Universidad **N**acional de **I**ngeniería

Recinto Universitario Simón Bolívar

Facultad de Electrotecnia y Computación

Ingeniería de Software. Una Perspectiva Didáctica

"Desarrollo de una herramienta interactiva para el apoyo de la enseñanza - aprendizaje de la asignatura ingeniería de software"

Trabajo Monográfico

Presentado por:

Br. Jorge Luis Hernández Morrison
Br. Alejandro Antonio Rojas Mena

Tutor:

Lic. Magda Auxiliadora Luna

Para optar al titulo de:

Ingeniero en Computación.

Managua Nicaragua
Julio, 2008

DEDICATORIA

Este trabajo constituye el producto de incansables días de sacrificios y entrega, pero, felices y con la satisfacción por haber llegado a la cima de la montaña a la que la mayoría le teme, dedicamos nuestro trabajo a:

- *DIOS Nuestro Padre: por habernos dado la vida, por permitirnos llegar hasta este momento gracias a su abundante amor y bondad, por lograr el éxito al haber coronado nuestra carrera universitaria.*

- *Nuestros padres y familiares que con sus abnegados esfuerzos y sacrificios nos han dado lo que tanto soñamos, ser unos profesionales, y por brindarnos mucho apoyo durante la realización de este trabajo.*

- *A cada uno de los docentes de esta Alma Mater que con sus consejos en el aula de clases nos guiaron hasta este punto.*

- *Nuestros amigos que directa o indirectamente influyeron en el desarrollo de este trabajo.*

AGRADECIMIENTOS

Sin duda alguna, esta página es la que más trabajo llevo redactar. Incluso se pensó en no ponerla; debido a que si se nombrara a cada uno de las personas que nos brindaron su apoyo para la realización de este trabajo, tendríamos que llenar otras cien páginas más.

Sin embargo, y a riesgo de cometer una injusticia con todos esos héroes anónimos, es nuestro deseo brindar especiales agradecimientos a:

- *Lic. Magda Auxiliadora Luna M. quien nos motivó y orientó el rumbo a seguir para el desarrollo de esta tesis monográfica. Muchas gracias por inyectarnos en las venas el "suero" de la Ingeniería de Software.*

- *Al Ing. Roberto David Cordero Moraga, quien fue, es y será inagotable fuente de motivación. Un modelo a seguir. Muchas gracias "Cordero" por acompañarnos todos estos días.*

Muchas Gracias a Todos.

RESUMEN DEL TEMA.

El trabajo que se presenta a continuación ha estado motivado por varios aspectos: una mejoría en la concepción pedagógico – metodológica de la impartición de la ingeniería de software (ISW) en la Universidad Nacional de Ingeniería (UNI) así como la introducción de un medio interactivo / digital como medio de transmisión de dicha asignatura dentro del estudio de la Ingeniería en Computación como especialidad técnica. Sus pilares pedagógicos descansan en dos de las categorías de la didáctica pedagógica: los medios de enseñanza y aprendizaje; así como la de métodos de enseñanza.

La UNI es una institución estatal de educación superior, autónoma, con excelencia académica, conciencia social, ética y humanismo; dedicada a formar profesionales en el campo de las ingenierías y arquitectura, contribuyendo así a la transformación y desarrollo tecnológico sustentable de Nicaragua.

Entre otras, ésta alma mater ofrece la carrera de Ingeniería en Computación, la que se desarrolla en tres amplias ramas de aplicación: hardware, software y sistemas de control. Dentro de la rama orientada al software, el pensum académico de la carrera contiene una asignatura denominada Ingeniería de Software la que tiene como objetivo principal dotar a los estudiantes de conocimientos para el establecimiento y uso de principios de ingeniería para la construcción de software confiables, de calidad, eficientes, funcionales y en el tiempo establecido para ello.

Esta cátedra se encuentra dividida en tres módulos: Ingeniería de Software I impartida en el octavo semestre, Ingeniería de Software II impartida en el noveno semestre e Ingeniería de Software III impartida en el décimo semestre.

Uno de los objetivos que persigue la cátedra es hacer que el estudiante sea capaz de reconocer y comparar la importancia y utilidad de los paradigmas de desarrollo de sistemas, y en qué tipo de circunstancias deben de ser utilizados de acuerdo a las características de los proyectos.

Puesto que son muy pocos los autores que escriben acerca de más de un paradigma de desarrollo las dificultades que se le presentan a los estudiantes y docentes (y lectores en general) en el transcurso de la clase son muchas. El contenido temático que abarca la clase se encuentra disperso entre diferentes textos o sitios Web, de tal manera que los unos complementan a los otros. Esto conlleva a que los estudiantes y docentes se enfrenten con serios problemas a la hora de recolectar información para el aprendizaje o enseñanza.

Ante este problema, surge la idea de construir una Herramienta Digital de Ingeniería de Software que sirva de apoyo en la enseñanza / aprendizaje de la misma. Esta herramienta abarcará el plan temático de la asignatura y estará estructurada de forma tal que se abordará de manera progresiva cada una de

las etapas de desarrollo de los proyectos informáticos d desarrollo de sistemas informáticos basados en computadoras. A demás se combinará los conocimientos teóricos con los prácticos, lo que dará como resultado una mejor asimilación del contenido de la clase por parte de los estudiantes.

INDICE DE CONTENIDO

ANEXOS

Primera Parte
Introducción

1.- INTRODUCCIÓN.

Actualmente se habla de ingeniería de software, pero la manera en la que desarrollamos software actualmente ¿se puede denominar, con propiedad, ingeniería? La palabra ingeniería tiene una connotación de prestigio que provoca que muchas disciplinas traten de autocalificarse como tal. En general, se concibe la ingeniería como una aplicación práctica y eficiente de los conocimientos científicos. Desde este punto de vista, el sector del software se podría encontrar en una fase de producción comercial en algunas organizaciones [Shaw,1994] pero seguramente resulta demasiado optimista hablar de la aplicación generalizada de una auténtica ingeniería.

Los cambios en el hardware en los últimos veinte años han sido notables, y podría parecer que los cambios en el software también han sido significativos. En honor a la verdad, nuestra capacidad para construir sistemas grandes y complejos ha mejorado drásticamente. En la construcción de sistemas de software se mezclan muchas tecnologías – J2EE, .NET, SAP (por mencionar algunas)– que permiten que aplicaciones grandes basadas en web sean desarrolladas mucho más rápido que en el pasado.

Sin embargo, a pesar de los cambios que ha habido en las dos últimas décadas, cuando miramos más allá de las tecnologías, hacia los procesos fundamentales de la ingeniería de software, éstos se han mantenido iguales.

Los métodos y técnicas actuales de ingeniería de software han hecho que la construcción de sistemas grandes y complejos sea mejor. A pesar de ello es común encontrar proyectos que se retrasan, que sobrepasan el presupuesto o que se entregan sin satisfacer las necesidades del cliente. El SIGA[1] es un proyecto desarrollado para proveer a las oficinas de compras y suministro de la corte suprema de justicia de Nicaragua de un sistema de software para mejorar la gestión de estas oficinas. La firma desarrolladora estimó el costo del sistema en $81,500 (Dólares) y fue planificado para ser entregado en Marzo del 2007. Para Abril del 2007 el costo ya había pasado los $90,000 y aún no estaba totalmente operativo. Hay, por lo tanto, una necesidad imperiosa de mejorar la educación en ingeniería de software.

En la Universidad Nacional de Ingeniería (UNI) hoy día, dada la cantidad cada vez más creciente de estudiantes en la carrera de ingeniería en computación y la necesidad de gran cantidad de profesores, por ser una carrera de relativa novedad – no más de 2 décadas – se hace necesario la utilización de otras alternativas pedagógicas con gran basamento didáctico para la enseñanza y la transmisión del conocimiento a tan creciente masa de estudiantes. Es

[1] SIGA Sistema de Integración de Gestión Administrativa

precisamente la Internet, como medio de transmisión masiva del conocimiento uno de esas posibles soluciones a dicha problemática, aunque no la única.

Dado lo anterior y con el fin de apoyar la enseñanza / aprendizaje de la ingeniería de software, en la carrera de ingeniería en computación, proponemos el desarrollo de una herramienta digital interactiva que sirva como material bibliográfico para docentes y estudiantes, y así ayudar a la formación profesional de éstos últimos.

2.- ANTECEDENTES.

En la carrera de ingeniería en computación ofertada en la UNI, la clase de ingeniería del software se imparte para que los estudiantes se preparen y se preocupen por la producción sistemática, la calidad y mantenimiento de los productos y proyectos informáticos que se desarrollarán y modificarán en un tiempo y presupuesto definido.

Por otra parte se pretende que los estudiantes queden aptos para la realización del análisis, especificación, diseño, desarrollo, pruebas y mantenimiento de los proyecto informáticos que se ponen en marcha en las distintas instituciones.

Pese a la claridad con la que están definidos los objetivos de la asignatura, los estudiantes tienden a confundirse al no contar con una bibliografía que les muestre los diferentes paradigmas de desarrollo de sistemas informáticos, sus etapas de desarrollo y su utilización de acuerdo a las características particulares de los proyectos.

Orientados por los docentes los estudiantes desarrollan sus proyecto de curso respaldados en la variada bibliografía alrededor de éste tema y de archivos procedentes vía Internet. Este método (si bien, sirve para documentarse) genera problemas ya que existen varios y muy distintos enfoques los que están en dependencia del punto de vista de los autores.

Dado esto, es comprensible que los libros que abordan la temática de desarrollo de proyectos informáticos difieran entre si. En algunos se tratan a profundidad etapas de desarrollo que en otros libros simplemente se omiten. La estructura que se le da al desarrollo de un proyecto contrasta de un libro a otro. Mientras unos son detallistas otros presentan la información de forma general, por lo que los estudiantes tienen que recurrir a la lectura de distintos libros e información de la web para conocer las muchas y variadas actividades que se relacionan con el desarrollo sistemas informáticos.

3.- JUSTIFICACIÓN.

Históricamente han surgido varios enfoques alrededor de la ISW los que buscan abordar de manera sistemática, la planificación, análisis, diseño, implementación y mantenimiento de los proyectos de desarrollo de software, sean estos de gran escala o pequeñas aplicaciones, software a la medida o productos de propósito general. Cada uno de estos enfoques tiene su raíz en las preconcepciones dominantes en su época y, sobre todo, en la búsqueda incesante de mejoras a los enfoques precedentes.

Esta variedad de enfoques en torno a la ingeniería del software ha generado serios problemas a los estudiantes de ingeniería en computación de la UNI (y lectores en general), ya que la perspectiva que se les da a cada una de las etapas del desarrollo de los proyectos informáticos varía de autor en autor; y como parte de las estrategias metodológicas y para alcanzar los objetivos definidos de la asignatura se recomienda que el docente asigne investigaciones sobre algunos temas del curso, las cuales, deberán ser presentadas por los grupos a los que se les asignó. Este hecho genera confusiones, debido a que el alumnado extrae la información de distintas unidades bibliográficas y sitios Web para realizar sus exposiciones y, al final, la concatenación de la información recabada se vuelve difícil debido a la misma diversidad de enfoques.

Por otra parte, el docente orienta el desarrollo de un proyecto de curso desde el inicio del semestre, en el cual, los estudiantes organizados en grupos de dos o tres integrantes presentan los avances sistemáticos de sus proyectos al resto del alumnado, de forma tal que pongan en práctica los conocimientos adquiridos en la clase. Cabe señalar que el avance del proyecto de curso se produce de forma sistemática y paralela al estudio de cada una de las etapas de desarrollo de los proyectos informáticos, conforme la programación elaborada por el docente. El desarrollo de este proyecto de curso se edifica sobre los conocimientos teóricos adquiridos en las clases conferencias impartidas en el aula de clase.

El poco desarrollado hábito de lectura en los estudiantes más la presión por cumplir con la entrega del proyecto en la fecha estipulada (en ésta y las otras clases) provoca que el proyecto se vea estancado en cierto momento. Los estudiantes “no quieren estudiar” ingeniería del software, máxime cuando se tiene que leer de dos a tres libros para poder comprender de forma clara los diferentes paradigmas de desarrollo de proyectos informáticos.

Este trabajo se realizará con el fin de desarrollar una herramienta digital de información que sirva como base teórica a alumnos y docentes para la asignatura ISW. La idea surge a partir de la poca bibliografía disponible en las bibliotecas y las limitaciones de acceso, principalmente por razones de tipo económico.

Con el desarrollo de esta herramienta se le brindará al alumnado la posibilidad de introducirse en los conceptos básicos y en los aspectos fundamentales de la ingeniería del software, describir las técnicas y herramientas actuales utilizadas para la construcción de proyectos informáticos.

Se estudiará las actividades asociadas con el desarrollo de proyectos informáticos. El estudiante dispondrá de información oportuna a cerca de cómo identificar los requerimientos de un sistema, incluyendo los métodos para recolectar requerimientos relacionados con los datos y como utilizar la información generada por esos datos, además de como documentar los detalles del sistema con ayuda de diversas técnicas.

La aplicación contendrá un sinnúmero de herramientas que muestran un panorama real del desarrollo de sistemas informáticos. Los conceptos y teorías sobre los que descansan el análisis y diseño de proyectos informáticos estarán entrelazados a lo largo de toda la aplicación. Se hará hincapié en los aspectos prácticos del desarrollo de los mismos, como en las decisiones que el analista debe de enfrentar de manera cotidiana cuando trabaja en un proyecto.

Con el desarrollo de este proyecto, se pretende integrar los aspectos relevantes del ciclo de vida de un proyecto informático, los enfoques estructurados al desarrollo de software que incluyen modelos de sistemas, notaciones, reglas, consejos del diseño y consejos del proceso de desarrollo de sistemas de información basados en computadoras. A demás se muestran las tendencias actuales que sigue la ingeniería de software.

Esta herramienta será de mucha utilidad, ya que permitirá a los lectores realizar consultas precisas del contenido pragmático de ISW. Por otra parte concientes del compromiso que tenemos con nuestra universidad, consideramos que es una forma de retribuirle algo a esta alma mater.

4.- OBJETIVOS.

Objetivo General.

- Desarrollar una herramienta digital que sirva como una guía de apoyo a la enseñanza / aprendizaje de la asignatura Ingeniería de Software en la Carrera de Ingeniería en Computación que oferta la UNI.

Objetivos Específicos.

- Proveer una herramienta que brinde información oportuna acerca de los conceptos fundamentales de Ingeniería de software.
- Instruir al usuario sobre las diferentes técnicas y herramientas disponibles que apoyan en el desarrollo del Software.
- Dotar al lector de habilidades necesarias para la evaluación de los sistemas informáticos mediante las técnicas de la auditoria informática o de software.
- Dar a conocer las tendencias actuales de la ingeniería de software.

Segunda Parte
Marco Teórico.

CAPITULO 1.
Ingeniería de software. Proceso y Métodos.

Actualmente todos los países dependen de complejos sistemas informáticos. Infraestructuras nacionales y utilidades dependen de sistemas informáticos, y la mayor parte de los productos eléctricos incluyen una computadora y software de control. Prácticamente la fabricación industrial y distribución está completamente informatizada, como el sistema financiero. Por lo tanto, producir software costeable se vuelve una necesidad de vital importancia para el funcionamiento de la economía nacional e internacional.

La ingeniería de software es una disciplina de la ingeniería cuya meta es el desarrollo costeable de sistemas de software. Este es abstracto e intangible. No esta restringido por materiales, o gobernado por leyes físicas o por procesos de manufactura. De alguna forma, esto simplifica la ingeniería de software ya que no existen limitaciones físicas del potencial del software. Sin embargo, esta falta de restricciones naturales significa que el software puede llegar a ser extremadamente complejo y, por lo tanto muy difícil de entender.

Cuando todo funciona bien las computadoras son de gran ayuda pero cuando no, el resultado puede ser nefasto, algo que se ha visto a lo largo de los años en múltiples ocasiones.

La noción de ingeniería de software fue propuesta inicialmente en 1968 en una conferencia para discutir lo que en ese entonces se llamó la “crisis del software”. Esta crisis del software fue el resultado de la introducción de las nuevas computadoras hardware basadas en circuitos integrados. Su poder hizo que las aplicaciones hasta ese entonces irrealizables fueran una propuesta factible. El software resultante fue de órdenes de magnitud más grande y más complejo que los sistemas de software previos.

La experiencia previa en la construcción de estos sistemas mostró que un enfoque informal para el desarrollo del software no era muy bueno. El desarrollo de software estaba en crisis. Los costos del hardware se tambaleaban mientras que los del software se incrementaban con rapidez. Se necesitaban nuevas técnicas y métodos para controlar la complejidad inherente a los sistemas grandes.

Estas técnicas han llegado a ser parte de la ingeniería de software y son ampliamente utilizadas. Sin embargo, cuanto más crezca nuestra capacidad para producir software, también lo hará la complejidad de los sistemas de software solicitados. Las nuevas tecnologías resultantes de la convergencia de las computadoras y de los sistemas de comunicación y complejas interfaces gráficas de usuarios impusieron nuevas demandas a los ingenieros de software.

Se puede afirmar que hemos hecho enormes progresos desde 1968 y que el desarrollo de esta ingeniería ha mejorado considerablemente nuestro software. Comprendemos mucho mejor las actividades involucradas en el desarrollo del mismo. Se han desarrollado métodos efectivos de especificación, diseño e implementación del software.
Ahora sabemos que no hay un enfoque "ideal" a la ingeniería de software. La amplia diversidad de diferentes tipos de sistemas y organizaciones que utilizan estos sistemas significa que necesitamos una diversidad de enfoques para el desarrollo de software. Sin embargo, las nociones fundamentales de procesos y la organización del sistema son la base de todas estas técnicas, y éstas son la esencia de la ingeniería de software.

1.1.-¿Qué es la ingeniería de software?

"La ingeniería de software es una disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después de que se utiliza."[1]

En esta definición existen dos frases claves:

1.- Disciplina de la Ingeniería. Los ingenieros hacen que las cosas funcionen. Aplican teorías, métodos y herramientas donde sean convenientes, pero las utilizan de manera selectiva y siempre tratando de descubrir soluciones a los problemas, aun cuando no existan teorías y métodos aplicables para resolverlos. Los ingenieros también saben que deben de trabajar con limitaciones de índoles económicas y organizacionales, por lo que las soluciones que encuentren deberán estar regidas por estas limitaciones.

2.- Todos los aspectos de la producción de software. La ingeniería de software, no sólo comprende los procesos técnicos del desarrollo de software, si no también con actividades tales como la gestión de proyectos software y el desarrollo de herramientas, métodos, teorías de apoyo a la producción del mismo.

1.2.- ¿Qué son los métodos de la ingeniería de software?

Un método de ingeniería de software es un enfoque estructurado para el desarrollo de software cuyo propósito es facilitar la producción de software de alta calidad de una forma costeable.

No existe un método ideal, y métodos diferentes Estos tienen diferentes áreas donde son aplicables. Por ejemplo los métodos orientados a objetos a menudo

[1] Ingeniería del Software, 7ma. Edición, Ian Sommerville.

son apropiados para sistemas interactivos, pero no para sistemas con requerimientos rigurosos de tiempo real.

1.3.- ¿Qué es un proceso de Software?

Un proceso de software es un conjunto de actividades y resultados asociados que producen un producto de software. Estas actividades son desarrolladas por los ingenieros de software".[2]
En este sentido, podemos nombrar cuatro actividades fundamentales de procesos que son comunes para todos los procesos del software. Estas actividades son:

1.- Especificación de software, donde los clientes e ingenieros definen el software que se desea producir y las restricciones sobre su operación.

2.- Desarrollo de software, donde el software se diseña y programa.

3.- Validación del software, donde el software es validado para asegurar que es lo que el cliente requiere.

4.- Evolución del software, donde el software se modifica para adaptarlo a los cambios requeridos por el cliente y el mercado.

Diferentes tipos de sistemas necesitan diferentes procesos de desarrollo. Por lo tanto estas actividades genéricas pueden organizarse de diferentes formas y describirse en diferentes niveles de detalle para diferentes tipos de software.

1.4.-Modelos del proceso de software.

¿Qué es un modelo de proceso de Software?

Un modelo de proceso de software es una descripción simplificada de un proceso de software que presenta una visión de ese proceso. Estos modelos pueden incluir actividades que son partes de los procesos y productos de software y el papel de las personas involucradas en la ingeniería de software. Un ejemplo de estos tipos de modelo que se pueden producir es:

1.- Un modelo de flujo de trabajo. El que muestra la secuencia de actividades en el proceso junto con sus entradas, salidas y dependencias. Las actividades en este modelo representan acciones humanas.

La mayor parte de los modelos de procesos de software se basan en uno de los tres modelos generales o paradigmas de desarrollo de software:

[2] Ingeniería del Software, 7ma. Edición, Ian Sommerville.

1.- El enfoque en cascada. Considera las actividades anteriores y las representa con fases de procesos separados, tales como la especificación de requerimientos, el diseño de software, la implementación, las pruebas, etc. Después de cada etapa queda definida "se firma" y el desarrollo continua con la siguiente etapa.

2.- Desarrollo iterativo. Este enfoque entrelaza las actividades de especificación, desarrollo y validación. Un sistema inicial se desarrolla rápidamente a partir de especificaciones muy abstractas. Este se refina basándose en las peticiones del cliente para producir un sistema que satisfaga las necesidades de dicho cliente. El sistema puede entonces ser entregado. De forma alternativa, se puede reimplementar utilizando un enfoque más estructurado para producir un sistema más sólido y mantenible.

3.- Ingeniería de Software basa en componentes. En esta técnica se supone que las partes del sistema existen. El proceso de desarrollo del sistema se enfoca en la integración de estas partes más que desarrollarlas desde el principio.

Estos tres modelos de procesos genéricos se utilizan ampliamente en la práctica actual de la ingeniería de software. No se excluyen mutuamente y a menudo se utilizan juntos, especialmente para el desarrollo de sistemas grandes. De hecho, el Proceso unificado de Rational combina elementos de todos estos modelos. Los subsistemas dentro de un sistema más grande pueden ser desarrollados usando enfoques diferentes. En este punto queremos dejar claro que si bien es necesario estudiar estos modelos de manera separada, debe de entenderse que, en la práctica, a menudo se combinan.

1.4.1.- El modelo en cascada.

