



**CORPORACIÓN
UNIVERSITARIA
REMINGTON**

ESCUELA DE CIENCIAS BÁSICAS E INGENIERÍA
Ingeniería de Sistemas
ASIGNATURA: Ingeniería del Software III

CORPORACIÓN UNIVERSITARIA REMINGTON
DIRECCIÓN PEDAGÓGICA

Este material es propiedad de la Corporación Universitaria Remington (CUR), para los estudiantes de la CUR en todo el país.

2011

CRÉDITOS



El módulo de estudio de la asignatura Ingeniería del Software III del Programa Ingeniería de Sistemas es propiedad de la Corporación Universitaria Remington. Las imágenes fueron tomadas de diferentes fuentes que se relacionan en los derechos de autor y las citas en la bibliografía. El contenido del módulo está protegido por las leyes de derechos de autor que rigen al país.

Este material tiene fines educativos y no puede usarse con propósitos económicos o comerciales.

AUTOR

Yolfaris Naidith Fuertes Arroyo

Ingeniera de Sistemas.

5 años de Experiencia docente

yolfaris.fuertes@remington.edu.co

Rodrigo Alcides Patiño Arango

Ingeniero de Sistemas.

12 años de Experiencia docente

Rodrigo.patino@remington.edu.co

Nota: el autor certificó (de manera verbal o escrita) No haber incurrido en fraude científico, plagio o vicios de autoría; en caso contrario eximió de toda responsabilidad a la Corporación Universitaria Remington, y se declaró como el único responsable.

RESPONSABLES

ESCUELA DE CIENCIAS BÁSICAS E INGENIERÍA

Director Dr. Mauricio Sepúlveda

ingenieria.director@remington.edu.co

Director Pedagógico

Octavio Toro Chica

dirpedagogica.director@remington.edu.co

Coordinadora de Medios y Mediaciones

Angélica Ricaurte Avendaño

mediaciones.coordinador01@remington.edu.co

GRUPO DE APOYO

Personal de la Unidad de Medios y Mediaciones

EDICIÓN Y MONTAJE

Primera versión. Febrero de 2011.

