Módulo
Ingeniería del SOFTWARE I

María Elena Penagos Garcés
Gladis Helena Vázquez Echevarria
John Jairo Monsalve Restrepo

POLITÉCNICO COLOMBIANO
JAIME ISAZA CADAVID

Alcaldía de Medellín

INTEGRACIÓN DEL MÓDULO POR UNIDADES

UNIDAD 0: SABERES PREVIOS

1. ¿Qué es un sistema?
2. ¿Qué es un sistema de información?
3. ¿Que es la Ingeniería del Software?
4. ¿Cuál es la importancia económica, social y política de la Ingeniería de software?
5. ¿Cuál es la importancia de la ingeniería del software para Colombia?
6. ¿Cuáles son las etapas de un proyecto de software?
7. ¿Cuáles son las metodologías y herramientas y que se utilizan en las diferentes etapas de un proyecto de software?
8. ¿Cuáles son los productos que se obtienen al terminar todas y cada una de las etapas del desarrollo de software?
9. ¿Cuáles son las principales aplicaciones de la Ingeniería del software?
10. ¿Cómo plantear un proyecto de software?

UNIDAD 1: INTRODUCCIÓN A LA INGENIERÍA DEL SOFTWARE

1. Evolución del software y crisis del software.
2. Características del Software.
3. ¿En qué consiste la actividad de resolver un problema?
4. ¿Qué son los conceptos?
5. ¿Qué es la Ingeniería de software?
6. ¿Qué es el diseño?
7. Componentes del software.
8. Aplicaciones del software

UNIDAD 2: PROCESO DE DESARROLLO DE SOFTWARE

1. Construcción de software
2. Procesos involucrados en el desarrollo de software
3. Modelos de procesos
4. Ciclo de vida de un proyecto

UNIDAD 3: PLANEACIÓN DE UN PROYECTO DE SOFTWARE

1. Conceptos básicos.
2. Pasos para el desarrollo de la planeación.
3. Investigación preliminar
4. Estudio de factibilidad.
5. Principios para la estimación
6. Estimaciones – técnicas y modelos empíricos.
7. Definición de requerimientos

UNIDAD 4: FORMULACIÓN DE UN PROYECTO DE SOFTWARE

Presentación del Proyecto y Cronograma de Actividades

BIBLIOGRAFÍA/CIBERGRAFÍA

UNIDAD 0

SABERES PREVIOS

CONDUCTA DE ENTRADA

¿Qué saberes previos se deben tener para entender este programa y por que ?

¿Qué considera Usted es la Ingeniería del software?

¿Cuáles aplicaciones de la Ingeniería del software conoce y con las cuales interactuamos todos los días ? Cite al menos tres.

Una vez desarrollado el programa ¿Qué se espera saber y que capacidades cree que debe haber adquirido con relación a la Ingeniería del software?

ESTRATEGIAS METODOLÓGICAS

- Presentación de la unidad por parte del profesor.
- Exposición magistral de los conceptos inherentes a la ingeniería de software, utilizando diferentes medios didácticos para apoyar el proceso de aprendizaje.
- Desarrollo de plenarias y exposiciones.
- Resolución en grupo de talleres y cuestionarios.
- Lecturas guiadas.
- Consultas e investigaciones en internet

PLAN DE TRABAJO

INTRODUCCIÓN

El programa de Ingeniería de Software a desarrollarse para grado 10°, comprende cuatro unidades de trabajo teórico practicas, con ellas, se pretende que el alumno posea una visión clara, sencilla sobre los principios generales de la Ingeniería del software.

OBJETIVO GENERAL

Introducir a los estudiantes en el conocimiento de la ingeniería del software sus Procesos o etapas, métodos y herramientas. Desarrollando en el estudiante habilidades y destrezas que le permitan al hacer uso de las técnicas, de forma que termine el curso planteando un proyecto .

OBJETIVOS ESPECÍFICOS

Adquirir una clara comprensión de los ciclos de vida de un proyecto de software .

Proporcionar los conocimientos y desarrollar las habilidades y destrezas que le permitan al estudiante, plantear, formular y modelar, para resolver problemas prácticos de los sistemas de información, aplicando las diferentes herramientas utilizadas en la Ingeniería del software.

Desarrollar un pensamiento sistémico que le permita solucionar un problema practico de un sistema de información.

Aprender a plantear un proyecto de desarrollo de software.

Conocer las diferentes aplicaciones de la Ingeniería del software y su importancia dentro de los planes de desarrollo empresarial para Colombia.

CONTENIDO

1. ¿Qué es un sistema?
2. ¿Qué es un sistema de información?
3. ¿Que es la Ingeniería del Software?
4. ¿Cuál es la importancia económica, social y política de la Ingeniería de software?
5. ¿Cuál es la importancia de la ingeniería del software para Colombia?
6. ¿Cuáles son las etapas de un proyecto de software?
7. ¿Cuáles son las metodologías y herramientas y que se utilizan en las diferentes etapas de un proyecto de software?
8. ¿Cuáles son los productos que se obtienen al terminar todas y cada una de las etapas del desarrollo de software?
9. ¿Cuáles son las principales aplicaciones de la Ingeniería del software?
10. ¿Cómo plantear un proyecto de software?

1. SISTEMA

Un sistema es un conjunto de elementos organizados que interactúan entre sí y con su ambiente, para lograr objetivos comunes, operando sobre materia energía o información u organismos para producir como salida información o energía materia u otros organismos.

2. SISTEMA DE INFORMACIÓN

Un sistema de información se puede definir como el conjunto de funciones y procedimientos encaminadas a la captación, desarrollo, recuperación, almacenamiento, etc., de información en el seno de una organización.

3. INGENIERÍA DEL SOFTWARE

Es el conjunto de métodos, técnicas y herramientas que controlan el proceso integral del desarrollo de software y suministra las bases para construir software de calidad de forma eficiente en los plazos adecuados.

4. IMPORTANCIA ECONÓMICA, SOCIAL Y POLÍTICA DE LA INGENIERÍA DEL SOFTWARE

Económicamente:

En los USA, el software contribuyó al 25% de todo el incremento del PIB durante los 90's (alrededor de 90,000 millones de dólares por año), y 1/6 de todo el crecimiento de productividad durante los últimos años de la década (alrededor de 33,000 millones de dólares por año). La ingeniería de software contribuyó con $1 billón dólares con de crecimiento económico y productividad en esa década. Alrededor del globo, el software contribuye al crecimiento económico en formas similares, aunque es difícil de encontrar estadísticas fiables.

Socialmente:

La ingeniería de software cambia la cultura del mundo debido al extendido uso de la computadora. El correo electrónico (E-mail), la WWW y la mensajería instantánea permiten a la gente interactuar en nuevas formas. El software baja el costo y mejora la calidad de los servicios de salud, los departamentos de bomberos, las dependencias gubernamentales y otros servicios sociales. Los proyectos exitosos donde se han usado métodos de ingeniería de software incluyen a Linux, el software del transbordador espacial, los cajeros automáticos y muchos otros.

5. IMPORTANCIA DE LA INGENIERÍA DEL SOFTWARE PARA COLOMBIA

Dentro de los planes de desarrollo del país y especialmente de Antioquia está el ubicarnos en una posición estratégica para convertirnos en productores y exportadores de software a nivel mundial.

Políticamente: Las consideraciones que se dan actualmente en el mundo obligan a nuestro país a alcanzar niveles competitivos para situarse en un lugar significativo en el concierto internacional.

6. ETAPAS DE UN PROYECTO DE SOFTWARE

1. Planificación y/o Levantamiento de requerimientos
2. Análisis y diseño
3. Implementación
4. Post-implementación

7. HERRAMIENTAS Y METODOLOGÍAS DE LAS ETAPAS DE UN PROYECTO DE SOFTWARE

Soporte automático o semiautomático a los métodos, orientadas a etapas particulares en el diseño de un software. Herramientas CASE.

Métodos:

Cómo se construye el software (planificación, análisis de los requisitos, diseño del sistema, codificación, prueba y mantenimiento).

8. PRODUCTOS QUE SE OBTIENEN AL TERMINAR TODAS Y CADA UNA DE LAS ETAPAS DEL DESARROLLO DE SOFTWARE

Planificación y/o Levantamiento de requerimientos

Productos que se obtienen de esta etapa:

- Requerimientos del sistema
- Especificaciones generales del sistema
- Costo y tiempos de ejecución del proyecto
- Recursos requeridos para el sistema
- Factibilidad del sistema

Análisis y diseño

Productos generales que se obtienen de esta etapa:

- Modelos de casos de uso (reglas del negocio)
- Interfaces de entrada y salida del sistema
- Modelo relacional de la base de datos

9. APLICACIONES DE LA INGENIERÍA DEL SOFTWARE

La Ingeniería de Software tiene que ver con muchos campos en diferentes formas:

Matemáticas:

Los programas tienen muchas propiedades matemáticas. Por ejemplo la corrección y la complejidad de muchos algoritmos son conceptos matemáticos que pueden ser rigurosamente probados. El uso de matemáticas en la IS (ingeniería de sistemas) es llamado métodos formales. Dijkstra ha dicho que la IS es una rama de las matemáticas.

Ciencias:

Los programas tienen muchas propiedades científicas que se pueden medir. Por ejemplo, el desempeño y la escalabilidad de programas bajo

diferentes cargas de trabajo puede ser medida. La efectividad de los cachés, procesadores más grandes, redes más rápidas, nuevas tecnologías de bases de datos tienen que ver con la ciencia. Se pueden deducir ecuaciones matemáticas de las medidas.

Ingeniería:

David Parnas ha argumentado que es una ingeniería.

La Ingeniería de Software es considerada por muchos como una disciplina ingenieril porque tiene los puntos de vistas pragmáticos y las características esperadas de los ingenieros. Análisis, documentación, y código comentado son signos de un ingeniero.

Manufactura:

Los programas son construidos en una secuencia de pasos. El hecho de definir propiamente y llevar a cabo estos pasos, como en una línea de ensamblaje, es necesario para mejorar la productividad de los desarrolladores y la calidad final de los programas. Este punto de vista inspira los diferentes procesos y metodologías que encontramos en la IS.

Manejo de Proyectos:

El software comercial (y mucho no comercial) requiere manejo de proyectos. Hay presupuestos y calendarizaciones establecidas. Gente para liderar. Recursos (espacio de oficina, computadoras) por adquirir. Todo esto encaja apropiadamente con la visión del Manejo de Proyectos.

Arte:

Los programas contienen muchos elementos artísticos. Las interfaces de usuario, la codificación, El diseño de sitios web etc. Incluso la decisión para un nombre de una variable o una clase. Donald Knuth es famoso porque ha argumentado que la programación es un arte.

10. ¿CÓMO PLANTEAR UN PROYECTO DE SOFTWARE?

Inicialmente definir el problema, justificar la necesidad de existencia del proyecto, determinar el alcance del mismo, el objetivo general y los específicos, el impacto social y las conclusiones. Esto se examinará en detalle en el capitulo 4 de este modulo cuando usted presente su propuesta de proyecto de grado.

Los anteriores interrogantes se resolverán en el desarrollo de este módulo en el mismo orden en el cual se plantearon, de la manera mas sencilla posible y didácticamente agrupados en unidades. Como es obvio se pretende que el estudiante mismo estructure las respuestas, tenga otros interrogantes y se capacite para resolverlos con propiedad. Además se complementa con la realización de talleres y practicas relacionadas con el tema.

ACTIVIDADES

1. Responder al siguiente cuestionario

   - De ejemplos de sistemas en general
   - De ejemplos de sistemas de información
   - Discuta la definición de la Ingeniería del software y argumente si es ciencia arte o conjunto de procesos
   - Investigue en Internet sobre el Plan de Desarrollo de Antioquia y el papel de la Ingeniería del software en el .
   - Busque en www.paginasamarillas.com cuantas empresas de software aparecen registradas y a que se dedican.

2. Investigue sobre como fueron creadas Oracle y Microsoft y comparta la historia con sus compañeros

3. Si UD fuera a crear su propia empresa de producción de software en ¿Cúal área lo haría y por que?

4. ¿Cuál es la diferencia entre herramientas y métodos ?

5. Supongamos que Ud. va a crear un software de video juegos, ¿qué tendría que considerar para hacer ese proyecto?

6. Elabore un grafico o mapa conceptual donde relacione las diferentes etapas del software con los productos que se obtienen en cada una de ellas y lo que contendrían esos productos para el caso del video juego.

UNIDAD 1

INTRODUCCIÓN A LA INGENIERÍA DEL SOFTWARE

INTRODUCCIÓN

En esta era tecnológica, nuestras vidas están regidos por las computadoras y el software que las controla. Las consecuencias generadas por el uso del software, tanto a favor como en contra, son cruciales. Cuando todo funciona bien, no nos percatamos de esto, pero cuando fallan, lo percibimos, ¡¡Y de qué manera!! El resultado puede ser nefasto, lo cual se ha corroborado en múltiples ocasiones.

En este curso se presenta la tecnología que deberían utilizar aquellos que construyen software informático, -aquellos que lo hacen bien-.

La tecnología acompaña el proceso, usando un juego de métodos y un conjunto o caja de herramientas que se llama Ingeniería del software.

JUSTIFICACIÓN

La era de la información posiciona los sistemas, además la necesidad básica de la empresa, es tener ventaja competitiva frente a la competencia, pues además de agilizar los procesos internos, la información permite la prestación de un mejor servicio a los clientes, generándoles mayor satisfacción menores costos y afianzamiento y fidelización con la empresa.