El primer modelo de proceso de desarrollo de software que se publicó se derivó de procesos de ingeniería de sistemas más generales. (Figura 1.1). Debido a la cascada de una fase a otra, dicho modelo se conoce como modelo en cascada o como ciclo de vida del software. Las principales etapas de este modelo se transforman en actividades fundamentales de desarrollo:

1.- Análisis y definición de requerimientos. Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Entonces, se definen en detalle y sirven como una especificación del sistema.

2.- Diseño del sistema y del software. El proceso de diseño del sistema divide los requerimientos en sistemas hardware o software. Establece una

arquitectura completa del sistema. El diseño del software identifica y describe las abstracciones fundamentales del sistema software y sus relaciones.

3.- Implementación y pruebas de unidades. Durante esta etapa, el diseño del software se lleva a cabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla su especificación.

4.- Integración y prueba del sistema. Los programas o las unidades individuales de programa se integran y prueban como un sistema completo para aseguran que cumplan con los requerimientos del software. Después de las pruebas, el sistema software se entrega al cliente.

5.- Funcionamiento y mantenimiento. Por lo general (pero no necesariamente), ésta es la fase más larga del ciclo de vida. El sistema se instala y se pone en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubren nuevos requerimientos.

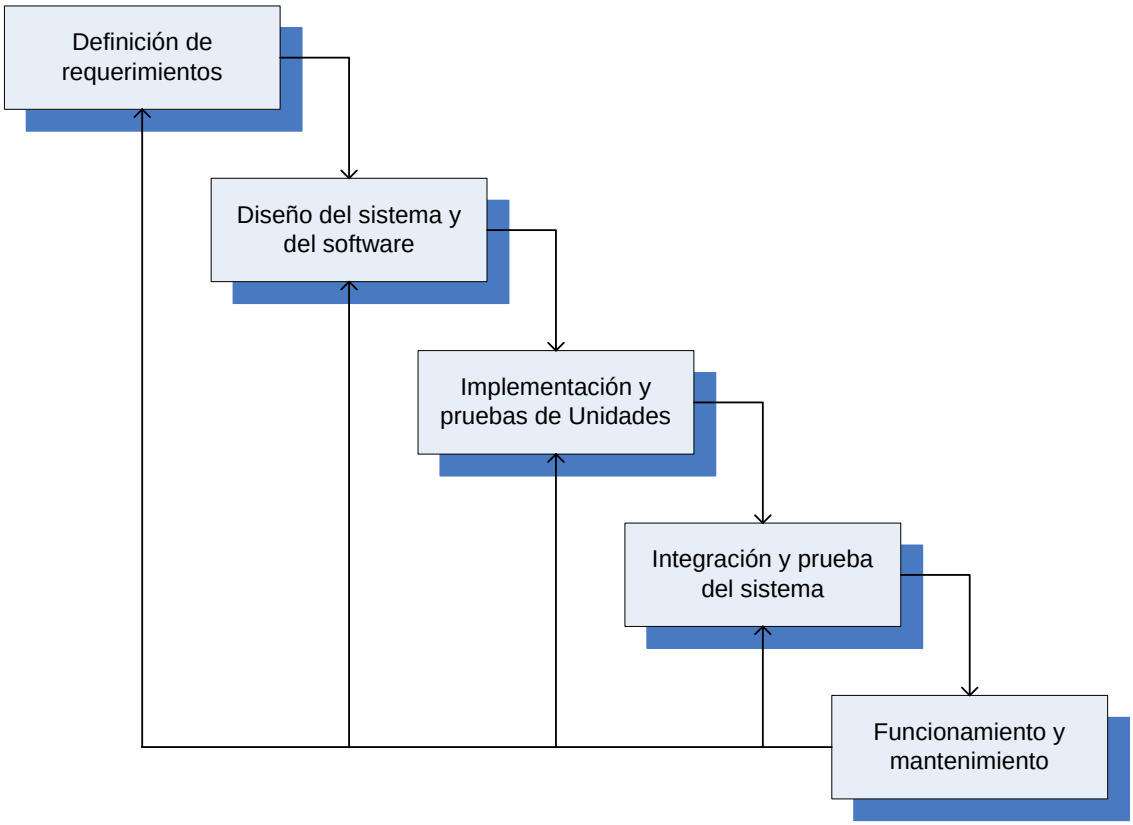


Figura 1.1.- Modelo Cascada o Ciclo de vida del software.

En principio, el resultado de cada fase es uno o más documentos aprobados (firmados). La siguiente fase no debe de empezar hasta que la fase previa haya terminado. En la práctica estas etapas se superponen y proporcionan información a las otras. Durante el diseño se identifican los problemas con los requerimientos; durante el diseño del código se encuentran problemas, y así sucesivamente. El proceso de software no es un modelo lineal simple, sino que implica una serie de iteraciones de las actividades de desarrollo.

La ventaja del modelo en cascada es que la documentación se produce en cada fase y que ésta cuadra con otros modelos de procesos de ingeniería. Su principal problema es su inflexibilidad al dividir el proyecto en distintas etapas. Se deben de hacer compromisos en las etapas iniciales, lo que hace difícil de responder a los cambios en los requerimientos del cliente.

Por lo tanto, el modelo en cascada sólo se debe de utilizar cuando los requerimientos se comprendan bien y el hecho de que cambien de forma radical durante el desarrollo del sistema sea improbable.

1.4.2.- Desarrollo evolutivo.

El desarrollo evolutivo se basa en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado (Figura 1.2). Las actividades de especificación, desarrollo y validación se entrelazan en lugar de separarse, con una rápida retroalimentación entre éstas.

Existen dos tipos de desarrollo evolutivo:

1.- Desarrollo exploratorio, donde el objetivo del proceso es trabajar con el cliente para explorar sus requerimientos y entregar un sistema final. El desarrollo empieza con las partes del sistema que se comprenden mejor. El sistema evoluciona agregando nuevos atributos propuestos por el cliente.

2.- Prototipos desechables, donde el objetivo del proceso de desarrollo evolutivo es comprender los requerimientos del cliente y entonces desarrollar una versión mejorada de los requerimientos para el sistema. El prototipo se centra en experimentar con los requerimientos del cliente que no se comprenden del todo.

Desde una perspectiva de ingeniería y de gestión, el enfoque evolutivo tiene dos problemas:

1.- El proceso no es visible. Los administradores tienen que hacer entregas regulares para medir el progreso. Si los sistemas se desarrollan

rápidamente, no es rentable producir documenten cada versión del sistema.

2.- A menudo los sistemas tienen una estructura deficiente. Los cambios continuos tienden a corromper la estructura del software. Incorporar cambios en él, se convierte cada vez más en una tarea difícil y costosa.

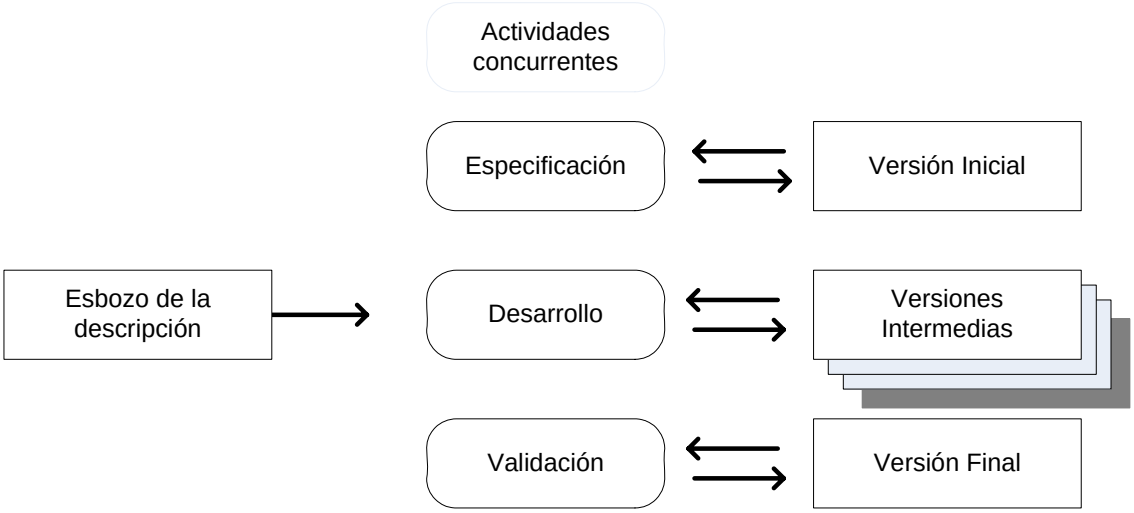


Figura 1.2.- Desarrollo evolutivo.

Para sistemas pequeños y de tamaño medio, el enfoque evolutivo de desarrollo es el mejor.

Para sistemas grandes, se recomienda un proceso mixto que incorpore las mejores características del modelo en cascada y del desarrollo evolutivo.

1.4.3.- Ingeniería de software basada en componentes.

Podríamos afirmar, con gran certeza, que en la mayoría de los proyectos de software existe algo de reutilización de software. Por lo general, esto sucede informalmente cuando las personas que trabajan en los proyectos conocen diseños o códigos similares al requerido. Los buscan, lo modifican según lo creen necesario y lo incorporan en el sistema. En el enfoque evolutivo, tratado anteriormente, la reutilización es a menudo indispensable para el desarrollo rápido de sistemas.

Esta reutilización informal es independiente del proceso de desarrollo que se utilice. Sin embargo, en los últimos años, ha surgido un enfoque de desarrollo de software denominado ingeniería de software basada en componentes (CBSE, por sus siglas en ingles) que se basa en la reutilización, el cual se está utilizando de manera amplia.

Este enfoque basado en reutilización se compone de una gran base de componentes software reutilizables y de algunos marcos de trabajo de integración para éstos. Algunas veces estos componentes son sistemas por sí mismos (COTS, por sus siglas en ingles o sistemas comerciales) que se pueden utilizar para proporcionar funcionalidad especifica, como dar formato al texto o efectuar cálculos numéricos. En la figura 1.3 se muestra el modelo del proceso genérico para la CBSE.

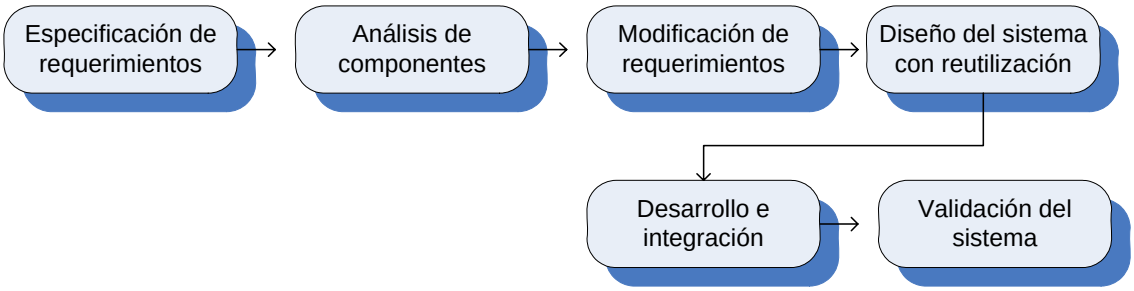


Figura 1.3.- Ingeniería de software basada en componentes.

Aunque la etapa de especificación de requerimientos y la de validación son comparables con otros procesos, las etapas intermedias en el proceso orientado a la reutilización son diferentes. Estas etapas son:

1.- Análisis de componentes. Dada la especificación de requerimientos, se buscan los componentes para implementar esta especificación. Por lo general, no existe una concordancia exacta y los componentes que se utilizan sólo proporcionan parte de la funcionalidad requerida.

2.- Modificación de requerimientos. En esta etapa, los requerimientos se analizan utilizando información acerca los componentes disponibles. Si las modificaciones no son posibles, la actividad de análisis de componentes se puede llevar a acabo nuevamente para buscar soluciones alternativas.

3.- Diseño del sistema con reutilización. En esta fase se diseña o reutiliza un marco de trabajo para el sistema. Los diseñadores tienen en cuenta los componentes que se reutilizan y organizan el marco de trabajo para que los satisfaga. Si los componentes reutilizables no están disponibles, se puede tener que diseñar nuevo software.

4.- Desarrollo e integración. Para crear el sistema, el software que no se puede adquirir externamente se desarrolla, y los componentes o sistemas COTS se integran, En este modelo, la integración de sistemas es parte del proceso de desarrollo, más que una actividad separada.

La ingeniería de software basada en componentes tiene la ventaja obvia de reducir la cantidad de software a desarrollarse y así reduce los costos y los

riesgos. Por lo general, también permite una entrega más rápida del sistema de software. Sin embargo, los compromisos en los requerimientos son inevitables, y esto puede dar lugar a un sistema que no cumpla las necesidades reales de los usuarios. Más aún: si las nuevas versiones de los componentes reutilizables no están bajo el control de la organización que los utiliza, se pierde parte del control sobre la evolución del sistema.

1.5.- Actividades del proceso.

Las cuatro actividades básicas del proceso de especificación, desarrollo, validación y evolución se organizan de forma distinta en diferentes procesos del desarrollo. En el enfoque en cascada, están organizadas en secuencia, mientras que en el desarrollo evolutivo se entrelazan. Como se llevan a cabo estas actividades depende del tipo de software, de las personas y de la estructura organizacional implicada. No hay una forma correcta o incorrecta de organizar estas actividades. A continuación mostramos una manera de como se pueden organizar estas actividades.

1.5.1.- Especificación del software.

La especificación del software o ingeniería de requerimientos es el proceso de comprensión y definición de ¿Qué? Servicios se requieren del sistema y de identificación de las restricciones de funcionamiento y de desarrollo del mismo. La ingeniería de requerimientos es una etapa particularmente crítica en el proceso del software ya que los errores en esta etapa originan, inevitablemente, problemas en las etapas posteriores en el diseño e implementación del sistema.

En la figura 1.4 se muestra el proceso de la ingeniería de requerimientos. Este conduce a la producción de un documento de requerimientos, que será la especificación del sistema. Normalmente en este documento los requerimientos se representan en dos niveles de detalle. Los usuarios finales y los clientes necesitan una declaración de alto nivel de los requerimientos, mientras que los desarrolladores del sistema necesitan una especificación más detallada de éste.

Existen cuatro fases principales en el proceso de la ingeniería de requerimientos:

1.- Estudio de viabilidad. Se estima si las necesidades del usuario se pueden satisfacer con las tecnologías actuales de software y hardware. El estudio analiza si el sistema propuesto será rentable desde un punto de vista de negocios y si se puede desarrollar dentro de las restricciones de presupuesto existentes. Este estudio puede ser relativamente económico y rápido de elaborar. El resultado debe de informar si se va a continuar con un análisis más detallado.

2.- Obtención y análisis de requerimientos. Es el proceso de obtener los requerimientos del sistema mediante la observación de los sistemas existentes, discusiones con los usuarios potenciales y proveedores, el análisis de tareas, etcétera. Esto puede implicar el desarrollo de uno o más modelos y prototipos del sistema que ayudan al analista a comprender el sistema a especificar.

3.- Especificación de requerimientos. Es la actividad de traducir la información recopilada durante la actividad de análisis en un documento que define un conjunto de requerimientos. En este documento se pueden incluir dos tipos de requerimientos: los requerimientos del usuario, que son declaraciones abstractas del cliente y usuario final del sistema, y los requerimientos del sistema, que son una descripción más detallada de la funcionalidad a proporcionar.

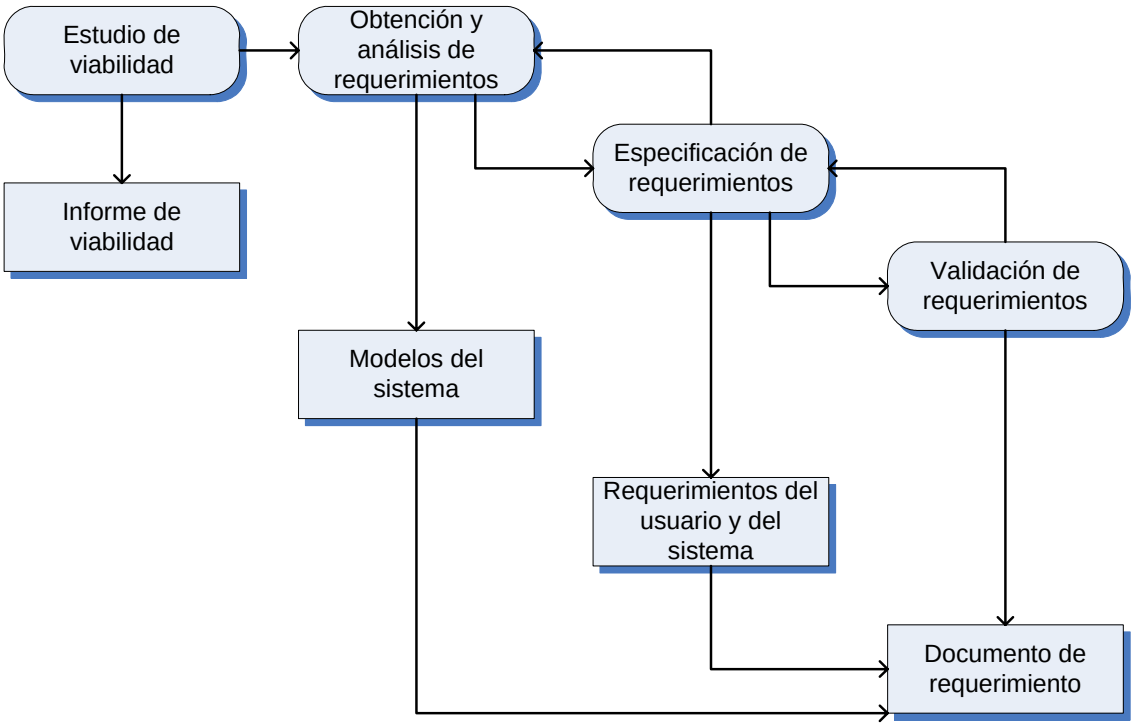


Figura 1.4.- El proceso de la ingeniería de los requerimientos.

4.- Validación de requerimientos. Esta actividad comprueba la veracidad, consistencia y completitud de los requerimientos. Durante este proceso, inevitablemente se descubren errores en el documento de requerimientos. Se debe de modificar entonces para corregir estos problemas.

Pro supuesto, las actividades en el proceso de requerimientos no se llevan a cabo de forma estrictamente secuencial. El análisis de requerimientos continúa durante la definición y especificación, y a lo largo del proceso surgen nuevos

requerimientos. Por lo tanto, las actividades de análisis, definición y especificación se entrelazan.

1.5.2.- Diseño e implementación del software.

La etapa de implementación del desarrollo de software es el proceso de convertir una especificación del sistema en un sistema ejecutable. Siempre implica los procesos de diseño y programación de software, pero, si se utiliza un enfoque evolutivo de desarrollo, también puede implicar un refinamiento de la especificación del software.

Un diseño de software es una descripción de la estructura de del software que se va a implementar, los datos que son parte del sistema, las interfaces entre los componentes del sistema y, algunas veces, los algoritmos utilizados. Los diseñadores no llegan inmediatamente a un diseño detallado, si no que lo desarrollan de manera iterativa a través de diversas versiones. El proceso de diseño conlleva agregar formalidad y detalle durante el desarrollo del diseño, y regresar a los diseños anteriores para corregirlos.

El proceso de diseño puede implicar el desarrollo de varios modelos del sistema con diferentes niveles de abstracción. Mientras se descompone un diseño, se descubren errores y omisiones de las etapas previas. Esta retroalimentación permite mejorar los modelos de diseños previos. La figura 1.5 es un modelo de este proceso que muestra las descripciones de diseño que pueden producirse en varias etapas del diseño. Este diagrama sugiere que las etapas son secuenciales. En realidad, las actividades del proceso de diseño se entrelazan. La retroalimentación entre las etapas y la consecuente repetición del trabajo es inevitable en todos los procesos de diseño.

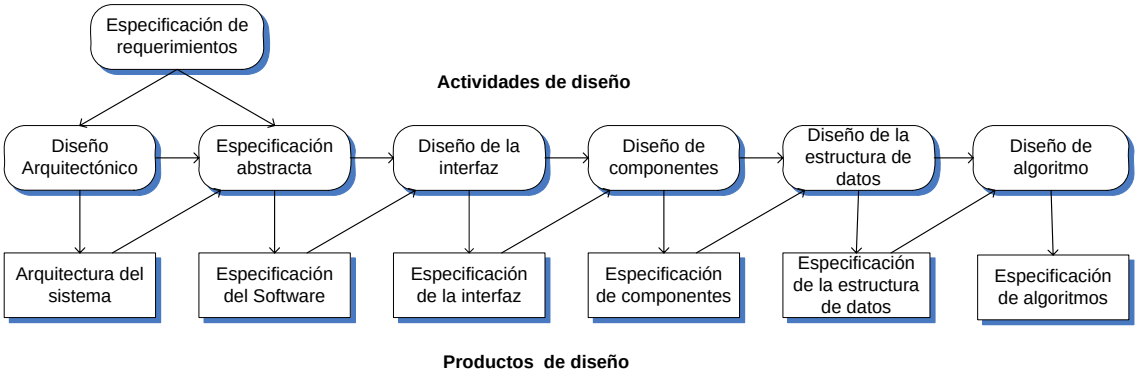


Figura 1.5.- Un modelo general del proceso de diseño.

Una especificación para la siguiente etapa es la salida de cada actividad de diseño. Esta especificación puede ser abstracta y formal, realizada para clarificar los requerimientos, o puede ser una especificación para determinar qué parte del sistema se va a construir. Durante todo el proceso de diseño, se destalla cada vez más esta especificación. El resultado final del proceso son especificaciones precisas de los algoritmos y estructuras de datos a implementarse.

Las actividades específicas del proceso de diseño son:

1.- Diseño arquitectónico. Los subsistemas que forman el sistema y sus relaciones se identifican y documentan.

2.- Especificación Abstracta. Para cada subsistema se produce una especificación abstracta de sus servicios y las restricciones bajo las cuales debe de funcionar.

3.- Diseño de la interfaz. Para cada subsistema se diseña y se documenta su interfaz con otros subsistemas. Esta especificación debe de ser inequívoca ya que permite que el subsistema se utilice sin conocimientos de su funcionamiento.