Derechos Reservados

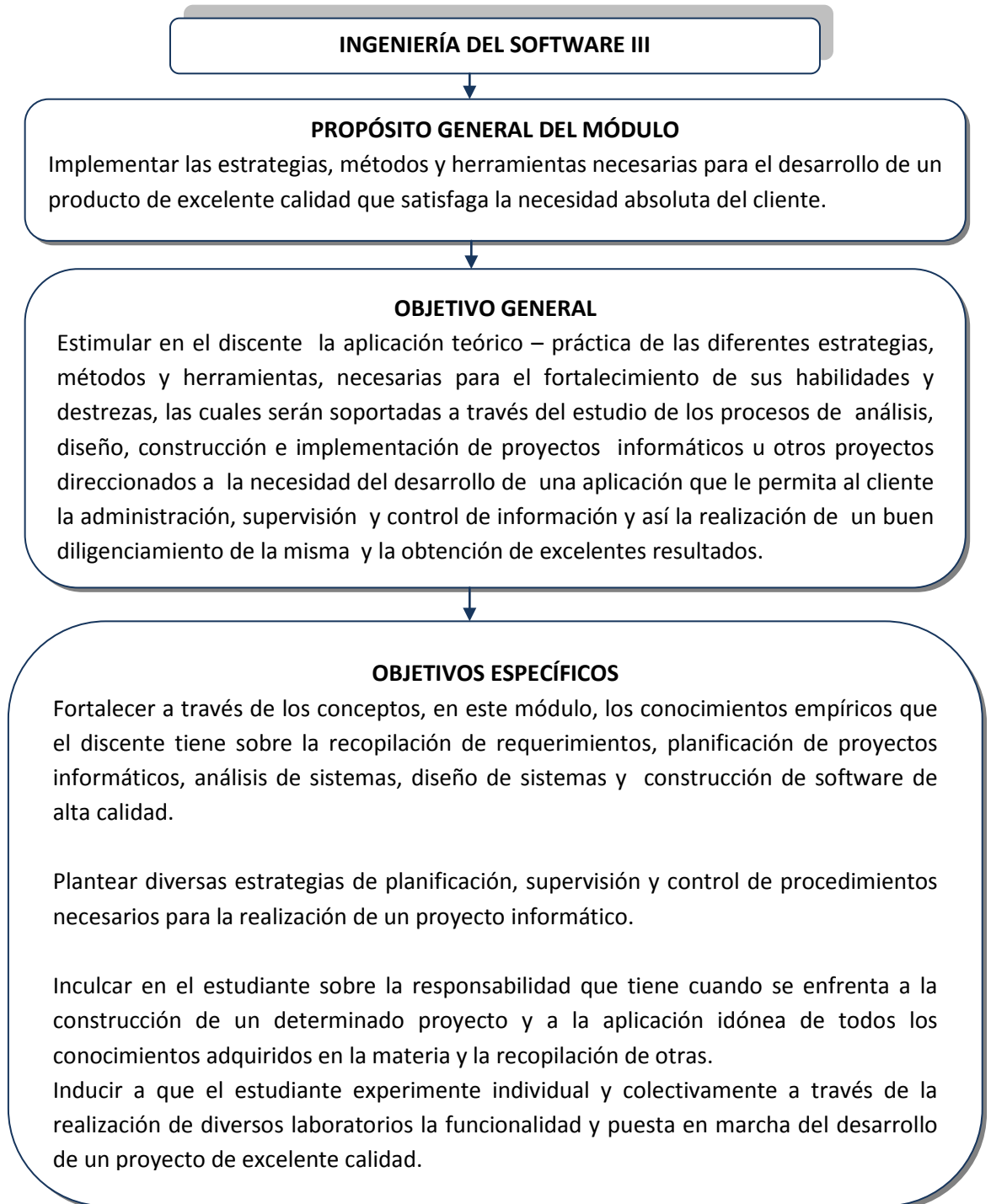


Esta obra es publicada bajo la licencia Creative Commons. Reconocimiento-No Comercial-Compartir Igual 2.5 Colombia.

TABLA DE CONTENIDO

1.	MAPA DE LA ASIGNATURA.....	7
2.	LOS MÉTODOS ANÁLISIS DE SOFTWARE	9
2.1.	Modelado de análisis	10
2.1.1.	Análisis de requisitos.....	10
3.	DISEÑO DE SOFTWARE.....	35
3.1.	Ingeniería del diseño	36
3.2.	Diseño arquitectónico	41
3.2.1.	Arquitectura del software	41
3.3.	Evaluación de diseños arquitectónicos alternos.....	45
3.4.	Diseño de la interfaz del usuario.....	46
3.4.1.	Las reglas de oro.....	46
3.5.	Diseño a nivel de componentes	50
3.5.1.	¿Qué es un componente?	50
4.	LAS PRUEBAS Y LA MEDICIÓN	55
4.1.	Estrategias y la medición.....	56
4.2.	Técnicas de prueba de software	61
4.2.1.	Técnicas de prueba de software	61
5.	MODELOS OPERATIVOS WEB	66
5.1.	Desarrollo Web	67
5.1.1.	Ingeniería web.....	67
5.2.	Pistas de Aprendizaje	78
5.3.	Glosario	79
5.4.	Bibliografía	80

1. MAPA DE LA ASIGNATURA



Unidades

UNIDAD 1

Capacidad para comprender y aplicar los diferentes métodos que facilitan el desarrollo de un buen análisis.

UNIDAD 2

Destreza para crear excelentes diseños que suplan las necesidades del usuario o cliente final.

UNIDAD 3

Habilidad e ingenio para desarrollar las pruebas necesarias que garanticen la calidad del producto.

UNIDAD 4

Habilidad y conocimiento para construir software fundamentado o en la ingeniería web

2. LOS MÉTODOS ANÁLISIS DE SOFTWARE

OBJETIVO GENERAL

Fortalecer a través de los conceptos, en este módulo, los conocimientos empíricos que el discente tiene sobre la recopilación de requerimientos, planificación de proyectos informáticos, análisis de sistemas, diseño de sistemas y construcción de software de alta calidad.

OBJETIVOS ESPECÍFICOS

- ◆ Comprender todos aquellos factores que intervienen dentro del modelado de análisis, el modelado de datos, la orientación a objetos y los diferentes casos de uso para que exista una comunicación directa entre el cliente y el desarrollador de la solución.

Prueba Inicial

Responda las siguientes preguntas que se describen a continuación:

- a. Que entiende usted por objetos de datos y de un **ejemplo**.
- b. Que es una asociación (**de ejemplo**)
- c. En un modelo de datos ¿qué es para usted dependencia?
- d. ¿Qué comprende usted por relación entre entidades?
- e. Mediante un diagrama de caso de uso y un diagrama de actividad, grafique el proceso de matrícula de un estudiante
- f. Indique mediante un diagrama entidad relación las entidades que puede utilizar para la matrícula del alumno.
- g. Realice un esquema de un diagrama de entrada y salida de información con el tema que facilite su aprendizaje. (Tema libre)

En el siguiente enunciado seleccione la respuesta correcta:

1. El orden en el que un estudio de factibilidad se presenta dentro de un proyecto de software es:
 - a. Factibilidad económica, factibilidad técnica y factibilidad operativa
 - b. Factibilidad operativa, factibilidad económica y factibilidad técnica
 - c. Factibilidad técnica, factibilidad operativa y factibilidad económica

2.1. Modelado de análisis

2.1.1. Análisis de requisitos

Ayuda a definir cuales es la manera o en el que el software va a funcionar, ayuda a definir la interfaz gráfica a utilizar y la manera como está interrelacionada con los elementos en los que estará implementado e identifica el nivel de acceso y las restricciones que debe manejar para garantizar la calidad del software.