Inicialmente se había pensado que diseñar es simplemente satisfacer las funcionalidades que nos pide el cliente, pero ahora existen mejores prácticas como son el uso de patrones que hacen nuestro diseño más robusto y más fácil de mantener.

OBJETIVO GENERAL

Identificar y argumentar como ha sido la evolución del software y la importancia de éste en el desarrollo de la industria.

OBJETIVOS ESPECÍFICOS

- Mostrar el amplio espectro del software y las características del mismo incluyendo una perspectiva histórica que permita al estudiante ubicarse en un contexto global .
- Aclarar los conceptos fundamentales de un sistema de información y sus componentes demostrando su importancia en el ámbito de la ingeniería del software.
- Identificar y diferenciar cada una de las aplicaciones que tiene el software en la industria.

CONTENIDO

1. Evolución del software y crisis del software.
2. Características del Software.
3. ¿En qué consiste la actividad de resolver un problema?
4. ¿Qué son los conceptos?
5. ¿Qué es la Ingeniería de software?
6. ¿Qué es el diseño?
7. Componentes del software.
8. Aplicaciones del software.

1. EVOLUCIÓN DEL SOFTWARE Y CRISIS DEL SOFTWARE

A principios de los años 80 la revista Business Magazine pregonó lo siguiente : “Software: Motor del Futuro “ y nadie tenía idea de lo bueno que eso podría ser, porque el software era desconocido en ese momento para la mayoría del público en general.

El software de computadora, se ha convertido en el alma mater. Es la máquina que conduce a la toma de decisiones comerciales; sirve como la base de la investigación científica moderna y la resolución de problemas de ingeniería. Es el factor clave que diferencia los productos y servicios modernos. Está inmerso en sistemas de todo tipo : de transporte, médicos, de telecomunicaciones, militares, procesos industriales, entretenimientos, productos de oficina…,¡¡ la lista es casi interminable!! A medida que nos adentramos en el siglo veintiuno, nos llevará a avances en todas las áreas desde la educación elemental hasta la genética y la biotecnología.

Todo esto ha cambiado la percepción pública del software. Los programas informáticos están omnipresentes, y el público los ve como un hecho tecnológico de la vida. En muchos ejemplos, las personas dejan su trabajo, bienestar, seguridad, entretenimiento, decisiones y sus propias vidas en manos del software informático.

Hoy en día el software tiene un doble papel. Es un producto y, al mismo tiempo, el vehículo para entregarlo. Como producto, hace entrega de la potencia informática que incorpora el hardware informático o, más ampliamente, una red de computadoras que es accesible por hardware local. Si reside dentro de un teléfono celular u opera dentro de una computadora central, el software es un transformador de información, reduciendo, gestionando, adquiriendo, modificando, mostrando o transmitiendo información que puede ser tan simple como un solo bit, o tan complejo como una presentación en multimedia. Como vehículo utilizado para hacer entrega del producto, el software actúa como la base de control de la computadora (sistemas operativos), la comunicación de información (redes) y la creación y control de otros programas (herramientas de software y entornos).

El papel del software informático ha sufrido un cambio significativo durante un periodo de tiempo superior a 50 años. Enormes mejoras en rendimiento del hardware, profundos cambios de arquitecturas informáticas, grandes aumentos de memoria y capacidad de almacenamiento y una gran variedad de opciones de entrada y salida han conducido a sistemas más sofisticados y más complejos basados en computadora. La sofisticación y la complejidad pueden producir resultados deslumbrantes cuando un sistema tiene éxito, pero también pueden suponer grandes problemas para aquellos que deben construir sistemas complejos.

Libros populares publicados durante los años 70 y 80 proporcionan una visión histórica útil dentro de la percepción cambiante de las computadoras y del software, y de su impacto en nuestra cultura. Osborne hablaba de una «nueva revolución industria, Toffler llamó a la llegada de componentes microelectrónicos la «tercera ola del cambio» en la historia de la humanidad, y Naisbitt " predijo la transformación de la sociedad industrial a una «sociedad de información». Feigenbaum y McCorduck sugirieron que la información y el conocimiento (controlados por computadora) serían el foco de poder del siglo veintiuno, y Stoll argumentó que la «comunidad electrónica» creada mediante redes y software es la clave para el intercambio de conocimiento alrededor del mundo.

Al comienzo de los años 90, Toffler describió un «cambio de poder» en el que las viejas estructuras de poder (gubernamentales, educativas, industriales, económicas y militares) se desintegrarían a medida que las computadoras y el software nos llevaran a la democratización del conocimiento». A Yourdon le preocupaba que las compañías en Estados Unidos pudieran perder su competitividad en empresas relativas al software y predijo «el declive y la caída del programador americano». Hammer y Champy argumentaron que las tecnologías de información iban a desempeñar el papel principal en la reingeniería de la compañía».
A medida que internet creció en importancia, su cambio de pensamiento demostró ser correcto. Al final del siglo veinte, el enfoque cambió una vez más. Aquí tuvo lugar el impacto de la «bomba de relojería» Y2K (año 2000) . Aunque muchos vieron las predicciones de los críticos del Y2K como reacciones, sus populares lecturas devolvieron la difusión del software a sus vidas. Hoy en día, «la computación omnipresente» ha producido una generación de aplicaciones de información que tienen conexión en banda

ancha a la Web para proporcionar «una capa de conexión sobre nuestras casas, oficinas, y autopistas». El papel del software continúa su expansión.

El programador solitario de antaño ha sido reemplazado por un equipo de especialistas del software, cada uno centrado en una parte de la tecnología requerida para entregar una aplicación concreta. Y de este modo, las cuestiones que se preguntaba el programador solitario son las mismas cuestiones que nos preguntamos cuando construimos sistemas modernos basados en computadoras:

¿Por qué lleva tanto tiempo terminar los programas?
¿Por qué son tan elevados los costes de desarrollo?
¿Por qué no podemos encontrar todos los errores antes de entregar el software a nuestros clientes?
¿Por qué nos resulta difícil constatar el progreso conforme se desarrolla el software?

2. CARACTERÍSTICAS DEL SOFTWARE

En 1970, menos del uno por ciento de las personas podría haber descrito inteligentemente lo que significaba «software de computadora». Hoy, la mayoría de los profesionales y muchas personas en general piensan en su mayoría que comprenden el software. ¿Pero lo entienden realmente? Para poder comprender lo que es el software (y consecuentemente la ingeniería del software), es importante examinar las características del software que lo diferencian de otras cosas que los hombres pueden construir. Cuando se construye hardware, el proceso creativo humano (análisis, diseño, construcción, prueba) se traduce finalmente en una forma física. Si construimos una nueva computadora, nuestro boceto inicial, diagramas formales de diseño y prototipo de prueba, evolucionan hacia un producto físico (chips, tarjetas de circuitos impresos, fuentes de potencia, entre otros).

El software es un elemento del sistema que es **lógico**, en lugar de **físico**. Por tanto el software tiene unas características considerablemente distintas a las del hardware:

- El software se desarrolla, no se fabrica en un sentido clásico
- El software no se estropea, se deteriora
- No tiene piezas de repuesto
- La mayoría del software se construye a la medida

3. ¿EN QUÉ CONSISTE LA ACTIVIDAD DE RESOLVER UN PROBLEMA?

Esta actividad consiste básicamente en considerar cuatro aspectos fundamentales :

- Identificar el problema
- Definir el problema

- Definir un método de desarrollo de software
- Dar una solución inteligente

4. ¿QUÉ SON LOS CONCEPTOS?

Desde temprana edad las personas nos formamos conceptos, y cada uno es una idea particular o una comprensión de nuestro mundo. Los conceptos adquiridos nos permiten sentir y razonar acerca de las cosas en el mundo.

A estas cosas que aplicamos nuestros conceptos las llamamos objetos y pueden ser reales o abstractos.

5. ¿QUÉ ES LA INGENIERÍA DE SOFTWARE?

La Ingeniería del Software, es una disciplina o área de la Informática o Ciencias de la Computación, que ofrece métodos y técnicas para desarrollar y mantener software de calidad que resuelven problemas de todo tipo.
Hoy día es cada vez más frecuente la consideración de la Ingeniería del Software como una nueva área de la ingeniería, y el ingeniero del software comienza a ser una profesión implantada en el mundo laboral internacional, con derechos, deberes y responsabilidades que cumplir, junto a una, ya, reconocida consideración social en el mundo empresarial y, por suerte, para esas personas con brillante futuro.

La Ingeniería del Software trata con áreas muy diversas de la informática y de las ciencias de la computación, tales como construcción de compiladores, sistemas operativos o desarrollos en Intranet o internet, abordando todas las fases del ciclo de vida del desarrollo de cualquier tipo de sistemas de información y aplicables a una infinidad de áreas tales como: negocios, investigación científica, medicina, producción, logística, banca, control de tráfico, meteorología, el mundo del derecho, la red de redes Intenet, redes Intranet y Extranet, etc.

El término ingeniería se define en el DRAE[1] como: Conjunto de conocimientos y técnicas que permiten aplicar el saber científico a la utilización de la materia y de las fuentes de energía.

Definición 1
Ingeniería de Software[2] es el estudio de los principios y metodoíogías para desarrollo y mantenimiento de sistemas de software .

[1] DRAE, Diccionario de la Real Academia Española de la Lengua
[2] En Hispanoamérica. el término utilizado normalmente es: Ingeniería de Software

***Definición* 2**
Ingeniería del Software es la aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadora y la documentación asociada requerida para desarrollar, operar (funcionar) y mantenerlos. Se conoce también como desarrollo de software o producción de software.

Definición 3
Ingeniería del Software trata del establecimiento de los principios y métodos de la ingeniería a fin de obtener software de modo rentable que sea fiable y trabaje en máquinas reales.

Definición 4
La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación (funcionamiento) y mantenimiento del software; es decir, la aplicación de ingeniería al software. 2. El estudio de enfoques como en (1) [IEEE, 19931

6. ¿QUÉ ES EL DISEÑO?

Es el proceso de planeación de un nuevo sistema dentro de la empresa para reemplazar o completar el existente.

La importancia del diseño del software se puede describir con una sola palabra “calidad”. El diseño es el lugar en donde se fomentará la calidad en la ingeniería del software. El diseño proporciona las representaciones del software que se pueden evaluar en cuanto a calidad. El diseño es la única forma de convertir exactamente los requisitos de un cliente en un producto o sistema de software finalizado. El diseño del software sirve como fundamento para todos los pasos siguientes del soporte del software y de la ingeniería del software.

Sin un diseño, corremos el riesgo de construir un sistema inestable, un sistema que fallará cuando se lleven a cabo cambios; un sistema que puede resultar difícil de comprobar; y un sistema cuya calidad no puede evaluarse hasta muy avanzado el proceso, sin tiempo suficiente y con mucho dinero gastado en él

7. COMPONENTES

Los componentes del software son básicamente los programas, los datos y los documentos

8. APLICACIONES DEL SOFTWARE

El software puede aplicarse en cualquier situación en la que se haya definido previamente un conjunto específico de pasos procedimentales (es decir, un algoritmo) (excepciones notables a esta regla son el software de los sistemas expertos y de redes neuronales). El

contenido y el determinismo de la información son factores importantes a considerar para determinar la naturaleza de una aplicación de software.

El contenido se refiere al significado y a la forma de la información de entrada y salida. Por ejemplo, muchas aplicaciones bancarias usan unos datos de entrada muy estructurados (una base de datos) y producen «informes» con determinados formatos.

El software que controla una máquina automática (por ejemplo: un control numérico) acepta elementos de datos discretos con una estructura limitada y produce órdenes concretas para la máquina en rápida sucesión.

El determinismo de la información se refiere a la predecibilidad del orden y del tiempo de llegada de los datos. Un programa de análisis de ingeniería acepta datos que están en un orden predefinido, ejecuta el algoritmo(s) de análisis sin interrupción y produce los datos resultantes en un informe o formato gráfico. Se dice que tales aplicaciones son determinadas.

Un sistema operativo multiusuario, por otra parte, acepta entradas que tienen un contenido variado y que se producen en instantes arbitrarios, ejecuta algoritmos que pueden ser interrumpidos por condiciones externas y produce una salida que depende de una función del entorno y del tiempo. Las aplicaciones con estas características se dice que son indeterminadas.

Algunas veces es difícil establecer categorías genéricas para las aplicaciones del software que sean significativas. Conforme aumenta la complejidad del software, es más difícil establecer compartimentos nítidamente separados. Las siguientes áreas del software indican la amplitud de las aplicaciones potenciales:

- **Software de sistemas.** El software de sistemas es un conjunto de programas que han sido escritos para servir a otros programas. Algunos programas de sistemas (por ejemplo: compiladores, editores y utilidades de gestión de archivos) procesan estructuras de información complejas pero determinadas. Otras aplicaciones de sistemas (por ejemplo: ciertos componentes del sistema operativo, utilidades de manejo de periféricos, procesadores de telecomunicaciones) procesan datos en gran medida indeterminados. En cualquier caso, el área del software de sistemas se caracteriza por una fuerte interacción con el hardware de la computadora; una gran utilización por múltiples usuarios; una operación concurrente que requiere una planificación, un compartir de recursos y una sofisticada gestión de procesos; unas estructuras de datos complejas y múltiples interfaces externas.