4.- Diseño de componentes. Se asignan servicios a los componentes y se diseñan sus interfaces.

5.- Diseño de la estructura de datos. Se diseña en detalle y especifica la estructura de datos utilizada en la implementación del sistema.

6.- Diseño de algoritmos. Se diseñan en detalle y especifican los algoritmos utilizados para proporcionar los servicios.

1.5.3.- Validación del software.

La validación del software, o de manera general, la verificación y validación se utilizan para verificar que el sistema se ajusta a su especificación y que cumple las expectativas del usuario que lo solicitó. Implica procesos de comprobación, como las inspecciones y revisiones, en cada etapa del proceso de software desde la especificación de los requerimientos hasta el desarrollo del programa. Sin embargo, la mayoría de los costos de validación aparecen después de la implementación, cuando se prueba el funcionamiento del sistema.

A excepción de los programas pequeños, los sistemas no se deben de probar como una simple unidad compacta. En la figura 1.6 muestra un proceso de pruebas de tres etapas en el cual se prueban los componentes del sistema, la integración del sistema y, finalmente, el sistema con los datos del cliente. En el mejor de los casos, los defectos se descubren en las etapas iniciales del

proceso y los problemas de interfaz, cuando el sistema se integra. Sin embargo, cuando se descubren defectos, el programa debe depurarse y esto puede implicar la repetición de otras etapas del proceso de pruebas. Los errores en los componentes pueden descubrirse durante las pruebas del sistema. Por lo tanto, el proceso es iterativo y se retroalimenta tanto de las últimas etapas como de la primera parte del proceso.

Las etapas del proceso de pruebas son:

1.- Prueba de componentes (o unidades). Se prueban los componentes individuales para asegurarse que funcionan correctamente. Cada uno se prueba de manera independiente, sin los otros componentes del sistema. Los componentes pueden ser entidades simples como funciones o clases de objetos, o pueden ser agrupaciones coherentes de estas entidades.

2.- Prueba del sistema. Los componentes se integran para conformar el sistema. Este proceso comprende encontrar errores que son resultados de interacciones no previstas entre los componentes y su interfaz. También comprende validar que el sistema cumpla sus requerimientos funcionales y no funcionales y probar las propiedades emergentes del sistema. Para sistemas grandes, esto puede ser un proceso gradual en el cual los componentes se integran para formar subsistemas que son probados individualmente antes de que ellos mismos se integren para formar el sistema final.

3.- Prueba de aceptación. Es la etapa final en el proceso de pruebas antes de que se acepte que el sistema se ponga en funcionamiento. Este se prueba con los datos proporcionados por cliente más que con datos de prueba simulados. Debido a la diferencia existente entre los datos reales y los de prueba, la prueba de aceptación puede revelar errores y omisiones en la definición de requerimientos del sistema. También puede revelar problemas en los requerimientos donde los recursos del sistema no cumplen las necesidades del usuario o donde el desempeño del sistema es inaceptable.

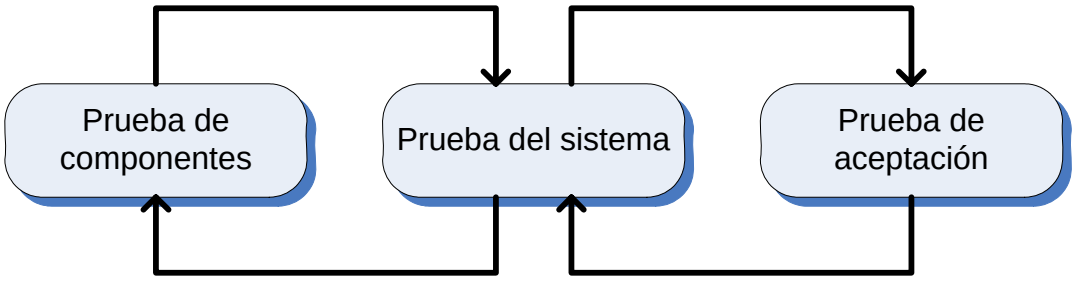


Figura 1.6.- El proceso de pruebas.

Normalmente, el desarrollo de componentes y las pruebas se entrelazan. Los programadores definen sus propios datos de prueba y de forma incremental prueban el código que se va desarrollando. Este es un enfoque económicamente razonable puesto que el programador es el que mejor conoce los componentes y es, por lo tanto, la mejor persona para generar los casos de prueba.

La prueba de aceptación algunas veces se denomina prueba alfa. Los sistemas personalizados se desarrollan para un único cliente. El proceso de prueba alfa continúa hasta que el desarrollador del sistema y el cliente acuerdan que el sistema que se va a entregar es una implementación aceptable de los requerimientos del sistema.

1.5.4.- Evolución del software.

La flexibilidad de los sistemas software es una de las principales razones por la que más y más software se incorpora a los sistemas grandes y complejos. Una vez que se decide adquirir hardware, es muy costoso hacer cambios en su diseño. Sin embargo, se puede hacer cambios al software en cualquier momento durante o después del desarrollo del sistema. Aún cambios importantes son todavía mucho más económicos que los correspondientes de los sistemas de hardware.

Históricamente, siempre ha existido una separación entre el proceso de desarrollo y el proceso de evolución del software (mantenimiento del software). La gente considera el desarrollo de software como una actividad creativa en la cual un sistema software se desarrolla desde un concepto inicial hasta que se pone en funcionamiento. Sin embargo, a veces consideran el mantenimiento del software como algo aburrido y sin interés. Aunque los costos de mantenimiento son con frecuencia varias veces los costos iniciales de desarrollo, el proceso de mantenimiento se considera a veces menos problemático que el desarrollo de software original.

Esta distinción entre el desarrollo y el mantenimiento es cada vez más irrelevante. Hoy en día, pocos sistemas software son completamente nuevos, lo que implica que tienen más sentido ver el desarrollo y el mantenimiento como actividades continuas, Más que dos procesos separados, es más realista considerar a la ingeniería de software como un proceso evolutivo (figura 1.7) en el cual el software se cambia continuamente durante su período de vida como respuesta a los requerimientos cambiantes y necesidades del usuario.

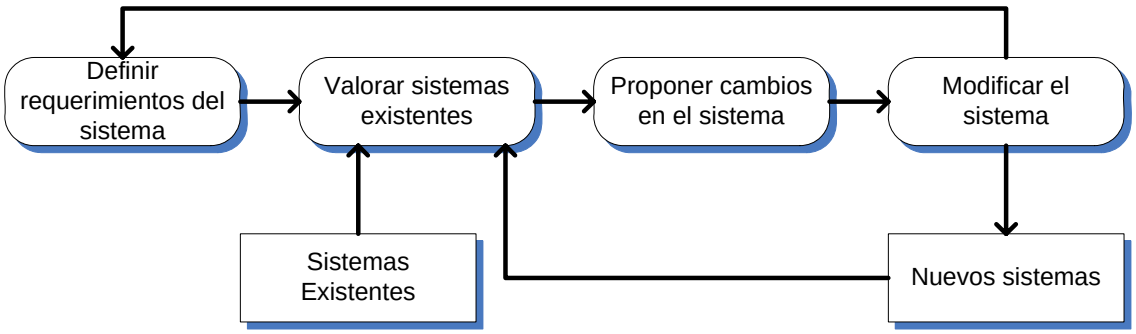


Figura 1.7.- Evolución del sistema.

CAPITULO 2.
La enseñanza de la Ingeniería de Software en la UNI desde un enfoque didáctico.

La educación constituye uno de los objetivos estratégicos de la sociedad. Con los niveles de desarrollo alcanzados hoy en día por la sociedad se revaloriza y considera con especial atención el papel de la educación en el progreso de la misma.

En la sociedad nicaragüense actual, desde finales del siglo pasado, se han estado generando cambios radicales en los conceptos de la educación. La universidad, como nivel superior del sistema de enseñanza nicaragüense, no ha estado ajena a dichas transformaciones. Se aplican hoy conceptos pedagógicos novedosos y es menester en todas las instituciones de este nivel, perfeccionar los procesos docentes y lograr mayor eficiencia en la formación de pregrado y postgrado.

El tercer milenio –desde su inicio- se ha caracterizado por la dinámica de los cambios en las más diversas direcciones. El profesional del siglo XXI vive lo que se ha denominado cultura del aprendizaje, ya no son los "titanes del renacimiento" capaces de incursionar en varias ramas del saber. Lo titánico es estar preparados para "navegar" en un mundo de altos niveles de información y conocimientos, orientarse en ellos con pensamiento propio y capacidad de asimilación e innovación, ser partícipes de una formación permanente, asumir una posición digna en el contexto social contradictorio y complejo que existe.

Los servicios que la Internet puede prestar a la educación quedaron evidenciados desde muy temprano en la historia de este medio.

Hoy día con el desarrollo tecnológico mundial alcanzado en los medios electrónicos para la difusión y recepción de la Internet y su "abaratamiento creciente" se hace muy económico el financiamiento y mantenimiento de la enseñanza por este medio, ventaja que muy eficazmente ha sido utilizada por muchas instituciones de educación superior y a la cual la universidad nacional de ingeniería no está ajena, y no lo puede estar ninguno de sus profesores.

Debe quedar completamente claro, para evitar confusiones, las diferencias entre la Internet educativa y la Internet didáctica; La primera se refiere al uso del Internet vinculado a la acción formadora sobre el individuo en el contexto social que no está enmarcada necesariamente en un plan de estudio, un grado o un currículo cualquiera. La Internet didáctica se asocia al marco de la escuela, se integra a los programas y planes de estudio de un nivel o grado, con carácter oficial, complemento a la labor del profesor o consolidación; su empleo está limitado en tiempo y espacio, y su repertorio está controlado; el público que la

recibe está cautivo, definido en edades, intereses, desarrollo psíquico y cognoscitivo.
Pedagógicamente debe tenerse claro que el uso de la Internet para los fines docentes debe ser completamente didáctica y apoyar constantemente el desarrollo de las habilidades potenciales que los egresados necesitan y la adquisición con mayor rapidez del conocimiento y su persistencia en el tiempo; por estar ligada su enseñanza a la utilización de un medio que armoniza eficientemente los sentidos, que mejor conceptualizan y formalizan el conocimiento en el ser humano: el visual y el auditivo.

Hay dos formas de asumir la Internet didáctica: la creativa, que requiere elaborar material propio para la escuela, universidad o centro de investigación, y la no creativa, que se limita al uso de materiales existentes, enlatados, realizados profesionalmente por instituciones especializadas.

Es objeto de estudio y campo de acción de los pedagogos el logro de las capacidades del estudiante a través del perfeccionamiento continuo de los procesos de enseñanza – aprendizaje en nuestras casas de altos estudios y la formación posgraduada.

2.1.- Situación problemática.

Los preparación de los profesionales egresados de la UNI debe ser de excelencia y estar vinculada a altas capacidades de autoaprendizaje, autoformación y con una sólida formación humanística que permita comprender el entorno social en el que él se desarrolla. Es en estos últimos señalamientos, donde las necesidades actuales de la educación, ganan significado para nosotros y forman parte de nuestro pensamiento pedagógico actual.

Hoy en la UNI, teniendo en cuenta lo planteado hasta el momento y sumándole las características particulares de la materia en cuestión, se hace necesario un análisis constante de la utilización de la Internet y los estudios pedagógicos para este uso. Es por este motivo que se hacen necesarios estudios metodológicos y pedagógicos de ¿cómo impartir la Ingeniería de Software en la UNI? de forma tal que los futuros egresados dominen y puedan aplicar con eficiencia en el ejercicio de su profesión las teorías y técnicas aprendidas en el aula y los laboratorios universitarios, mejorando considerablemente nuestra forma de hacer software.

¿Cómo lograr un mejor desarrollo del proceso de enseñanza – aprendizaje de la Ingeniería de Software en la UNI, apoyado en el uso de la Internet didáctica, que responda a los fines y objetivos de la docencia en esta rama de la informática previstos en el Plan de Estudios actual?

2.2.- Actualidad y necesidad del trabajo.

El reto de los educadores en la actualidad, es lograr en los educandos que estos sean capaces de: *aprender a conocer, aprender a hacer, aprender a ser, aprender a vivir juntos, aprender a vivir con los demás* y *aprender a desaprender*. Definitivamente es menester de los profesionales de la educación, lograr el desarrollo de dichas habilidades en nuestros centros universitarios. (Ver Anexo 1).

La Universidad Nacional de Ingeniería (UNI), debe ser una universidad de constantes cambios pedagógicos; en la cual la investigación y la experimentación de nuevos conceptos, técnicas, métodos, procedimientos y contenidos sean lo cotidiano. Hoy la enseñanza de la Ingeniería de Software –como ya ha sido dicho- encierra una gran cantidad de incógnitas en el mundo entero, de ahí la necesidad de profundizar en este tema y buscar nuestras propias soluciones a esas preguntas, aún no contestadas del todo.

El proceso de enseñanza – aprendizaje ha sido históricamente caracterizado de formas diferentes, que van desde su identificación como proceso de enseñanza, con un marcado acento en el papel central del maestro como transmisor de conocimientos, hasta las concepciones más actuales en las que se concibe el proceso de enseñanza – aprendizaje como un todo integrado, en el cual se pone de relieve el papel protagónico del alumno.

En cualquier nivel de la enseñanza, y máxime en el universitario, es necesario que el estudiante desarrolle las habilidades y se apropie eficientemente de los conocimientos que necesitará para el ejercicio de su profesión, logrando los enlaces con el conocimiento de los niveles previos, así como resultado también de su interacción con el entorno social universitario y la sociedad toda.

La clase tiene sus códigos y sus lenguajes; la Internet también tiene los suyos, que son muy diferentes. Para introducir la Internet a la clase hay que hacer una adaptación del lenguaje del medio, en que se pueda sopesar a partes iguales el “arte de la navegación Web” y el rigor pedagógico. Cuando la balanza se va del lado del arte, podemos tener un material verdaderamente bello, obras acabadas y perfectas, pero donde ningún estudiante aprenderá absolutamente nada; si la balanza se va al lado pedagógico, obtendremos un “ladrillo” que aburrirá a los estudiantes antes de que terminen d recorrer el sitio web.

En el actual contexto de nuestro país, se hace eminente una modificación de nuestros métodos de enseñanza, que convoquen en mayor eficiencia del proceso docente – educativo, mayor rapidez en la adquisición de los conocimientos y habilidades por los estudiantes y que esto se trasmita a una mayor calidad de vida y de desarrollo tecnológico de nuestra sociedad en esta era del conocimiento. Es en este contexto donde hoy la UNI forma ingenieros en una rama que tiene un gran impacto en el logro de estos objetivos y apuesta al

uso de la ya bien establecida Internet como medio por excelencia para la transmisión del conocimiento.

2.3.- Métodos de Enseñanza.

2.3.1.- Métodos y técnicas de enseñanza.

Constituyen recursos necesarios de la enseñanza; son los vehículos de realización ordenada, metódica y adecuada de la misma. Los métodos y técnicas tienen por objeto hacer más eficiente la dirección del aprendizaje. Gracias a ellos, pueden ser elaborados los conocimientos, adquiridas las habilidades e incorporados con menor esfuerzo los ideales y actitudes que la escuela pretende proporcionar a sus alumnos.

1.- Método. Es el planeamiento general de La acción de acuerdo con un criterio determinado y teniendo en vista determinadas metas.

2.- Técnica de enseñanza. Tiene un significado que se refiere a la manera de utilizar los recursos didácticos para hacer efectivo el aprendizaje del educando. Conviene al modo de actuar, objetivamente, para alcanzar una meta.

3.- Método de enseñanza. Es el conjunto de momentos y técnicas lógicamente coordinados para dirigir el aprendizaje del alumno hacia determinados objetivos. El método es quien da sentido de unidad a todos los pasos de la enseñanza y del aprendizaje y como principal ni en lo que atañe a la presentación de la materia y a la elaboración de la misma.

4.- Método didáctico. Es el conjunto lógico y unitario de los procedimientos didácticos que tienden a dirigir el aprendizaje, incluyendo en él desde la presentación y elaboración de la materia hasta la verificación y competente rectificación del aprendizaje.

Los métodos, de un modo general y según la naturaleza de los fines que procuran alcanzar, pueden ser agrupados en tres tipos:

1.- Métodos de Investigación: Son métodos que buscan acrecentar o profundizar nuestros conocimientos.

2.- Métodos de Organización: Trabajan sobre hechos conocidos y procuran ordenar y disciplinar esfuerzos para que hay eficiencia en lo que se desea realizar.

3.- Métodos de Transmisión: Destinados a transmitir conocimientos, actitudes o ideales también reciben el nombre de métodos de enseñanza, son los intermediarios entre el profesor y el alumno en la acción educativa que se ejerce sobre éste último.

2.3.2.- Clasificación General de los Métodos de Enseñanza.

Veremos ahora la clasificación general de los métodos de enseñanza, tomando en consideración una serie de aspectos, algunos de los cuales están implícitos en la propia organización de la entidad educadora.

Estos aspectos realzan las posiciones del educador, del educando, de la disciplina y de la organización educacional en el proceso educativo. Los aspectos tenidos en cuenta son:

1.- En cuanto a la forma de razonamiento. Método Deductivo, Método Inductivo, Método Analógico o Comparativo.

2.- Coordinación de la materia. Método Lógico, Método Psicológico.

3.- Concretización de la enseñanza. Método Simbólico o Verbalístico, Método Intuitivo.

4.- Sistematización de la materia. Métodos de Sistematización (Rígida y Semirígida), Método Ocasional.

5.- Actividades del alumno. Método Pasivo (Dictados, Lecciones marcadas en el libro de texto, Preguntas y respuestas, Exposición Dogmática), Método Activo.

6.- Globalización de los conocimientos. Método de Globalización, Método no globalizado o de Especialización, Método de Concentración.

7.- Relación del profesor con el alumno. Método Individual, Método Recíproco, Método Colectivo.

8.- Aceptación de lo que es enseñado. Método Dogmático, Método Heurístico.

9.- Trabajo del alumno. Método de Trabajo Individual, Método de Trabajo Colectivo, Método Mixto de Trabajo.

10.- Métodos en cuanto al abordaje del tema de estudio. Método Analítico, Método Sintético.

Si desea profundizar a cerca de los métodos de enseñanza, abocarse al Anexo2

2.4.- Enseñanza Virtual.

En las sociedades modernas, las nuevas tecnologías de la información y la comunicación, adquieren un significado predominante. Podría decirse que, la

abundancia de información es casi infinita e imposible de asimilar en su totalidad, la complejidad del conocimiento es cada vez más grande, el cambio vertiginoso en los aspectos de la sociedad vuelve caducos los conocimientos y hábitos con gran velocidad, el tiempo de los individuos se convierte en recurso escaso y costoso, los valores y actitudes de las nuevas generaciones cambian con las tendencias de la internacionalización, de los saberes y la globalización de la economía.

Con el transcurso de los años las formas alternativas de enseñanza, como la enseñanza a distancia, han evolucionado de los cursos por correspondencia a los cursos por videoconferencia o satélite. Sin embargo, nunca han llegado a los niveles de refinamiento de la enseñanza impartida en las aulas. Las posibilidades de conexión a Internet y una nueva generación de programas informáticos hacen posible un nuevo modelo de enseñanza en línea de mucha mayor calidad y flexibilidad que podría recibir el nombre, más apropiado, de enseñanza virtual.

Es así como la educación virtual, surge como una necesidad de los tiempos modernos, donde el estudiante debe capacitarse en forma permanente, para lo cual requiere aprender a regular su propio ritmo de aprendizaje conciliando su tiempo de trabajo, de estudio, de socialización, de diversión y recreación, así como seleccionando por sí mismo las temáticas e información de su interés, de acuerdo con su propia necesidad, utilizando los diferentes medios de auto instrucción y comunicación que ofrece el mundo moderno.

La enseñanza virtual, en la que participan tecnologías diversas, métodos de enseñanza, técnicas de colaboración e instructores, eleva la enseñanza a niveles inalcanzables con los métodos tradicionales, sobre todo en lo que respecta a flexibilidad y a disponibilidad (en cualquier momento y desde cualquier lugar). La enseñanza virtual alcanza su apogeo si se desarrolla la tecnología hasta el punto de que pueda integrar los tres métodos de enseñanza: asíncrona, síncrona y autoformación.

La enseñanza virtual se está configurando como una herramienta de gran utilidad porque presenta productos formativos:

1.- Interactivos. En los que el usuario puede adoptar un papel activo en relación al ritmo y nivel de trabajo.

2.- Multimedia. Ya que incorpora textos, imágenes fijas, animaciones, vídeos, sonido.

3.- Abiertos. Ya que permite una actualización de los contenidos y las actividades de forma permanente, algo que los libros de texto no poseen.

4.- Sincrónicos y Asincrónicos. Que permiten que los alumnos puedan participar en tareas o actividades en el mismo momento independientemente del lugar en que se encuentren (sincrónico), o bien la realización de trabajo y estudio individual en el tiempo particular de cada alumno (asincrónico).

5.- Accesibles. Significa que no existen limitaciones geográficas, ya que utiliza todas las potencialidades de la Red Internet, de manera que los mercados de la formación son abiertos.

6.- Con recursos on-line. Los alumnos los pueden recuperar en sus propios ordenadores personales.

Distribuidos, de manera que los recursos para la formación no tienen por qué concentrarse en un único espacio o institución. Las potencialidades de la red permiten que los alumnos puedan utilizar recursos y materiales didácticos esparcidos por el mundo en diferentes servidores de Internet. También permite poder recurrir a formadores que no necesariamente tienen que estar en el mismo espacio geográfico donde se imparte el curso.

Con un alto seguimiento del trabajo de los alumnos, ya que los formadores organizan la formación en base a tareas que los alumnos deben realizar y remitir en tiempo y forma establecida.

Comunicación horizontal entre los alumnos, debido a que la colaboración forma parte de las técnicas de formación.

Los procesos de cambio sociales, tecnológicos y científicos exigen una permanente actualización en todo profesional productivo. Estos acelerados cambios llevan implícitos tres claves para adaptarse a las nuevas situaciones: una atención específica al cambio y a la innovación, unos proyectos de mejora y formación permanente del capital humano, y por último, el empleo de Nuevas Tecnologías de la Comunicación y de la Información asociadas a la producción.