El analista de requisitos tiene la posibilidad de diseñar una manera de visualizar la entrada y salida de información, indicar la funcionalidad y la manera cómo se comportará la información hasta que llegue a su diseño final que satisfaga las necesidades del usuario.

Con el modelado de análisis el desarrollador siempre responde al qué y no al cómo, dentro de los cuales se puede enunciar por ejemplo, que hará el sistema, que información debe tener, que presentación se le da al sistema que agrade al cliente y que controles debe tener.

Si no se tiene claro y bien estructurado lo que se quiere que el sistema realice, no se llegará a un resultado efectivo y que soluciones las dificultades en las empresas u organizaciones sobre las que se está trabajando.

◆ Filosofía y objetivos generales

Para tener un modelo de análisis que cumpla con las expectativas esperadas debe cumplir tres objetivos tales como, reconocer lo que el cliente necesita, definir la base sobre la cual se implementará el software y realizar la validación del desarrollo que se entregará al usuario final.

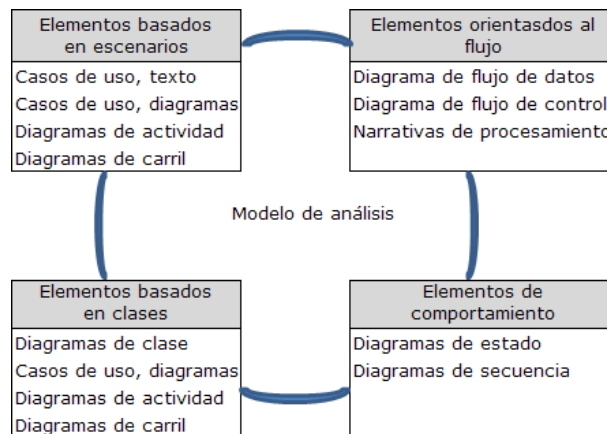
Entre el análisis y el diseño se van solucionando las dificultades que se presentan entre cada uno de ellos y con esto se logra que el trabajo final alcance un buen nivel de calidad.

◆ Enfoques de modelado del análisis

Este es llamado análisis estructurado, ya que todo aquello que transforman los datos se toman como entidades separadas, se organizan adecuadamente los campos que conforman a cada entidad para su utilización desde los distintos niveles y los objetos de datos permanecen activos para que utilizados en cualquier momento por los distintos objetos que conforman al sistema.

Dentro de este enfoque también se tiene presente el análisis orientado a objetos, los cuales están basados en clases que facilitan el control en el manejo de la información para satisfacer las necesidades el cliente.

El modelado de análisis conduce a la derivación de cada uno de los elementos que se muestran a continuación, no obstante, el contenido específico de cada elemento puede ser diferente en cada proyecto.



◆ Conceptos del modelado de datos

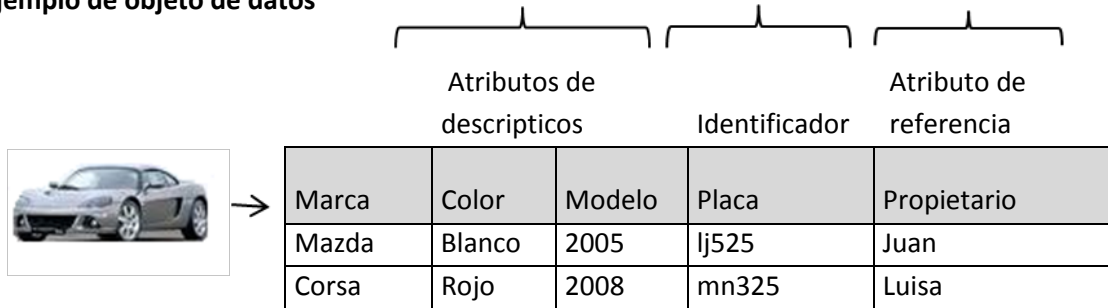
Se definen todos los campos u objetos de datos que se van a utilizar dentro del sistema y se definen las relaciones que son necesarias para la interacción los datos.