- **Software de tiempo real.** El software que coordina/ analiza/ controla sucesos del mundo real conforme ocurren, se denomina de tiempo real. Entre los elementos del software de tiempo real se incluyen: un componente de adquisición de datos que recolecta y da formato a la información recibida del entorno externo, un componente de análisis que transforma la información según lo requiera la aplicación, un componente de control Vs salida que responda al entorno externo, y un componente de monitorización que coordina todos los demás componentes, de forma que pueda mantenerse la repuesta en tiempo real (típicamente en el rango de un milisegundo a un segundo).

- **Software de gestión.** El proceso de la información comercial constituye la mayor de las áreas de aplicación del software. Los «sistemas» discretos (por ejemplo: nóminas, cuentas de haberes-débitos, inventarios, etc.) han evolucionado hacia el software de sistemas de información de gestión **(SIG)** que accede a una o más bases de datos que contienen información comercial. Las aplicaciones en esta área reestructuran los datos existentes para facilitar las operaciones comerciales o gestionar la toma de decisiones. Además de las tareas convencionales de procesamientos de datos, las aplicaciones de software de gestión también realizan cálculo interactivo (por ejemplo: el procesamiento de transacciones en puntos de ventas).

- **Software de ingeniería y científico.** El software de ingeniería y científico está caracterizado por los algoritmos de «manejo de números». Las aplicaciones van desde la astronomía a la vulcanología, desde el análisis de la presión de los automotores a la dinámica orbital de las lanzaderas espaciales y desde la biología molecular a la fabricación automática. Sin embargo, las nuevas aplicaciones del área de ingeniería/ciencia se han alejado de los algoritmos convencionales numéricos. El diseño asistido por computadora (del inglés CAD), la simulación de sistemas y otras aplicaciones interactivas, han comenzado a coger características del software de tiempo real e incluso del software de sistemas.

- **Software empotrado.** Los productos inteligentes se han convertido en algo común en casi todos los mercados de consumo e industriales. El software empotrado reside en memoria de sólo lectura y se utiliza para controlar productos y sistemas de los mercados industriales y de consumo. El software empotrado puede ejecutar funciones muy limitadas y curiosas (por ejemplo: el control de las teclas de un horno de microondas) o suministrar una función significativa y con capacidad de control (por ejemplo: funciones digitales en un automóvil, tales como control de la gasolina, indicadores en el salpicadero, sistemas de frenado, etc)

- **Software de computadoras personales.** El mercado del software de computadoras personales ha germinado en las pasadas dos décadas. El procesamiento de textos, las hojas de cálculo, los gráficos por computadora, multimedia, entretenimientos, gestión de bases de datos, aplicaciones financieras, de negocios y personales y redes o acceso a bases de datos externas son algunas de los cientos de aplicaciones.

- **Software basado en Web.** Las páginas Web buscadas por un explorador son software que incorpora instrucciones ejecutables (por ejemplo, CGI, HTML, Perl, o Java), y datos (por ejemplo, hipertexto y una variedad de formatos de audio y visuales). En esencia, la red viene a ser una gran computadora que proporciona un recurso software casi ilimitado que puede ser accedido por cualquiera con un modem.

- **Software de inteligencia artificial.** El software de inteligencia artificial (IA) hace uso de algoritmos no numéricos para resolver problemas complejos para los que no son adecuados el cálculo o el análisis directo. Los sistemas expertos, también

llamados sistemas basados en el conocimiento, reconocimiento de patrones (imágenes y voz), redes neuronales artificiales, prueba de teoremas, y los juegos son representativos de las aplicaciones de esta categoría.

ACTIVIDADES

- Elaborar un mapa conceptual del contenido de este capitulo.
- Hacer un ensayo de tres páginas sobre la perspectiva de la Ingeniería del software y la importancia que tiene dentro de tu formación académica.

ESTRATEGIAS METODOLÓGICAS

- Presentación de la unidad por parte del profesor.
- Exposición magistral de los conceptos inherentes a la ingeniería de software, utilizando diferentes medios didácticos para apoyar el proceso de aprendizaje.
- Desarrollo de plenarias y exposiciones.
- Resolución en grupo de talleres y cuestionarios.
- Lecturas guiadas.
- Consultas e investigaciones en internet

RECURSOS

- Humanos: profesor y alumnos.
- Institucionales: Salón de clases.
- Materiales: Módulo de Ingeniería del Software

INDICADORES DE EVALUACIÓN

- Evaluación escrita sobre la unidad
- Evaluación del mapa conceptual elaborado
- Revisión del ensayo

CRITERIOS DE EVALUACIÓN

¿Qué es Ingeníería de software ?
¿Cuáles son los componentes del software?
¿Porqué es importante la etapa de diseño para un producto de Software?
¿En qué consiste la actividad de resolver un problema?
¿Cuál es la diferencia entre datos, información, sistema, programas y documentos?
¿Cuáles son las características del software?

UNIDAD 2

PROCESO DE DESARROLLO DE SOFTWARE

INTRODUCCIÓN

Esta unidad busca que el estudiante distinga los factores que deben considerarse en el proceso de desarrollo de un proyecto de sistema de información, tales como el aspecto más apropiado de la computadora o la tecnología de comunicaciones que se va a utilizar, el impacto del nuevo sistema sobre los empleados de la empresa, las características específicas que el sistema debe tener y el modelo de ciclo de vida del proyecto a implementar.

JUSTIFICACIÓN

En la actualidad para muchas organizaciones, los sistemas de información basados en computadoras son el corazón de las actividades cotidianas y objeto de gran consideración en la toma de decisiones, las empresas consideran con mucho cuidados las capacidades de sus sistemas de información cuando deciden ingresar o no en nuevos mercados o cuando planean la respuesta que darán a la competencia.

Al establecer los sistemas de información basados en computadoras deben tener la certeza de que se logren dos objetivos principales: Que sea un sistema correcto y que este correcto el sistema. Ningún sistema que deje satisfacer ambos objetivos será completamente útil para la gerencia u organización.

Si los dispositivos de un sistema de información no se adaptan a su población de clientes, no lograra sus objetivos potenciales. Al mismo tiempo, aun cuando se identifiquen precisamente las necesidades del usuario, un sistema de información va tener un valor único si funciona en forma adecuada.

Los informes y las salidas producidas por el sistema deben ser precisos, confiables y completos. La función del Análisis puede ser: Dar soporte a las actividades de un negocio o desarrollar un producto que pueda venderse para generar beneficios.

Aunque la estimación, es más un arte que una Ciencia, es una actividad importante que no debe llevarse a cabo de forma descuidada. Existen técnicas útiles para la estimación de costes de tiempo. Y dado que la estimación es la base de todas las demás actividades de planificación del proyecto sirve como guía para una buena Ingeniería de Software.

Al estimar tomamos en cuenta no solo el procedimiento técnico a utilizar en el proyecto, sino que se toma en cuenta los recursos, costos y planificación. El Tamaño del proyecto es otro factor importante que puede afectar la precisión de las estimaciones.

A medida que el tamaño aumenta, crece rápidamente la interdependencia entre varios elementos del Software. La disponibilidad de información Histórica es otro elemento que determina el riesgo de la estimación.

OBJETIVO GENERAL

Proporcionar los conocimientos que permitan al estudiante aclarar los conceptos fundamentales de un sistema de información y sus componentes demostrando su importancia en el ámbito de la ingeniería del software.

OBJETIVOS ESPECÍFICOS

- Identificar los procesos utilizados para la construcción de software
- Analizar la pertinencia en la utilización de un modelo de procesos
- Determinar la importancia de la planeación en el ciclo de vida de un proyecto

CONTENIDO

1. Construcción de software
2. Procesos involucrados en el desarrollo de software
3. Modelos de procesos
4. Ciclo de vida de un proyecto

1. CONSTRUCCIÓN DE SOFTWARE

El Proceso de gestión para la creación de un Sistema o software, encierra un conjunto de actividades, una de las cuales es la estimación, estimar es echar un vistazo al futuro y aceptar un cierto grado de incertidumbre. Aunque la estimación, es mas un arte que una Ciencia, es una actividad importante que no debe llevarse a cabo de forma descuidada. Existen técnicas útiles para la estimación de costes y de tiempos
Al estimar tomamos en cuenta no solo del procedimiento técnico a utilizar en el proyecto, sino que se toma en cuenta los recursos, costos y planificación. El Tamaño del proyecto es otro factor importante que puede afectar la precisión de las estimaciones. A medida que el tamaño aumenta, crece rápidamente la interdependencia entre varios elementos del Software.

La disponibilidad de información Histórica es otro elemento que determina el riesgo de la estimación.

2. PROCESOS INVOLUCRADOS EN EL DESARROLLO DE SOFTWARE

Es el conjunto estructurado de actividades requeridas para desarrollar un sistema de software.

Las actividades varían dependiendo de la organización y del tipo de sistema a desarrollarse.

El objetivo de la Planificación del proyecto de Software es proporcionar un marco de trabajo que permita al gestor hacer estimaciones razonables de recursos costos y planificación temporal. Estas estimaciones se hacen dentro de un marco de tiempo limitado y al comienzo de un proyecto de software, y deberían actualizarse regularmente a medida que progresa el proyecto. Además, las estimaciones deberían definir los escenarios del mejor caso, y peor caso, de modo que los resultados del proyecto pueden limitarse.

El Objetivo de la planificación se logra mediante un proceso de descubrimiento de la información que lleve a estimaciones razonables.

ACTIVIDADES ASOCIADAS AL PROYECTO DE SOFTWARE.

1. **Ámbito del Software.**

   Es la primera actividad llevada a cabo durante la planificación del proyecto de Software.

   En esta etapa se deben evaluar la función y el rendimiento que se asignaron al Software durante la Ingeniería del Sistema de Computadora para establecer un

ámbito de proyecto que no sea ambiguo, e incomprensible para directivos y técnicos.

Describe la función, el rendimiento, las restricciones, las interfaces y la fiabilidad, se evalúan las funciones del ámbito y en algunos casos se mejoran para dar mas detalles antes del comienzo de la estimación. Las restricciones de rendimiento abarcan los requisitos de tiempo de respuesta y procesamiento, identifican los límites del software originados por el hardware externo, por la memoria disponible y por otros sistemas existentes.

El Ámbito se define como un prerequisito para la estimación y existen algunos elementos que se debe tomar en cuenta como es:

La Obtención de la Información necesaria para el software. Para esto el analista y el cliente se reúnen sobre las expectativas del proyecto y se ponen de acuerdo en los puntos de interés para su desarrollo.

2. Recursos:

La segunda tarea de la planificación del desarrollo de Software es la estimación de los recursos requeridos para acometer el esfuerzo de desarrollo de Software, esto simula a una pirámide donde las Herramientas (hardware y Software), son la base que proporciona la infraestructura de soporte al esfuerzo de desarrollo, en segundo nivel de la pirámide se encuentran los Componentes reutilizables. Y en la parte mas alta de la pirámide se encuentra el recurso primario, las personas (el recurso humano).

Cada recurso queda especificado mediante cuatro características:

- Descripción del Recurso.
- Informes de disponibilidad.
- Fecha cronológica en la que se requiere el recurso.
- Tiempo durante el que será aplicado el recurso.

- Recursos Humanos: La Cantidad de personas requeridas para el desarrollo de un proyecto de software solo puede ser determinado después de hacer una estimación del esfuerzo de desarrollo (por ejemplo personas mes o personas años), y seleccionar la posición dentro de la organización y la especialidad que desempeñara cada profesional.

- Recursos o componentes de software reutilizables: Cualquier estudio sobre recursos de software estaría incompleto sin estudiar la reutilización, esto es la creación y la reutilización de bloques de construcción de Software. Tales bloques se deben establecer en catálogos para una consulta más fácil, estandarizarse para una fácil aplicación y validarse para la también fácil integración.

El Autor Bennatan sugiere cuatro categorías de recursos de software que se deberían tener en cuenta a medida que se avanza con la planificación:

- Componentes ya desarrollados.
- Componentes ya experimentados.
- Componentes con experiencia Parcial.
- Componentes nuevos.

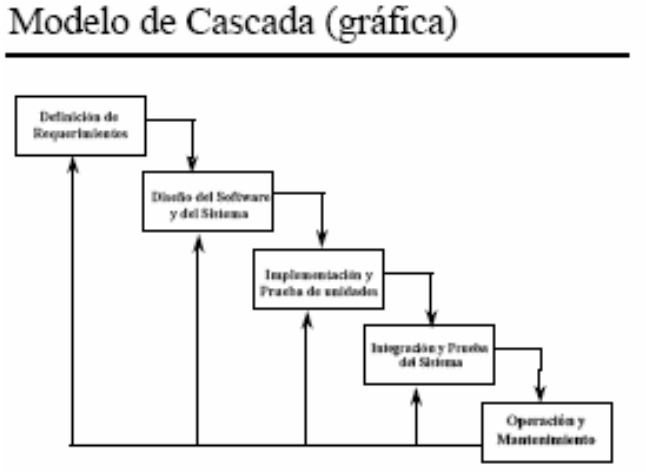
- Recursos de entorno: El entorno es donde se apoya el proyecto de Software, llamado a menudo entorno de Ingeniería de Software, incorpora Hardware y Software.

