiene sus respuestas en un tiempo más corto.

JavaScript es aún un lenguaje en evaluación, algunas de sus características y comandos pueden variar en el futuro.

2.2 ¿QUÉ ES JAVASCRIPT?

JavaScript se basa en el potente lenguaje Java en su uso y sintaxis, pero es interpretado, no compilado. Esto quiere decir que el código de aplicación de JavaScript es transferido como texto al navegador junto con el texto HTML. Dicho código se ejecuta dentro del navegador, con lo que se capacita al usuario para desarrollar aplicaciones sencillas que puedan interactuar con el usuario y ayudarle.

Con JavaScript es posible responder a determinadas acciones del usuario, tales como pulsaciones del ratón, movimientos del ratón sobre un vinculo, y crear una entrada de datos (input). También se pueden crear paginas dinámicas que cambian a solicitud del usuario, e incluso ejecutar sonidos o miniaplicaciones cuando un usuario entra o abandona la página. Este tipo de acomodar a nivel del cliente permite una enorme capacidad de interactividad con los usuarios de sus páginas Web.

El lenguaje JavaScript se parece a Java, pero es más sencillo y más fácil de aprender. Una aplicación en JavaScript puede tener tan sólo una línea u ocupar varias páginas. Su complejidad depende del grado en que sus instrucciones hayan de interaccionar con la página en que se encuentra dicha aplicación. Para la mayoría de los creadores de paginas Web, una de las primeras utilidades de JavaScript se da en la validación de formularios. Se llama así a la capacidad de un formulario en HTML para comprobar los datos introducidos por un usuario antes de presentarlos, hecho que mejora notablemente las prestaciones en su servidor y que a la vez disminuye la espera del usuario.

COMPARACIÓN ENTRE JAVA Y JAVASCRIPT

JavaScript y Java se apoyan en los mismos principios básicos, ambos son lenguajes de programación con comandos y sintaxis similares, los dos están orientados al objeto, y los dos son abiertos y de plataforma cruzada.

JavaScript es muy parecido a Java en su sintaxis, pero ahí acaban muchas de sus similitudes. JavaScript es un lenguaje interpretado, lo que significa que el código JavaScript se ejecuta directamente en el navegador y no requiere compilación anterior a la ejecución, además JavaScript tiene un conjunto de tipos de datos más reducido.

JavaScript no tiene clases ni herencia, sin embargo tiene un conjunto de objetos incorporados que requieren un esfuerzo mínimo en su creación, que pueden ser ampliados de forma que se acomoden a las necesidades del programador.

Por ser interpretado, todas las referencias que se hacen a los objetos se comprueban en el momento de la ejecución, mientras que Java requiere que todas las referencias existan en tiempo de compilación. Además, en JavaScript las variables no necesitan declaración.

Otra diferencia importante es que el código JavaScript se encuentra incluido dentro de la página HTML, mientras que las miniaplicaciones Java son referenciadas mediante la etiqueta APPLET, pero el código compilado se encuentra en un archivo separado.

SEGURIDAD EN JAVASCRIPT

JavaScript fue pensado para que fuera un lenguaje seguro. No permite el uso de punteros, que es la causa más frecuente de violaciones de la seguridad.

Además al ser un lenguaje interpretado, no hay problemas de tiempo de compilación y asignación de memoria, que es otra fuente potencial de violaciones de la seguridad.

Por último, para minimizar la eficiencia de programas malintencionadamente creados, JavaScript no permite incorporar códigos grabados en disco.

JavaScript tiene el inconveniente de que el código fuente viaja dentro de la página HTML por lo que cualquier usuario puede leer y modificar dicho código; así, la única forma de proteger ese código fuente es insertando una nota de copyright en el código fuente de la aplicación.

INTEGRACIÓN DE JAVASCRIPT Y JAVA

Una de las características más importantes de JavaScript es su capacidad para interaccionar directamente con las aplicaciones Java. En el código Java, pueden encontrarse útiles propiedades que permiten que una aplicación en JavaScript alcance o fije tales propiedades y modifique así el estado o las prestaciones de la aplicación Java. También puede solicitar o modificar las prestaciones de los plug-ins.

Estas funcionalidades no están incluidas en la versión 2.0 de Netscape, pero se espera que aparezcan en la versión 2.1 de Netscape Navigator.

Con la integración de todas estas tecnologías "HTML, JavaScript y Java", el desarrollo y la implementación de paginas Web altamente interactivas se convierte en una tarea mucho más fácil y rápida.

REQUISITOS DE JAVASCRIPT

Los recursos necesarios para poder utilizar y programar en JavaScript son pocos. Este es uno sus aspectos más atractivos, ya que cualquiera puede emplearlo directamente para crear sus propias aplicaciones.

Los requisitos de hardware, necesarios para ejecutar JavaScript son los mismos que se necesitan para ejecutar Netscape Navigator. Es decir, que si se puede utilizar Navigator, se tiene todo el hardware y software necesario para ejecutar aplicaciones JavaScript.

Para crear páginas Web que contengan funciones JavaScript es posible emplear cualquier editor de texto. Y como JavaScript no necesita compilación, en cuanto el programa esté archivado podrá ser probado directamente en el navegador.

CREACIÓN DE PROGRAMAS CON JAVASCRIPT

Para que el navegador reconozca que un archivo contiene funciones escritas en JavaScript, dichas funciones se han de colocar entre dos etiquetas especiales : script ... /script

Entre estas dos etiquetas se colocaran las funciones que luego el navegador reconocerá cuando cargue la página. Estas etiquetas tienen atributos como LANGUAJE="JavaScript".

Ejecutar código JavaScript es tan fácil como visualizar una página HTML, una vez cargada la página en el navegador, observará los resultados de las funciones JavaScript de forma inmediata. Las aplicaciones JavaScript pueden comprobarse localmente, sin necesidad de estar conectado a ninguna red, con lo que los tiempos de carga son muy cortos.