Objetos de datos

Un objeto de datos se define como una representación de información de un sistema debe de interpretar e identificar, estos datos tienen diferentes propiedades o atributos que los diferencian entre sí. Este objeto de datos puede ser una entidad externa como puede ser una cosa, un evento, una estructura, entre otros.

Los objetos de datos para su manipulación están representados en una tabla, por ejemplo un vehículo tiene marca, color, modelo, placa, propietario, entre otros atributos que los identifican.

Ejemplo de objeto de datos



Marca	Color	Modelo	Placa	Propietario
Mazda	Blanco	2005	lj525	Juan
Corsa	Rojo	2008	mn325	Luisa

Atributos

Estos definen las propiedades de un objeto dentro de los cuales se debe definir uno o varios de ellos como el principal por medio del cual se va acceder a la información cuando se esté haciendo uso de ella en los formularios o consultas.

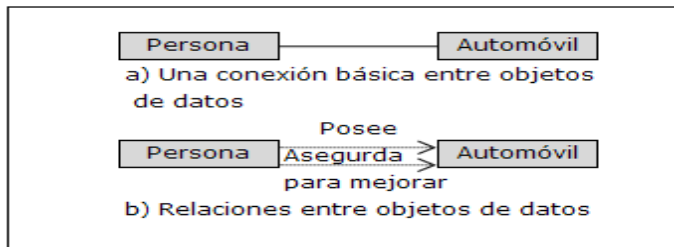
Para poder definir los atributos que debe llevar una entidad, es necesario analizar claramente el problemas a solucionar de tal manera que el cliente puede tener un control adecuado a la información que está utilizando.

Para tener un ejemplo más claro se pueden tomar los atributos que se utilizaron el texto anterior como es marca, color, entre otros.

Relaciones

Las relaciones es la forma de interconectar las entidades y dentro de estas relaciones se tienen diferentes maneras de ser leídas para entender la comunicación que hay entre ellas.

Dentro del manejo de relaciones que se pueden dar entre las entidad se tiene un término que se denomina cardinalidad que se refiere al número de veces que se pueden presentar ocurrencias entre las entidades, dentro de lo cual se puede utilizar las siguiente relaciones de acuerdo al proyecto que se esté implementando, tales como relación uno a uno, relación uno a muchos, relación muchos a muchos y con algunos ejemplos se puede interpretar un poco más sobre esta cardinalidad, como puede ser una persona puede tener uno o varios vehículos, un cliente puede llevar una o varias facturas, una factura puede tener uno o varios comprobante de pago o abono, una factura puede tener uno o varios detalles de factura, entre otros.



Análisis orientado a objetos

Uno de los objetivos principales de la orientación a objetos es manejo adecuado de las clases que permitan solucionar de manera óptima las dificultades en el manejo y control de información, pero esto se logra teniendo una buena comunicación con el cliente para extraer la información más relevante, luego al clasificar y organizar lo extraído se definen las clases que van a ser utilizadas, se establece una jerarquía de las mismas, las respectivas relaciones en las que la información va a ser utilizada e interpretar como va a responder el objeto ante cualquier evento externo.

Dentro de la orientación a objetos se debe entender con claridad algunos términos que son la base para realizar una buena utilización del mismo tales como:

Atributo: Es un conjunto de valores de datos que representan a las clases

Clase: es una colección de objetos de características idénticas que permiten la obtención de un resultado específico por ejemplo plantillas, patrones de trabajo, etc.

Objetos: Representación detallada y particular de algo de la realidad¹. Todo objeto tiene una identidad o nombre, estado y comportamiento. Los objetos heredan los atributos y operaciones de una clase.

Operaciones: Estos son los métodos y servicios que hacen que la clase puede mostrar los resultados esperados por el usuario final.

Subclase: Es la que puede heredar los atributos y las superclase

Superclase: conjunto de clases que estas relacionas con esta superclase.

◆ Modelado basado en escenarios

Los diagramas de casos de uso, diagramas de actividad y los diagramas de carril, que orientan al desarrollador de software en la manera como debe planear sus tareas para satisfacer las necesidades del cliente y no desviarse de la realidad porque es así como el cliente interactúa con el sistema de una manera familiar y puede sacarle el mejor provecho a su información ayudándole a tomar las mejores decisiones dentro de su empresa u organización.