Dada la naturaleza investigadora de la universidad, es conocida su especial dedicación a la investigación y al desarrollo tecnológico. De la misma forma, es competencia de ella estar en vanguardia de las últimas innovaciones científico-técnicas. A esta tradición se une en la actualidad una frecuente atención a los procesos de mejora de la gestión y de la docencia con su informatización.

La informatización de la universidad, con la creación de espacios y servicios virtuales a través de la red, pretende mejorar y optimizar el conjunto de sus actividades y funciones. Este enfoque necesita diferenciarse de la Universidad Virtual, en la que todos los espacios y comunicaciones entre seres humanos están exclusivamente y necesariamente mediados por las redes y los ordenadores (CMC).

El objetivo que se pretende con el uso de los recursos informáticos y las redes en la universidad, no es sólo desarrollar una enseñanza a distancia, una enseñanza virtual..., sino, y además, una docencia que utiliza las posibilidades de los recursos tecnológicos, especialmente las ofrecidas por las redes con sus rupturas del espacio y el tiempo, para mejorar los procesos de enseñanza y aprendizaje presénciales.

El concepto de virtual posee una polisemia muy diversa en estos momentos cuando es utilizado en el mundo educativo, y es utilizada en muchos casos sin ninguna definición clara o explícita. En el sistema universitario definimos los "campus virtuales" como aquella infraestructura de redes y ordenadores que generan nuevos espacios para que la comunidad universitaria desempeñe sus procesos de comunicación, gestión y servicios, investigación, enseñanza y aprendizaje.

2.5.- Ingeniería de Software una perspectiva didáctica.

Todo proceso de enseñanza – aprendizaje está regido por las seis categorías de la didáctica: *objetivos de la educación* (¿qué se quiere con la enseñanza?), *contenido* (¿qué conocimientos enseñar que permitan el cumplimiento de los objetivos? y ¿qué debe aprender a hacer el estudiante?), *métodos y procedimientos de enseñanza* (¿cómo regular las actividades del profesor y el estudiante para el logro de los objetivos propuestos?), *medios de enseñanza* (¿con qué enseñar y aprender?), *formas de organización* (¿cómo organizar el enseñar y el aprender?) y *evaluación* (¿en qué medida se cumplen los objetivos?). Todas estas categorías conforman un sistema en el cual al modificar una de estas es necesario el re-análisis y la redefinición del resto, que garanticen la continuidad del funcionamiento del sistema.

La Ingeniería de Software se puede decir, que es una materia de reciente surgimiento en el planeta, ha sido el resultado de la necesidad de aumentar en el proceso de producción de software, la eficiencia y la calidad del producto final de este, así como de su proceso productivo en general. Es una gran incógnita en cualquier universidad del mundo el *¿cómo lograr impartir de una mejor manera la Ingeniería de Software de forma tal que se logren la formación de las habilidades y la asimilación de los conocimientos por los estudiantes? La UNI* no está ajena a esta problemática. Si se le suma a lo planteado, que es un área del conocimiento a la cual constantemente se le suman los resultados obtenidos en las investigaciones de los propios procesos productivos y que se modifica no solo desde el punto de vista del tipo de software que se esté desarrollando, se entiende mucho mejor la complejidad de su enseñanza.

Ha sido un consenso a largo de las últimas décadas en las universidades Nicaragüenses en las cuales se estudia la Ingeniería Informática (en cualquiera de sus ramas); así como la de la mayoría de las del mundo entero donde se

estudia esta carrera, de que los objetivos principales están vinculados a lograr en los estudiantes el conocimiento de procedimientos, métodos, técnicas y herramientas que le permitan planificar, estimar y controlar la producción de software. Idea que no difiere con el propio objeto de estudio de la Ingeniería de Software a nivel mundial por sus principales e iniciales impulsores, como rama de la Informática.

La Ingeniería de Software -como ya ha sido mencionado- es una materia en la cual el contenido está sometido a una frecuencia de cambio o de generación de nueva información bastante grande; así como que es de trascendental importancia para el futuro Ingeniero. Por este motivo, la categoría *Métodos de enseñanza y aprendizaje*, así como los *procedimientos didácticos* en la impartición de la Ingeniería de Software deben tener un enfoque mayoritariamente práctico y lograr, basados en un gran análisis educativo – metodológico, el cumplimento de los objetivos y la asimilación de los contenidos por parte de los estudiantes.

Por otra parte, los medios de enseñanza deben contribuir a la apropiación del contenido por los estudiantes, haciendo el proceso productivo de una manera que involucre y motive a los estudiantes, permitiéndoles *penetrar en la esencia*, en lo fundamental del contenido.

Es en el proceso de enseñanza y aprendizaje de la Ingeniería de Software, por ser una asignatura técnica de la especialidad y de gran necesidad para el ejercicio de la profesión del egresado; donde los *medios de enseñanza* deben constantemente permitir al estudiante llegar a la esencia de los problemas a resolver.

Consideramos que de forma clásica se puede clasificar a los medios de enseñanza en el proceso de enseñanza – aprendizaje de la Ingeniería de software en tres tipos: los *tradicionales* (láminas, dibujos, transparencias, maquetas, esquemas, etc.), los generados a partir del desarrollo de las *Tecnologías de la Información y las Comunicaciones (TIC)* (láminas en Power Point, Internet, correo electrónico, etc.) y las propias *herramientas software* de la asignatura (herramientas CASE, herramientas para la planificación, estimación y seguimiento de proyectos, entre otros). En la actualidad del mundo, y de la UNI se hace necesario adicionar a esta clasificación un nuevo medio de enseñanza, que como la computadora, ha pasado a ser no solo un trasmisor de conocimiento, sino que para muchos es el único al que tienen acceso por su abaratamiento y nos referimos al Internet. Debemos señalar que un uso planificado, eficiente y balanceado de la Internet, apunta a ser (y casi se ha establecido como tal), la única fórmula eficiente para lograr una educación desarrolladora en el caso de la impartición de la docencia de la ISW en la UNI.

Obviamente, no todo es color de rosa; la Internet tiene también algunas limitaciones que debemos asumir cuando pensamos en ella para su uso

educacional o didáctico. Ella tiende a la pasividad, a menos que los sitios se realicen teniendo en cuenta un alto nivel de participación racional del estudiante; para que la atención no se disperse fácilmente hacia otros sitios, su carácter unidireccional impide una corrección inmediata de los contenidos o el estilo de la clase en dependencia de las necesidades del receptor, lo cual debe suplirse posteriormente con un trabajo correccional; tiende a la mediocridad, pues quien prepara las clases lo hace teniendo como población destinataria al alumno "promedio" del grupo, es decir, el que en la campana de probabilidades se encuentra ubicado en la zona mayoritaria, en términos de inteligencia, aptitudes, motivaciones, conocimientos precedentes y actuales, vocabulario y otros elementos, de los cuales quedan excluidos los dos extremos, donde se encuentran los alumnos menos aventajados y los más aventajados.

Por todo lo planteado y analizando el actual plan de estudios de la asignatura en cuestión; consideramos que una de las asignaturas que más utilidad pudiese obtener del medio en cuestión es la ISW, teniendo en cuenta los contenidos a trabajar, así como las habilidades a formar, estando la mayoría ligadas a capacidades de dirección, liderazgo, planificación, modelación, síntesis como las fundamentales a mencionar. Es por esta razón que todas las conferencias que aparecen en el *Plan Calendario* de la asignatura pueden ser apoyadas por emisiones didácticas.

CAPITULO 3.
Bases temática para la estructura de la aplicación.

La consolidación teórica de los diferentes planteamientos de ingeniería del software, contribuirá a estudiar, seleccionar, y aplicar un mecanismo para el desarrollo, mantenimiento y seguimiento satisfactorio de un sistema de aplicación informática.

El universo de este trabajo giró entorno a toda la documentación bibliográfica recolectada así como información, debidamente verificada, procedente de Internet. De este universo se retomó una muestra que contempla las unidades bibliográficas que contengan la información más sólida y completa.

3.2.- Programas analíticos actuales.

Como punto de partida retomamos los programas analíticos de la asignatura Ingeniería de software (I, II, III) en los que se propone una organización de las unidades temáticas de forma tal, que se aborde progresivamente cada etapa de desarrollo de sistemas. La metodología recomendada para lograr una asimilación de los conocimientos de cada unidad, es combinando teoría y práctica.

3.2.1.- Programa analítico de asignatura. Ingeniería de Software I.

I. INFORMACION GENERAL

UNIVERSIDAD:	Universidad Nacional de Ingeniería (UNI)
FACULTAD:	Facultad de Electrotecnia y Computación (FEC)
CARRERA:	Ingeniería en Computación
PLAN:	1995
DISCIPLINA:	Ingeniería de Software
ASIGNATURA:	Ingeniería de Software I
TIPO DE ASIGNATURA:	Obligatoria
CÓDIGO:	
AÑO:	Cuarto
SEMESTRE:	Octavo
PRE-REQUISITO:	Administradores de Bases de Datos
PRECEDENCIA:	Economía de Proyectos Informáticos
EQUIVALENCIA PLAN 87:	Ingeniería de Software I
CRÉDITOS:	4
HORAS:	84
FRECUENCIA SEMANAL:	6 Horas

II.- OBJETIVOS

Generales

Que el estudiante sea capaz de:

Conocer los diferentes paradigmas de desarrollo de sistemas, sus etapas y su utilización de acuerdo a las características de los proyectos.

Modelar un Sistema de Información, haciendo uso de un paradigma de desarrollo de sistema, siguiendo cada una de sus etapas hasta llegar a la etapa del diseño.

Específicos

Que el estudiante sea capaz de:

Reconocer la importancia y utilidad de los paradigmas de desarrollo de sistemas y en qué tipo de circunstancias deben ser utilizados.

Identificar los requerimientos de un sistema, para elaborar el estudio de factibilidad que incluya las recomendaciones acorde a las características y necesidades del sistema, tanto en el aspecto económico, técnico, operativo como legal.

Modelar los requerimientos del sistema mediante la metodología de Análisis y Diseño que mejor se ajuste a la naturaleza de la situación planteada.

Comprender la importancia de documentar el Análisis y el Diseño del sistema.

III.- RECOMENDACIONES METODOLÓGICAS Y DE ORGANIZACIÓN

Para alcanzar los objetivos definidos, se propone una organización de las unidades de forma tal, que se aborde progresivamente cada etapa de desarrollo de sistemas. La metodología recomendada para lograr una asimilación de los conocimientos de cada unidad, es combinando teoría y práctica de la forma que sigue:

Cada unidad debe contener:

Clases Conferencias. Las que consistirán de Clases Magistrales a impartirse por el docente. Se recomienda el uso de datashow o cañón, o en su defecto, proyector para la presentación de filminas.

Seminarios. Se recomienda que el docente asigne investigaciones sobre algunos temas del curso, los cuales, deberán ser presentados por los grupos asignados, según programación elaborada por el docente al inicio del curso. Los grupos deben estar organizados por 2 ó 3 estudiantes.

El documento de la investigación debe constar de 15 páginas y contemplar como mínimo lo siguiente:

a) Base conceptual (Científico / Metodológica)
b) Ejemplos prácticos
c) Conclusiones y Recomendaciones
d) Proyección al futuro
e) Bibliografía
f) Anexos

Para el desarrollo de los incisos (b), (c) y (d) los estudiantes deben basarse, además del estudio de la teoría, en entrevistas realizadas en el ámbito nacional.

El documento debe ser entregado una semana antes de la exposición, para que el docente puntualice detalles de enfoque. Se sugiere como tiempo promedio 20 minutos para la exposición del tema y 10 minutos para discusión con la participación de los demás estudiantes.

También, el docente puede organizar seminarios a impartirse por profesores invitados y/o profesionales del área.

Clases Prácticas. Las que consistirán de clases de discusión o de casos prácticos a ser trabajados por grupos, para luego presentar los resultados en la pizarra.

Laboratorios. En los cuales el docente deberá presentar algunas herramientas CASE para Análisis y Diseño de sistemas. Se recomienda para los laboratorios la ubicación de no más de dos estudiantes por computadora, así como el uso de datashow o cañón, de manera que facilite las demostraciones y prácticas.

El estudiante deberá reforzar el uso de las herramientas CASE en tiempo extra clase, de manera que pueda utilizarlas en la documentación del proyecto de curso.

Proyecto de curso. Donde los grupos, formados por 2 ó 3 estudiantes, podrán presentar los avances de su proyecto al resto de la clase, así como recibir retroalimentación del mismo, según programación elaborada por el docente.

Se sugiere que el docente oriente el desarrollo del proyecto de curso desde el inicio, pudiendo utilizar como base los proyectos de las asignaturas de la disciplina de Ingeniería de Software previamente desarrollados, o bien, orientar otros casos para poner en práctica los conocimientos adquiridos en la clase.

Se recomienda que el contenido del documento del proyecto de curso contenga lo siguiente:

I Introducción
Antecedentes
Planteamiento de la situación
Objetivos (Generales y Específicos)
II Estudio de Factibilidad
Requerimientos del sistema
Factibilidad Técnica
Factibilidad Económica
Factibilidad Operativa
Factibilidad Legal
Alternativa propuesta
III Análisis del Sistema
(en dependencia de la metodología)
IV Diseño del Sistema
(en dependencia de la metodología)
V Conclusiones y Recomendaciones
Bibliografía
Anexos

Para el desarrollo del proyecto los estudiantes deben basarse preferiblemente en un caso de la vida real.

El documento debe ser entregado una semana antes de la defensa, para que el docente estudie el caso en particular. Se sugiere como tiempo promedio 20 minutos para la presentación del proyecto de curso y 10 minutos para preguntas y respuestas, en las que pueden participar los demás estudiantes.

La defensa de los proyectos, generalmente se realiza en el período de los II exámenes parciales, por lo que se recomienda reconocerle al docente un número no menor de 15 horas de clase.

Se recomienda que la defensa de los proyectos se realice ante todo el grupo de clase, de manera que todos los estudiantes puedan adquirir conocimientos sobre los diversos casos abordados y los modelos propuestos.

Siendo que el proyecto de curso tiene un peso considerable en el sistema de evaluación, el estudiante deberá aprobar el mismo para tener derecho a

realizar la I convocatoria. La nota del proyecto de curso formará parte de la I convocatoria, en las proporciones establecidas en la Nota Final. No se realiza II convocatoria ni examen a título de suficiencia.

IV.- PLAN TEMATICO

UNIDAD	TEMA	C	S	CP	LAB	PRC	TOTAL
I	Introducción a la Ingeniería de Software	4	2	2	0	0	8
II	Factibilidad de Proyectos	4	4	4	0	4	16
III	Análisis de Sistemas de Información	10	4	4	4	8	30
IV	Diseño de Sistemas de Información	10	4	4	4	8	30
TOTALES		**28**	**14**	**14**	**8**	**20**	**84**

V.- CONTENIDOS POR TEMAS

UNIDAD I: INTRODUCCION A LA INGENIERIA DE SOFTWARE

I.1 Qué es Ingeniería de Software
- I.2.1 Conceptos
- I.2.2 Papel del desarrollador de sistemas
- I.2.3 Ciclo de vida clásico
- I.2.4 Otros paradigmas de Desarrollo de Sistemas

UNIDAD II: ESTUDIO DE FACTIBILIDAD DE PROYECTOS

II.1 Definición
- II.1.1 Técnicas de recolección de información
- II.1.2 Especificación de requerimientos del sistema

II.1.3 Confiabilidad, Protección y Garantía como categorías de Análisis

II.1.4 Estructura del Estudio de Factibilidad

II.2 Plataformas de Desarrollo

II.2.1 Parámetros de selección y evaluación de Hardware y Software
- II.2.2 Establecimiento de prioridades (Identificar/evaluar)
- II.2.3 Características operativas

II.3 Establecimiento de Costos
- II.3.1 Medición y Métricas (Tiempo y complejidad del proyecto)
- II.3.2 Recursos Humanos
- II.3.3 Recursos Materiales
- II.3.4 Elaboración del Plan de Actividades

UNIDAD III: ANALISIS DE SISTEMAS DE INFORMACION

III.1 Análisis
- III.1.1 Definición
- III.1.2 Actividades comunes

III.2 Tipos de Especificaciones
- III.2.1 Formales
- III.2.2 Algebraicas
- III.2.3 Basadas en modelo

III.3 Análisis Orientado a los Procesos
- III.3.1 Conceptos
- III.3.2 Actividades
- III.3.3 Herramientas

III.4 Análisis Orientados a los Datos
- III.4.1 Conceptos
- III.4.2 Actividades
- III.4.3 Herramientas

III.5 Análisis Orientado a los Objetos
- III.5.1 Conceptos
- III.5.2 Actividades
- III.5.3 Herramientas

UNIDAD IV: DISEÑO DE SISTEMAS DE INFORMACION

IV.1 Diseño
- IV.1.1 Definición
- IV.1.2 Actividades comunes
- IV.1.3 Diseño Lógico
- IV.1.4 Diseño Físico
- IV.1.5 Especificación de procedimientos
- IV.1.6 Pseudocódigo

IV.2 Diseño Orientado a los Procesos
- IV.2.1 Conceptos
- IV.2.2 Actividades
- IV.2.3 Herramientas

IV.3 Diseño Orientado a los Datos
- IV.3.1 Conceptos
- IV.3.2 Actividades
- IV.3.4 Herramientas

IV.4 Diseño Orientado a los Objetos
- IV.4.1 Conceptos

IV.4.2 Actividades
IV.4.3 Herramientas

VI.- SISTEMA DE EVALUACION

Debido a que esta asignatura conlleva un gran contenido práctico, se recomienda la siguiente evaluación:

Documento de la Investigación	10%
Exposición de la Investigación	10%
Evaluaciones Sistemáticas	30%
Clases Prácticas	10%
Documento Proyecto	20%
Defensa Proyecto	20%
Total	**100%**

El documento – de la investigación y del proyecto – se puede evaluar considerando los siguientes aspectos: presentación, base conceptual / dominio del paradigma y ejemplos prácticos, entre otros.

La exposición de la investigación se puede evaluar tomando en consideración los siguientes aspectos: dominio del tema, tiempo de desarrollo de la exposición, uso de medios/recursos y capacidad de respuesta.

Para la evaluación de las clases prácticas se podrá tomar en cuenta el dominio de comprensión del caso planteado.
La defensa del proyecto se puede evaluar considerando los siguientes aspectos: planteamiento de la situación, dominio del paradigma, tiempo de desarrollo de la exposición, uso de medios/recursos, ejemplos prácticos y capacidad de respuesta.

Cada unidad tendrá su evaluación sistemática, la cual consistirá de evaluaciones escritas y planteamientos prácticos de situaciones reales a ser analizadas, discutidas y resueltas utilizando los conceptos y paradigmas aprendidos.

VII.- LITERATURA DOCENTE

1 Texto Básico

Ingeniería de Software: Un enfoque práctico, Róger S. Pressman, McGraw Hill, 1993, 3a. Edición, España o USA

2 Textos Auxiliares

Análisis y Diseño de Sistemas, Kendall y Kendall, Prentice Hall, 1991, México

Análisis y Diseño de Sistemas de Información, James A. Senn, MacGraw Hill, 1993, 2da. Edición, México

3 Textos Complementarios

The New Software Engineering, Sue Conger, 1994, 1a. Edición

Object Oriented Software Engineering, Iván Jacobson, Addison Wesley, 1994, 2a. Edición, USA

Análisis y Diseño Orientado a Objetos, James Martin - James J. Odell, Prentice Hall, 1994, México.

VIII.- RELACION DE AUTORES

Elaborado por: Ing. Raquel Rodríguez Lara
Lic. Melania Miranda Quintero

Revisado en Agosto 2001 por Lic. Magda Auxiliadora Luna Molina
Revisado en Noviembre 2001 por Ing. Flor de María Valle Izaguirre

3.2.2.- Programa analítico de asignatura. Ingeniería de Software II.

I.- INFORMACION GENERAL

UNIVERSIDAD:	Universidad Nacional de Ingenierías (UNI)
FACULTAD:	Facultad de Electrotecnia y Computación (FEC)
CARRERA:	Ingeniería en Computación
PLAN:	1995
DISCIPLINA:	Ingeniería de Software
ASIGNATURA:	Ingeniería de Software II
TIPO DE ASIGNATURA:	Obligatoria
CÓDIGO:	
AÑO:	Quinto
SEMESTRE	Noveno
PRE – REQUISITOS:	Ingeniería de Software I
PRECEDENCIA:	Ninguna
EQUIVALENCIA PLAN 87:	Ingeniería de Software II
CRÉDITOS:	4
HORAS:	84
FRECUENCIA SEMANAL:	6 Horas

II.- OBJETIVOS

Generales

Que el estudiante sea capaz de:

Consolidar las primeras etapas del desarrollo de sistemas.

Desarrollar un Sistema de Información siguiendo las técnicas de garantía de calidad del software, hasta llegar a la etapa de mantenimiento.

Específicos

Que el estudiante sea capaz de:

Comparar los paradigmas de desarrollo de sistemas entre sí, basados en el conocimiento y utilización de los mismos.

Reforzar los conocimientos adquiridos sobre análisis y diseño de sistemas de información.

Conocer e implementar las actividades que deben realizarse para garantizar un software de calidad.

Reconocer la importancia de la elaboración de Documentación consistente durante el Desarrollo y Vida Útil del Sistema.

Entender la importancia de realizar y ejecutar planes de prueba, así como el uso de principios de Ingeniería de Software durante la etapa de Mantenimiento del Sistema.

III.- RECOMENDACIONES METODOLOGICAS Y DE ORGANIZACION

Para alcanzar los objetivos definidos, se propone una organización de las unidades de forma tal que se vaya abordando progresivamente cada etapa y componente involucrado en el Proyecto de curso. La metodología recomendada para lograr una asimilación de los conocimientos de cada unidad, es combinando teoría y práctica de la forma que sigue:

Cada unidad podrá contener:

Clases Conferencias. Las que consistirán de Clases Magistrales a impartirse por el docente. Se recomienda el uso de datashow o cañón, o en su defecto, proyector para la presentación de filminas.

Clases Prácticas. Las que consistirán de clases de discusión o de casos de estudio que estén relacionados con Proyectos de desarrollo de sistemas, a ser trabajados por grupos no mayores de tres estudiantes.

