INSTITUTO NACIONAL GENERAL FRANCISCO MORAZÁN

Integrantes:

Karla Marcela Mendoza Ruíz N° 5
Alejandra Patricia Oviedo Guardado N° 9
Ingrid Melissa Ponce Aguilar N° 16
Mónica Yamileth Ponce Menjívar N° 17
Emily Hazel Ramírez Chamagua N° 18

Profesora: Inga. Mayra González

Grado: 2B ITSI

Tema de exposición: Procedimiento y construcción de diagramas de flujo, simbología, diccionario de datos, otros.

Asignatura: Módulo Informático.

Año: 2015

ÍNDICE

INTRODUCCIÓN

El propósito de Desarrollo de Software es la realización sistemática de las actividades de planeación, diseño, codificación, pruebas, lanzamiento de productos de software nuevos cumpliendo con los requisitos especificados y con las normativas de seguridad de información. El proceso de Desarrollo de Software apoyado sobre la metodología del desarrollo ágil adaptando la programación Extrema (XP) la cual se compone de uno o más ciclos de desarrollo. Cada ciclo está compuesto de las siguientes fases

OBJETIVOS

- ✓ Conocer el ciclo de vida de desarrollo de software.
- ✓ Identificar procesos y construcción del desarrollo de software
- ✓ Investigar con interés las técnicas y conceptos de desarrollo de software
- ✓ Indagar sobre los diferentes métodos del desarrollo de software
- ✓ Conocer sobre la lógica computacional, procedimiento de diagrama de entidad-relación y diagramas de flujo.

1. REALIZAR UNA INVESTIGACIÓN DOCUMENTAL SOBRE DESARROLLO DE SOFTWARE.

Un **proceso para el desarrollo de software**, también denominado **ciclo de vida del desarrollo de software** es una estructura aplicada al desarrollo de un producto de software. Hay varios modelos a seguir para el establecimiento de un proceso para el desarrollo de software, cada uno de los cuales describe un enfoque diferente para diferentes actividades que tienen lugar durante el proceso. Algunos autores consideran un modelo de ciclo de vida un término más general que un determinado proceso para el desarrollo de software. Por ejemplo, hay varios procesos de desarrollo de software específicos que se ajustan a un modelo de ciclo de vida de espiral.

Generalidades

La gran cantidad de organizaciones de desarrollo de software implementan metodologías para el proceso de desarrollo. Muchas de estas organizaciones pertenecen a la industria armamentística, que en los Estados Unidos necesita un certificado basado en su modelo de procesos para poder obtener un contrato.

El estándar internacional que regula el método de selección, implementación y monitoreo del ciclo de vida del software es **ISO 12207.**

ISO/IEC 12207 Information Technology / Software Life Cycle Processes, es el estándar para los procesos de ciclo de vida del software de la organización ISO.

Planificación

La importante tarea a la hora de crear un producto de software es obtener los requisitos o el análisis de los requisitos. Los clientes suelen tener una idea más bien abstracta del resultado final, pero no sobre las funciones que debería cumplir el software.

Una vez que se hayan recopilado los requisitos del cliente, se debe realizar un análisis del ámbito del desarrollo. Este documento se conoce como especificación funcional.

Implementación, pruebas y documentación

La implementación es parte del proceso en el que los ingenieros de software programan el código para el proyecto.

Las pruebas de software son parte esencial del proceso de desarrollo del software. Esta parte del proceso tiene la función de detectar los errores de software lo antes posible.

La documentación del diseño interno del software con el objetivo de facilitar su mejora y su mantenimiento se realiza a lo largo del proyecto. Esto puede incluir la documentación de un API, tanto interior como exterior.

Despliegue y mantenimiento

El despliegue comienza cuando el código ha sido suficientemente probado, ha sido aprobado para su liberación y ha sido distribuido en el entorno de producción.

El mantenimiento o mejora del software de un software con problemas recientemente desplegado, puede requerir más tiempo que el desarrollo inicial del software. Es posible que haya que incorporar código que no se ajusta al diseño original con el objetivo de solucionar un problema o ampliar la funcionalidad para un cliente. Si los costes de mantenimiento son muy elevados puede que sea oportuno rediseñar el sistema para poder contener los costes de mantenimiento.

¿Cuál es la Importancia del Desarrollo de Software?

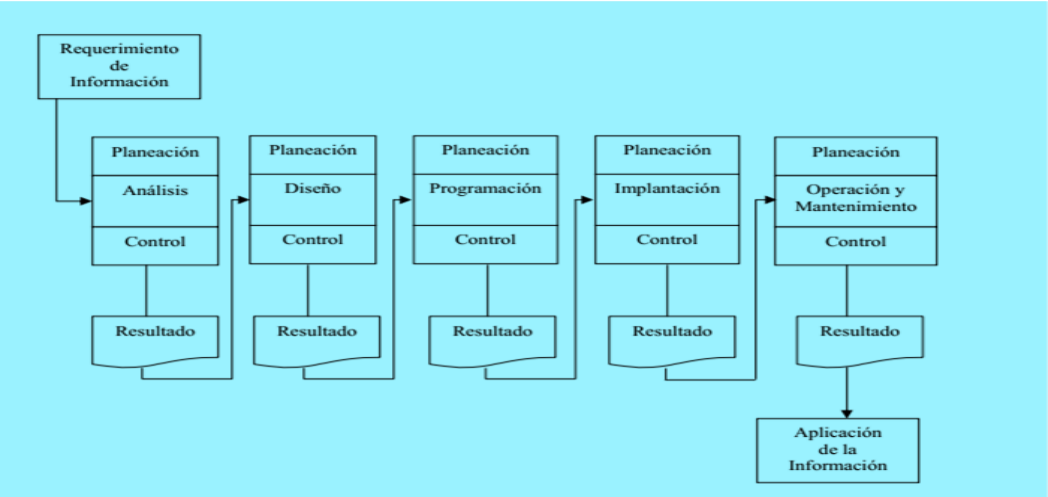
Actualmente la transición que estamos viviendo hacia una sociedad del conocimiento a cambiado profundamente las relaciones entre las personas, empresas y gobiernos: las empresas usan la red para comunicarse con los clientes, utilizan también herramientas de gestión del conocimiento para hacer masa eficientes, los gobiernos mejoran su presencia en Internet y los servicios a los ciudadanos a través de la red, los usuarios usan las herramientas para sus relaciones personales, etc. Se va de forma imparable hacia una sociedad altamente interconectada donde el eje fundamental es la información.
El software es el intermediario cada vez más grande entre la información y la inteligencia humana. De la misma manera que preocupa para poder acceder a la información, si existe la censura, es tema de preocupación de quien controla este intermediario y las garantías de su transparencia y con fiabilidad.
El desarrollo del software y la programación es uno de los pilares fundamentales de la informática y al cual se dedican muchas horas de esfuerzos en empresas, colegios, academias y universidades. Conforme a la tecnología va avanzando, van apareciendo nuevas soluciones, nuevas formas de programación, nuevos lenguajes y un sin fin de herramientas que intentan realizar el trabajo del desarrollador un poco más fácil. La programación orientadas a objetos o los compiladores basados en maquinas virtuales (en muchos casos, multiplataforma), también a sus puestos unas renovación en la manera de programar.