Estructura de casos de uso

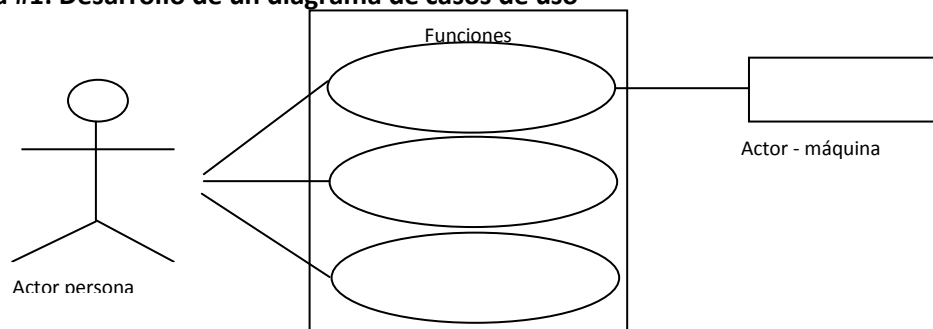
Un caso de uso muestra la manera en que interactuar el usuario (actor) con el sistema o la máquina.

Los casos de uso enuncian en primera instancia en forma de narración la información que se va a utilizar dentro del diagrama.

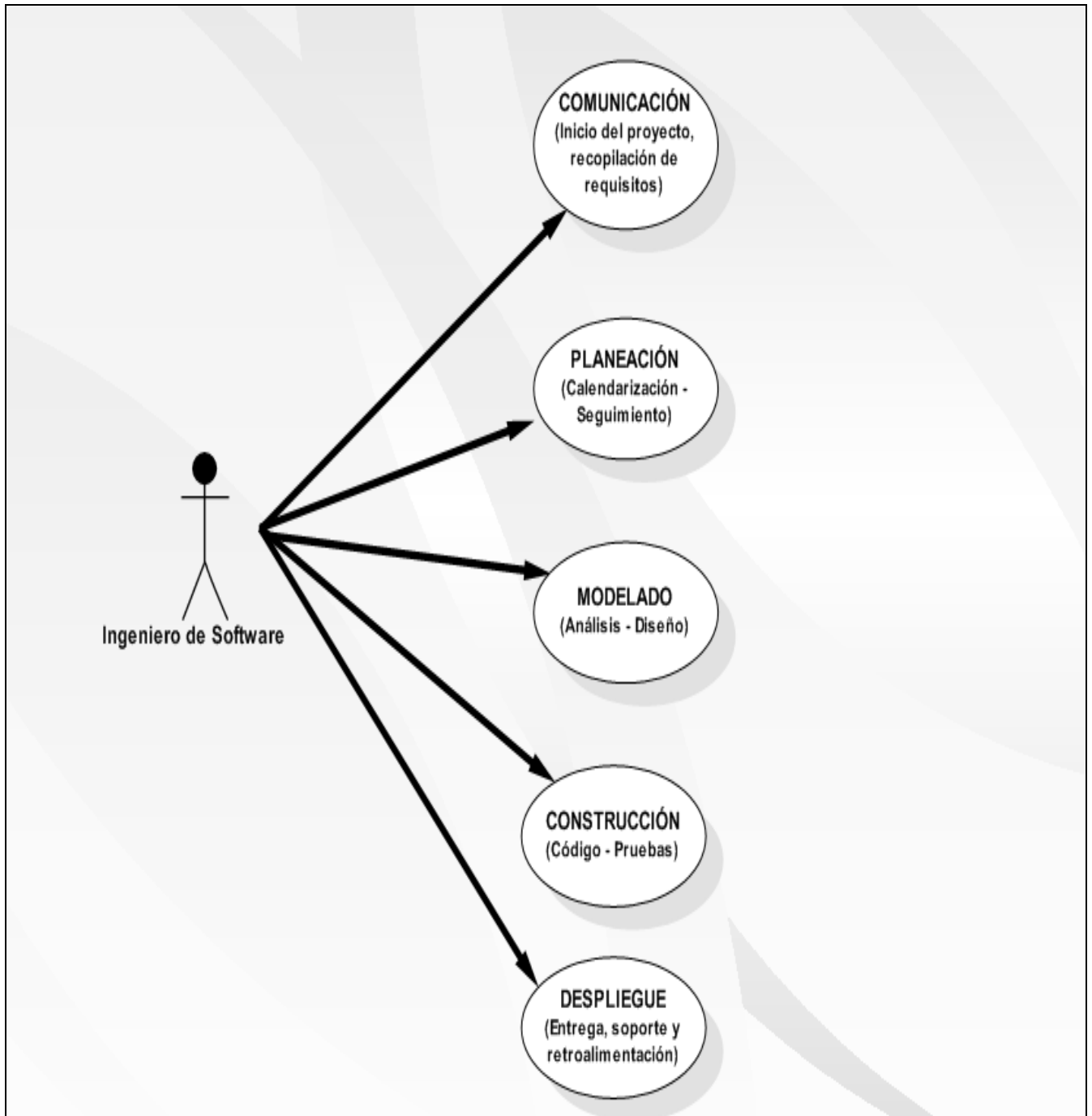
Por ejemplo en la vigilancia de hogar seguro se identifican las siguientes funciones