  El Hardware proporciona una plataforma con las herramientas (Software) requeridas para producir los productos que son el resultado de la buena practica de la Ingeniería del Software, un planificador de proyectos debe determinar la ventana temporal requerida para el Hardware y el Software, y verificar que estos recursos estén disponibles. Muchas veces el desarrollo de las pruebas de validación de un proyecto de software para la composición automatizada puede necesitar un compositor de fotografías en algún punto durante el desarrollo. Cada elemento de hardware debe ser especificado por el planificador del Proyecto de Software.

3. MODELOS DE PROCESOS

1. MODELO LINEAL SECUENCIAL.

Llamado ciclo de vida clásico o básico o modelo en cascada, consta de 6 etapas que son: Investigación preliminar, determinación de requisitos, diseño del sistema, desarrollo del sistema, prueba del sistema e implantación y evaluación. Algunos autores lo sintetizan en 5 fases que son: análisis de requisitos, diseño, generación de código, pruebas y mantenimiento



2. EL MODELO DE CONSTRUCCIÓN DE PROTOTIPOS

En casos en que el cliente define un conjunto de objetivos generales pero no detalla los requisitos de entrada, proceso o salida; o no se está seguro de la eficacia de un algoritmo, de la capacidad de adaptación a un sistema operativo, es mejor no arriesgar tiempo, dinero y trabajo haciendo un software "totalmente terminado", sino hacer un prototipo. Un prototipo es un programa que funciona y que al ponerse a prueba sirve para identificar nuevos requisitos de software.

3. EL MODELO DRA (DESARROLLO RÁPIDO DE APLICACIONES)

Es un modelo de desarrollo de software lineal secuencial que enfatiza un ciclo de desarrollo extremadamente corto. Es una adaptación a alta velocidad del modelo lineal secuencial en el que se logra el desarrollo rápido utilizando una construcción basada en componentes. Si se comprende bien los requisitos y se limita el ámbito del proyecto, el proceso DRA permite al equipo de desarrollo crear un "sistema completamente funcional" dentro de períodos cortos de tiempo (por ejemplo 60 a 90 días).

4. EL MODELO INCREMENTAL

Combina elementos del lineal secuencial (aplicados repetidamente) con la filosofía interactiva de construcción de prototipos.

5. EL MODELO ESPIRAL

Es un modelo de software evolutivo que combina el modelo de construcción de prototipos con el modelo secuencial lineal. El software se desarrolla en una serie de versiones incrementales, cada una con 6 regiones de tareas: Comunicación con el cliente, planificación, análisis de riesgos, ingeniería, construcción y adaptación y evaluación del cliente.

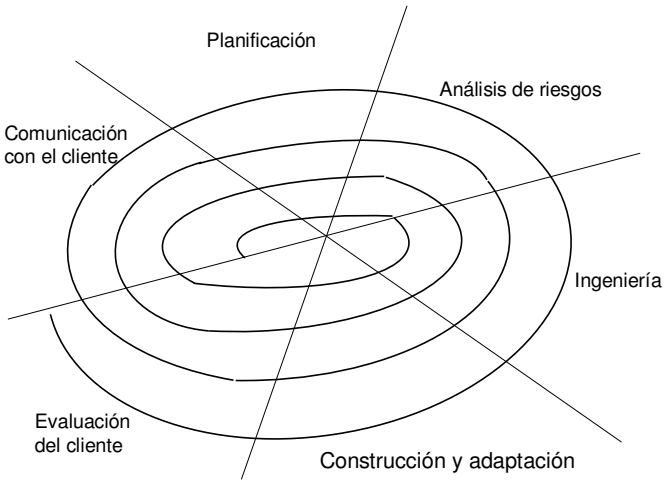


Figura 4. Modelo de la espiral

6. DESARROLLO BASADO EN COMPONENTES

La tecnología de objetos proporciona el marco de trabajo técnico para un modelo de proceso basado en componentes para la ingeniería del software. El paradigma orientado a objetos enfatiza la creación de clases que encapsulan tanto los datos como los algoritmos que se utilizan para manejarlos. Si se diseñan e implementan adecuadamente, las clases orientadas a objetos son reutilizables por las diferentes aplicaciones y arquitecturas de sistemas basados en computadora.

El modelo de desarrollo basado en componentes incorpora muchas de las características del modelo en espiral. Es evolutivo e iterativo para la creación de software, pero a diferencia del modelo en espiral, configura aplicaciones desde componentes preparados de software llamados “clases”.

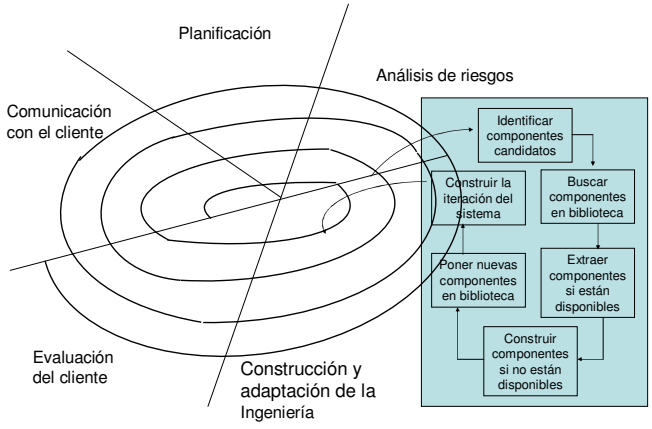


Figura 4. Modelo de desarrollo basado en componentes

4. CICLO DE VIDA DE UN PROYECTO

El desarrollo de software va unido a un ciclo de vida compuesto por una serie de etapas que comprenden todas las actividades, desde el momento en que surge la idea de crear un nuevo producto software, hasta aquel en que el producto deja definitivamente de ser utilizado por el último de sus usuarios.

El método del ciclo de vida para desarrollo de sistemas es el conjunto de actividades que los analistas, diseñadores y usuarios realizan para desarrollar e implantar un sistema de información.

El método del ciclo de vida para el desarrollo de sistemas consta de las siguientes actividades:

1. INVESTIGACIÓN PRELIMINAR O PREANÁLISIS

Consta de 3 subetapas (aclaración de la solicitud, estudio de factibilidad y confirmación de la solicitud).

En la aclaración de la solicitud, el equipo de software define con los usuarios el alcance del sistema.

El estudio de factibilidad consta de tres tipos de estudio, la factibilidad económica (¿traerá beneficios económicos al cliente el hecho de hacer el software o traerá perjuicios el hecho de no hacerlo?, la técnica (¿tiene el cliente los medios técnicos de hardware y de software para implementar el programa que se le desarrolle?) y la operacional (¿será operado, es decir, se utilizará el software desarrollado?)

En la confirmación de la solicitud, se adquiere un compromiso bilateral entre cliente e ingeniero del software en el cual cada uno tiene que cumplir con su parte: el ingeniero a desarrollar el software requerido y el cliente a entregar todos los requerimientos del software y retribuir el costo del proyecto. Esto se hace solo para un proyecto específico, el cual se hará a la medida para determinado cliente.

2. DETERMINACIÓN DE REQUERIMIENTOS O REQUISITOS (TAMBIÉN LLAMADO ANÁLISIS)

El ingeniero del software, utilizando herramientas como las encuestas, las entrevistas, la revisión de registros y la observación, recolecta los requerimientos (características del nuevo sistema) de los usuarios. Además, de acuerdo con sus experiencias en proyectos similares, puede anticipar algunos de los requisitos y recomendárselos al usuario. Esta fase es la que se llama en el modelo en espiral “comunicación con el cliente”

3. DISEÑO DEL SISTEMA:

Llamado también diseño lógico, consiste en bosquejar los elementos constitutivos del software como interfaz con el usuario, base de datos con sus tablas, consultas y reportes, además de los datos que se ingresarán, los que serán calculados y los que se almacenarán. El diseño del sistema produce los detalles que establecen la forma en que el sistema cumplirá con los requerimientos identificados durante la fase del análisis.

4. DESARROLLO DEL SISTEMA:

Los ingenieros del software se encargan de desarrollar el software o supervisar el desarrollo por parte de los analistas programadores, los cuales deben hacer parte del equipo de software. Además se debe producir la documentación del mismo.

5. PRUEBAS DEL SISTEMA:

En la fase de prueba, el software se emplea de manera experimental, para asegurarse de que el software no tenga fallas, que se haya desarrollado de acuerdo a los requisitos del usuario y de acuerdo a los estándares de calidad.

6. IMPLANTACIÓN Y EVALUACIÓN:

Es el proceso de instalar el software en el equipo, entrenar a los usuarios y construir los archivos y bases de datos necesarios para utilizarlo. Una vez hecho esto, se lleva a cabo la evaluación para identificar puntos débiles y fuertes y evaluar la operatividad del software, el impacto en la organización y la opinión de los administradores.

ACTIVIDAD…

- Elaborar un mapa conceptual que ilustre el ciclo de vida del proceso de desarrollo de un proyecto de software.
- Consultar con la industria más cercana al lugar de vivienda o de estudio que factores que tienen en cuenta cuando deciden implementar un sistema de información. Entre los factores a tener en cuenta pueden estar, por ejemplo, los costos, el factor humano, el tiempo, el técnico o tecnológico.

ESTRATEGIAS METODOLÓGICAS

- Presentación de la unidad a cargo del profesor.
- Discusión sobre la importancia de cada una de las etapas del ciclo de vida.

RECURSOS

- Humanos: Profesor y alumnos
- Institucionales: Salón de clase
- Materiales: Texto guía

INDICADORES DE EVALUACIÓN

- Evaluación escrita sobre la unidad
- Evaluación de un mapa conceptual elaborado.

CRITERIOS DE EVALUACIÓN

- ¿Qué factores se deben tener en cuenta en la estimación para el desarrollo de un proyecto de software?
- ¿Cuáles son las actividades asociadas a un proyecto de software?
- ¿Cuáles son los modelos de procesos?
- ¿Cuáles son las herramientas utilizadas en la determinación de requerimientos?
- ¿Cuáles son las actividades a realizar en el ciclo de vida para desarrollo de sistemas?

UNIDAD 3

PLANEACIÓN DE UN PROYECTO DE SOFTWARE

INTRODUCCIÓN

Esta unidad busca que el estudiante comprenda las técnicas para la abstracción de la información de la realidad relevante de un problema y de expresar dicha realidad en términos de un formato de especificación.

JUSTIFICACIÓN

Para resolver un problema, independiente del tipo de problema que este sea, es necesario primero entenderlo. Por esto la primera etapa en todo proceso de desarrollo de software consiste en tratar de entender el problema que tiene el cliente y expresar toda la información que este suministre, de manera que cualquier persona pueda entender lo que se espera de la solución.

Cuando nos enfrentamos a un problema podemos dividir este en tres aspectos, por una parte, se debe identificar lo que el cliente espera de la solución, esto se denomina requerimientos funcionales. El segundo aspecto que conforma el problema es el contexto en el que ocurre, es decir, el mundo en el que se encuentra inmerso. El tercer aspecto son los requerimientos no funcionales, que corresponden a las restricciones o condiciones que impone el cliente al programa que se va a construir.

Estas actividades iniciales del proceso de construcción de un software son críticas, puesto que cuando más tarde se detecta un error, más costoso es corregirlo.

OBJETIVO GENERAL

Desarrollar metodológicamente y aplicar las fases que se requieren en el proceso de la investigación preliminar de un proyecto de desarrollo de software.

Justificar la realización de un proyecto de software mediante los estudios de factibilidad en la etapa de investigación preliminar.

Determinar los requerimientos necesarios para construir un sistema de información.

OBJETIVOS ESPECÍFICOS

- Identificar los pasos que se llevan en la planeación de un proyecto de software.
- Conceptuar y aplicar cada una de las factibilidades de un proyecto
- Identificar las diferentes técnicas de estimación de software
- Determinar cuánto dinero, esfuerzo, recursos y tiempo requerirá la construcción de un sistema de información.
- Identificar los requerimientos operacionales de un sistema de información
- Aplicar técnicas de levantamiento de requerimientos.

CONTENIDO

1. Conceptos básicos.
2. Pasos para el desarrollo de la planeación.
3. Investigación preliminar
4. Estudio de factibilidad.
5. Principios para la estimación
6. Estimaciones – técnicas y modelos empíricos.
7. Definición de requerimientos

1. CONCEPTOS BÁSICOS

La planeación es un conjunto o disposición de procedimientos o programas relacionados de manera que juntos forman una sola unidad. Un conjunto de hechos, principios y reglas clasificadas y dispuestas de manera ordenada mostrando un plan lógico en la unión de las partes. Un método, plan o procedimiento de clasificación para hacer algo. También es un conjunto o arreglo de elementos para realizar un objetivo predefinido en el procesamiento de la Información. Esto se lleva a cabo teniendo en cuenta ciertos principios:

1. Debe presentarse y entenderse el dominio de la información de un problema.
2. Defina las funciones que debe realizar el Software.
3. Represente el comportamiento del software a consecuencias de acontecimientos externos.
4. Divida en forma jerárquica los modelos que representan la información, funciones y comportamiento.

El proceso debe partir desde la información esencial hasta el detalle de la Implementación.