2. INVETIGAR SOBRE LOS PROCESOS, MÉTODOS Y HERRAMIENTAS DE SOFTWARE UTILIZADOS EN LOS SISTEMAS DE INFORMACIÓN.

Procesos:

CICLO DE DESARROLLO 2.1 DESCRIPCIÓN La construcción de un sistema de información implica la conjugación de esfuerzos, conocimientos, experiencias, recursos y tiempo muy valiosos; por lo que es necesario contar con un adecuado rumbo de acción que garantice el éxito del proyecto, empleado al máximo los elementos disponibles. Por esta razón es conveniente apoyarse en una metodología que establezca las etapas con objetivos, actividades y técnicas necesarias en la creación de un sistema. La ejecución de las etapas lleva normalmente un orden cronológico, en donde los resultados producidos por una, alimentan las funciones de la siguiente y así sucesivamente. Además se aplican los conceptos del proceso administrativo, que regula las acciones de cada etapa y por lo tanto, de todo el proyecto. Por ello es necesario planear y controlar la realización de las actividades. Para ilustrar lo anterior, se presenta una figura en donde aparecen las etapas que componen el ciclo de desarrollo de los sistemas, sus interrelaciones y el papel del proceso administrativo.

• ETAPAS DE DESARROLLO El papel que juega cada una de las etapas que conforman el ciclo de desarrollo de los sistemas de información es el siguiente: Análisis: define los requerimientos de información y la forma más adecuada de atenderlos. Diseño: describe la estructura, funciones e interrelaciones de los componentes del sistema. Programación: elabora los elementos del sistema (programas, archivos, reportes, etc.). Implantación: prueba e instala el sistema construido. Operación: hace uso del sistema. Mantenimiento: modifica el sistema en función a los nuevos requerimientos que se van presentando. Asimismo, en cuanto a la participación del proceso administrativo se refieren: • Planeación: se establecen los objetivos, estrategias, programas de acción y determinación de recursos. • Control: se evalúan los resultados conforme a las metas esperadas, se realizan las correcciones necesarias influenciando la ejecución de la siguiente capa.



Métodos de ingeniería de software:
Los métodos son herramientas computacionales que están destinadas a asistir en los procesos de ciclo de vida de un software, estos son estructurados para el desarrollo del software, también facilitan la producción del software y se basan principalmente en la idea de un modelo gráfico. No existe un método ideal para la elaboración de un software. Son enfoques estructurados para el desarrollo del software.

Existen diferentes procesos en el tema ingeniería de software, que tienen como objetivo presentar diferentes técnicas que consisten en la combinación de procedimientos que permiten guiar el diseño y el desarrollo de sistemas de software a un producto final de calidad. Algunos de esos procesos son: modelo secuencial, mode lo en espiral, modelo win win, rational unified process (RUP), extreme programming (XP), método delphi, métrica versión 3, modelo de madurez de capacidad (CMM), proceso personal de software (PSP) y proceso en equipo de software (TSP).

1. Modelo en espiral
El modelo espiral, propuesto originalmente por B. Boehm, es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo en cascada [1], añadiendo un nuevo elemento: el análisis de riesgo. El modelo, representado mediante la espiral de la figura

2. Modelo en espiral (win-win)
El modelo en espiral tratado anteriormente sugiere una actividad del marco de trabajo que aborda la comunicación con el cliente.
El objetivo de esta actividad es mostrar los requisitos del cliente.

3. Método Delphi
El método Delphi consiste en una serie de interrogaciones repetidas, usualmente por medio de cuestionarios a un grupo de individuos, cuyas opiniones o juicios son de interés. Después de un interrogatorio inicial a cada individuo, cada subsiguiente interrogatorio es acompañado por información considerando las respuestas precedentes, usualmente presentadas anónimamente.

4. Modelo de madurez de capacidad (CMM)
Al final de los años ochenta y a comienzos de los noventa el SEI desarrolló el modelo de madurez de capacidad CMM que capturó las mejores prácticas de las organizaciones para el desarrollo de software.

La siguiente clasificación es la más habitual basada en las fases del ciclo de desarrollo que cubren:

- *Upper CASE (U-CASE)*, herramientas que ayudan en las fases de planificación, análisis de requisitos y estrategia del desarrollo, usando, entre otros diagramas UML.
- *Middle CASE (M-CASE)*, herramientas para automatizar tareas en el análisis y diseño de la aplicación.

- *Lower CASE (L-CASE)*, herramientas que semi-automatizan la generación de código, crean programas de detección de errores, soportan la depuración de programas y pruebas. Además automatizan la documentación completa de la aplicación. Aquí pueden incluirse las herramientas de Desarrollo rápido de aplicaciones.

Existen otros nombres que se le dan a este tipo de herramientas, y que no es una clasificación excluyente entre sí, ni con la anterior:

- *Integrated CASE (I-CASE)*, herramientas que engloban todo el proceso de desarrollo software, desde análisis hasta implementación.
- *MetaCASE*, herramientas que permiten la definición de nuestra propia técnica de modelado, los elementos permitidos del metamodelo generado se guardan en un repositorio y pueden ser usados por otros analistas, es decir, es como si definiéramos nuestro propio UML, con nuestros elementos, restricciones y relaciones posibles.
- *CAST (Computer-Aided Software Testing)*, herramientas de soporte a la prueba de software.
- *IPSE (Integrated Programming Support Environment)*, herramientas que soportan todo el ciclo de vida, incluyen componentes para la gestión de proyectos y gestión de la configuración activa.