- Tener acceso a la cámara de vigilancia vía internet
- Seleccionar la cámara que desea ver
- Solicitar vistas en miniatura de todas las cámaras
- Desplegar vistas de cámara en una ventana de una pc
- Controlar la visión panorámica y de acercamiento en una cámara específica
- Registrar la salida de la cámara
- Repetir la salida de la cámara

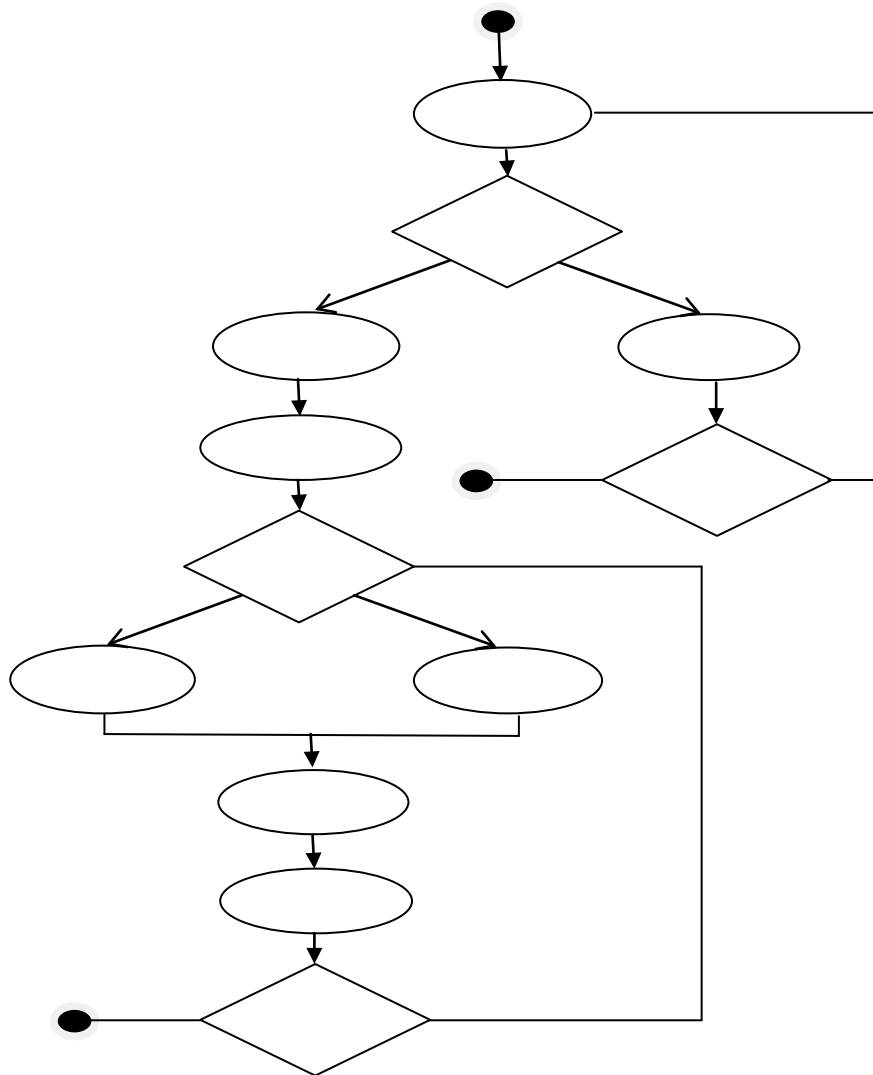
Gráfica #1: Desarrollo de un diagrama de casos de uso