Los casos de estudio deben ser definidos por el docente, el cual entregará la guía correspondiente al inicio de cada unidad. Los resultados deben ser expuestos y discutidos por todo el grupo en el aula de clases, tomando notas de las conclusiones relevantes.

Laboratorios. En los cuales el docente y los estudiantes podrán presentar/observar diversas interfaces de usuario, revisiones y pruebas de software, entre otros.

Proyecto, para su desarrollo, se sugiere organizar los grupos con 2 ó 3 estudiantes.

Se recomienda darle seguimiento al proyecto desarrollado en la asignatura de Ingeniería de Software I. Para facilitar su compresión se deben exponer los resultados del proyecto, de manera que el docente pueda hacer

recomendaciones para llevarlo a un estado optimo, durante el curso de Ingeniería de Software II. Es por ello, que se asignan 8 (ocho) horas de clase para el proyecto al inicio del curso (en el capítulo I).

El proyecto debe dividirse en etapas, las que serán expuestas por los grupos de acuerdo a calendario previamente establecido. Se recomienda que la presentación de cada etapa vaya acompañada de su documentación correspondiente.

Además del contenido del proyecto sugerido en el programa analítico de Ingeniería de Software I, se recomienda insertar los siguientes puntos relevantes:

V Diseño de la interfaz de usuario

VI Actividades realizadas para garantizar la calidad del software

VII Documentación del diseño de pruebas
Especificación del diseño de pruebas
Documentación de la ejecución de pruebas
Informe resumen de las pruebas

VIII Mantenimiento
Estimación de costos de mantenimiento
Planificación de la gestión de configuración

IX Documentación del Sistema (como producto)
Manual de Instalación
Manual del usuario

El documento debe ser entregado una semana antes de la defensa, para que el docente estudie el caso en particular. Se sugiere como tiempo promedio 20 minutos para la presentación del proyecto de curso y 10 minutos para preguntas y respuestas, en las que pueden participar los demás estudiantes.

La defensa de los proyectos, generalmente se realiza en el período de los II exámenes parciales, por lo que se recomienda reconocerle al docente un número no menor de 15 horas de clase.

Se recomienda que la defensa de los proyectos se realice ante todo el grupo de clase, de manera que todos los estudiantes puedan adquirir conocimientos sobre los diversos sistemas presentados.

Siendo que el proyecto de curso tiene un peso considerable en el sistema de evaluación, el estudiante deberá aprobar el mismo para tener derecho a realizar la I convocatoria. La nota del proyecto de curso formará parte de la I convocatoria, en las proporciones establecidas en la Nota Final. No se realiza II convocatoria ni examen a título de suficiencia.

IV.- PLAN TEMATICO

UNIDAD	TEMA	C	CP	LAB	PRC	TOTAL
I	Introducción	4	0	0	8	12
II	Otros aspectos de Diseño	4	0	2	4	10
III	Garantía de Calidad del Software	6	2	2	4	14
IV	Documentación	6	2	0	4	12
V	Prueba	8	4	2	6	20
VI	Mantenimiento	4	2	0	4	10
VII	Herramientas CASE	2	0	4	0	6
TOTALES		**34**	**10**	**10**	**30**	**84**

V.- CONTENIDOS POR TEMAS

UNIDAD I: INTRODUCCION

I.1 Evaluación de los paradigmas de desarrollo de sistemas
- I.1.1 Síntesis sobre los Modelos de Ciclos de Vida
- I.1.2 Estrategias para comparar paradigmas de desarrollo de sistemas
- I.1.3 Impacto de los Prototipos en la Ingeniería de Software
- I.1.4 Diseño estructurado vs diseño orientado a objetos

UNIDAD II: OTROS ASPECTOS DEL DISEÑO

II.1 Diseño de Procedimientos de usuario

II.2 Diseño de la Interfaz de Usuario
- II.2.1 Introducción
- II.2.2 Directrices para el diseño de interfaces
- II.2.3 Estándares de interfaz

UNIDAD III: GARANTÍA DE CALIDAD DEL SOFTWARE

III.1 Calidad del software y garantía de calidad del software
- III.1.1 Conceptos
- III.1.2 Factores que determinan la calidad del software
- III.1.3 Actividades de la garantía de calidad del software

III.2 Revisiones del software
- III.2.1 Conceptos

III.2.2 Revisiones técnicas formales

UNIDAD IV: DOCUMENTACION

IV.1 Clasificación de Documentación
- IV.1.1 Documentación del Proceso
- IV.1.2 Documentación del Producto

IV.2 Preparación de la Documentación
- IV.2.1 Estructura, Políticas y Producción

UNIDAD V: PRUEBA

V.1 Terminología

V.2 Técnicas de diseño de casos de prueba
- V.2.1 Enfoque estructural o caja blanca
- V.2.2 Enfoque funcional o caja negra

V.3 Documentación del diseño de pruebas
- V.3.1 Plan de pruebas
- V.3.2 Especificación del diseño de pruebas
- V.3.3 Especificación de caso de prueba
- V.3.4 Especificación de procedimiento de prueba

V.4 Ejecución de pruebas
- V.4.1 El proceso de ejecución
- V.4.2 Documentación de la ejecución de pruebas
- V.4.3 Histórico de pruebas
- V.4.4 Informe de incidente
- V.4.5 Informe resumen de las pruebas
- V.4.6 Depuración

UNIDAD VI: MANTENIMIENTO

VI.1 Mantenimiento de Software
- VI.1.1 Tipos de Mantenimiento
- VI.1.2 Costos de Mantenimiento
- VI.1.3 Estimación de Costos de Mantenimiento
- VI.1.4 Reingeniería de Software

VI.2 Gestión de Configuración
- VI.2.1 Planificación
- VI.2.2 Control de Cambios
- VI.2.3 Manejo de Versión

UNIDAD VII: HERRAMIENTAS CASE

VII.1 Clasificación de las herramientas CASE

VII.2 Integración de herramientas CASE

VII.3 Plan para la adopción de herramientas CASE

VI.- SISTEMA DE EVALUACION

Se recomienda que el docente organice las evaluaciones sistemáticas, de manera que evalúe equitativamente el contenido del curso, mediante evaluaciones escritas y planteamientos prácticos de situaciones reales a ser analizadas, discutidas y resueltas, utilizando los conocimientos adquiridos.

Para la evaluación de las clases prácticas se podrá tomar en cuenta el dominio de comprensión del caso planteado
El documento del proyecto se puede evaluar considerando los siguientes aspectos: presentación, mejoras a las etapas de estudio de factibilidad, análisis y diseño del sistema, documentación del proceso, plan de prueba, manual de instalación y manual del usuario, entre otros.

La defensa del proyecto se puede evaluar considerando los siguientes aspectos: planteamiento de la situación, diseño de la interfaz de usuario, informe resumen de las pruebas, calidad del software, tiempo de desarrollo de la exposición, uso de medios/recursos, y capacidad de respuesta.

Debido a que esta asignatura conlleva un gran contenido práctico, se recomienda la siguiente evaluación:

Evaluaciones Sistemáticas	40%
Clases Prácticas	10%
Documento Proyecto	25%
Defensa Proyecto	25%
Total	**100%**

VII.- LITERATURA DOCENTE

1 Texto Básico

Ingeniería de Software: Un enfoque práctico, Róger S. Pressman, McGraw Hill, 1993, 3ª Edición, España o USA

2 Textos Auxiliares

Análisis y diseño detallado de Aplicaciones Informáticas de Gestión, Mario Piattini, José Calvo, Joaquín Cervera, Luis Fenández, Alfaomega, 2000, 1ª Edición, México

Análisis y Diseño de Sistemas, Kendall y Kendall, Prentice Hall, 1991, México

4 Textos Complementarios

Análisis y Diseño de Sistemas de Información, James A. Senn, MacGraw Hill, 1993, 2da. Edición, México

Object Oriented Software Engineering, Iván Jacobson, Addison Wesley, 1994, 2a. Edición, USA

VIII.- RELACION DE AUTORES

Elaborado por: Ing. Raquel Rodríguez Lara
Lic. Melania Miranda Quintero
Revisado en Septiembre 2001 por Lic. Magda Auxiliadora Luna Molina.
Revisado en Noviembre 2001 por Ing. Flor de María Valle Izaguirre

3.2.3.- Programa analítico de asignatura. Ingeniería de Software III.

I.- INFORMACION GENERAL

UNIVERSIDAD:	Universidad Nacional de Ingenierías (UNI)
FACULTAD:	Facultad de Electrotecnia y Computación (FEC)
CARRERA:	Ingeniería en Computación
PLAN:	1995
DISCIPLINA:	Ingeniería de Software
ASIGNATURA:	Ingeniería de Software III
TIPO DE ASIGNATURA:	Optativa
CÓDIGO:	
AÑO:	Quinto
SEMESTRE	Décimo
PRE – REQUISITOS:	Ingeniería de Software II
PRECEDENCIA:	Ninguna
EQUIVALENCIA PLAN 87:	Ninguna
CRÉDITOS:	5
HORAS:	84
FRECUENCIA SEMANAL:	6 Horas

II.- OBJETIVOS

Generales

Tomar conciencia de la importancia de construir y mantener la calidad en los servicios informáticos.

Conocer y aplicar las técnicas de control de calidad y su aseguramiento.

Conocer las condiciones bajo las cuales un proyecto informático puede ser auditado.

Específicos

Conocer y aplicar diferentes técnicas de prueba de software para evaluar la calidad de los productos y sistemas informáticos.

Conocer y aplicar estrategias para el aseguramiento de la calidad de los servicios, para que los proyectos informáticos cuenten con este atributo.

Conocer y aplicar métodos de auditoría informática en cada una de las diferentes actividades que se realizan en el ámbito informático.
Conocer y aplicar métodos de evaluación de sistemas ajustados a cada una de las actividades informáticas que se realizan en el ámbito informático.

III.- RECOMENDACIONES METODOLOGICAS Y DE ORGANIZACIÓN

Esta asignatura esta compuesta de conocimientos complementarios al quehacer informático, aunque no por ello son los menos importantes. Se recomienda para alcanzar la asimilación de los temas propuestos, combinar teoría y práctica de forma que se desarrolle el contenido teórico a través de clases conferencias, seminarios, foros – debates, y concluyan con el desarrollo de un caso de estudio que abarque los contenidos planteados para cada una de ellas.

Para el desarrollo de los seminarios y foros – debates, el docente asignado deberá de hacer lo posible por invitar a profesionales del ramo con el objetivo de enriquecer y relacionar los conocimientos con el ejercicio práctico. Esta modalidad se realizará en horas destinadas a clases prácticas por lo que los estudiantes deberán de organizarse en grupos de trabajo no menores de cuatro y no mayores de siete, para elaborar preguntas, planteamientos y conclusiones las que se discutirán en clase con el objetivo de medir el nivel de asimilación de los temas. Por considerarse clase práctica el docente deberá definir al inicio de cada unidad la guía correspondiente.

Al final de cada unidad se propone el desarrollo de un caso de estudio en el que se reflejen el dominio de los conocimientos adquiridos. Estos trabajos deben de ser reflejados en un documento y expuestos al grupo de clase. Para ello se conformaran grupos de estudiantes no menores de dos y no más de cuatro participantes. El docente deberá de definir al inicio del semestre los casos de estudio de cada unidad, la organización y guía de desarrollo del mismo, al igual que los límites tanto de páginas en el documento como de tiempo para exponerlo. Los estudiantes también podrán plantear propuestas de casos de estudio.

Se recomienda para un mejor desarrollo y aprovechamiento de cada sesión de clase (magistral, seminario, expositiva, adiestramiento) el uso de medios audiovisuales (retroproyector, datashow) y de ser posible el uso de computadoras para hacer demostraciones en los casos necesarios. También será de mucha utilidad que el docente retome documentos y sistemas reales desarrollados como proyectos en otras asignaturas o documentación o sistemas implementados en instituciones que deseen colaborar con la universidad, para el apoyo al desarrollo práctico de la asignatura.

IV.- PLAN TEMATICO

UNIDAD	TEMA	C	CP	S	TOTAL
I	Control de calidad	16	4	10	30
II	Evaluación y auditoría informática.	36	4	14	54
TOTALES		**52**	**8**	**24**	**84**

V.- CONTENIDOS POR TEMAS

UNIDAD I: Control de calidad.

I.1 Garantía de la calidad del software.
- I.1.1 Garantía de la calidad de procesos.
- I.1.2 Establecimientos de Estándares.
- I.1.3 Métricas de software.

I.2 Validación y Verificación.
- I.2.1 Validación y Verificación.
- I.2.2 Fiabilidad del software.
- I.2.3 Seguridad del Software.

I.3 Estrategias de Prueba.
- I.3.1 Prueba Caja Blanca.
- I.3.2 Prueba Caja Negra.
- I.3.3 Prueba Top – Dow.
- I.3.4 Prueba bottom Up.
- I.3.5 Prueba de integración.

UNIDAD II: Evaluación y Auditoría informática.

II.1 Conceptos de auditoría.
- II.1.1 Concepto.
- II.1.2 Tipos.
- II.1.3 Planeación.
- II.1.4 Riesgos informáticos. Tipología de Riesgos.

II.2 Controles informáticos y medidas de prevención.

II.3 Metodología y Fases de la auditoría informática.
- II.3.1 Métodos.
- II.3.2 Organización.

II.4 Evaluación.
- II.4.1 Procedimientos.

II.4.2 Sistemas.
II.4.3 Seguridad.

VI.- SISTEMA DE EVALUACION

El docente definirá la forma de realizar la evaluación escrita, la que podrá realizarse ya sea a través de sistemáticos o la realización del examen Parcial y Final.

Evaluaciones Sistemáticas	30%
Participación Clases Prácticas	10%
Casos de estudio - Documentos	25%
Casos de estudio –Exposición y Foros	25%
Total	**100%**

VII.- LITERATURA DOCENTE

1 Texto Básico

Ingeniería de Software: Un enfoque práctico, Róger S. Pressman, McGraw Hill, 1993, 3ª Edición, España o USA

Software Engineering, Ian Sommerville, Addison Wesley, USA 1998, 4ta edición.

2 Texto Auxiliar

Auditoría informática, Un enfoque práctico. Mario Piattini, Emilio del Peso. Editorial Alfaomega, 2da edición, 2000.

Auditoría informática, 1990, José Antonio Echenique, Mc Graw Hill, Edición Unica.

Textos Complementarios

The New Software Engineering, Sue Conger, Internactional Thomson Publishing Company ITP, 1994, 1ra edición.

CAPITULO 4
Herramientas de desarrollo.

El desarrollo de este estudio está influenciado en algunos aspectos de diseño, forma y funcionalidad por:

1.- CD´s multimedia elaborados para Enciclopedias.
2.- Sitios Web de fines didácticos.
3.- Libros digitales varios.
4.- Libros bibliográficos de la asignatura.
5.- Observaciones del tutor y de los estudiantes.

Los elementos multimedia utilizados fueron; imagen, sonido, texto, así como también las herramientas para desarrollarlos.

4.1.- Adobe Flash® (Fl).

Es una aplicación que trabaja sobre "Fotogramas" destinado a la producción de animación. Es actualmente escrito y distribuido por Adobe Systems, y utiliza gráficos vectoriales e imágenes ráster, sonido, código de programa, flujo de vídeo y audio bidireccional (el flujo de subida sólo está disponible si se usa conjuntamente con Macromedia Flash Communication Server). En sentido estricto, Flash es el entorno y Flash Player es el programa de máquina virtual utilizado para ejecutar los archivos generados con Flash.

Los archivos de Flash, que tienen generalmente la extensión de archivo SWF, pueden aparecer en una página web para ser vista en un navegador, o pueden ser reproducidos independientemente por un reproductor Flash. Los archivos de Flash aparecen muy a menudo como animaciones en páginas Web y sitios Web multimedia, y más recientemente Aplicaciones de Internet Ricas. Son también ampliamente utilizados en anuncios de la web.

4.2.- Adobe Reader.

Anteriormente conocido como Adobe Acrobat Reader, fue el primer programa en soportar el formato PDF (Portable Document Format). El Adobe Acrobat Reader de Adobe Systems, actualmente conocido solamente como Adobe Reader es una aplicación que permite visualizar e imprimir archivos en formato PDF y está disponible gratuitamente para descargar desde el sitio Web de Adobe. Está disponible para los sistemas operativos Linux, Mac OS y Microsoft Windows. El uso del formato PDF está ampliamente extendido para mostrar texto con un diseño visual ordenado.

4.3.- Adobe Dreamweaver® (Dw).

Es una aplicación en forma de estudio (Basada por supuesto en la forma de estudio de Adobe Flash®) pero con más parecido a un taller destinado para la edición WYSIWYG de páginas web, creado inicialmente por Macromedia (actualmente es propiedad de Adobe Systems). Es el programa de este tipo más utilizado en el sector del diseño y la programación web, por sus funcionalidades, su integración con otras herramientas como Adobe Flash y, recientemente, por su soporte de los estándares del World Wide Web Consortium. Su principal competidor es Microsoft Expression Web. Tiene soporte tanto para edición de imágenes como para animación a través de su integración con otras herramientas

4.4.- WinRAR.

Es la versión para Windows de RAR, un potente programa compresor y descompresor de datos multi-función (tiene varios modos de compresión). Se trata de una herramienta útil para ahorrar espacio de almacenamiento y tiempo de transmisión al enviar y recibir archivos a través de Internet o al realizar copias de seguridad. Se distribuye como shareware, esto significa que puede probarlos gratuitamente durante 40 días, pasado este periodo de prueba deberá comprar una licencia o desinstalarlos de su ordenador.

4.5.- Sitio web (en inglés: website).

Es un conjunto de páginas web, típicamente comunes a un dominio de Internet o subdominio la World Wide Web en Internet.

4.6.- Página web.

Es un documento HTML/XHTML accesible generalmente mediante el protocolo HTTP de Internet.

4.7.- HTML.

Siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

Tercera Parte
Análisis y presentaión de resultados.

1.- ANÁLISIS DE ENCUESTA.

La demanda de un proyecto como este debía de ser reflejada de alguna manera, por lo cual se procedió a la realización de encuestas dirigidas hacia estudiantes que llevaban o llevaron la asignatura de ingeniería de software. Gracias a esto se logró reunir las bases suficientes que respaldan la concepción de está idea - el desarrollo de una herramienta digital,- mediante el uso de preguntas que resultaron claves en el planteamiento de las necesidades y vacíos de los involucrados en el desarrollo de la clase.

La encuesta original realizada a los estudiantes, constó de un total de doce (12) preguntas y tuvo como objetivo principal establecer el nivel de desarrollo en que se encuentra la asignatura de ingeniería de software, mediante el análisis del contenido y documentación utilizado para impartirla. De esta encuesta tomamos las cinco principales preguntas (5) que nos revelan información más significativa. La encuesta fue aplicada a una población estudiantil integrada por estudiantes de la carrera de Ingeniería en Computación de la Universidad Nacional de Ingeniería, que cursan o cursaron la asignatura. (Ver anexo 4)

1.- El análisis de la pregunta No 1, nos muestra que más del 80% por ciento de los entrevistados se encuentra inconforme – en alguna manera - con el contenido abarcado en la asignatura, en comparación a las expectativas que ellos tenían.

2.- Las respuestas obtenidas a partir de la pregunta No 2, nos revelan que más del 85% de los entrevistados presentó cierto grado de dificultad al momento de cursar la clase de Ingeniería de Software.

3.- Para saber el motivo o los motivos de esas dificultades se formuló la pregunta No. 3 de la encuesta. Del porcentaje de entrevistados que presentaron dificultad al cursar la clase se obtuvo que: 15% dijo que era debido al contenido del plan temático de la clase. El 13% dijo que fue por la Bibliografía. Un 7% dijo que fue por la complejidad de la asignatura, un 54% afirmó que fue debido a la metodología utilizada por el docente, y el restante 11% se debió a problemas varios.

4.- Para determinar que medios bibliográficos utilizaban los encuestados para obtener información a cerca de la asignatura se planteo la pregunta No 5, de la cual se produjeron los siguientes resultados. Un 40% afirmó utilizar Internet como medio bibliográfico, contra un 48% que afirmó utilizar bibliotecas (Libros) como medio bibliográfico. El restante 12% se dividió entre consultas a Profesores o Amigos y otros medios de conseguir información.

5.- Abarcando de forma directa la idea de una herramienta digital para apoyar la enseñanza / aprendizaje, se planteo la Pregunta No 7 de la

encuesta. El 85% consideró que es una buena idea, contra el restante 15% que opino que era regular.

2.- Estructura temática propuesta.

En base a los Programas analíticos anteriores y mediante el análisis de diferentes bibliografías, la estructura temática propuesta para la aplicación es la siguiente:

Primera Parte.

Introducción a la Ingeniería de software.

Capitulo I Principios de Ingeniería de software.

1.1.- Introducción
1.2.- ¿Qué es la ingeniería de Software?
1.3.- La resolución de problemas
1.4.- Dónde encaja el ingeniero de software
1.5.- ¿Cuánto éxito hemos logrado?
1.7.-Otras Definiciones. La Ingeniería de Software
1.8.- Software de alta calidad.
1.9.- ¿Quién hace ingeniería de software?
1.10.- ¿Cuáles son los costos de ingeniería de software?

Capitulo II Responsabilidades del analista de sistemas.

2.- Introducción
- 2.1.-La información como recursos de las organizaciones
  - 2.1.1.- Manejo de la información generada por computadora
  - 2.1.2.-Tipos de sistemas informáticos
    - 2.1.2.1.- Sistemas de procesamiento de datos
    - 2.1.2.2.- Sistemas informáticos para la administración
    - 2.1.2.3.- Sistema de apoyo para la toma decisión.
    - 2.1.2.4.- Sistemas expertos
    - 2.1.2.5.- Sistemas automatizados.
    - 2.1.2.6.- Sistemas en línea.
    - 2.1.2.7.- Sistemas de tiempo real.
- 2.2.-Ejemplos
- 2.3.- Tipos de Usuarios de Sistemas
  - 2.3.1.- Los usuarios operacionales
  - 2.3.2.- Los usuarios supervisores.
  - 2.3.3.- Los usuarios de nivel ejecutivo
  - 2.3.4.- Clasificación en categoría por nivel de experiencia

2.3.5.-El Analista de Sistemas
2.4.- Papeles del Analista de Sistemas.
2.4.1.- El analista de sistema como consultor
2.4.2.- El analista de sistema como especialista de apoyo
2.4.3.- El analista de sistema como agente de cambio
2.5.- Cualidades del analista de sistemas
2.6.- Orígenes del analista en sistemas.
2.7.-Descripción de trabajo del analista moderno
2.7.1.- Definición de analista de sistemas
2.7.2.- El analista como solucionador de problemas
2.8.- Departamento De Informática
2.8.1.- Compañía De Consultoría De Gestión.
2.9.- Casas De Software
2.10.- Preparación del analista de sistema
2.11.- Experiencia y dominio de la programación informática
2.11.1.- Conocimientos generales de empresa
2.11.2.- Capacidad para resolver problemas
2.11.3.- Técnicas de comunicación interpersonal
2.11.4.- Capacidad de relación interpersonal
2.11.5.- Flexibilidad y capacidad de adaptación
2.11.6.- Carácter y ética.
2.11.7.-Análisis de sistemas y técnicas de diseño

Capitulo III Proyectos Informáticos.