3. INVESTIGAR CONCEPTOS Y TÉCNICAS DE DESARROLLO DE SORFTWARE (AGILE SORFTWARE DEVEPLOMENT)

- **Software de sistema:** Su objetivo es desvincular adecuadamente al usuario y al programador de los detalles del sistema informático en particular que se use, aislándolo especialmente del procesamiento referido a las características internas de: memoria, discos, puertos y dispositivos de comunicaciones, impresoras, pantallas, teclados, etc. El software de sistema le procura al usuario y programador

adecuadas interfaces de alto nivel, controladores, herramientas y utilidades de apoyo que permiten el mantenimiento del sistema global. Incluye entre otros:

- Sistemas operativos
- Controladores de dispositivos
- Herramientas de diagnóstico
- Herramientas de Corrección y Optimización
- Servidores
- Utilidades

- **Software de programación:** Es el conjunto de herramientas que permiten al programador desarrollar programas informáticos, usando diferentes alternativas y lenguajes de programación, de una manera práctica. Incluyen básicamente:
  - Editores de texto
  - Compiladores
  - Intérpretes
  - Enlazadores
  - Depuradores
  - Entornos de Desarrollo Integrados (IDE): Agrupan las anteriores herramientas, usualmente en un entorno visual, de forma tal que el programador no necesite introducir múltiples comandos para compilar, interpretar, depurar, etc. Habitualmente cuentan con una avanzada interfaz gráfica de usuario (GUI).
- **Software de aplicación**: Es aquel que permite a los usuarios llevar a cabo una o varias tareas específicas, en cualquier campo de actividad susceptible de ser automatizado o asistido, con especial énfasis en los negocios. Incluye entre muchos otros:
  - Aplicaciones para Control de sistemas y automatización industrial
  - Aplicaciones ofimáticas
  - Software educativo
  - Software empresarial
  - Bases de datos
  - Telecomunicaciones (por ejemplo Internet y toda su estructura lógica)
  - Videojuegos
  - Software médico
  - Software de cálculo numérico y simbólico.
  - Software de diseño asistido (CAD)

- Software de control numérico (CAM)

Técnicas de desarrollo de software:
Existen 3 tipos de técnicas dentro del proceso de desarrollo de software que son:

Técnicas para la recopilación de datos:

Observación: «La observación es una técnica que consiste en visualizar o captar mediante la vista, en forma sistemática, cualquier hecho, fenómeno o situación que se produzca en la naturaleza o en la sociedad, en función de unos objetivos de investigación preestablecidos

Entrevista: «La entrevista, más que un simple interrogatorio, es una técnica basada en un diálogo o conversación «cara a cara», entre el entrevistador y el entrevistado acerca de un tema previamente determinado, de tal manera que el entrevistador pueda obtener la información requerida

Encuesta: «Se define la encuesta como una técnica que pretende obtener información que suministra un grupo o muestra de sujetos acerca de sí mismos, o en relación con un tema en partícula

Cuestionario: «Es la modalidad de encuesta que se realiza de forma escrita mediante un instrumento o formato en papel contentivo de un aserie de preguntas

Técnica de Costo-Beneficio

Mejora de Procesos: Conducen a reducción de tiempo y recursos
Disponer de Sistemas de Información: Mejora la toma de decisiones y obtención de ingresos
Personal Motivado: Creciente moral del personal al funcionar en un entorno de herramientas modernas para el negocio.
Intangibles: Otros beneficios intangibles que sean identificados y cuantificables

Técnica de Planificación y Control de Proyectos

1) Planificación
2) Descomponer el proyecto en actividades distintas. Luego, se determinan las estimaciones de tiempo para cada actividad y se
3) construyen diagramas de red para estas actividades.
4) Programación Construir un gráfico de tiempo donde se muestran los tiempos de iniciación y terminación para cada actividad y la relación con el resto de las actividades del proyecto.
5) Control Comprende el uso del diagrama de flechas y la gráfica de tiempo para hacer

reportes periódicos del progreso.
6) Se debe analizar la secuencia de las actividades y, si es necesario, determinar un nuevo programa para la parte restante del proyecto.

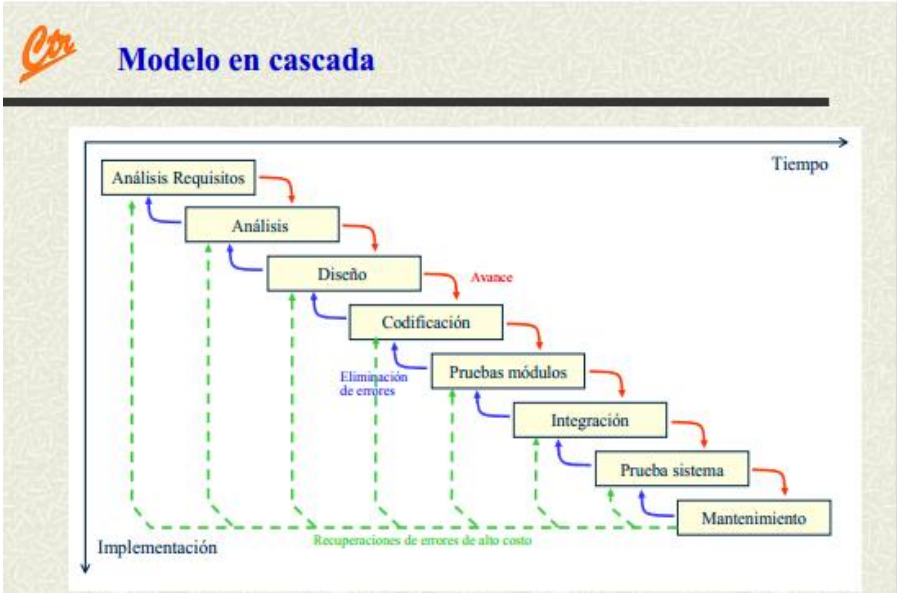
4. PROCEDIMIENTOS DE CONSTRUCCIÓN DE APLICACIONES DE SOFTWARE

Los elementos básicos de un proceso de desarrollo de software es definir los papeles que juegan los trabajadores, las actividades que desarrollan y los productos que deben generarse. En un plan de desarrollo cada trabajador debe tener su papel dentro de él, lo que define las actividades que debe realizar y los productos que debe generar. Las actividades son las tareas que deben realizar los trabajadores para cumplir sus obligaciones. A alto nivel, estas actividades son concebidas como las fases del proceso (especificación, análisis, ..), mientras que a mas bajo nivel son tareas más concretas (crear cierto diagramas, escribir código,..). Los productos son los documentos o información que debe ser creada como consecuencia de la actividad que se desarrolla. El producto último es el sistema que se desarrolla, pero en las fases intermedias deben generarse una amplia gama de documentos intermedios. Cada actividad debe tener siempre como principal objetivo generar ciertos productos bien definidos y especificados. Los procesos deben estar condicionados por el tipo de producto que se desarrolla y por la tradición y experiencia de la empresa que lo desarrolla.