La función de la planeación puede ser dar soporte a las actividades de un negocio, o desarrollar un producto que pueda venderse para generar beneficios. Para conseguir este objetivo, un Sistema basado en computadoras hace uso de seis (6) elementos fundamentales:

1. Software, que son Programas de computadora, con estructuras de datos y su documentación que hacen efectiva la logística metodología o controles de requerimientos del Programa.
2. Hardware, dispositivos electrónicos y electromecánicos, que proporcionan capacidad de cálculos y funciones rápidas, exactas y efectivas (Computadoras, Censores, maquinarias, bombas, lectores, etc.), que proporcionan una función externa dentro de los Sistemas.
3. Personal, son los operadores o usuarios directos de las herramientas del Sistema.
4. Base de Datos, una gran colección de informaciones organizadas y enlazadas al Sistema a las que se accede por medio del Software.
5. Documentación, Manuales, formularios, y otra información descriptiva que detalla o da instrucciones sobre el empleo y operación del Programa.
6. Procedimientos, o pasos que definen el uso específico de cada uno de los elementos o componentes del Sistema y las reglas de su manejo y mantenimiento.

2. PASOS PARA EL DESARROLLO DE LA PLANEACIÓN

La planeación del sistema se lleva a cabo teniendo en cuenta los siguientes objetivos:

1. Identificar las necesidades del Cliente.
2. Evaluar que conceptos tiene el cliente del sistema para establecer su viabilidad.
3. Realizar un Análisis Técnico y económico.
4. Asignar funciones al Hardware, Software, personal, base de datos, y otros elementos del Sistema.
5. Establecer las restricciones de presupuestos y planificación temporal.
6. Crear una definición del sistema que forme el fundamento de todo el trabajo de Ingeniería.

Para lograr estos objetivos se requiere tener un gran conocimiento y dominio del Hardware y el Software, así como de la Ingeniería humana (Manejo y Administración de personal), y administración de base de datos.

3. INVESTIGACIÓN PRELIMINAR

La solicitud para recibir ayuda de un sistema de información pueden originarse por una persona, cuando se formula la solicitud comienza la primera actividad del sistema.

Antes de considerar cualquier investigación de sistemas, la solicitud de proyecto debe examinarse para determinar con precisión lo que el solicitante desea; ya que muchas solicitudes que provienen de empleados y usuarios no están formuladas de manera clara.

4. ESTUDIO DE FACTIBILIDAD

En la investigación preliminar un punto importante es determinar que el sistema solicitado sea factible. Existen cuatro aspectos relacionados con el estudio de factibilidad, que son realizados por lo general por analistas capacitados o directivos:

1. Factibilidad técnica: Estudia si el trabajo para el proyecto, puede desarrollarse con el software y el personal existente, y si en caso de necesitar nueva tecnología, cuales son las posibilidades de desarrollarla (no solo el hardware).

2. Factibilidad económica: Investiga si los costos se justifican con los beneficios que se obtienen, y si se ha invertido demasiado, como para no crear el sistema si se cree necesario.

3. Factibilidad operacional: Investiga si será utilizado el sistema, si los usuarios usaran el sistema, como para obtener beneficios.

4. Factibilidad Legal: Es determinar cualquier posibilidad de infracción, violación o responsabilidad legal en que se podría incurrir al desarrollar el Sistema.

Aprobación de la solicitud

Algunas organizaciones reciben tantas solicitudes de sus empleados que sólo es posible atender unas cuantas. Sin embargo, aquellos proyectos que son deseables y factibles deben incorporarse en los planes. En algunos casos el desarrollo puede comenzar inmediatamente, aunque lo común es que los miembros del equipo de sistemas estén ocupados en otros proyectos. Cuando esto ocurre, la administración decide que proyectos son los más importantes y el orden en que se llevarán acabo.

Después de aprobar la solicitud de un proyecto se estima su costo, el tiempo necesario para terminarlo y las necesidades de personal

5. PRINCIPIOS PARA LA ESTIMACIÓN

En el principio el costo del Software constituía un pequeño porcentaje del costo total de los sistemas basados en Computadoras. Hoy en día el Software es el elemento más caro de la mayoría de los sistemas informáticos.

Un gran error en la estimación del costo puede ser lo que marque la diferencia entre beneficios y perdidas, la estimación del costo y del esfuerzo del software nunca será una ciencia exacta, son demasiadas las variables: humanas, técnicas, de entorno, políticas, que pueden afectar el costo final del software y el esfuerzo aplicado para desarrollarlo.

Para realizar estimaciones seguras de costos y esfuerzos tienen varias opciones posibles:

- Deje la estimación para más adelante (obviamente podemos realizar una estimación al cien por cien fiable después de haber terminado el proyecto.
- Base las estimaciones en proyectos similares ya terminados.
- Utilice técnicas de descomposición relativamente sencillas para generar las estimaciones de costos y esfuerzo del proyecto.
- Desarrolle un modelo empírico para él cálculo de costos y esfuerzos del Software.

Desdichadamente la primera opción, aunque atractiva no es práctica.

La Segunda opción puede funcionar razonablemente bien si el proyecto actual es bastante similar a los esfuerzos pasados y si otras influencias del proyecto son similares. Las opciones restantes son métodos viables para la estimación del proyecto de software. Desde el punto de vista ideal, se deben aplicar conjuntamente las técnicas indicadas usando cada una de ellas como comprobación de las otras.

Antes de hacer una estimación, el planificador del proyecto debe comprender el ámbito del software a construir y generar una estimación de su tamaño.

Estimación basada en el Proceso.

Es la técnica más común para estimar un proyecto es basar la estimación en el proceso que se va a utilizar, es decir, el proceso se descompone en un conjunto relativamente pequeño de actividades o tareas, y en el esfuerzo requerido para llevar a cabo la estimación de cada tarea.

Al igual que las técnicas basadas en problemas, la estimación basada en el proceso comienza en una delineación de las funciones del software obtenidas a partir del ámbito del proyecto. Se mezclan las funciones del problema y las actividades del proceso. Como ultimo paso se calculan los costos y el esfuerzo de cada función y la actividad del proceso de software.

6. ESTIMACIONES – TÉCNICAS Y MODELOS EMPÍRICOS

Existen diferentes modelos de estimación como son:

Los Modelos Empíricos

Donde los datos que soportan la mayoría de los modelos de estimación obtienen una muestra limitada de proyectos. Por esta razón, el modelo de estimación no es adecuado para todas las clases de software y en todos los entornos de desarrollo. Por lo tanto los resultados obtenidos de dichos modelos se deben utilizar con prudencia.

El Modelo COCOMO.

Barry Boehm, en su libro clásico sobre economía de la Ingeniería del Software, introduce una jerarquía de modelos de estimación de Software con el nombre de COCOMO, por su nombre en Ingles (Constructive, Cost, Model) modelo constructivo de costos. La jerarquía de modelos de Boehm esta constituida por los siguientes:

1. Modelo I. El Modelo COCOMO básico calcula el esfuerzo y el costo del desarrollo de Software en función del tamaño del programa, expresado en las líneas estimadas.
2. Modelo II. El Modelo COCOMO intermedio calcula el esfuerzo del desarrollo de software en función del tamaño del programa y de un conjunto de conductores de costos que incluyen la evaluación subjetiva del producto, del hardware, del personal y de los atributos del proyecto.
3. Modelo III. El modelo COCOMO avanzado incorpora todas las características de la versión intermedia y lleva a cabo una evaluación del impacto de los conductores de costos en cada caso (análisis, diseño, etc.) del proceso de ingeniería de Software.

Herramientas Automáticas De Estimación.

Las herramientas automáticas de estimación permiten al planificador estimar costos y esfuerzos, así como llevar a cabo análisis del tipo, que pasa si, con importantes variables del proyecto, tales como la fecha de entrega o la selección del personal. Aunque existen muchas herramientas automáticas de estimación, todas exhiben las mismas características generales y todas requieren de una o más clases de datos.

A partir de estos datos, el modelo implementado por la herramienta automática de estimación proporciona estimaciones del esfuerzo requerido para llevar a cabo el proyecto, los costos, la carga de personal, la duración, y en algunos casos la planificación temporal de desarrollo y riesgos asociados.

En resumen el planificador del Proyecto de Software tiene que estimar tres cosas antes de que comience el proyecto: cuanto durara, cuanto esfuerzo requerirá y cuanta gente estará implicada. Además el planificador debe predecir los recursos de hardware y software que va a requerir y el riesgo implicado.

Para obtener estimaciones exactas para un proyecto, generalmente se utilizan al menos dos de las tres técnicas referidas anteriormente. Mediante la comparación y la conciliación de las estimaciones obtenidas con las diferentes técnicas, el planificador puede obtener una estimación más exacta. La estimación del proyecto de software nunca será una

ciencia exacta, pero la combinación de buenos datos históricos y técnicas puede mejorar la precisión de la estimación.

7. DEFINICIÓN DE REQUERIMIENTOS

1. **Identificación de Necesidades (Análisis de Requisitos).**

   Definición: Es el conjunto de técnicas y procedimientos que nos permiten conocer los elementos necesarios para definir un proyecto de software.

   Es la etapa más crucial del desarrollo de un proyecto de software.

   La IEEE los divide en funcionales y no funcionales:

   1. **Funcionales**: Condición o capacidad de un sistema requerida por el usuario para resolver un problema o alcanzar un objetivo.

   2. **No Funcionales**: Condición o capacidad que debe poseer un sistema par satisfacer un contrato, un estándar, una especificación u otro documento formalmente impuesto.

   Para realizar bien el desarrollo de software es esencial realizar una especificación completa de los requerimientos de los mismos. Independientemente de lo bien diseñado o codificado que esté, un programa pobremente especificado decepcionará al usuario y hará fracasar el desarrollo.

   La tarea de análisis de los requerimientos es un proceso de descubrimiento y refinamiento, El ámbito del programa, establecido inicialmente durante la ingeniería del sistema, es refinado en detalle. Se analizan y asignan a los distintos elementos de los programas las soluciones alternativas.

   Tanto el que desarrolla el software como el cliente tienen un papel activo en la especificación de requerimientos. El cliente intenta reformular su concepto, algo nebuloso, de la función y comportamiento de los programas en detalles concretos, El que desarrolla el software actúa como interrogador, consultor y el que resuelve los problemas.

   El análisis y especificación de requerimientos puede parecer una tarea relativamente sencilla, pero las apariencias engañan. Puesto que el contenido de comunicación es muy alto, abundan los cambios por mala interpretación o falta de información. El dilema con el que se enfrenta un ingeniero de software puede ser comprendido repitiendo la sentencia de un cliente anónimo: “Sé que crees que comprendes lo que piensas que he dicho, pero no estoy seguro de que lo que creíste oír sea lo que yo quise decir”.

2. Análisis de Requerimientos

El análisis de requerimientos es la tarea que plantea la asignación de software a nivel de sistema y el diseño de programas (Figura 1). El análisis de requerimientos facilita al ingeniero de sistemas especificar la función y comportamiento de los programas, indicar la interfaz con otros elementos del sistema y establecer las ligaduras de diseño que debe cumplir el programa. El análisis de requerimientos permite al ingeniero refinar la asignación de software y representar el dominio de la información que será tratada por el programa. El análisis de requerimientos de al diseñador la representación de la información y las funciones que pueden ser traducidas en datos, arquitectura y diseño procedimental. Finalmente, la especificación de requerimientos suministra al técnico y al cliente, los medios para valorar la calidad de los programas, una vez que se haya construido.

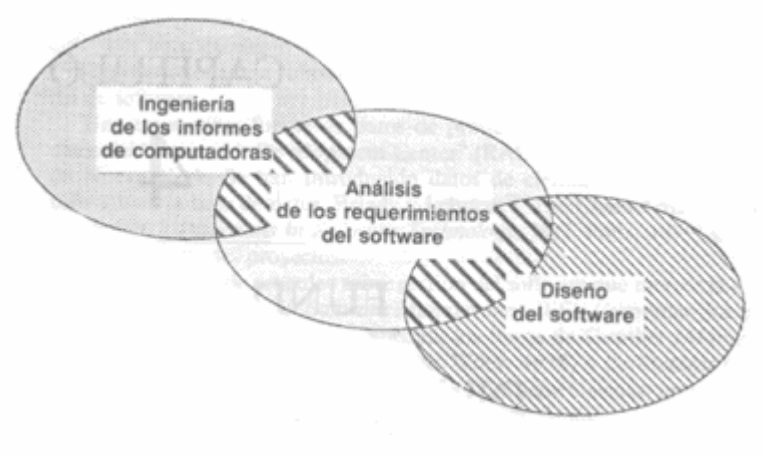


Figura 1.

- **Tareas del Análisis**

El análisis de requerimientos puede dividirse en cuatro áreas:

1. Reconocimiento del problema
2. Evaluación y síntesis
3. Especificación
4. Revisión.

Inicialmente, el analista estudia la especificación del sistema (si existe) y el plan de proyecto. Es importante comprender el contexto del sistema y revisar el ámbito de los programas que se usó para generar las estimaciones de la planificación. A continuación, debe establecerse la comunicación necesaria para el análisis, de forma que se asegure el reconocimiento del problema.

Las formas de comunicación requeridas para el análisis se ilustran en la Figura 2. El analista debe establecer contacto con el equipo técnico y de gestión del

usuario/cliente y con la empresa que vaya a desarrollar el software. El gestor del programa puede servir como coordinador para facilitar el establecimiento de los caminos de comunicación. El objetivo del analista es reconocer los elementos básicos del programa tal como lo percibe el usuario/cliente.