3.- Introducción
3.2.- Definición de proyecto informático
3.3.1.- Administración general del proyecto
3.3.2.- Administración de la actividad del equipo de trabajo
3.4.- Fases de la planificación de u proyecto
3.4.1.- Definición de los objetivo
3.4.2.- Selección y asignación de los miembros del equipo y sus roles
3.4.3.- Desarrollo de un proyecto informático
3.5.- Organización para el trabajo en equipo
3.5.1.- Administración participativa
3.5.2.- Administración autocrática
3.5.3.- Administración jerárquica
3.6.- Tamaño del equipo de trabajo
3.7.- Comunicación con el equipo de trabajo
3.8.- Etapas de desarrollo de un grupo y las necesidades
3.9.- Software administrador de proyecto
3.10.- Características de los integrantes
3.11.- El papel de director de proyecto
3.11.1.- Alianzas como estrategias de trabajo
3.11.2- Disminuir el costo de desarrollo
3.11.3.- Penetración a otros mercados.

3.11.4.- Estimulación de la inversión extranjera
3.12.- Evaluación y control del proyecto

Segunda Parte.

Estudio de factibilidad e Ingeniería de los requerimientos.

Capitulo IV Recolección de la información.

4.- Introducción.
4.1 Técnicas para hallar datos
4.2 La entrevista
4.2.1 Preparación de la Entrevista
4.2.2 Conducción de la Entrevista
4.2.3 Secuela de la Entrevista
4.2.4 Recabar datos mediante la Entrevista
4.2.5 Determinación del tipo de Entrevista
4.2.6 Selección de Entrevistados
4.2.7 Realización de Entrevista.-
4.3 Cuestionario
4.3.1 Recabación de datos mediante cuestionarios
4.3.2 Selección de formas para cuestionarios
4.3.2.1 Cuestionario Abierto
4.3.2.2 Cuestionario Cerrado
4.4 La observación
4.4.1.- Tipos de Observación.
4.4.2 Preparación para la observación
4.4.3 Conducción de la observación
4.4.4 Secuela de la observación
4.5.-Diagrama de Flujo
4.5.1 ¿Cuándo se utiliza un Diagrama De Flujo?
4.5.2 ¿Cómo se Utiliza?
4.6.- Diccionario de datos
4.6.1.- Descripción de los Datos en el Diccionario
4.6.1.1.- Nombre de los Datos
4.6.1.2.- Descripción de los Datos
4.6.1.3.- Alias
4.6.1.4.- Longitud de campo
4.6.1.5.- Valores de los datos
4.6.1.6.- Registro de las descripciones de datos
4.7.- Conclusión

Capitulo V Requerimientos del software.

5.- Introducción
- 5.1.-¿Por qué importan los requerimientos?
  - 5.1.1.- Definición de Requerimientos
  - 5.1.2.- Especificación de los requerimientos del sistema
- 5.2 Requerimientos funcionales y no funcionales.
  - 5.2.1 Requerimientos funcionales.
  - 5.2.2 Requerimientos no Funcionales
  - 5.2.3 Los Requerimientos del Dominio.
- 5.3 Requerimientos del Usuario.
- 5.4 Requerimientos del Sistema.
  - 5.4.1 Especificaciones en Lenguaje Estructurado.
  - 5.4.2 Especificación de requerimientos utilizando un PDL.
  - 5.4.3 Las Especificaciones de Interfaces.
- 5.5 El Documento de Requerimientos del Software
- 5.6 Puntos clave.

Capitulo VI Procesos de la ingeniería de requerimientos.

6.- Introducción.
- 6.1 Estudios de factibilidad.
- 6.2 Obtención y análisis de requerimientos.
- 6.3.- Validación de los requerimientos.
- 6.4.- Revisiones de requerimientos.
- 6.5.- Administración de requerimientos.
  - 6.5.1.- Requerimientos duraderos y volátiles.
  - 6.5.2 Planeación de la administración de requerimientos.
  - 6.5.3 Administración del cambio de los requerimientos.
- 6.6.- Conclusión.

Capitulo VII: Plataformas de desarrollo

7.-Introducción.
- 7.1- Hardware: Metodología de adquisición y evaluación.
  - 7.1.1.- Utilización de la tecnología.
  - 7.1.2.- Tecnología existente.
- 7.2.- Adquisición del hardware.
  - 7.2.1 Licitaciones y compra.
    - 7.2.1.1.- Licitación publica.
    - 7.2.1.2.- Licitación por registro.
    - 7.2.1.3.- Licitaciones restringidas.
    - 7.2.1.4.- Contratación directa.
- 7.3.- Evaluación. Generalidades.
  - 7.3.1.- Características básicas a evaluar.
    - 7.3.1.1 Tipo de procesador.
    - 7.3.1.2.- Rendimiento del equipo.

7.3.1.3.- Memoria.
7.3.1.4.- Dispositivos de almacenamiento.
7.3.1.5.- Comunicaciones.
7.3.2 Impacto en el usuario
7.4.- Software: Metodología de adquisición y evaluación
7.4.1 Determinación de las necesidades.
7.4.2 Impacto en el usuario.
7.4.3 Fuentes que pueden sugerir un cambio.
7.4.4 Presentación de la necesidad.
7.5.- Levantado de requerimiento
7.5.1.- Determinación de los requerimientos
7.5.2.-Criterios relacionados con la función que se necesita
7.6.- Tipo de software
7.6.1.- Donde obtener software
7.6.2.- Pasos a seguir para obtener software
7.7.- Consideraciones a tomar

Tercera Parte.

Análisis y diseño de sistemas de información.

Capitulo XI Metodologías para el análisis de sistemas de informáticos.

11.1.- Introducción
11.2.- Análisis Orientado a Procesos.
11.2.1.- El punto de partida del análisis orientado a proceso (AOP)
11 2.2.-El primer paso en el AOP
11.2.2.1.- Convenciones utilizadas en el DFD
11.2.2.2.- Pasos para desarrollar el diagrama de flujo de datos (DFD)
11.2.3.- Elaboración del diagrama de contexto
11.2.3.1.- Elaboración del diagrama cero
11.2.3.2.- Elaboración de nivel de explosión.
11.2.3.3.- Errores comunes en el diagrama
11.2.4.- Los diagramas de flujo de datos lógicos y físicos
11.2.4.1.- El diagrama de flujo de datos
11.2.4.2.- El diagrama de flujo de datos físicos
11.3.- Análisis Orientado a Datos
11.3.1.- La ingeniería de la información.
11.3.2.- Simbología utilizada en el desarrollo del AOD.
11.3.2.1 - Diagrama Entidad – Relación
11.3.2.2.- Descomposición Funcional
11.3.2.3.- Dependencia de Procesos
11.3.2.4.- Diagrama de Flujo de Datos – Procesos
11.3.2.5.- Matriz CLAB

11.3.3.- Desarrollo del diagrama entidad-relación.
11.3.3.1.- Ejemplo Diagrama entidad relación video ABC
11.3.4.- Descomposición Funcional.
11.3.5.-Desarrollo del diagrama de dependencia de proceso.
11.3.6.- Desarrollo del diagrama de flujo de datos.
11.3.7.- Desarrollo y análisis de la Matriz entidad/Proceso.
11.3.7.1.- Ejemplo Matriz Entidad/Proceso –Video ABC
11.4.- Análisis Orientado a Objetos.
11.4.1.- Conceptos básicos para el AOO.
11.4.2.- Características del AOO.
11.4.3.- Que puede hacer con la metodología AOO.
11.4.4.- Beneficios de la utilización del AOO.
11.4.5.- Elementos del modelado del análisis
11.4.5.1.- Modelos de datos.
11.4.5.2.- Objeto de datos, atributos y relaciones.
11.4.5.3.- Cardinalidad y modalidad.
11.4.6.- Diagrama entidad relación.
11.4.6.1.- Modelo funcional.
11.4.6.2.- Diagrama de descomposición funcional.
11.4.6.3.- Diagrama de flujo de datos.
11.4.6.4.- Modelo de comportamiento.
11.4.7.- Mecanismo del análisis estructurado.
11.4.7.1.- Elaboración del diagrama entidad relación.
11.4.7.2.- Elaboración del modelo de flujo de datos.
11.4.7.3.- Elaboración del modelo de flujo de control.
11.4.7.4.- Especificaciones de control.
11.4.7.5.-Especificación del proceso.
11.4.8.- Diccionario de datos.
11.4.9.- Análisis del dominio.
11.4.10.- El proceso de análisis orientado a objeto.
11.4.10.1.- Casos de Uso.
11.4.10.2.- Especificación de un caso de uso.
11.4.10.3.- Diagrama de casos de uso.
11.4.11.- El modelo CRC.
11.4.11.1.- Colaboradores.
11.4.12.- Diagrama de Estructura.
11.4.12.1.- Descripción.
11.4.12.2.- Estructuras del diagrama.
11.4.13.- Modelo objeto-relación.
11.4.14.- El modelo objeto comportamiento.
11.4.15.- Identificación de eventos a través de los casos de uso.
11.4.15.1.- Traza de eventos
11.4.16.- Diagrama de flujo de eventos.
11.4.17.- Diagrama de comunicación

Capitulo XII Metodologías para el diseño de sistemas informáticos.

12.1.- Introducción.
12.2.- Diseño Orientado a Procesos
12.2.1.- Diseño de datos.
12.2.2.- Diseño arquitectónico
12.2.2.1. Áreas de aplicación
12.2.3.- El proceso del diseño arquitectónico.
12.2.3.1.-Flujo de transformación.
12.2.3.2.- Flujo de transacción.
12.2.4.- Análisis de las transformaciones.
12.2.4.1.- Un ejemplo.
12.2.4.2.- Pasos para el Diseño.
12.2.5.- Análisis de las Transacciones
12.2.5.1.- Un ejemplo
12.2.5.2.- Pasos del diseño.
12.2.6.- Postproceso de Diseño.
12.2.7.- Optimización del Diseño Arquitectónico.
12.2.8.- Diseño de la Interfaz.
12.2.8.1. Diseño de la Interfaz Interna y Externa.
12.2.8.2.- Diseño de la interfaz de usuario.
12.2.9.- Diseño de la Interfaz Hombre- Máquina.
12.2.9.1.- Modelos de diseño de interfaz.
12.2.9.2.- Análisis y modelado de tareas.
12.2.9.3.- Aspectos del diseño.
12.2.9.4.- Herramientas de implementación.
12.2.10.- Directrices para el diseño de interfaces.
12.2.10.1.- Interacción general.
12.2.10.2.- Visualización de la información.
12.2.10.3.- Entrada de datos.
12.2.11.- Diseño procedimental.
12.2.11.1.- Programación estructurada.
12.2.11.2.- Notación gráfica del diseño.
12.2.11.3.- Notación tabular de diseño.
12.2.11.4.- Lenguaje de diseño de programas.
12.2.11.5.- Un ejemplo de LDP.
12.3.- Diseño Orientado a Datos.
12.3.1.- Fundamentos conceptuales.
12.3.2.- Diseño de la ingeniería de la información.
12.3.2.1- Análisis de uso y distribución de datos. Guía para el análisis de uso y distribución de datos.
12.3.2.1.1- Ejemplo de uso y distribución de datos para el video ABC.
12.3.3- Definición de seguridad, recuperación y controles de revisión.
12.3.3.1- Seguridad
12.3.3.2- Recuperación
12.3.3.3- Controles de Revisión.
12.3.4.- Diagrama de Acción. Guía de desarrollo.

12.3.4.1- Ejemplo del diagrama de acción para el vídeo ABC.
12.3.5.- Definición de la estructura del menú influjo de diálogo. Guía para definir la estructura del menú y el flujo de diálogo.
12.3.5.1-Ejemplo de Estructura de Menú y el Flujo de Diálogo para el Video ABC.
12.3.6.- Plan de instalación y prueba de hw y sw
12.4.-Diseño orientado a objetos.
12.4.1.- Diseño de sistemas orientados a objetos.
12.4.2.- El enfoque convencional y el enfoque oo.
12.4.3.- Asuntos de diseño.
12.4.4.- La visión del doo.
12.4.4.1.- El método de booch.
12.4.4.2.- El método de coad y yourdon.
12.4.4.3.- El método de jacobson.
12.4.4.4.- El método de rumbaugh.
12.4.4.5.- El método de wirfs-brock.
12.4.5.- Los componentes genéricos del modelo de diseño oo.
12.4.6.- El proceso de diseño del sistema.
12.4.6.1.- El proceso de diseño del sistema.
12.4.6.2.- Concurrencia y asignación de subsistemas.
12.4.6.3.- El componente para la gestión de tareas.
12.4.6.4.- El componente para la gestión de datos.
12.4.6.5.- El componente para la gestión de recursos.
12.4.6.6.- El componente de interfaz hombre-máquina.
12.4.6.7.- Comunicación entre subsistemas.
12.4.7- El proceso de diseño de objetos.
12.4.7.1- Descripción de objeto.
12.4.7.2.- Diseño de algoritmos y estructuras de datos.
12.4.7.3.- Componentes de programas e interfaces.
12.4.8.- Patrones de diseño.
12.4.8.1.- Descripción de un patrón de diseño.
12.4.8.2.- Uso de patrones en el diseño.
12.4.9.- Programación orientada a objetos.
12.4.10.- Resumen.
12.4.11.- Pruebas orientadas a objetos.
12.4.12.- Modelos de pruebas aoo y doo.
12.4.12.1.- Diseño de sistemas orientados a objetos.
12.4.12.2.- Consistencia de los modelos de aoo y doo.
12.4.12.3.- Estrategias de pruebas orientadas a objetos.
12.4.12.4.- Prueba de unidad en el contexto oo.
12.4.12.5.- Prueba de integración en el contexto oo.
12.4.12.6.- Prueba de validación de un contexto oo.
12.4.13.- diseño de casos de prueba para software oo.
12.4.13.1.- Implicaciones de los conceptos oo para el diseño de casos de prueba.

12.4.13.2.- Implicaciones de los conceptos oo para el diseño de casos de prueba.
12.4.13.3.- Pruebas basadas en fallos.
12.4.13.4.- El impacto de la programación oo en la realización de pruebas.
12.4.13.5.- Casos de prueba y jerarquía de clases.
12.4.13.6.- Diseños de pruebas basadas en escenarios.
12.4.13.7.- Probando la estructura superficial y la estructura profunda.
12.4.14.- Métodos de prueba aplicables al nivel de clase.
12.4.14.1.- Métodos de prueba aplicables al nivel de clase.
12.4.14.2.- Pruebas de partición al nivel de clase.
12.4.14.3.- Pruebas derivadas de modelos de comportamiento.
12.4.15.- Resumen.

Capitulo XIII Otros aspectos del diseño.

13.1.- Introducción
13.2.- Interacción hombre-máquina.
13.3.- Diseño de la interfaz hombre maquina
13.4.- Principios para el diseño
13.5.- Tendencias en el diseño
13.5.1- Estilos de aprendizaje.
13.5.2.- Factores que influyen en los procesos cognitivos.
13.6.- Diseño de interfaz de usuario.
13.6.1.- Principios de la interfaz de usuario
13.6.2.- Interacción del usuario.
13.6.3.- Presentación de la información.
13.6.4- Diseño del sistema de ayuda
13.6.5.- Evaluación de la interfaz de usuario.
13.6.6.- Técnicas simples para la evaluación.
13.7.- Diseño de entrada efectiva.
13.7.1.- Diseño de formas.
13.7.2.- Diseño de pantallas.
13.7.2.1.- ¿Cómo mantener una pantalla simple?
13.7.2.2.- ¿Cómo mantener las pantallas consistentes?
13.7.2.3.- ¿Cómo facilitar el movimiento?
13.7.3 ¿Cómo diseñar pantallas atractivas?
13.8.- Diseño de salidas efectivas.
13.8.1.- ¿Cómo diseñar salidas que sirvan al propósito deseado?
13.8.2.- ¿Cómo diseñar salidas que se ajusten al usuario?
13.8.3.- ¿Cómo entregar la salida adecuada?
13.8.4.- ¿Cómo asegurarse que la salida se encuentre donde se necesite?
13.8.4.1.- Entrega de salida a tiempo.
13.8.4.2.- Selección del método adecuado.
13.8.4.3.- Factores a considerar en la selección de la tecnología de salida

13.8.4.4.- Diseño de salidas impresas
13.8.4.5.- Convenciones para el diseño de reporte.
13.9.- Diseño para la captura de datos
13.9.1.- La codificación efectiva.
13.9.2.- ¿Por qué hacer el seguimiento de algo?
13.9.3.- Códigos de clasificación
13.9.3.1.- Código de secuencia en bloque
13.9.3.2.- Códigos de subconjuntos de dígitos significativos
13.10.- Captura de datos efectivas y eficientes.
13.10.1.- Selección del método de captura.
13.10.2.- Aseguramiento de la calidad de los datos por medio de la validación de la entrada.
13.10.3- Validación de las transacciones de entrada.
13.10.4.- Validación de los datos de entrada.

Capitulo XIV Herramientas CASE

14.1.- Introducción.
14.2.- Herramientas Case.
14.5.- Componentes de una herramienta case.
14.6.- Estructura general de una herramienta case.
14.7.- Estado Actual.
14.8.- Integración de las herramientas case en el futuro.
14.9.- Clasificación de las herramientas case.
14.10.- Características Deseables De Una Case.
14.11.- Factores asociados a la implantación de las herramientas case.
14.12.- Conclusión.
14.13.- Bibliografía.

Cuarta Parte.

Documentación, pruebas y mantenimiento de los sistema informáticos.

Capitulo XV Documentación.

15.- Introducción.
15.1.- Importancia De La Documentación De Sistemas.
15.2.- Estandarización y normalización.
15.2.2.- Normalización.
15.2.3.- Teoría general de los manuales de documentación.
15.3.-Manual Administrativo.
15.3.1.- Resumen Administrativo.
15.3.2.- Planteamiento.
15.3.3.-Objetivos Del Sistema.

15.3.4.- Entradas Del Sistema (Información A Captar).
15.3.5.- Salidas Del Sistema (Resultados A Obtener).
15.3.5.- Diagramación General Del Sistema.
15.3.6.- Explicaciones De Las Fases Del Sistema.
15.3.7.- Requerimientos Del Sistema.
15.3.8.- Estimación de la fecha probable de implementación del sistema.
15.4.- Manual De Usuario.
15.4.1.- Objetivos.
15.4.2.- Pasos a seguir para definir como desarrollar el manual de usuario.
15.4.2.1.- Importancia Del Manual De Usuario.
15.5.- Manual De Captación.
15.5.1.- Objetivos.
15.5.2.-Contenido.
15.5.3.- Diagramas De Pantalla.
15.5.4.-Explicación Genérica De Las Fases Del Sistema.
15.5.5.- Equipo Utilizado Para La Captación.

Capitulo XVI Garantía de la calidad del software.

16.- Introducción.
16.1.- Marco histórico de las normas de calidad.
16.2.- Importancia y necesidad de la adopción de las normas de calidad.
16.3- Origen de la calidad del software.
16.4.- El impacto de la calidad en las Organizaciones.
16.5.- Estándares y garantía de la calidad.
16.6.- El aprendizaje Organizacional.
16.7.- La gestión de calidad en las Organizaciones y su formación.
16.8.- Las principales motivaciones.
16.9.- Sugerencias al iniciar el camino.
16.10.- Cambios ocasionados en el ambiente y la cultura organizacional.
16.11.- Papel de la dirección en el proceso.
16.12.- Beneficios captados en la utilización de la calidad en el producto software.
16.13.- Desafíos que plantea el proceso de gestión de calidad.
16.14.- Definiciones relevantes.
16.14.1.- Calidad.
16.14.2.- Política de calidad.
16.14.3.- Sistema de calidad.
16.14.4.- Control de la calidad.
16.14.5.- Proceso.
16.14.6.- Procedimiento.
16.15.- Elementos de las Normas ISO 9001.
16.15.1.- Gestión ejecutiva.
16.15.2.- Gestión de los procesos.
16.15.2.1.- Gestión de los procesos.
16.15.2.2.- Recursos Humanos.
16.15.2.3.-Relación con el cliente.

16.15.2.4.- Procesos relacionados con el producto, tales como:
16.15.2.5.- Medida.
16.15.2.6.- Evaluación y mejora.
16.16.- Plan de garantía de calidad.
16.17.- Calidad del proceso.
16.18.- Calidad del servicio.
16.19.- Calidad del producto.
16.20.- Procedimientos y responsabilidades del Plan de Garantía de Calidad.
16.21.- Atributos de calidad del software.
16.22.- Modelo de la calidad de software.