Una de las propiedades que deben ser exigidas a un proceso de desarrollo de aplicaciones software es la escalabilidad, lo que hace posible que sea aplicable tanto a sistemas complejos como a sistemas sencillos. En general la propiedad de escalabilidad representa que si para desarrollar un proyecto de complejidad (y) es necesario realizar un esfuerzo (x), para desarrollar un proyecto de complejidad (100y) se requiere un esfuerzo (100cx) (donde c es una constante). Cuando un proyecto crece, se produce que: •Sus objetivos se hacen menos concretas y mas globales. •El sistema

gana en niveles de abstracción (tales como subsistemas, subsistemas, y así). •Líneas potenciales de comunicación crece exponencialmente con el número de miembros del proyecto. •El costo de los errores que hay que corregir se incrementa ya que aumentan las posibilidades de interferencias. En general, existen dos soluciones para conseguir la escalabilidad de un proceso: •El proceso es visto desde diferentes escalas de tiempo: macro, micro y nano escala, y en función de que el proyecto crezca más relevancia adquieren la escala macro a fin de organizar y gestionar el proceso de desarrollo más global. •Muchas de la fase y mecanismos del proceso se hacen opcionales en función de que la complejidad de la aplicación se requiera.

La siguiente es una secuencia común de las actividades para un proyecto software: 1. Es necesario comprender la naturaleza del proyecto. Esto parece obvio, pero casi siempre lleva tiempo entender lo que desean los clientes, en especial cuando ellos mismos no saben por completo que quieren. 2. Los proyectos requieren documentación desde el principio; es muy probable que esta documentación sufra muchos cambios. Por esta razón, desde el principio debe disponerse de una estrategia para mantener los documentos que se generen. Este proceso es denominado “Gestión de la Configuración”. 3. Hay que reunir los requisitos que ha de cumplir la aplicación. Gran parte de esta actividad es conversar con los “interesados”. 4. Hay que analizar el problema, diseñar la solución y codificar los programas. 5. El producto inicial y final debe probarse en forma exhaustiva en todos sus aspectos. 6. Una vez entregado el producto, entra el modo “mantenimiento, que incluye reparaciones y mejoras. Es una actividad que consume muchos recursos, a veces hasta el 65% de los recursos utilizados en el desarrollo de la aplicación. Ello hace hace que ha sea considerado un objetivo fundamental.



5. ELABORAR UN COMPENDIO SOBRE MÉTODOS DE DESARROLLO DE SOFTWARE.

El **desarrollo ágil de software** refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan mediante la colaboración de grupos auto organizado y multidisciplinario. Existen muchos métodos de desarrollo ágil; la mayoría minimiza riesgos desarrollando software en lapsos cortos. El software desarrollado en una unidad de tiempo es llamado una iteración, la cual debe durar de una a cuatro semanas. Cada iteración del ciclo de vida incluye: planificación, análisis de requisitos, diseño, codificación, revisión y documentación. Una iteración no debe agregar demasiada funcionalidad para justificar el lanzamiento del producto al mercado, sino que la meta es tener una «demo» (sin errores) al final de cada iteración. Al final de cada iteración el equipo vuelve a evaluar las prioridades del proyecto.

Los métodos ágiles enfatizan las comunicaciones cara a cara en vez de la documentación. La mayoría de los equipos ágiles están localizados en una simple oficina abierta, a veces llamadas "plataformas de lanzamiento" (*bullpen* en inglés). La oficina debe incluir revisores, escritores de documentación y ayuda, diseñadores de iteración y directores de proyecto. Los métodos ágiles también enfatizan que el software funcional es la primera medida del progreso. Combinado con la preferencia por las comunicaciones cara a cara, generalmente los métodos ágiles son criticados y tratados como "indisciplinados" por la falta de documentación técnica.

MÉTODOS ÁGILES

Algunos métodos ágiles de desarrollo de software:

- Adaptive Software Development (ASD)
- Agile Unified Process (AUP)
- Crystal Clear
- Feature Driven Development (FDD)
- Lean Software Development (LSD)
- Kanban
- Open Unified Process (OpenUP)
- Programación Extrema (XP)
- Método de desarrollo de sistemas dinámicos (DSDM)
- Scrum

METODOLOGÍAS ÁGILES

En una reunión celebrada en febrero de 2001 en Utah-EEUU, nace el término "ágil" aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas. Varias de las denominadas metodologías ágiles ya estaban siendo utilizadas con éxito en proyectos reales, pero les faltaba una mayor difusión y reconocimiento.

Tras esta reunión se creó *The Agile Alliance*[3], una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida es fue el Manifiesto Ágil, un documento que resume la filosofía "ágil".

6. PREPARAR UNA FICHA PARA ESPECÍFICAR EL PROCEDIMEINTO DE LA INGENIERÍA DE SOFTWARE Y CADA EQUIPO ANALIZA EL PROCESO LOS DATOS OBTENIDOS DE LA INGENIERÍA DE SOFTWARE

Etapas del proceso

La ingeniería de software requiere llevar a cabo numerosas tareas agrupadas en etapas, al conjunto de estas etapas se le denomina ciclo de vida. Las etapas comunes a casi todos los modelos de ciclo de vida son las siguientes:

Obtención de los requisitos

Se debe identificar sobre que se está trabajando es decir, el tema principal que motiva el inicio del estudio y creación del nuevo software o modificación de uno ya existente. a su vez identificar los recursos que se tienen, en esto entra el conocer los recursos humanos y materiales que participan en el desarrollo de las actividades. Es importante entender el contexto del negocio, para identificar adecuadamente los requisitos.

Se tienen que tener dominio de información de un problema lo cual incluye los datos fuera del software (usuarios finales, otros sistemas o dispositivos externos), los datos salen del sistema (por la interfaz de usuario, interfaces de red, reportes, gráficas y otros medios) y los almacenamientos de datos que recaban y organizan objetos persistentes de datos (por ejemplo, aquellos que se conservan de manera permanente).