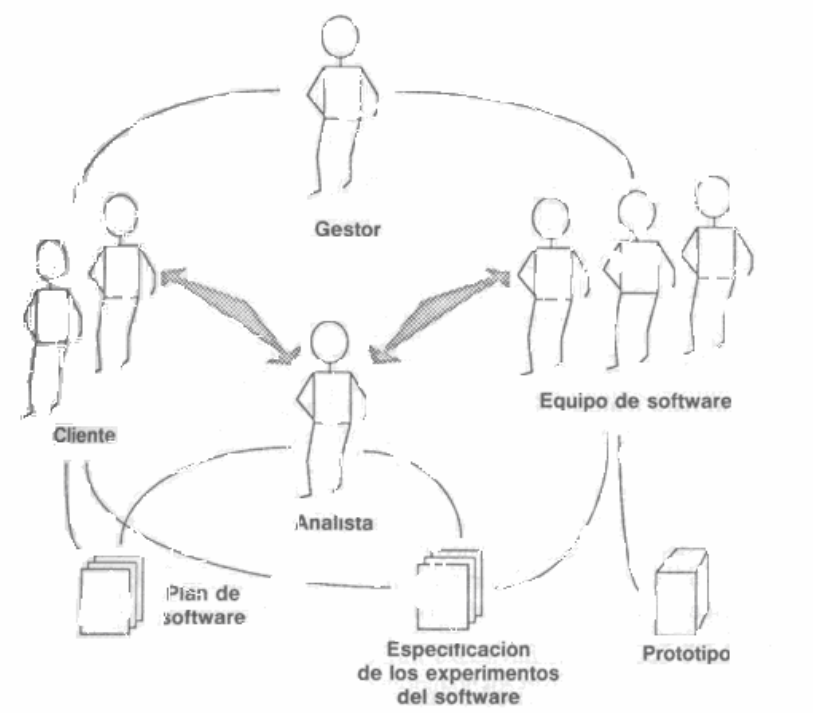


Figura 2

La evaluación del problema y la síntesis de la solución es la siguiente área principal de trabajo del análisis. El analista debe evaluar el flujo y estructura de la información, refinar en detalle todas las funciones del programa, establecer las características de la interfase del sistema y descubrir las ligaduras del diseño, Cada una de las tareas sirve para descubrir el problema de forma que pueda sintetizarse un enfoque o solución global.

Las tareas asociadas con el análisis y especificación existen para dar una representación del programa que pueda ser revisada y aprobada por el cliente. En un mundo ideal el cliente desarrolla una especificación de requerimientos del software completamente por sí mismo. Esto se presenta raramente en el mundo real. En el mejor de los casos, la especificación se desarrolla conjuntamente entre el cliente y el técnico.

Una vez que se hayan descrito las funcionalidades básicas, comportamiento, interfase e información, se especifican los criterios de validación para demostrar una comprensión de una correcta implementación de los programas. Estos criterios sirven como base para hacer una prueba durante el desarrollo de los programas. Para definir las características y atributos del software se escribe una especificación de

requerimientos formal. Además, para los casos en los que se desarrolle un prototipo se realiza un manual de usuario preliminar.

Puede parecer innecesario realizar un manual de usuario en una etapa tan temprana del proceso de desarrollo, Pero de hecho, este borrador del manual de usuario fuerza al analista a tomar el punto de vista del usuario del software. El manual permite al usuario / cliente revisar el software desde una perspectiva de ingeniería humana y frecuentemente produce el comentario: "La idea es correcta pero esta no es la forma en que pensé que se podría hacer esto". Es mejor descubrir tales comentarios lo más tempranamente posible en el proceso.

Los documentos del análisis de requerimiento (especificación y manual de usuario) sirven como base para una revisión conducida por el cliente y el técnico. La revisión de los requerimientos casi siempre produce modificaciones en la función, comportamiento, representación de la información, ligaduras o criterios de validación. Además, se realiza una nueva apreciación del plan del proyecto de software para determinar si las primeras estimaciones siguen siendo validas después del conocimiento adicional obtenido durante el análisis.

- **Principios del Análisis**

En la pasada década, se desarrollaron varios métodos de análisis y especificación del software. Los investigadores han identificado los problemas y sus causas y desarrollando reglas y procedimientos para resolverlos. Cada método de análisis tiene una única notación y punto de vista. Sin embargo, todos los métodos de análisis están relacionados por un conjunto de principios fundamentales:

- El dominio de la información, así como el dominio funcional de un problema debe ser representado y comprendido.
- El problema debe subdividirse de forma que se descubran los detalles de una manera progresiva (o jerárquica)
- Deben desarrollarse las representaciones lógicas y físicas del sistema.

Aplicando estos principios, el analista enfoca el problema sistemáticamente. Se examina el dominio de la información de forma que pueda comprenderse su función más completamente. La partición se aplica para reducir la complejidad. La visión lógica y física del software, es necesaria para acomodar las ligaduras lógicas impuestas por los requerimientos de procesamiento, y las ligaduras físicas impuestas por otros elementos del sistema.

- **El dominio de la Información**

Todas las aplicaciones del software pueden colectivamente llamarse procesamiento de datos. Este término contiene la clave de lo que entendemos por requerimientos del software. El software se construye para procesar datos; para transformar datos de una forma a otra; esto es, para aceptar entrada, manipularla de alguna forma y producir una salida. Este establecimiento fundamental de los objetivos es verdad tanto si construimos software por lotes para un sistema de nominas, como software

empotrado en tiempo real para controlar el flujo de la gasolina de un motor de automóvil; el dominio de la información contiene tres visiones diferentes de los datos que se procesan por los programas de computadoras:

1. El flujo de información;
2. El contenido de la información y
3. La estructura de la información. Para comprender completamente el dominio de la información, deben considerarse cada una de estas tres partes.

El flujo de la información representa la manera en la que los datos cambian conforme pasan a través de un sistema. Refiriéndonos a la Figura 3, la entrada se transforma en datos intermedios y más adelante se transforma en la salida.

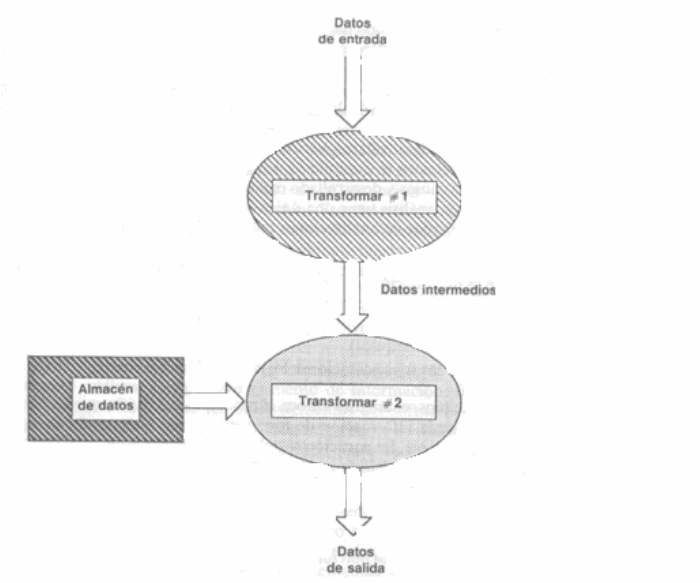


Figura 3.

3. Partición

Normalmente los problemas son demasiado grandes y complejos para ser comprendidos como un todo. Por esta razón, tendemos a particionar (dividir) tales problemas en partes que puedan ser fácilmente comprendidas, y establecer interfases entre las partes, de forma que se realice la función global. Durante el análisis de requerimientos, el dominio funcional y el dominio de la información del software pueden ser particionados.

En esencia la partición descompone un problema en sus partes constituyentes. Conceptualmente, establecemos una representación jerárquica de la función o información y luego partimos el elemento superior mediante:

1. Incrementando los detalles, moviéndonos verticalmente en la jerarquía, o
2. Descomponiendo funcionalmente el problema, moviéndonos horizontalmente en la jerarquía.

4. Visiones Lógicas y Físicas

La visión lógica de los requerimientos del software presenta las funciones que han de realizarse y la información que ha de procesarse independientemente de los detalles de implementación.

La visión física de los requerimientos del software presenta una manifestación del mundo real de las funciones de procesamiento y las estructuras de información. En algunos casos se desarrolla una representación física como el primer paso del diseño del software. Sin embargo la mayoría de los sistemas basados en computador, se especifican de forma que se dictan ciertas recomendaciones físicas.

5. Construcción de Prototipos de Software

El análisis debe ser conducido independientemente del paradigma de ingeniería de software aplicado. Sin embargo, la forma que ese análisis tomará puede variar. En algunos casos es posible aplicar los principios de análisis fundamental y derivar a una especificación en papel del software desde el cual pueda desarrollarse un diseño. En otras situaciones, se va a una recolección de los requerimientos, se aplican los principios de análisis y se construye un modelo de software, llamado un prototipo, según las apreciaciones del cliente y del que lo desarrolla. Finalmente, hay circunstancias que requieren la construcción de un prototipo al comienzo del análisis, puesto que el modelo es el único mediante el que los requerimientos pueden ser derivados efectivamente.

- **Un escenario para la construcción de prototipos**

  Todos los proyectos de ingeniería de software comienzan con una petición del cliente. La petición puede estar en la forma de una memoria que describe un problema, un informe que define un conjunto de objetivos comerciales o del producto, una petición de propuesta formal de una agencia o compañía exterior, o una especificación del sistema que ha asignado una función y comportamiento al software, como un elemento de un sistema mayor basado en computadora. Suponiendo que existe una petición para un programa de una de las formas dichas anteriormente, para construir un prototipo del software se aplican los siguientes pasos:

  PASO 1. Evaluar la petición del software y determinar si el programa a desarrollar es un buen candidato para construir un prototipo.

  Debido a que el cliente debe interaccionar con el prototipo en los últimos pasos, es esencial que: 1) el cliente participe en la evaluación y refinamiento del prototipo, y 2) el cliente sea capaz de tomar decisiones de requerimientos de una forma oportuna. Finalmente, la naturaleza del

proyecto de desarrollo tendrá una fuerte influencia en la eficacia del prototipo.

PASO 2. Dado un proyecto candidato aceptable, el analista desarrolla una representación abreviada de los requerimientos.
Antes de que pueda comenzar la construcción de un prototipo, el analista debe representar los dominios funcionales y de información del programa y desarrollar un método razonable de partición. La aplicación de estos principios de análisis fundamentales, pueden realizarse mediante los métodos de análisis de requerimientos.

PASO 3. Después de que se haya revisado la representación de los requerimientos, se crea un conjunto de especificaciones de diseño abreviadas para el prototipo.

El diseño debe ocurrir antes de que comience la construcción del prototipo. Sin embargo, el diseño de un prototipo se enfoca normalmente hacia la arquitectura a nivel superior y a los aspectos de diseño de datos, en vez de hacia el diseño procedimental detallado.

PASO 4. El software del prototipo se crea, prueba y refina

Idealmente, los bloques de construcción de software preexisten se utilizan para crear el prototipo de una forma rápida. Desafortunadamente, tales bloques construidos raramente existen.

Incluso si la implementación de un prototipo que funcione es impracticable, es escenario de construcción de prototipos puede aun aplicarse. Para las aplicaciones interactivas con el hombre, es posible frecuentemente crear un prototipo en papel que describa la interacción hombre-maquina usando una serie de hojas de historia.

PASO 5. Una vez que el prototipo ha sido probado, se presenta al cliente, el cual "conduce la prueba" de la aplicación y sugiere modificaciones.

Este paso es el núcleo del método de construcción de prototipo. Es aquí donde el cliente puede examinar una representación implementada de los requerimientos del programa, sugerir modificaciones que harán al programa cumplir mejor las necesidades reales.

PASO 6. Los pasos 4 y 5 se repiten iterativamente hasta que todos los requerimientos estén formalizados o hasta que el prototipo haya evolucionado hacia un sistema de producción.

El paradigma de construcción del prototipo puede ser conducido con uno o dos objetivos en mente: 1) el propósito del prototipado es establecer un conjunto de requerimientos formales que pueden luego ser traducidos en la producción de programas mediante el uso de métodos y técnicas de ingeniería de programación, o 2) el propósito de la construcción del prototipo es suministrar un continuo que

pueda conducir al desarrollo evolutivo de la producción del software. Ambos métodos tienen sus meritos y amos crean problemas.

6. Especificación

No hay duda de que la forma de especificar tiene mucho que ver con la calidad de la solución. Los ingenieros de software que se han esforzado en trabajar con especificaciones incompletas, inconsistentes o mal establecidas han experimentado la frustración y confusión que invariablemente se produce. Las consecuencias se padecen en la calidad, oportunidad y completitud del software resultante.

Hemos visto que los requerimientos de software pueden ser analizados de varias formas diferentes. Las técnicas de análisis pueden conducir a una especificación en papel que contenga las descripciones graficas y el lenguaje natural de los requerimientos del software. La construcción de prototipos conduce a una especificación ejecutable, esto es, el prototipo sirve como una representación de los requerimientos. Los lenguajes de especificación formal conducen a representaciones formales de los requerimientos que pueden ser verificados o analizados.

- **Principios de Especificación**

  La especificación, independientemente del modo en que se realice, puede ser vista como un proceso de representación. Los requerimientos se representan de forma que conduzcan finalmente a una correcta implementación del software.