Capitulo XVII Mantenimiento del software

17.- Introducción.
17.1.- Definición de mantenimiento.
17.1.1.- Mantenimiento correctivo.
17.1.2.- Mantenimiento adaptativo.
17.1.3.- Mantenimiento perfectivo.
17.1.4.- Mantenimiento preventivo.
17.2.- Actividades de mantenimiento.
17.3.- Costos del mantenimiento.
17.4.- Dificultades del mantenimiento.
17.4.1.- Código heredado.
17.5.- Problemas de mantenimiento.
17.5.1.- Efectos secundarios del mantenimiento.
17.5.2.- Efectos secundarios sobre el código.
17.5.4.- Efectos secundarios sobre la documentación.
17.5.5.- Soluciones al problema del mantenimiento.
17.6.- Gestión de calidad.
17.6.1.- Gestión estructurada del manteniendo.
17.6.2.- Organización del equipo humano.
17.7.- Documentación de cambio.
17.8.- Soluciones técnicas.
17.9.- Mantenibilidad.
17.10.- Actividades y tareas del mantenimiento de software.
17.10.1.- Análisis de problema y modificaciones.
17.10.2.- Implementación de las modificaciones.
17.10.3.- Revisión y aceptación del mantenimiento.
17.10.4.- Migración.
17.10.5.- Retirada del software.
17.11.- Descripción del proceso de mantenimiento.
17.11.1.- Implementación del proceso.
17.11.2.- Análisis de problema y modificaciones.
17.11.3.- Implementación de la modificación.
17.11.4.- Revisión y aceptación del mantenimiento.
17.11.5.- Migración.

17.12.- Mantenibilidad del software.
17.12.1.- Medidas externas de la mantenibilidad.

Quinta Parte.

Evaluación de los proyectos informáticos. Auditoría informática.

Capitulo XVIII Auditoría informática.

18.-Introducción.
18.1.- Concepto de auditoria informática.
18.2.- Clases de auditoria.
18.2.1.- Concepto de consultora.
18.2.2.- Control interno informático.
18.2.3.- Tipos de controles internos.
18.3.- Implementación de los controles internos informáticos.
18.3.1.- Estructura de los controles internos dentro de la empresa.
18.3.1.1.- Controles generales organizativos.
18.3.1.2.- Controles de desarrollo, adquisición y mantenimiento de sistemas de información.
18.3.1.3.- Explotación y mantenimiento.
18.3.1.4.- Controles de explotación de sistemas de información.
18.3.1.5.- Controles para usar de manera efectiva los recursos en la computadora.
18.3.1.6.- Controles de aplicaciones.
18.3.1.7.- Control especifico a determinadas tecnologías.
18.4.- Tipo de metodología aplicada a la ingeniería de software.
18.4.- Metodología cuantitativa.
18.4.1.- Metodología cualitativa subjetiva.
18.5.- PRIMA (Prevención de riesgos informáticos con metodología abierta).
18.6.- El plan de contingencia.
18.6.1.- Fase I. Análisis y diseño.
18.6.2.- Fase II. Desarrollo del plan.
18.6.3.- Fase III. Pruebas y mantenimiento.
18.7.- Tipos de evidencias
18.8.- El informe de auditoria.
18.8.1.- Resultados (informes largo y otros informes).
18.8.2.- Informes previos.
18.8.3.- Fecha del informe.
18.8.4.- Identificación y firma del auditor.
18.8.5.- Distribución del informe.
18.9.- Perfil del auditor.
18.11.- Consideraciones para la contratación de una auditoria externa.
18.10.- Fuentes de información de la auditoria de sistemas informáticos.

Capitulo XIX Principales áreas de la auditoría informática.

19.- Introducción.
- 19.1.- Auditoria de seguridad física.
  - 19.1.1.- Desarrollo.
    - 19.1.1.1.-Conclusión.
- 19.2.- Auditoria de la dirección.
  - 19.2.1.- Desarrollo.
  - 19.2.2.- Conclusión.
- 19.3.- Auditoria de técnicas de sistemas.
  - 19.3.1.- Desarrollo.
  - 19.3.2.- Conclusiones.
- 19.4.- auditoria de la explotación de sistemas de información.
  - 19.4.1.- Desarrollo.
    - 19.4.1.1.- Planificación Estratégica.
    - 19.4.1.2.- Procedimientos objetivos.
    - 19.4.1.3.- Planificación Administrativa.
    - 19.4.1.4.- Planificación técnica.
  - 19.4.2.- Conclusión.
- 19.5 Auditoria de desarrollo.
  - 19.5.1.- Desarrollo.
  - 19.5.2.- Análisis de los requisitos del sistema.
    - 19.5.2.1.- Especificaciones funcionales del sistema.
  - 19.5.3.- Auditoria de la fase de diseño.
  - 19.5.4.- Auditoria de la fase de construcción.
  - 19.5.4.- Auditoria del Desarrollo de los procedimientos de usuarios.
  - 19.5.5.- Auditando la fase de implantación.
  - 19.5.6.- Conclusión.
- 19.6.- Auditoria de mantenimiento de software.
  - 19.6.1.- Desarrollo.
  - 19.6.2.- Modelo de mantenimiento COCOMO.
    - 19.6.2.1.- Esfuerzo de desarrollo.
    - 19.6.2.2.- Esfuerzo de mantenimiento.
    - 19.6.2.3.- Función de la Mantenibilidad.
  - 19.6.3.- Conclusión.
- 19.7.- Auditoria de la calidad de sistemas de información.
  - 19.7.1.- Desarrollo.
  - 19.7.2.- Conclusión.
- 19.8.- Auditoria de la seguridad informática.
  - 19.8.1.- Desarrollo.
  - 19.8.2.- Conclusión.
- 19.9.- Auditoria de base de datos.
  - 19.9.1.- Desarrollo.
    - 19.9.1.1.- Estudio y plan de trabajo.
    - 19.9.1.2.- Concepción de la DB y la selección del equipo.
    - 19.9.1.3.- Diseño de carga.
    - 19.9.1.4.- Explotación y mantenimiento de los datos.

19.9.1.5.- Revisión post-implantación.
19.9.2.- Conclusión.
19.10.- Auditoria de redes informáticas.
19.10.1.- Desarrollo.
19.10.2.- Auditoria de los componentes físicos de la red.
19.10.3.- Auditoria de la red lógica.
19.10.4.- Conclusión.
19.11.- Auditoria jurídica para el entorno informático.
19.11.1.- Desarrollo.
19.11.2.- Conclusión.

Sexta Parte.

Tendencias de la Ingeniería de Software.

Capitulo XX: Desarrollo Ágil de Software

20.- Desarrollo Ágil.
20.1.- ¿Qué es el desarrollo Ágil?.
20.2.- Cómo lograr el desarrollo Ágil.
20.3.- Estrategia para el desarrollo Ágil.
20.3.1.- Estrategia general para el desarrollo ágil.
20.3.2.-Cuatro dimensiones de la velocidad de desarrollo.
20.3.2.1. Personas.
20.3.2.2.- Proceso.

Capitulo XXI Ingeniería del Software Libre.

21.- Introducción.
21.1.- Metodología próxima al software libre: eXtreme Programming.
21.1.1.- Planificación.-
21.1.2.- Diseño.
21.1.3.- Codificación.
21.1.4.- Pruebas.
21.1.5.- Conclusiones.
21.2.- Metodología Clásica: Métrica V3.
21.2.1.- Planificación de sistemas de información.
21.2.1.1.- Inicio del plan de sistemas de información.
21.2.1.2.- Definición y organización del plan.
21.2.1.3.- Estudio de l información relevante.
21.2.1.4.- Identificación de requisitos.
21.2.1.5.- Estudio de los sistemas de información actuales.
21.2.1.6.- Diseño del modelo del sistema de información.
21.2.1.7.- Definición de la arquitectura tecnológica.

21.2.1.8.- Definición del plan de acción.
21.2.1.9.- Revisión y aprobación del plan.
21.2.2.- Desarrollo de sistemas de información.
21.2.2.1.- Estudio de viabilidad.
21.2.2.2.- Análisis del sistema de información.
21.2.2.3.- Diseño del sistema de información.
21.2.2.4.- Construcción del sistema de información.
21.2.2.5.- Implantación y aceptación del sistema.
21.2.3.- Mantenimiento de sistemas de información.
21.3.- Control de calidad y pruebas.
21.3.1.- Términos Comunes.
21.3.2.- Principios de la comprobación de software.
21.3.3. Técnicas manuales de comprobación de software.
21.3.4. Técnicas automáticas de comprobación de software.
21.3.4.1. White-box testing
21.3.4.2. Black-box testing
21.3.4.3. Unidades de comprobación.
21.3.5. Sistemas de control de errores.
21.3.5.1. Reportado en errores.
21.3.5.2. Anatomía de un informe de error.
21.3.5.3. Ciclo de vida de un error.
21.3.5.4. Bugzilla
21.3.5.5. Gnats
21.4. Control de versiones
21.4.1. Sistemas de control de versiones
21.4.1.1. Algunos términos comunes.
21.4.1.2. Características de los sistemas de control de versiones.
21.4.1.3. Principales sistemas de control de versiones.
21.4.1.4 Sistemas de compartición
21.5. Sistemas de creación de documentación.
21.5.1. Documentación libre: estándares y automatización.
21.5.1.1. La documentación del software.
21.5.1.2. El problema de documentar software libre.
21.5.1.3. Licencias libres para la documentación.
21.5.1.4. Formatos libres y propietarios.
21.5.1.5. Herramientas de control y administración de versiones.
21.5.2. Creación de páginas de manual.
21.5.2.1. Secciones de las páginas de manual.
21.5.2.2. Camino de búsqueda de páginas man.
2.1.5.2.3. Jerarquía de capítulos de las páginas man.
21.5.2.4. Generación de páginas man usando herramientas estándares.
21.5.2.5. Generación de páginas de manual usando perlpod.
21.5.3. SGML.
21.5.3.1. Documentación en formato html.
21.5.3.2. Documentación en formato Docbook.
21.5.3.3. Doxygen. Documentación de código fuente.

3.- CONCLUSIONES.

El plan temático de la asignatura ISW esta organizado de tal manera que permite a los estudiantes quedar aptos para la realización del análisis, especificación, diseño, desarrollo, pruebas, y mantenimiento de los proyectos informáticos de desarrollo de software; a pesar de esto más del ochenta por ciento (80%) del alumnado manifestó estar inconforme – en alguna manera - con el contenido abarcado en la asignatura, en comparación a las expectativas que ellos tenían. (Ver anexo 4)

Partiendo de la hipótesis de que [la consolidación teórica de los diferentes planteamientos de ISW, contribuirá en gran medida al proceso de enseñanza – aprendizaje de esta asignatura][1], y la opinión de los alumnos (encuestados) a cerca del mejor medio bibliográfico utilizado (Ver anexo 4), se desarrolló e implantó una aplicación informática (página Web) que sirve de apoyo a este proceso.

Para llegar a esta meta, se estudió el plan temático actual de la asignatura, y diferentes bibliografías que abarcan los diferentes temas relacionados con ISW, así como planes temáticos de otras universidades (extranjeras). Además se contó con el valioso apoyo de los docentes que imparten la asignatura en la UNI.

La ISW, como una asignatura dentro de la UNI, demanda especial atención, ya que la investigación y el desarrollo de técnicas y métodos de ISW son constantes y suelen suponer interesantes avances en la resolución de problemas de desarrollo de software. Sin embargo, es habitual que en la práctica diaria profesional no se incluya ninguna de las recomendaciones más elementales de la ingeniería de software.

Por último, queremos señalar que la ingeniería de software tiene respuestas (aunque hay que seguir trabajando en refinarlas y ponerlas en práctica) correspondientes con los avances y nuevos paradigmas de desarrollo de proyectos de software; y es por eso que el proceso enseñanza – aprendizaje de esta materia debe de estar en constante evolución. Por otra parte, hace falta realizar un esfuerzo para que se garantice la aplicación de las técnicas de aseguramiento de calidad y de adaptación de las técnicas de análisis y diseño. Son dos campos donde la ingeniería del software, como asignatura dentro de la UNI debe fortalecerse para solventar muchos de los problemas diarios de los profesionales del software

Es por eso que esperamos que este trabajo deba considerarse un aporte inicial a tan importante asignatura.

[1] Tomada del Protocolo Monográfico “Ingeniería de Software. Una perspectiva Didáctica”.

4.- RECOMENDACIONES FINALES.

1.- Poner en línea el sitio web, en la página de la UNI, ubicando el enlace principal en la página de la carrera de ingeniería en computación.

2.- Promoción del sitio web, como una herramienta auxiliar a la clase, por parte de los docentes de la universidad.

3.- Garantizar la constante actualización del sitio, mediante la formación de grupos de estudiantes destacados en dicha asignatura en conjunto con profesores que la impartan. Estos últimos serán los encargados de revisar semestralmente los temas de actualidad de ISW para luego incluirlos en el sitio web.

4.- Sugerir a los estudiantes de la carrera, la elaboración de este tipo de proyecto en otras asignaturas que requieran recopilación de una base bibliográfica, y así obtener una consolidación de enfoques o planteamientos enmarcados dentro de un Plan Temático.

5.- Tomar como guía el presente documento y código fuente para futuros desarrollos que estén orientados a la implementación de aplicaciones similares que permita a los usuarios el fácil acceso a una base bibliográfica amplia enmarcada dentro de un plan temático.

6.- Implementación de un medio de interacción en línea entre usuarios para fortalecer la “cultura” de Ingeniería de Software (chat y foros).

7.- Publicar encuestas dirigidas a los usuarios para obtener información que sirva de retroalimentación a la sitio.

ANEXOS

ANEXO 1

Pilares de la Educación.

1.1.- Aprender a conocer.

Proceso infinito que requiere, sobre todo, de estrategias de aprendizaje: de búsqueda de información a través de los recursos analógicos o digitalizados; el desarrollo de habilidades de lectura que ayuden en la selección y rapidez de la comprensión de lo leído, a la distinción de lo esencial, al poder de la síntesis, a la asimilación crítica de lo leído y a su uso en la solución de problemas con un pensamiento creativo, humanista y el estímulo a la curiosidad permanente.

1.2.- Aprender a hacer.

Aprendizaje para el cambio, que estimule la capacidad emprendedora, la iniciativa, la innovación. Vínculo teoría-práctica. Da sentido al quehacer didáctico, que no debe confundirse son la rutina o la resistencia al cambio.

1.3.- Aprender a ser.

Existir en la correspondencia con los principios fundamentales de una ética humana, aplicados a cada profesión y a todos los ámbitos de la actividad humana. El respeto al desarrollo humano sustentable, la preservación de la identidad cultural, es un criterio que conjuga autonomía y responsabilidad.

1.4.- Aprender a vivir juntos, aprender a vivir con los demás.

Emana no solo del carácter social del hombre, sino de las peculiaridades del trabajo profesional actual, en grupos multidisciplinarios e interdisciplinarios. Plantea una tarea especial: la habilidad de comunicarse con las personas, ajustándose al medio que se usa. Significa a aprender a ocupar diferentes puestos en el colectivo y que el desarrollo pleno de la individualidad coincida con la participación en la vida en sociedad.

1.5.- Aprender a desaprender.

Dejar en el cajón de lo obsoleto, todo aquello que sirvió y ya no es válido, según palabras de M. A. Escotet citadas por C. Tünnerman (1996) a lo que conviene

agregar, no con un sentido nihilista, sino con el poder de la dialéctica, de superación necesaria de momentos anteriores para responder a las nuevas exigencias.

ANEXO 2

Clasificación General de los métodos de enseñanza.

1.1.- En cuanto a la forma de razonamiento.

1.1.1.- Método Deductivo.

Es cuando el asunto estudiado procede de lo general a lo particular.

1.1.2.- Método Inductivo.

Es cuando el asunto estudiado se presenta por medio de casos particulares, sugiriéndose que se descubra el principio general que los rige.

1.1.3.- Método Analógico o Comparativo.

Cuando los datos particulares que se presentan permiten establecer comparaciones que llevan a una conclusión por semejanza.

1.2- En cuanto a la coordinación de la materia.

1.2.1.- Método Lógico.

Es cuando los datos o los hechos son presentados en orden de antecedente y consecuente, obedeciendo a una estructuración de hechos que van desde lo menos hasta lo más complejo.

1.2.2.- Método Psicológico.

Es cuando la presentación de los métodos no sigue tanto un orden lógico como un orden más cercano a los intereses, necesidades y experiencias del educando.

1.3.- En cuanto a la concretización de la enseñanza.

1.3.1.- Método Simbólico o Verbalístico.

Se da cuando todos los trabajos de la clase son ejecutados a través de la palabra. El lenguaje oral y el lenguaje escrito adquieren importancia decisiva, pues son los únicos medios de realización de la clase.

1.3.2.- Método Intuitivo.

Se presenta cuando la clase se lleva a cabo con el constante auxilio de objetivaciones o concretizaciones, teniendo a la vista las cosas tratadas o sus sustitutos inmediatos.

1.4.- En cuanto a la sistematización de la materia.

1.4.1.- Métodos de Sistematización.

1.- Rígida: Es cuando el esquema de a clase no permite flexibilidad alguna a través de sus ítems lógicamente ensamblados, que no dan oportunidad de espontaneidad alguna al desarrollo del tema de la clase.

2.- Semirígida: Es cuando el esquema de la lección permite cierta flexibilidad para una mejor adaptación a las condiciones reales de la clase y del medio social al que la escuela sirve.

1.4.2.- Método Ocasional.

Se denomina así al método que aprovecha la motivación del momento, como así también los acontecimientos importantes del medio. Las sugestiones de los alumnos y las ocurrencias del momento presente son las que orientan los temas de las clases.

1.5.- En cuanto a las actividades de los alumnos.

1.5.1.- Método Pasivo.

Se le denomina de este modo cuando se acentúa la actividad del profesor, permaneciendo los alumnos en actitud pasiva y recibiendo los conocimientos y el saber suministrado por aquél, a través de: Dictados, Lecciones marcadas en el

libro de texto, que son después reproducidas de memoria, Preguntas y respuestas, con obligación de aprenderlas de memoria, Exposición Dogmática

1.5.2.- Método Activo.

Es cuando se tiene en cuenta el desarrollo de la clase contando con la participación del alumno. La clase se desenvuelve por parte del alumno, convirtiéndose el profesor en un orientado, un guía, un incentivador y no en un transmisor de saber, un enséñate.

1.6.- En cuanto a la globalización de los conocimientos.

1.6.1.- Método de Globalización.

Es cuando a través de un centro de interés las clases se desarrollan abarcando un grupo de disciplinas ensambladas de acuerdo con las necesidades naturales que surgen en el transcurso de las actividades.

1.6.2.- Método no globalizado o de Especialización.

Este método se presenta cuando las asignaturas y, asimismo, parte de ellas, son tratadas de modo aislado, sin articulación entre sí, pasando a ser, cada una de ellas un verdadero curso, por la autonomía o independencia que alcanza en la realización de sus actividades.

1.6.3.- Método de Concentración.

Este método asume una posición intermedia entre el globalizado y el especializado o por asignatura. Recibe también le nombre de *método por época* (o enseñanza epocal). Consiste en convertir por un período una asignatura en materia principal, funcionando las otras como auxiliares. Otra modalidad de este método es pasar un período estudiando solamente una disciplina, a fin de lograr una mayor concentración de esfuerzos, benéfica para el aprendizaje.

1.7.- En cuanto a la relación entre el profesor y el alumno.

1.7.1.- Método Individual.

Es el destinado a la educación de un solo alumno. Es recomendable en alumnos que por algún motivo se hayan atrasado en sus clases.

1.7.2.- Método Recíproco.

Se llama así al método en virtud del cual el profesor encamina a sus alumnos para que enseñen a sus condiscípulos.

1.7.3.- Método Colectivo.

El método es colectivo cuando tenemos un profesor para muchos alumnos. Este método no sólo es más económico, sino también más democrático.

1.8.- En cuanto al trabajo del alumno.

1.8.1.- Método de Trabajo Individual.

Se le denomina de este modo, cuando procurando conciliar principalmente las diferencias individuales el trabajo escolar es adecuado al alumno por medio de tareas diferenciadas, estudio dirigido o contratos de estudio, quedando el profesor con mayor libertad para orientarlo en sus dificultades.

1.8.2.- Método de Trabajo Colectivo.

Es el que se apoya principalmente, sobre la enseñanza en grupo. Un plan de estudio es repartido entre los componentes del grupo contribuyendo cada uno con una parcela de responsabilidad del todo. De la reunión de esfuerzos de los alumnos y de la colaboración entre ellos resulta el trabajo total. Puede ser llamado también Método de Enseñanza Socializada.

1.8.3.- Método Mixto de Trabajo.

Es mixto cuando planea, en su desarrollo actividades socializadas e individuales. Es, a nuestro entender, el más aconsejable pues da oportunidad para una acción socializadora y, al mismo tiempo, a otra de tipo individualizador.

1.9.- En cuanto a la aceptación de lo enseñado.

1.9.1- Método Dogmático.

Se le llama así al método que impone al alumno observar sin discusión lo que el profesor enseña, en la suposición de que eso es la verdad y solamente le cabe absorberla toda vez que la misma está siéndole ofrecida por el docente.

1.9.2- Método Heurístico.

Del griego heurístico = yo encuentro. Consiste en que el profesor incite al alumno a comprender antes de fijar, implicando justificaciones o fundamentos lógicos y teóricos que pueden ser presentadas por el profesor o investigadas por el alumno.

1.10.- En cuanto al abordaje del tema de estudio.

1.10.1.- Método Analítico.