También hay que ver los puntos críticos lo que significa tener de una manera clara los aspectos que entorpecen y limitan el buen funcionamiento de los procedimientos actuales, los problemas más comunes y relevantes que se presentan, los motivos que crean insatisfacción y aquellos que deben ser cubiertos a plenitud. Por ejemplo: ¿El contenido de los reportes generados, satisface realmente las necesidades del usuario? ¿Los tiempos de respuesta ofrecidos, son oportunos?, etc.

Hay que definir las funciones que realizara el software ya que estas ayudan al usuario final y al funcionamiento del mismo programa.

Se tiene que tener en cuenta cómo será el comportamiento del software antes situaciones inesperadas como lo son por ejemplo una cantidad de usuarios enormes usando el software o una gran cantidad de datos entre otros.

Análisis de requisitos

Extraer los requisitos de un producto software es la primera etapa para crearlo. Durante la fase de análisis, el cliente plantea las necesidades que se presenta e intenta explicar lo que debería hacer el software o producto final para satisfacer dicha necesidad mientras que el desarrollador actúa como interrogador, como la persona que resuelve problemas. Con este análisis, el ingeniero de sistemas puede elegir la función que debe realizar el software y establecer o indicar cuál es la interfaz más adecuada para el mismo.

El análisis de requisitos puede parecer una tarea sencilla, pero no lo es debido a que muchas veces los clientes piensan que saben todo lo que el software necesita para su buen funcionamiento, sin embargo se requiere la habilidad y experiencia de algún especialista para reconocer requisitos incompletos, ambiguos o contradictorios. Estos requisitos se determinan tomando en cuenta las necesidades del usuario final, introduciendo técnicas que nos permitan mejorar la calidad de los sistemas sobre los que se trabaja.

El resultado del análisis de requisitos con el cliente se plasma en el documento ERS (especificación de requisitos del sistema), cuya estructura puede venir definida por varios estándares, tales como CMMI. Asimismo, se define un diagrama de entidad/relación, en el que se plasman las principales entidades que participarán en el desarrollo del software.

La captura, análisis y especificación de requisitos (incluso pruebas de ellos), es una parte crucial; de esta etapa depende en gran medida el logro de los objetivos finales. Se han ideado modelos y diversos procesos metódicos de trabajo para estos fines. Aunque aún no está formalizada, ya se habla de la ingeniería de requisitos.

La IEEE Std. 830-1998 normaliza la creación de las especificaciones de requisitos de software (Software Requirements Specification).

Finalidades del análisis de requisitos:

- Brindar al usuario todo lo necesario para que pueda trabajar en conjunto con el software desarrollado obteniendo los mejores resultados posibles.
- Tener un control más completo en la etapa creación del software, en cuanto a tiempo de desarrollo y costos.
- Utilización de métodos más eficientes que permitan el mejor aprovechamiento del software según sea la finalidad de uso del mismo.
- Aumentar la calidad del software desarrollado al disminuir los riesgos de mal funcionamiento.

No siempre en la etapa de "análisis de requisitos" las distintas metodologías de desarrollo llevan asociado un estudio de viabilidad y/o estimación de costes. El más conocido de los modelos de estimación de coste del software es el modelo COCOMO

Limitaciones

El software tiene la capacidad de emular inteligencia creando un modelo de ciertas características de la inteligencia humana pero sólo posee funciones predefinidas que abarcan un conjunto de soluciones que en algunos campos llega a ser limitado. Aun cuando tiene la capacidad de imitar ciertos comportamientos humanos no es capaz de emular el pensamiento humano porque actúa bajo condiciones.

Otro aspecto limitante de los software proviene del proceso totalmente mecánico que requiere de un mayor esfuerzo y tiempos elevados de ejecución lo que lleva a tener que implementar el software en una máquina de mayor capacidad.

Especificación

La especificación de requisitos describe el comportamiento esperado en el software una vez desarrollado. Gran parte del éxito de un proyecto de software radicará en la identificación de las necesidades del negocio (definidas por la alta dirección), así como la interacción con los usuarios funcionales para la recolección, clasificación, identificación, priorización y especificación de los requisitos del software.

El proceso

Como el software, al igual que el capital, es el conocimiento incorporado, y puesto que el conocimiento está inicialmente disperso, el desarrollo del software implícito, latente e incompleto en gran medida, es un proceso social de aprendizaje.
El proceso es un diálogo en el que se reúne el conocimiento y se incluye en el software.

El proceso proporciona una interacción entre los usuarios y los diseñadores, entre los usuarios y las herramientas de desarrollo, y entre los diseñadores y las herramientas de desarrollo [tecnología]. Es un proceso interactivo donde la herramienta de desarrollo se usa como medio de comunicación, con cada iteración del diálogo se obtiene mayor conocimiento de las personas involucradas.
Cuando se trabaja para construir un producto o un sistema, es importante seguir una serie de pasos predecibles, un mapa de carreteras que le ayude a obtener el resultado oportuno de calidad. El mapa de carreteras a seguir es llamado proceso del software. Lo construyen los ingenieros del software y sus gestores adaptan el proceso a sus necesidades y entonces lo siguen. Además las personas que han solicitado el software tienen un papel a desempeñar en el proceso del software. Es importante porque proporciona estabilidad, control y organización a una actividad que puede, si no se controla, volverse caótica.
Los pasos son a un nivel detallado, el proceso que adoptemos depende del software que estamos construyendo. Un proceso puede ser apropiado para crear software de un sistema de aviación, mientras que un proceso diferente por completo puede ser adecuado para la creación de un sitio web.
Desde el punto de vista de un ingeniero de software, los productos obtenidos son programas, documentos y datos que se producen como consecuencia de las actividades ingenieriles definidas por el proceso.
Hay una cantidad de mecanismos de evaluación del proceso de software que permiten a las organizaciones determinar la madurez de su proceso. Sin embargo, la calidad, oportunidad y viabilidad a largo plazo del producto que se está construyendo, son los mejores indicadores de la eficiencia del proceso que estamos utilizando.

7. ENLISTA REQUERIMIENTOS DEL SISTEMA DE INFORMACIÓN DEFINIENDO LOS DATOS DE ENTRADA, INFORMACIÓN DE SALIDA Y CONTROLES DE VALIDACIÓN.

Definición de requerimientos:
En ingeniería del software y el desarrollo de sistemas, un requerimiento es una necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio.

¿Que son los requerimientos?
Son declaraciones que identifican atributos, capacidades, características y/o cualidades que necesita cumplir un sistema (o un sistema de software) para que tenga valor y utilidad para el usuario. En otras palabras, los requerimientos muestran qué elementos y funciones son necesarias para un proyecto.