  Baltzer y Goldman proponen ocho principios para una buena especificación:

  PRINCIPIO #1. Separar funcionalidad de implementación. Primero, por definición, una especificación es una descripción de lo que se desea, en vez de cómo se realiza (implementa). Las especificaciones pueden adoptar dos formas muy diferentes. La primera forma es la de funciones matemáticas: dado algún conjunto de entrada, producir un conjunto particular de salida. La forma general de tales especificaciones es encontrar [un/el/todos] resultado tal que P (entrada), donde P representa un predicado arbitrario. En tales especificaciones, el resultado a ser obtenido ha sido expresado enteramente en una forma sobre el que (en vez de cómo). En parte, esto es debido a que el resultado es una función matemática de la entrada (la operación tiene puntos de comienzo y parada bien definidos) y no esta afectado por el entorno que le rodea.

  PRINCIPIO #2. Se necesita un lenguaje de especificación de sistemas orientado al proceso. Considerar una situación en la que el entorno sea dinámico y sus cambios afecten al comportamiento de alguna entidad que interactúe con dicho entorno. Su comportamiento no puede ser expresado como una función matemática de su entrada. En vez de ello, debe emplearse una descripción orientada al proceso, en la cual la especificación

del que se consigue mediante la especificación de un modelo del comportamiento deseado en términos de respuestas funcionales, a distintos estímulos del entorno.

PRINCIPIO #3. Una especificación debe abarcar el sistema del cual el software es una componente. Un sistema está compuesto de componentes que interactúan. Solo dentro del contexto del sistema completo y de la interacción entre sus partes puede ser definido el comportamiento de una componente específica. En general, un sistema puede ser modelado como una colección de objetos pasivos y activos. Estos objetos están interrelacionados y dichas relaciones entre los objetos cambian con el tiempo. Estas relaciones dinámicas suministran los estímulos a los cuales los objetos activos, llamados agentes, responden. Las respuestas pueden causar posteriormente cambios y, por tanto, estímulos adicionales a los cuales los agentes deben responder.

PRINCIPIO #4. Una especificación debe abarcar el entorno en el que el sistema opera. Similarmente, el entorno en el que opera el sistema y con el que interactúa debe ser especificado. Afortunadamente, esto tan solo necesita reconocer que el propio entorno es un sistema compuesto de objetos que interactúan, pasivos y activos, de los cuales el sistema especificado es una agente, Los otros agentes, los cuales son por definición inalterables debido a que son parte del entorno, limitan el ámbito del diseño subsecuente y de la implementación. De hecho, la única diferencia entre el sistema y su entorno es que el esfuerzo de diseño e implementación subsecuente opera exclusivamente sobre la especificación del sistema. La especificación del entorno facilita que se especifique la interfaz del sistema de la misma forma que el propio sistema, en vez de introducir otro formalismo.

PRINCIPIO #5. Una especificación de sistema debe ser un modelo cognitivo. La especificación de un sistema debe ser un modelo cognitivo, en vez de un modelo de diseño o implementación. Debe describir un sistema tal como es percibido por su comunidad de usuario. Los objetivos que manipula deben corresponderse con objetos reales de dicho dominio; los agentes deben modelar los individuos, organizaciones y equipo de ese dominio; y las acciones que ejecutan deben modelar lo que realmente ocurre en el dominio. Además, debe ser posible incorporar en la especificación las reglas o leyes que gobiernan los objetos del dominio. Algunas de estas leyes proscriben ciertos estados del sistema (tal como “dos objetos no pueden estar en el mismo lugar al mismo tiempo”), y por tanto limitan el comportamiento de los agentes o indican la necesidad de una posterior elaboración para prevenir que surjan estos estados.

PRINCIPIO #6. Una especificación debe ser operacional. La especificación debe ser completa y lo bastante formal para que pueda usarse para determinar si una implementación propuesta satisface la especificación de pruebas elegidas arbitrariamente. Esto es, dado el resultado de una implementación sobre algún conjunto arbitrario de datos elegibles, debe ser

posible usar la especificación para validar estos resultados. Esto implica que la especificación, aunque no sea una especificación completa del como, pueda actuar como un generador de posibles comportamientos, entre los que debe estar la implementación propuesta. Por tanto, en un sentido extenso, la especificación debe ser operacional.

PRINCIPIO #7. La especificación del sistema debe ser tolerante con la incompletitud y aumentable. Ninguna especificación puede ser siempre totalmente completa. El entorno en el que existe es demasiado complejo para ello. Una especificación es siempre un modelo, una abstracción, de alguna situación real (o imaginada). Por tanto, será incompleta. Además, al ser formulad existirán muchos niveles de detalle. La operacionalidad requerida anteriormente no necesita ser completa. Las herramientas de análisis empleadas para ayudar a los especificadores y para probar las especificaciones, deben ser capaces de tratar con la incompletitud. Naturalmente esto debilita el análisis, el cual puede ser ejecutado ampliando el rango de comportamiento aceptables, los cuales satisfacen la especificación, pero tal degradación debe reflejar los restantes niveles de incertidumbre.

PRINCIPIO #8. Una especificación debe ser localizada y débilmente acoplada. Los principios anteriores tratan con la especificación como una entidad estática. Esta surge de la dinámica de la especificación. Debe ser reconocido que aunque el principal propósito de una especificación sea servir como base para el diseño e implementación de algún sistema, no es un objeto estático compuesto, sino un objeto dinámico que sufre considerables modificaciones. Tales modificaciones se presentan en tres actividades principales: formulación, cuando se está creando una especificación inicial; desarrollo, cuando la especificación se esta elaborando durante el proceso iterativo de diseño e implementación; y mantenimiento, cuando la especificación se cambia para reflejar un entorno modificado y/o requerimientos funcionales adicionales.

7. Métodos de Análisis de Requerimientos

Las metodologías de análisis de requerimientos combinan procedimientos sistemáticos con una notación única para analizar los dominios de información y funcional de un problema de software; suministra un conjunto de heurísticas para subdividir el problema y define una forma de representación para las visiones lógicas y físicas. En esencia, los métodos de análisis de requerimientos del software, facilitan al ingeniero de software aplicar principios de análisis fundamentales, dentro del contexto de un método bien definido.

La mayoría de los métodos de análisis de requerimientos son conducidos por la información. Estos es, el método suministra un mecanismo para representar el dominio de la información. Desde esta representación, se deriva la función y se desarrollan otras características de los programas.

El papel de los métodos de análisis de requerimientos, es asistir al analista en la construcción de "una descripción precisa e independiente" del elemento software de un sistema basado en computadora.

- **Metodologías de Análisis de Requerimientos**

  Las metodologías de análisis de requerimientos facilitan al analista la aplicación de los principios fundamentales del análisis de una manera sistemática.

  Características Comunes

  Aunque cada método introduce nueva notación y heurística de análisis, todos los métodos pueden ser evaluados en el contexto de las siguientes características comunes:

  1. Mecanismos para el análisis del dominio de la información
  2. Método de representación funcional
  3. Definición de interfaces
  4. Mecanismos para subdividir el problema
  5. Soporte de la abstracción
  6. Representación de visiones físicas y lógicas

  Aunque el análisis del dominio de la información se conduce de forma diferente en cada metodología, pueden reconocerse algunas guías comunes. Todos los métodos se enfocan (directa o indirectamente) al flujo de datos y al contenido o estructura de datos. En la mayoría de los casos el flujo se caracteriza en el contexto de las transformaciones (funciones) que se aplican para cambiar la entrada en la salida. El contenido de los datos puede representarse explícitamente usando un mecanismo de diccionario o, implícitamente, enfocando primero la estructura jerárquica de los datos.

  Las funciones se describen normalmente como transformaciones o procesos de la información. Cada función puede ser representada usando una notación específica. Una descripción de la función puede desarrollarse usando el lenguaje natural, un leguaje procedimental con reglas sintácticas informales o un lenguaje de especificación forma.}

8. Métodos de Análisis Orientados al Flujo de Datos

La información se transforma como un flujo a través de un sistema basado en computadora. El sistema acepta entrada de distintas formas; aplica un hardware, software y elementos humanos para transformar la entrada en salida; y produce una salida en distintas formas. La entrada puede ser una señal de control transmitida por un transductor, una serie de números escritos por un operador humano, un paquete de información transmitido por un enlace a red, o un voluminoso archivo de datos almacenado en memoria secundaria. La transformación puede comprender una sencilla comparación lógica, un complejo algoritmo numérico, o un método de inferencia basado en regla de un sistema

experto. La salida puede encender un sencillo led o producir un informe de 200 páginas. En efecto, un modelo de flujo de datos puede aplicarse a cualquier sistema basado en computadora independientemente del tamaño o complejidad.

Una técnica para representar el flujo de información a través del sistema basado en computadora se ilustra en la figura 4. La función global del sistema se representa como una transformación sencilla de la información, representada en la figura como una burbuja. Una o más entradas. Representadas como flechas con etiqueta, conducen la transformación para producir la información de salida. Puede observarse que el modelo puede aplicarse a todo el sistema o solo a un elemento de software. La clave es representar la información dada y producida por la transformación.

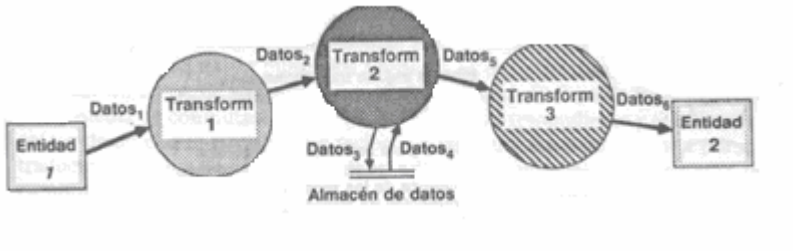


Figura 4.

9. Diagramas de Flujos de Datos

Conforme con la información se mueve a través del software, se modifica mediante una serie de transformaciones. Un diagrama de flujos de datos (DFD), es una técnica grafica que describe el flujo de información y las transformaciones que se aplican a los datos, conforme se mueven de la entrada a la salida. La forma básica de un DFD se ilustra en la figura 5. El diagrama es similar en la forma a otros diagramas de flujo de actividades, y ha sido incorporado en técnicas de análisis y diseños propuesto por Yourdon y Constantine, DeMarco y Gane y Sarson. También se le conoce como un grafo de flujo de datos o un diagrama de burbujas.

Figura 5.

10. Diccionario de Datos

Un análisis del dominio de la información puede ser incompleto si solo se considera el flujo de datos. Cada flecha de un diagrama de flujo de datos

representa uno o más elementos de información. Por tanto, el analista debe disponer de algún otro método para representar el contenido de cada flecha de un DFD.

Se ha propuesto el diccionario de datos como una gramática casi formal para describir el contenido de los elementos de información y ha sido definido da la siguiente forma: El diccionario de datos contiene las definiciones de todos los datos mencionados en el DFD, en una especificación del proceso y en el propio diccionario de datos. Los datos compuestos (datos que pueden ser además divididos) se definen en términos de sus componentes; los datos elementales (datos que no pueden ser divididos) se definen en términos del significado de cada uno de los valores que puede asumir. Por tanto, el diccionario de datos esta compuesto de definiciones de flujo de datos, archivos [datos almacenados] y datos usados en los procesos [transformaciones]...

- **Descripciones Funcionales**

  Una vez que ha sido representado el dominio de la información (usando un DFD y un diccionario de datos), el analista describe cada función (transformación) representada, usando el lenguaje natural o alguna otra notación estilizada. Una de tales notaciones se llama ingles estructurado (también llamado lenguaje de diseño del programa o proceso (LDP)). El ingles estructurado incorpora construcciones procedimentales básicas –secuencia, selección y repetición- junto con frases del lenguaje natural, de forma que pueden desarrollarse descripciones procedimentales precisas de las funciones representadas dentro de un DFD.

11. Métodos Orientados a la Estructura de Datos

Hemos ya observado que el dominio de la información para un problema de software comprende el flujo de datos, el contenido de datos y la estructura de datos. Los métodos de análisis orientados a la estructura de datos representan los requerimientos del software enfocándose hacia la estructura de datos en vez de al flujo de datos. Aunque cada método orientado a la estructura de datos tiene un enfoque y notación distinta, todos tienen algunas características en común: 1) todos asisten al analista en la identificación de los objetos de información clave (también llamados entidades o ítems) y operaciones (también llamadas acciones o procesos); 2)todos suponen que la estructura de la información es jerárquica; 3)todos requiere que la estructura de datos se represente usando la secuencia, selección y repetición; y 4) todos dan una conjunto de pasos para transformar una estructura de datos jerárquica en una estructura de programa.

Como los métodos orientados al flujo de datos, los métodos de análisis orientados a la estructura de datos proporcionan la base para el diseño de software. Siempre puede extenderse un método de análisis para que abarque el diseño arquitectural y procedimental del software.

12. Desarrollo de Sistemas de Jackson

El desarrollo de sistema de Jackson (DSJ) se obtuvo a partir del trabajo de M.A. Jackson sobre el análisis del dominio de la información y sus relaciones con el diseño de programas y sistemas. En palabras de Jackson: "El que desarrolla el software comienza creando un modelo de la realidad a la que se refiere el sistema, la realidad que proporciona su materia objeto [del sistema]..."

Para construir un DSJ el analista aplica los siguientes pasos:

1. Paso de las acciones y entidades. Usando un método muy similar a la técnica de análisis orientada al objeto, en este paso se identifican las entidades (persona, objetos u organizaciones que necesita un sistema para producir o usar información) y acciones (los sucesos que ocurren en el mudo real que afectan a las entidades).
2. Paso de estructuración de las entidades. Las acciones que afectan a cada entidad son ordenadas en el tiempo y representadas mediante diagramas de Jackson (una notación similar a un árbol).
3. Paso de modelación inicial. Las entidades y acciones se representan como un modelo del proceso; se definen las conexiones entre el modelo y el mundo real.
4. Paso de las funciones. Se especifican las funciones que corresponden alas acciones definidas.
5. Paso de temporización del sistema. Se establecen y especifican las características de planificación del proceso.
6. Paso de implementación. Se especifica el hardware y software como un diseño.