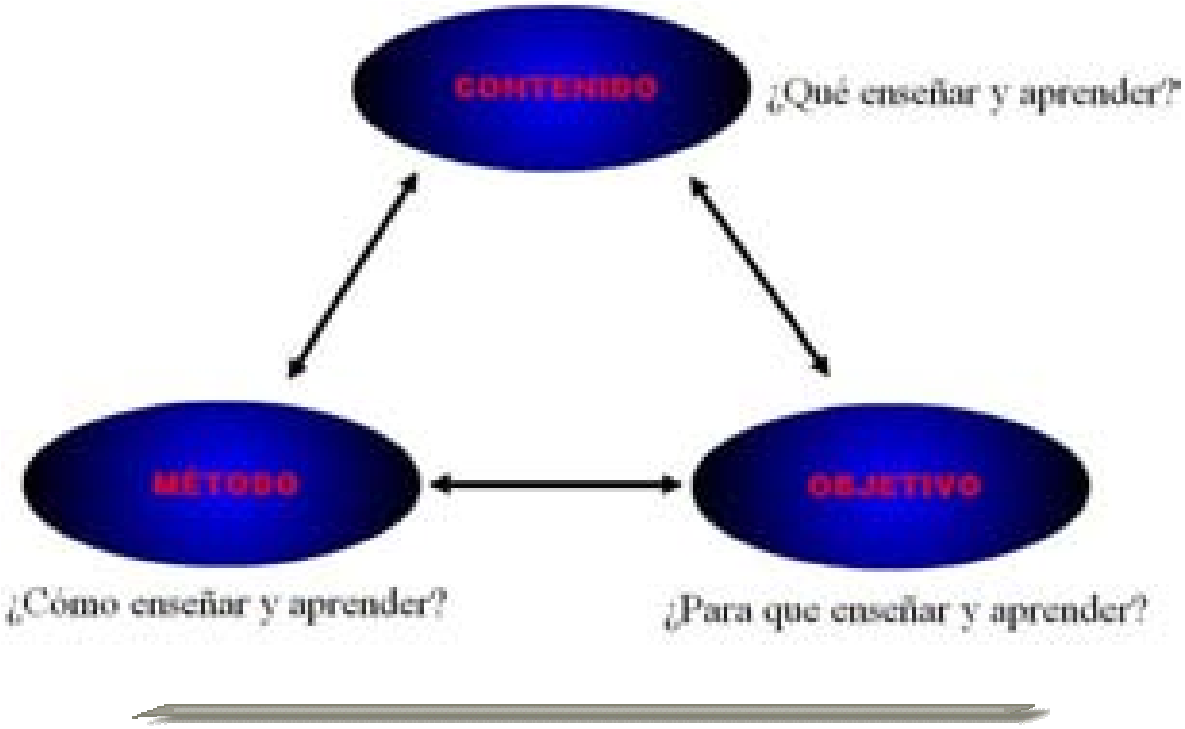
Este método implica el análisis (del griego análisis, que significa descomposición), esto es la separación de un tono en sus partes o en sus elementos constitutivos. Se apoya en que para conocer un fenómeno es necesario descomponerlo en sus partes.

1.10.2.- Método Sintético.

Implica la síntesis (del griego synthesis, que significa reunión), esto es, unión de elementos para formar un todo.

ANEXO 3

Relación entre las categorías didácticas objetivo – contenido – método.



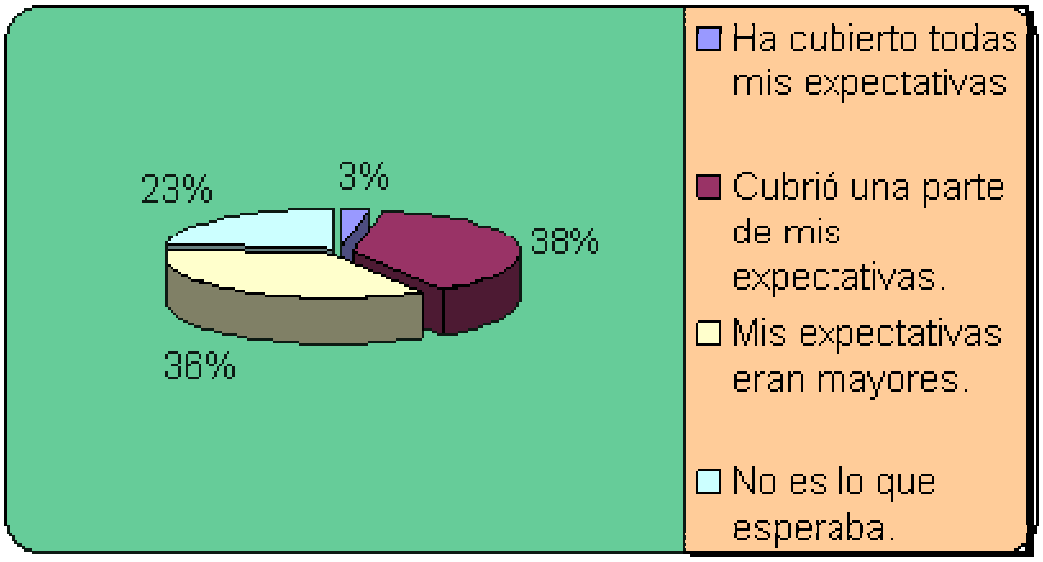
ANEXO 4

Encuesta realizada a estudiantes que cursaban o habían cursado la asignatura ingeniería de software.

1.- ¿El contenido de la asignatura, ha llenado las expectativas que tenías de la clase?

a.- Ha cubierto todas mis expectativas.
b.- Cubrió una parte de mis expectativas.
c.- Mis expectativas eran mayores.
d.- No es lo que esperaba.

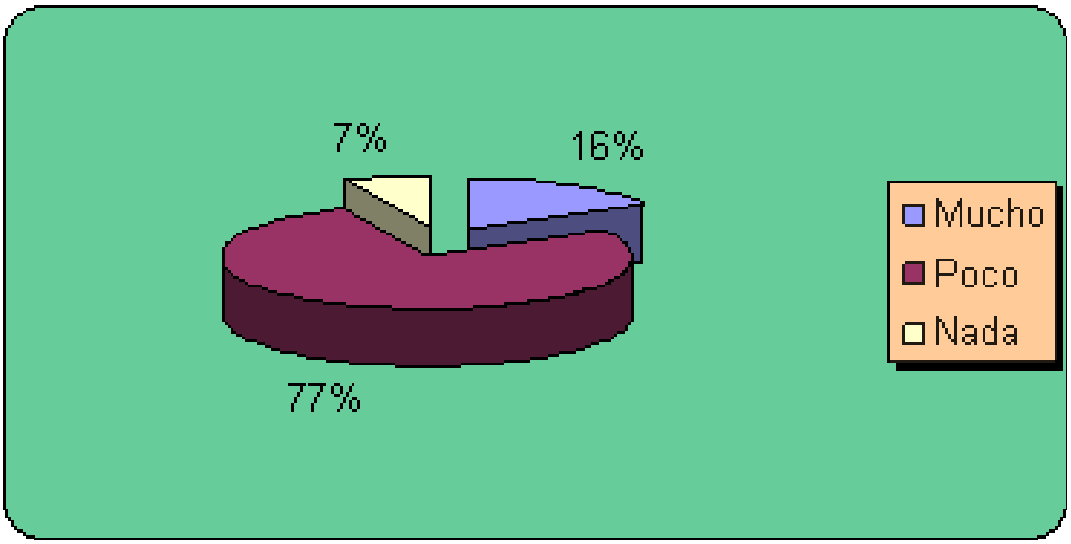
Categoría	# Respuesta	%
a	2	2,73972603
b	28	38,3561644
c	26	35,6164384
d	17	23,2876712
Total	73	100



2.- ¿Presentaste algún grado de dificultad al cursar la asignatura Ingeniería de Software?

a.- Mucho
b.- Poco
c.- Nada

Categoría	# Respuesta	%
Mucho	12	16,438356
Poco	56	76,712329
Nada	5	6,8493151
Total	73	100



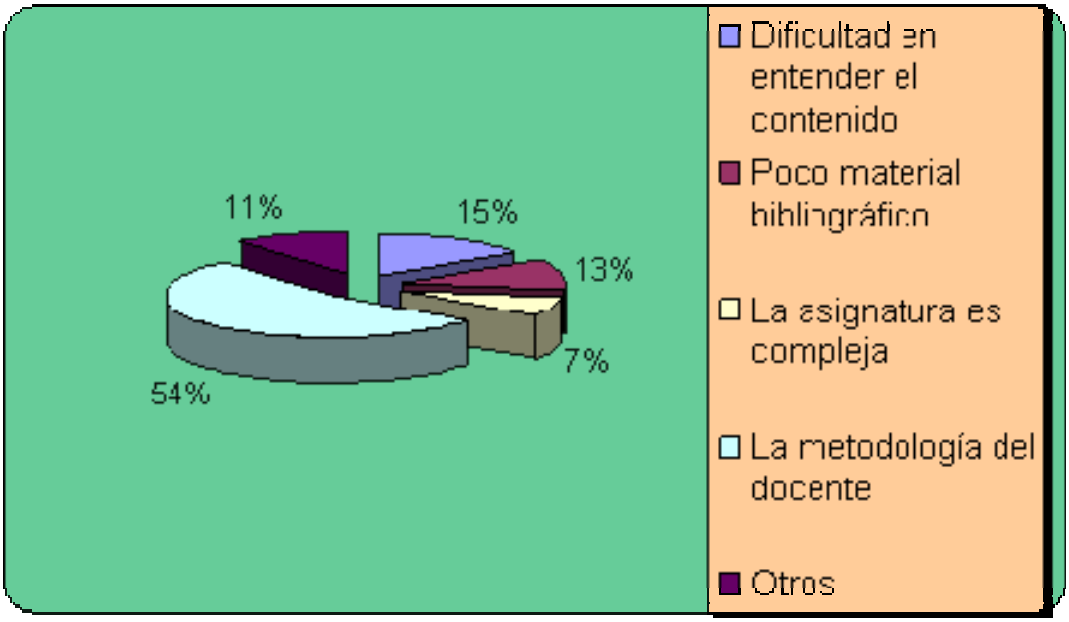
3.- Sí presentaste algún grado de dificultad al cursar la materia, ¿Cuál crees que fue la causa? (En caso de que no haya presentado dificultad omita esta pregunta y pase a la siguiente)

a.- Dificultad en entender el contenido.
b.- Poco material bibliográfico
c.- La asignatura es compleja
d.- La metodología del docente
e.-

Otros_____

_____

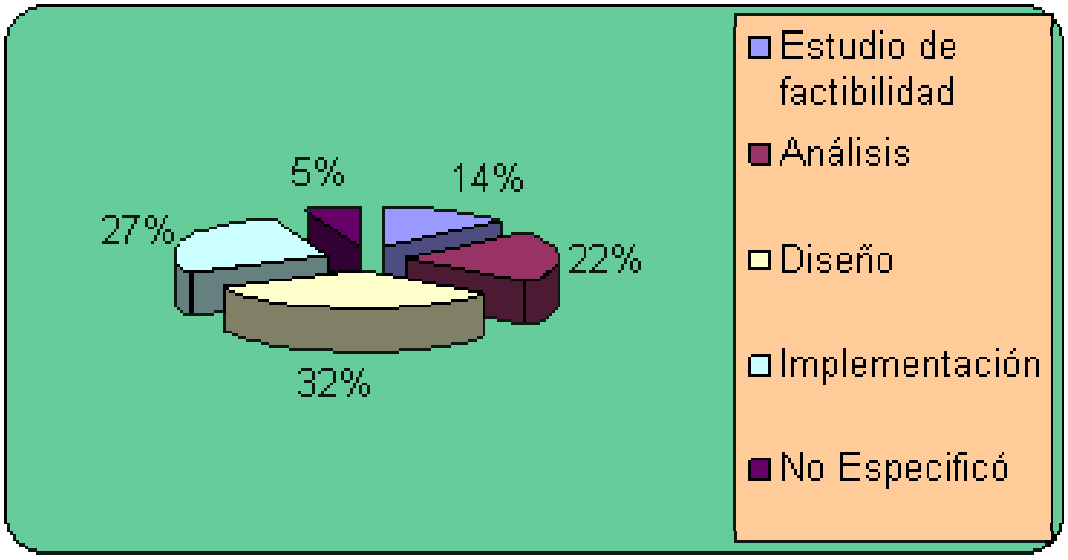
Categoría	# Respuesta	%
a	11	15,068493
b	9	12,328767
c	5	6,8493151
d	39	53,424658
e	8	10,958904
Total	72	98,630137



4.- ¿En cuales de las siguientes unidades temáticas de la asignatura presentaste dificultades?

Estudio de factibilidad
a.- Análisis
b.- Diseño
c.- Implementación

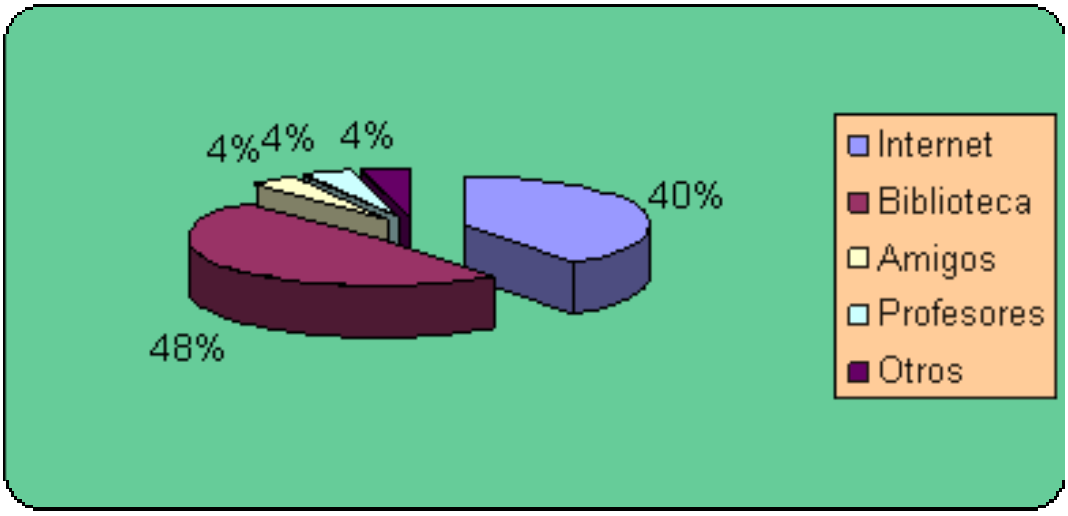
Categoría	# Respuesta	%
a	10	13,69863
c	16	21,917808
b	23	31,506849
d	20	27,39726
No Especificó	4	5,4794521
Total	73	100



5.- ¿Cuáles fueron los medios bibliográficos que utilizaste para la asignatura?

a- Internet
b- Biblioteca
c- Amigos
d- Profesores
e- Otros _____

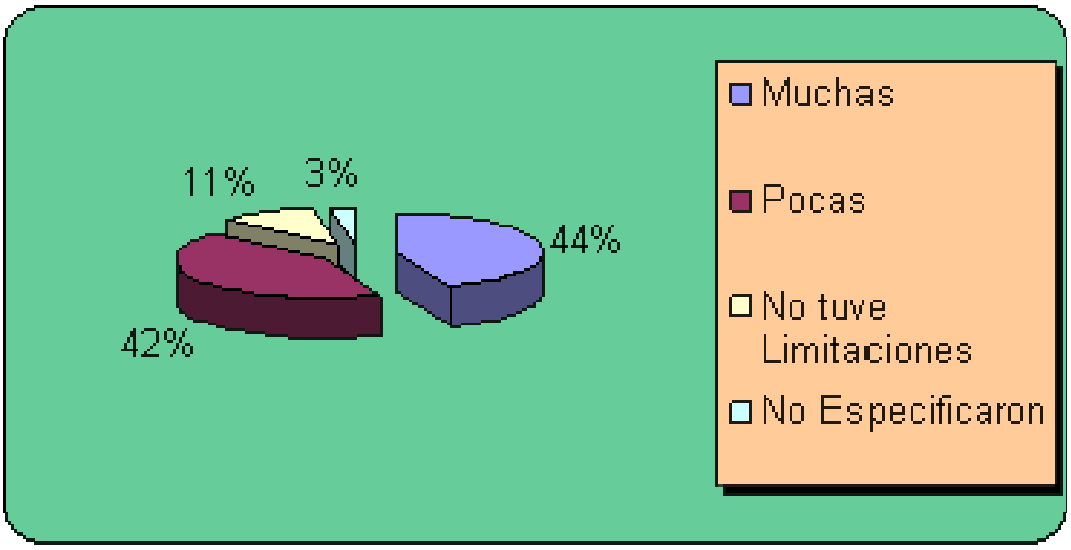
Categoría	# Respuesta	%
a	29	39,726027
b	35	47,945205
c	3	4,109589
d	3	4,109589
e	3	4,109589
Total	73	100



6.- En cuanto a limitaciones bibliográficas, podrías decir que fueron:

a.- Muchas
b.- Pocas
c.- No tuve limitaciones

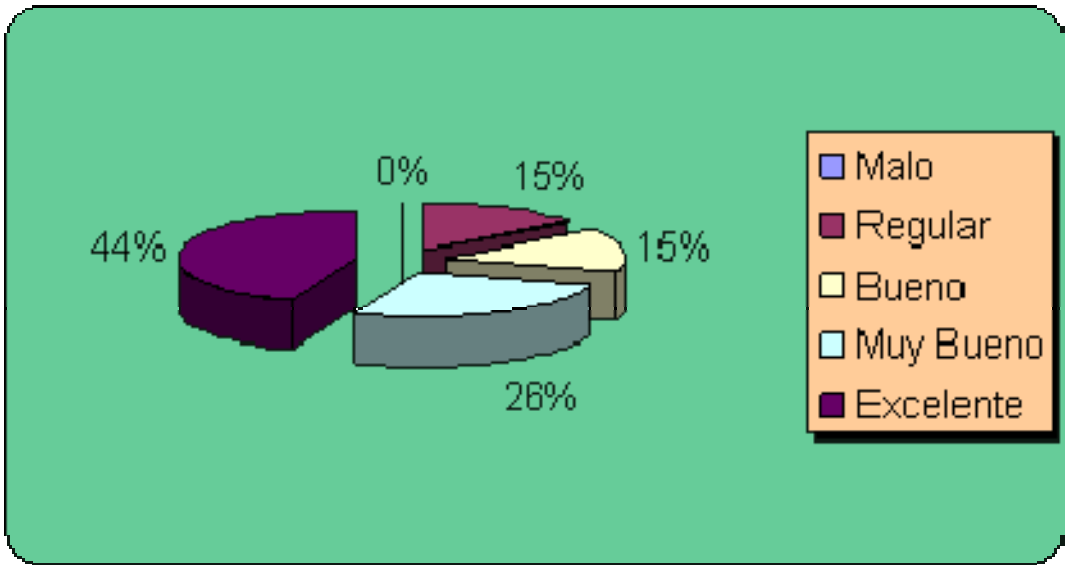
Categoría	# Respuesta	%
Muchas	32	43,835616
Pocas	31	42,465753
No tuve Limitaciones	8	10,958904
No Especificaron	2	2,739726
Total	73	100



7.- Si tuvieras que evaluar el considerar crear un medio interactivo (sitio web, libro digital) que cuente con la información necesaria para el desarrollo de la asignatura, ¿que calificación le darías a esta alternativa?:

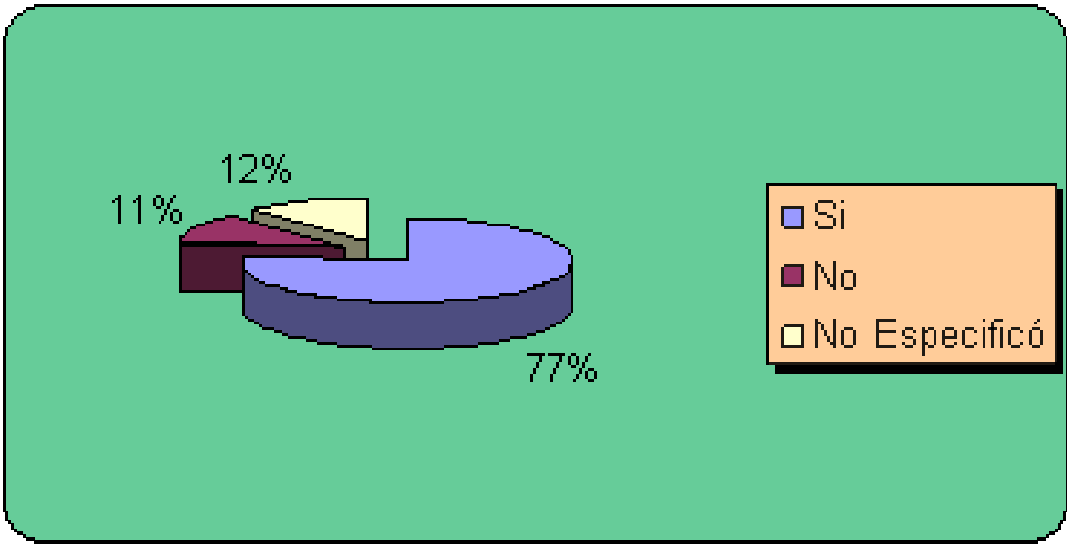
a.- Malo
b.- Regular
c.- Bueno
d.- Muy bueno
e.- Excelente

Categoría	# Respuestas	%
Malo	0	0
Regular	11	15,068493
Bueno	11	15,068493
Muy Bueno	19	26,027397
Excelente	32	43,835616
Total	73	100



8.- ¿Cree que la carrera de Ingeniería en Computación debería de promover o crear un medio interactivo para apoyar la enseñanza y el aprendizaje de las Ingenierías de Software?

Categoría	# Respuestas	%
Si	56	76,712329
No	8	10,958904
No Especificó	9	12,328767
Total	73	100



ANEXO 5

Principales pantallas del sitio Web.

1.- Pantallas de bienvenida al sitio Web.

Esta es la pantalla que se carga al entrar al sitio Web. Desde esta pantalla el usuario se puede desplazar hacia cualquiera de las otras páginas de este sitio.

2.- Página de Inicio.

En esta página se muestra una breve información a cerca de los autores del sitio, y donde éstos muestran especiales agradecimientos a las personas que se vieron involucradas y que fueron de vital importancia en el desarrollo este proyecto. Esta página es omitible para los usuarios.

3.- Pagina principal.

Esta es la pagina principal del sitio, desde donde el usuario puede acceder a la información contenida dentro del sitio.

3.1.- Página de introducción.

En esta página se mencionan de forma breve, los motivos que conllevaron a la puesta en marcha de este proyecto.

3.2.- Capítulos.

En esta parte del sitio, el usuario puede observar y acceder a la información contenida en este sitio. La información se muestra a manera de un “libro”, dividido en partes y éstas a su vez en capítulos.

3.4.- Referencias.

En este link se presentan los diferentes sitios web visitados y los libros que fueron utilizados como bases teóricas para la estructuración de la información contenida en el sitio.

4.- Fundamentos estadísticos.

En esta parte del sitio, el usuario puede observar las encuestas realizadas y cuyo análisis y resultados respaldaron la idea que culminó en el desarrollo de esta herramienta.

5.- Zona de descarga.

En esta página se puede descargar, a manera de archivos de extensión .rar, la información contenida en el sitio.

6.- Comentarios.

Esta página podría decirse que es la despedida, en ella se presentan a manera de link, los correos electrónicos de los autores del sitio. Dejando la posibilidad de contactarse con ellos para comentarios referentes al sitio web