Clasificación de los requerimientos
* Requerimientos funcionales: qué debe hacer el sistema o software.
* Requerimientos no funcionales: cómo debe funcionar el sistema o software (no su implementación), por ej. Calidad, rendimiento, facilidad de uso, etc.
* Requerimientos externos: a qué se debe atener el sistema o software con respecto a su entorno: compatibilidad con otros sistemas, adecuación a determinadas leyes, etc.

Características que deberían cumplir los requerimientos

* Actual: el requerimiento no debe volverse obsoleto con el paso del tiempo.
* Cohesión: el requerimiento debe dirigirse a solo una única cosa.
* Completo: el requerimiento debe estar completamente declarado en un único lugar, sin información faltante.
* Consistente: el requerimiento no debe contradecir ningún otro requerimiento y debe ser completamente consistente con toda la documentación.
* Correcto/necesario: el requerimiento debe cumplir con la necesidad declarada por los interesados en el sistema/software.
* Factible/viable: el requerimiento debe poder ser implementado.
* No ambiguo: el requerimiento debe estar concisamente declarado. Debe expresar hechos objetivos, no opiniones subjetivas. Debe poder ser interpretado de una única manera.
* Obligatorio: el requerimiento debe representar una característica definida por el grupo interesado en el desarrollo del sistema/software, su ausencia no puede ser reemplazada.
* Observable externamente: el requerimiento debe especificar una característica observable externa o experimentable por el usuario del producto.
* Verificable/demostrable: La implementación del requerimiento debe poder ser resuelta en alguno de estos cuatro métodos: inspección, análisis, demostración o prueba

Datos de entrada:

El diseño de la entrada es el enlace que une al sistema de información con el mundo y sus usuarios. Algunos aspectos del diseño cambian, lo que depende si el sistema está orientado hacia lotes o en línea. Pero sin considerar el sistema, existen aspectos generales en la entrada que todos los analistas deben tener en cuenta.

Información de salida:

El término salida se usa para denotar cualquier información producida por un sistema de información, ya sea impresa o en una pantalla que será entregada a los usuarios. Algunos datos requieren un procesamiento extenso antes de que se conviertan en salida adecuada, y otros datos son guardados y considerados salida cuando se les recupera con poco o ningún procesamiento. La salida puede tomar muchas formas, la permanente tradicional de los reportes impresos y la fugaz, tal como la de las pantallas VDT, micro-formas y sonido. Los usuarios dependen de la salida para realizar sus tareas, y frecuentemente juzgan el mérito de un sistema únicamente por su salida.

Controles de validación:

Son sistemas de aseguramiento de la calidad mediante los cuales se establecen evidencias documentadas para demostrar que un proceso conduce a resultados de calidad consistentes dentro de las especificaciones predeterminadas. El proceso de Validación se inicia con las actividades de pre-validación, las cuales consisten en la recopilación de la información relacionada con el proceso, en la revisión de las evaluaciones de riesgos realizadas en el pasado, las materias primas e insumos usados, la verificación de la realización de una calificación técnica a las instalaciones locativas y a los equipos, existencia de procedimientos para las tareas u operaciones y el entrenamiento a los trabajadores.

8. LÓGICA COMPUTACIONAL, VARIABLE Y CONSTANTES PARA ALMACENAMIENTO DE DATOS Y SENTENCIAS DE CONTROL.

La lógica computacional la misma lógica matemática aplicada al contexto de las ciencias de la computación. Su uso es fundamental a varios niveles: en los circuitos computacionales, en la programación lógica y en el análisis y optimización (de recursos temporales y espaciales) de algoritmos.

Las variables y constantes son los espacios de memorias reservados para almacenar los datos en un programa de computador. Dichos datos pueden ser de varios tipos. El presente objeto de aprendizaje permite identificar de manera práctica las variables y constantes, así como tres de los principales tipos de datos que se pueden almacenar en las variables y constantes.

Sentencias de control

Un programa es una sucesión de **sentencias** que son ejecutadas secuencialmente.

Por ejemplo, el siguiente programa tiene cuatro sentencias:

```
n = int(raw_input('Ingrese n: '))
m = int(raw_input('Ingrese m: '))
suma = n + m
print 'La suma de n y m es:', suma
```

Las primeras tres son asignaciones, y la última es una llamada a función. Al ejecutar el programa, cada una de estas sentencias es ejecutada, una después de la otra, una sola vez.

Además de las sentencias simples, que son ejecutadas en secuencia, existen las **sentencias de control** que permiten modificar el flujo del programa introduciendo ciclos y condicionales.

9. PROCEDIMIENTOS Y CONSTRUCCIÓN DE DIAGRAMAS ENTIDAD RELACIÓN Y SUS ELEMENTOS: ENTIDADES, ATRIBUTOS, RELACIONES, CARDINALES Y SIMBOLOGÍA.

Un **diagrama o modelo entidad-relación** (a veces denominado por sus siglas en inglés, *E-R* "Entity relationship", o del español *DER* "Diagrama de Entidad Relación") es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades.

Entidades

Las entidades son el fundamento del modelo entidad relación. Podemos adoptar como definición de entidad cualquier cosa o parte del mundo que es distinguible del resto. Por ejemplo, en un sistema bancario, las personas y las cuentas bancarias se podrían interpretar como entidades. Las entidades pueden representar entes concretos, como una persona o un avión, o abstractas, como por ejemplo un préstamo o una reserva. Se representan por medio de un rectángulo. Que pueden ser de tipo: maestras, transaccionales, históricas y temporales

Atributos

Se representan mediante un círculo o elipse etiquetado mediante un nombre en su interior. Cuando un atributo es identificativo de la entidad se suele subrayar dicha etiqueta.

Por motivos de legibilidad, los atributos suelen no aparecer representados en el diagrama entidad-relación, sino descritos textualmente en otros documentos adjuntos.

Relaciones

Se representan mediante un rombo etiquetado en su interior con un verbo. Este rombo se debe unir mediante líneas con las entidades (rectángulos) que relaciona, para así saber cuál es la relación que lleva cada uno.