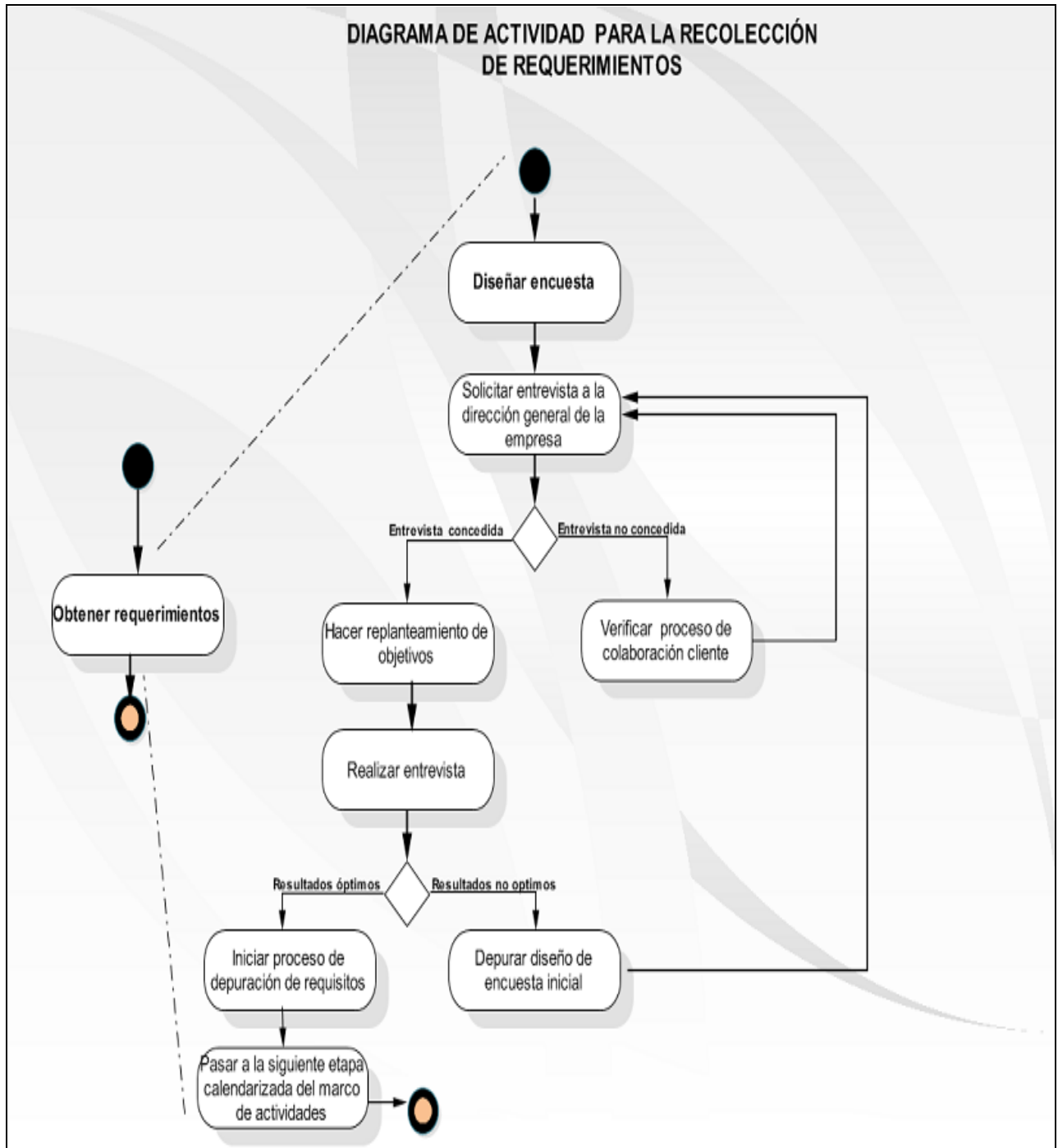
Gráfica #2: Diagrama de casos de uso para el proceso de desarrollo de software