Los últimos tres pasos del DSJ están muy relacionados con el diseño de sistemas.

13. Requerimientos de las Bases de Datos

El análisis de requerimientos para una base de datos incorpora las mismas tareas que el análisis de requerimientos del software. Es necesario un contacto estrecho con el cliente; es esencial la identificación de las funciones e interfaces; se requiere la especificación del flujo, estructura y asociatividad de la información y debe desarrollarse un documento formal de los requerimientos.

- **Características de las bases de datos**

El termino base de datos se ha convertido en uno de los muchos tópicos del campo de las computadoras. Aunque existen muchas definiciones elegantes, definiremos una base de datos como: una colección de información organizada de forma que facilita el acceso, análisis y creación de informes. Una base de datos contiene entidades de información que están relacionadas vía organización y asociación. La arquitectura lógica de una base de datos se define mediante un esquema que representa las definiciones de las relaciones entre las entidades de información. La arquitectura física de una base de datos depende de la configuración del hardware residente. Sin embargo, tanto el esquema (descripción

lógica) como la organización (descripción física) deben adecuarse para satisfacer los requerimientos funcionales y de comportamiento para el acceso al análisis y creación de informes.

14. Análisis Económico y Técnico.

El análisis económico incluye lo que llamamos, el análisis de costos – beneficios, significa una valoración de la inversión económica comparado con los beneficios que se obtendrán en la comercialización y utilidad del producto o sistema.

Muchas veces en el desarrollo de Sistemas de Computación estos son intangibles y resulta un poco dificultoso evaluarlo, esto varía de acuerdo a las características del Sistema. El análisis de costos – beneficios es una fase muy importante de ella depende la posibilidad de desarrollo del Proyecto.

En el Análisis Técnico, el Analista evalúa los principios técnicos del Sistema y al mismo tiempo recoge información adicional sobre el rendimiento, fiabilidad, características de mantenimiento y productividad.

Los resultados obtenidos del análisis técnico son la base para determinar sobre si continuar o abandonar el proyecto, si hay riesgos de que no funcione, no tenga el rendimiento deseado, o si las piezas no encajan perfectamente unas con otras.

15. Modelado de la arquitectura del Sistema.

Cuando queremos dar a entender mejor lo que vamos a construir en el caso de edificios, Herramientas, Aviones, Maquinas, se crea un modelo idéntico, pero en menor escala (mas pequeño).

Sin embargo cuando aquello que construiremos es un Software, nuestro modelo debe tomar una forma diferente, deben representar todas las funciones y subfunciones de un Sistema. Los modelos se concentran en lo que debe hacer el sistema no en como lo hace, estos modelos pueden incluir notación gráfica, información y comportamiento del Sistema.

Todos los Sistemas basados en computadoras pueden modelarse como transformación de la información empleando una arquitectura del tipo entrada y salida.

16. Especificaciones del Sistema.

Es un Documento que sirve como fundamento para la Ingeniería Hardware, software, Base de datos, e ingeniería Humana. Describe la función y rendimiento de un Sistema basado en computadoras y las dificultades que estarán presentes durante su desarrollo. Las Especificaciones de los requisitos del software se producen en la terminación de la tarea del análisis.
En Conclusión un proyecto de desarrollo de un Sistema de Información comprende varios componentes o pasos llevados a cabo durante la etapa del análisis, el cual ayuda a traducir las necesidades del cliente en un modelo de Sistema que utiliza

uno más de los componentes: Software, hardware, personas, base de datos, documentación y procedimientos.

ACTIVIDAD…

- Aplicar las diferentes etapas del proceso a un caso de estudio presentado por el docente
- Aplicar las diferentes etapas del proceso a un caso presentado por el estudiante

ESTRATEGIAS METODOLÓGICAS

- Presentación de la unidad a cargo del profesor.
- Desarrollo del proceso aplicado a un caso específico a cargo del docente.

RECURSOS

- Humanos: Profesor y alumnos
- Institucionales: Salón de clase
- Materiales: Texto guía

INDICADORES DE EVALUACIÓN

- Evaluación escrita sobre la unidad
- Evaluación del taller proporcionado

CRITERIOS DE EVALUACIÓN

- ¿Qué objetivos tiene la planeación del software?
- ¿Qué aspectos se deben tener en cuenta en el estudio de factibilidad?
- ¿Cuál es el principio en el que se basa la estimación?
- ¿Cuáles son los modelos de estimación?
- ¿Cuál es la importancia de la definición de requerimientos?
- ¿Cuáles son las tareas a realizar en el análisis de requerimientos?

UNIDAD 4

FORMULACIÓN DE UN PROYECTO DE SOFTWARE

INTRODUCCIÓN

Esta unidad busca que el estudiante de la Media Técnica en el Área de Programación desarrolle la capacidad de analizar, diseñar y plantear un proyecto, utilizando las tecnologías adecuadas para el fin buscado.

JUSTIFICACIÓN

La apropiación del conocimiento por parte del estudiante se logra con la praxis. Por ello se requiere la aplicación de las herramientas teóricas de la Ingeniería del software a un caso practico real, para lo cual se planteará un proyecto de grado, que se implementará en grado 11 y el cual será requisito para poderse graduar en la modalidad de la Medía Técnica en programación.

OBJETIVO GENERAL

Proporcionar los conocimientos que permitan al estudiante Plantear un proyecto de desarrollo de software

OBJETIVOS ESPECÍFICOS

- Identificar la necesidades de información en una organización o empresa real
- Definir los requerimientos y utilizar los procesos conocidos durante el presente curso para analizar la pertinencia en la utilización de un desarrollo de software o pagina web dinámica en la solución de un problema de información real.
- Determinar y justificar la importancia de la solución, así como su impacto social y / o empresarial.

El proyecto será revisado y aprobado por un comité que el Coordinador general del Proyecto de parte de Politécnico Colombiano JIC determinará.

El proyecto será presentado en el formato que se detalla y anexa a continuación.

MEDIA TÉCNICA EN PROGRAMACIÓN
Convenio
Politécnico Colombiano JIC Alcaldía de Medellín.

PROYECTO DE TRABAJO DE GRADO

Colegio :_____

Años para los cuales aplica 200_ y 200_

Nombre del estudiante	Doc. Identidad	e-mail
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

Nombre del posible proyecto : _____

Nombre de la empresa o institución donde se efectuará
_____

NIT de la empresa: _____

Actividad económica: _____

Dirección: _____
Teléfono (s): _____ e- mail: _____
Área práctica: _____
Responsable de la empresa: _____
Asesor : _____

Justificación: _____
_____
_____
_____

Objetivo general:
_____
_____
_____
_____
_____
_____
_____
_____

Objetivos específicos:

1. _____
_____

2. _____
_____

3. _____
_____

Alcance del proyecto :
_____
_____
_____
_____

Impacto social: _(población beneficiada por el proyecto)
_____
_____
_____
_____
_____

Intensidad horaria semanal de la práctica: _____
Fecha de iniciación: _____
Fecha de terminación esperada: _____

OBSERVACIONES:
_____
_____
_____
_____
_____

FIRMA DEL ESTUDIANTE: FIRMA RESPONSABLE EMPRESA

_____ _____

FIRMA ASESOR PRÁCTICA:

_____

Favor entregar este formato en original y copia al Comité de Evaluación adjuntando el cronograma de actividades del proyecto

CRONOGRAMA DE ACTIVIDADES										
ACTIVIDADES	MESES									
	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										
15										

NOMBRE DEL POSIBLE PROYECTO

El nombre debe estar de máximo 12 palabras, debe terminar con el nombre de la empresa o institución para la cual se desarrolla.
El nombre debe ser corto, claro y conciso de forma que se entienda muy bien de que trata el proyecto.
Ejemplo: Sitio web colegio San Martín

NOMBRE DE LA EMPRESA O INSTITUCIÓN

Es el nombre de la empresa o instución para la cual se hará el trabajo de grado

NIT DE LA EMPRESA

Aquí se coloca el numero de identificación tributaria NIT de la empresa

ACTIVIDAD ECONÓMICA

Aquí se coloca el objeto social o área económica de generación de productos o servicios a los cuales se dedica la empresa o institución.

DIRECCIÓN

Aquí figura la dirección que tiene registrada la empresa en el certificado de existencia y representación legal en cámara de comercio.

TELÉFONO

Aquí se coloca el teléfono de la empresa

E-MAIL

Aquí se coloca el correo electrónico de la empresa

ÁREA PRÁCTICA

Aquí se coloca el lugar especifico donde se va a manejar el producto de dicha empresa, por ejemplo: área de ventas, producción, sistemas.

RESPONSABLE DE LA EMPRESA

Es el encargado de dirigir de que la empresa marche bien y de que los productos ofertados vayan a un destinatario (clientes, usuario final).

ASESOR

Es el responsable de dar orientación a la persona encargada de elaborar el producto, que cumpla con unos estándares en dicha elaboración para la entrega final.

JUSTIFICACIÓN

Aquí se coloca los argumentos que demuestren la importancia del estudio, para que se logren los resultados, como se aplicaran, que necesidades resuelven, cuales son las innovaciones.

En la justificación del proyecto se debe dar respuesta a las siguientes preguntas:

¿Por qué se hace el trabajo?
¿Cuál es la razón, el motivo del trabajo?
¿Qué importancia tiene?
La importancia puede ser por que ayuda en:

- La solución de problemas
- La toma de decisiones.
- La creación de instrumentos de control
- La articulación de conocimientos teóricos o prácticos
- La aplicación de conocimientos teóricos
- El cumplimiento de un requisito para la obtención de un título

OBJETIVO GENERAL

Es el propósito final del trabajo y expresa los resultados amplios que se pretenden lograr.

El *objetivo general* ofrece propósitos amplios. Pueden ser varios.

El alcance de los objetivos debe estar dentro de las posibilidades de quien realiza el trabajo y referirse a la consecución de resultados.

La presentación formal de los objetivos pueden plantearse mediante verbos en infinitivo que señalen la acción de quien lo formula. Ejemplo: identificar, plantear, realizar, describir, estudiar, examinar, evaluar, recomendar, revisar, calcular, diseñar, inventariar, presupuestar, verificar

OBJETIVOS ESPECÍFICOS

Dan cuentan de lo que se pretende realizar en cada una de las etapas del proyecto para lograr el objetivo general.

Los *objetivos específicos* se refieren a situaciones particulares que inciden o forman parte de situaciones propias de los objetivos generales.

Se deben plantear varios objetivos específicos por cada objetivo general. El resultado de los objetivos específicos permite alcanzar el objetivo general.

Responden a las preguntas:
¿Qué se quiere hacer?
¿Qué propósitos tiene el trabajo?
¿Qué se busca conocer?
¿A donde se quiere llegar?

ALCANCE DEL PROYECTO

Aquí se define hasta donde se llevará técnica y tecnolóogicamente el trabajo a realizar, esta muy ligado a los objetivos y la formulación o descripción el área problemática a solucionar.

IMPACTO SOCIAL

Son las repercuciones sociales del proyecto, en el se explica que personas se van a ver beneficiadas por el proyecto y de que forma.

Ejemplo: en un sistema de liquidación para vendedores el impacto social esta relacionado con sus 3000 vendedores y 5 personas del personal que elabora la liquidación y que verá facilitada sus labores y hará uso mas eficiente de su tiempo de trabajo aumentando con ello su productividad.

BIBLIOGRAFÍA / CIBERGRAFÍA

www.wikipedia.org /wiki/Sistema - Definición dentro de contexto
www.itver.edu.mx/areas/deptos/sc/ - 1k - En caché - Páginas similares
FUENTE : PSL Productora de Software
Corporación Colombia Digital
Carrera 13 No.90-36, Of. 405 Tel (571)611-3059, Fax (571) 6113028. Bogotá, Colombia
www.Colombiadigital.net info@ColombiaDigital.net
www.conocimientosweb.net/dcmt/ficha1643.html - 15k - En caché - Páginas similares
www.intersoft.com.bo/portal/servicios/gestioncalidad.php - 12k - En caché - Páginas similares
www.cecam.sld.cu/pages/rcim/revista_1/articulos_pdf/r0100a01.pdf - Páginas similares
Fowler, Martin. UML Distilled. Addison -Wesley Longman Inc.1997
Sommerville, Ian. Ingeniería de Software. 6ª Edición. Addison Wesley. 2002
Pressman, Roger. Ingeniería del Software. 5ª Edición. Mc Graw Hill. 2002
Ruble, David. Análisis y Diseño Práctico de Sistemas Cliente/Servidor con GUI. Prentice - Hall, 1998.
Larman, Craig. UML y patrones. Introducción al análisis y diseño orientado a objetos. Ed. Prentice – Hall, 1999.
Rumbaugh James. Modelado y diseño orientado a objetos. Ed. Prentice – Hall, 1991.
Booch Grady, Rumbaugh James, Jacobson Ivar. El lenguaje Unificado de Modelado. Ed. Addison Wesley, 1999.
McCONNEL, Steve. Desarrollo y gestión de proyectos informáticos. McGraw Hill. España, 1997.