Cardinalidad de las relaciones

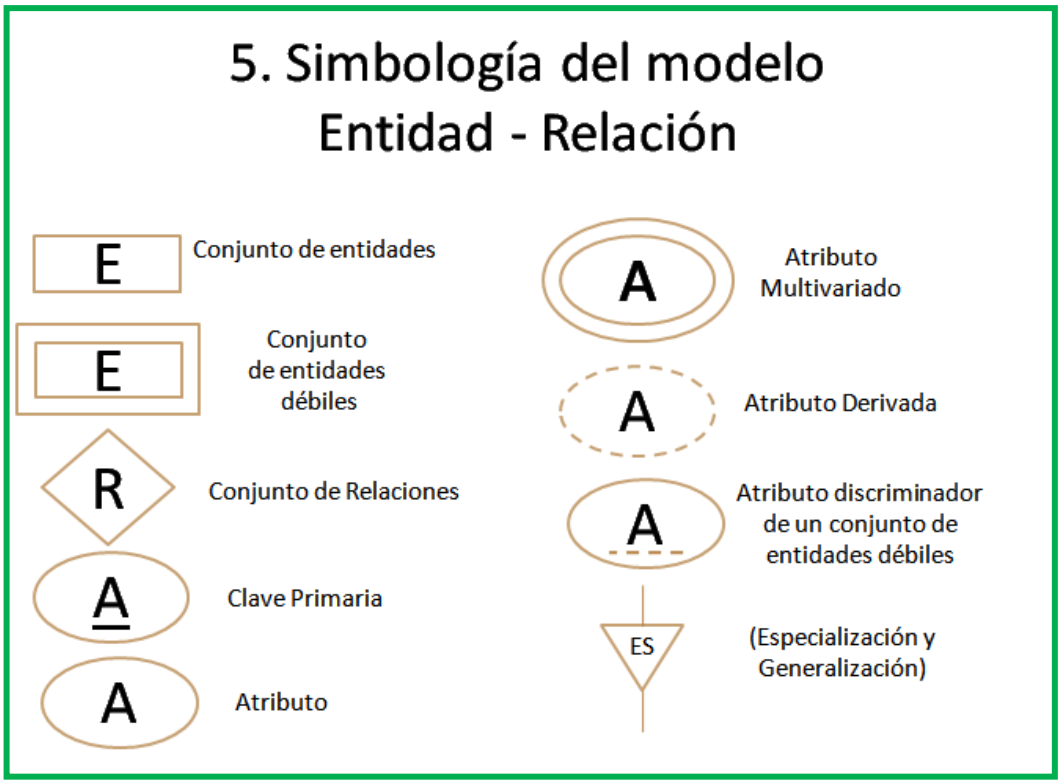
El tipo de cardinalidad se representa mediante una etiqueta en el exterior de la relación, respectivamente: "1:1", "1:N" y "N:M", aunque la notación depende del lenguaje utilizado, la que más se usa actualmente es el unificado. Otra forma de expresar la cardinalidad es situando un símbolo cerca de la línea que conecta una entidad con una relación:

- **"0"** si cada instancia de la entidad no está obligada a participar en la relación.
- **"1"** si toda instancia de la entidad está obligada a participar en la relación y, además, solamente participa una vez.
- **"N" , "M", ó "*"** si cada instancia de la entidad no está obligada a participar en la relación y puede hacerlo cualquier número de veces.

Ejemplos de relaciones que expresan cardinalidad:

- Un policía (entidad) tiene (relación) un arma (entidad) siempre y cuando no realice funciones de oficina, pudiendo entonces tenerla o no asignada. Es una relación 0:1.
- Cada esposo (entidad) está casado (relación) con una única esposa (entidad) y viceversa. Es una relación 1:1.
- Una factura (entidad) se emite (relación) a una persona (entidad) y sólo una, pero una persona puede tener varias facturas emitidas a su nombre.Cardinalidad es el número de entidades con la cual otra entidad puede asociar mediante una relación considerando una relación binaria entre el juego de entidades A y el B, la cardinalidad puede ser:Uno a unoUno a muchos ó muchos a unoMuchos a muchos Todas las facturas se emiten a nombre de alguien. Es una relación 1:N.
- Un cliente (entidad) puede comprar (relación) varios servicios (entidad) y un servicio puede ser comprado por varios clientes distintos. Es una relación N:M.

Simbología:



10. PROCEDIMIENTO Y CONSTRUCCIÓN DE DIAGRAMAS DE FLUJO, SIMBLOGÍA, DICCIONARIO DE DATOS, OTROS.

El **diagrama de flujo** o **diagrama de actividades** es la representación gráfica del algoritmo o proceso. Se utiliza en disciplinas como programación, economía, procesos industriales y psicología cognitiva.

Dentro de los símbolos fundamentales para la creación de diagramas de flujo, los símbolos gráficos son utilizados específicamente para operaciones aritméticas y relaciones condicionales. La siguiente es una lista de los símbolos más comúnmente utilizados:

+	Sumar
-	Menos
*	Multiplicación
/	División
±	Mas o menos
=	Equivalente a
>	Mayor que
<	Menor que
³	Mayor o igual que
£	Menor o igual que
¹ o <>	Diferente de
	Si
	No
	True
	False

Los pasos a seguir para construir el diagrama de flujo son:

1. Establecer el alcance del proceso a describir. De esta manera quedará fijado el comienzo y el final del diagrama. Frecuentemente el comienzo es la salida del proceso previo y el final la entrada al proceso siguiente.

2. Identificar y listar las principales actividades/subprocesos que están incluidos en el proceso a describir y su orden cronológico.

3. Si el nivel de detalle definido incluye actividades menores, listarlas también.
4. Identificar y listar los puntos de decisión.

5. Construir el diagrama respetando la secuencia cronológica y asignando los correspondientes símbolos.

6. Asignar un título al diagrama y verificar que esté completo y describa con exactitud el

proceso elegido.

7. Ventajas de los diagrama de flujo

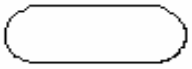
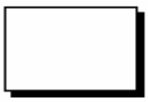
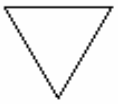
8. Favorecen la comprensión del proceso a través de mostrarlo como un dibujo. El cerebro humano reconoce fácilmente los dibujos. Un buen diagrama de flujo reemplaza varias páginas de texto.

9. Permiten identificar los problemas y las oportunidades de mejora del proceso. Se identifican los pasos redundantes, los flujos de los re-procesos, los conflictos de autoridad, las responsabilidades, los cuellos de botella, y los puntos de decisión.

10. Muestran las interfaces cliente-proveedor y las transacciones que en ellas se realizan, facilitando a los empleados el análisis de las mismas.

11. Son una excelente herramienta para capacitar a los nuevos empleados y también a los que desarrollan la tarea, cuando se realizan mejoras en el proceso.

SIMBOLOGÍA:

SÍMBOLO	REPRESENTA	SÍMBOLO	REPRESENTA
	**Terminal**: Indica el inicio o la terminación del flujo del proceso.		**Actividad:** Representa una actividad llevada a cabo en el proceso.
	**Decisión**: Indica un punto en el flujo en que se produce una bifurcación del tipo "SI" – "NO"		**Documento:** Se refiere a un documento utilizado en el proceso, se utilice, se genere o salga del proceso.
	**Multidocumento**: Refiere a un conjunto de documentos. Un ejemplo es un expediente, que agrupa a distintos documentos.		**Inspección / Firma:** Empleado para aquellas acciones que requieren una supervisión (como una firma o "visto bueno").
	**Conector de proceso**: Conexión o enlace con otro proceso diferente, en la que continúa el diagrama de flujo.		**Archivo Manual:** Se utiliza para reflejar la acción de archivo de un documento y/o expediente.
	**Base de datos/aplicación:** Empleado para representar la grabación de datos.		**Línea de Flujo.** Proporciona indicación sobre el sentido de flujo del proceso.

REGLAS BÁSICAS DEL DIAGRAMA DE FLUJO

1. Todos los diagramas deben tener un inicio y un final.

2. El diagrama de flujo debe ser construido de arriba hacia abajo y de izquierda a

derecha.

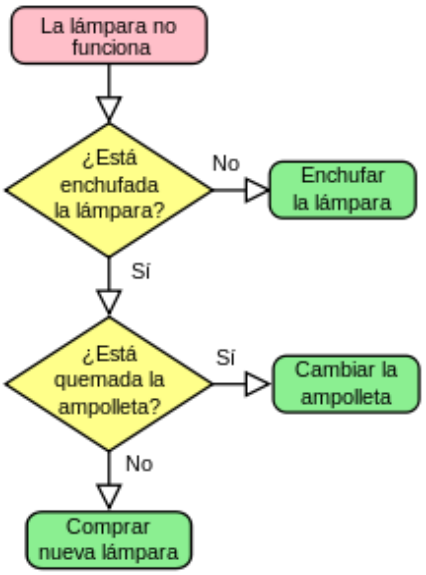
3. La notación utilizada en el diagrama de flujo debe ser independiente del lenguaje de programación. La solución presentada en el diagrama puede escribirse posteriormente y fácilmente en diferentes lenguajes de programación.

4. Es conveniente cuando realizamos una tarea compleja poner comentarios que expresen o ayuden a en tender lo que hicimos.

5. Si el diagrama de flujo requiriera más de una hoja para su construcción, debemos utilizar los conectores adecuados y enumerar las páginas convenientemente.

6. No puede llegar más de una línea a un símbolo.

EJEMPLO:



¿Qué es un Diccionario de Datos?

Un diccionario de datos contiene las características lógicas de los datos que se van a utilizar en un sistema, incluyendo nombre, descripción, alias, contenido y organización.

Estos diccionarios se desarrollan durante el análisis de flujo de datos y ayuda a los analistas que participan en la determinación de los requerimientos del sistema, evitando así malas interpretaciones o ambigüedades, su contenido también se emplea durante el diseño del proyecto.

En un diccionario de datos se encuentra la lista de todos los elementos que forman parte del flujo de datos de todo el sistema. Los elementos más importantes son flujos de datos, almacenes de datos y procesos. El diccionario de datos guarda los detalles y descripción de todos estos elementos.

Desde el punto de vista estadístico, este diccionario debe de tener la variable, el tipo de variable, su definición como también su delimitación espacial.

Para comprender mejor el significado de un diccionario de datos, puede considerarse su contenido como "datos acerca de los datos"; es decir, descripciones de todos los demás objetos (archivos, programas, informes, sinónimos...) existentes en el sistema. Un diccionario de datos almacena la totalidad de los diversos esquemas y especificaciones de archivos, así como sus ubicaciones. Si es completo incluye también información acerca de qué programas utilizan qué datos, y qué usuarios están interesados en unos u otros informes. Por lo general, el diccionario de datos está integrado en el sistema de información que describe.

Tipos de Diccionarios

Diccionario Off-Line
• Se ocupa de mantener el diccionario en condiciones.
• No tiene ingerencia en el uso dinámico de los datos.

Diccionarios On-Line
• Trabaja junto con el compilador.
• Impide que el programador defina los datos en el programa y los toma directamente del diccionario.
• Verifica que los datos nombrados existan en el diccionario.
• Incorpora al programa, desde el diccionario la definición de los datos.
• Inconveniente: si uno se olvida de recompilar, estarán conviviendo datos en la correcta versión actual con otros en una versión superada.

Diccionarios In-Line
• No incorpora la definición de datos en el programa, sino que las carga cuando se ejecuta.

CONCLUSIONES

El desarrollo de software son tal cual muy importantes para el desarrollo futuro de alguna aplicación que fuese a desarrollarse pues, este contenido de técnicas y procedimientos de construcciones son tan importantes para valerse de el desarrollo de tal aplicación, para no tener fallos y que sea bien planeada y estructurada desde un principio para no caer en el mantenimiento de actualizaciones y parches, pues esto quita tiempo y hay pérdida económica de parte de la empresa.

RECOMENDACIONES

- ✓ Tener en cuenta la planeación del desarrollo de software en alguna aplicación desde un principio.
- ✓ Utilizar diagramas de flujo y de entidad-relación para una estructuración más exacta de la aplicación.
- ✓ Observar detalladamente toda estructura y modificación de la planeación de la aplicación.
- ✓ Revisar posibles fallos dentro de la aplicación.

BIBLIOGRAFIAS

Enlaces

http://es.wikipedia.org/wiki/Proceso_para_el_desarrollo_de_software

Nombre de la página: Wikipedia
Fecha de visita: 10 de febrero

http://es.slideshare.net/chidochido020/proceso-modelos-y-metodos-de-ingenieria-de-software

Nombre de la página: Slideshare
Fecha de visita: 10 de febrero

http://www.ctr.unican.es/asignaturas/MC_OO/Doc/OO_08_I2_Proceso.pdf

Nombre de la página: Documento pdf, Santander 2008
Fecha de visita: 12 de febrero

http://es.slideshare.net/gualsema/tecnicas-y-herramientas-de-desarrollo-de-software1

Nombre de la página: Slideshare
Fecha de visita: 12 de febrero

http://es.wikipedia.org/wiki/Modelo_entidad-relaci%C3%B3n

Nombre de la página: Wikipedia
Fecha de visita: 13 de febrero