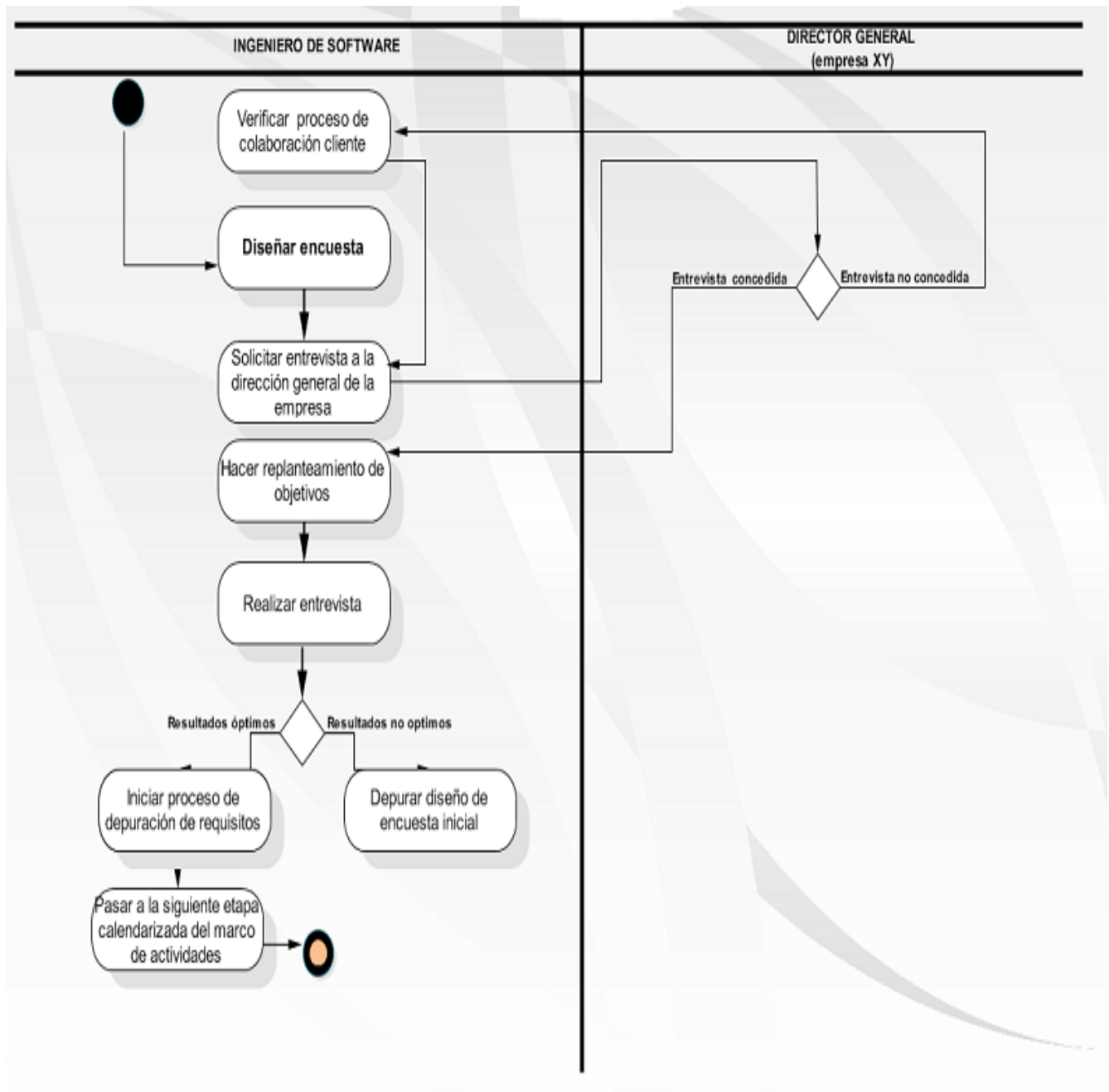
Gráfica #3: Desarrollo de un diagrama de actividad



Gráfica #4:



Gráfica #6: Diagrama de carril para el proceso de recolección de requerimientos



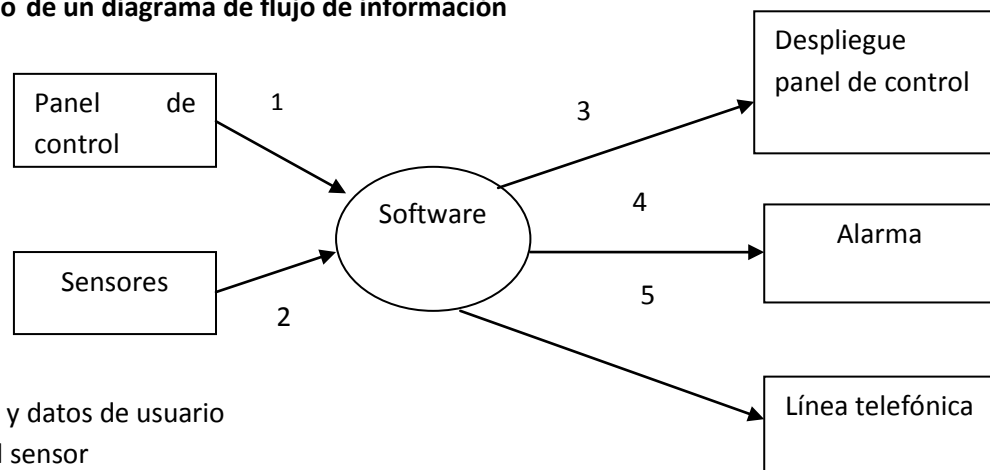
◆ Modelado orientado al flujo

El diagrama de flujo de datos tiene como base las entradas, procesos y salidas de información y es a través de estos que el cliente obtiene unos resultados óptimos para tener una mejor seguridad y control con respecto a todo lo que tiene a su disposición

Cuando se entiende claramente el manejo de la información, el DFD ayuda a comprender con detalle lo que se necesita para lo cual se debe tener presente lo siguiente.

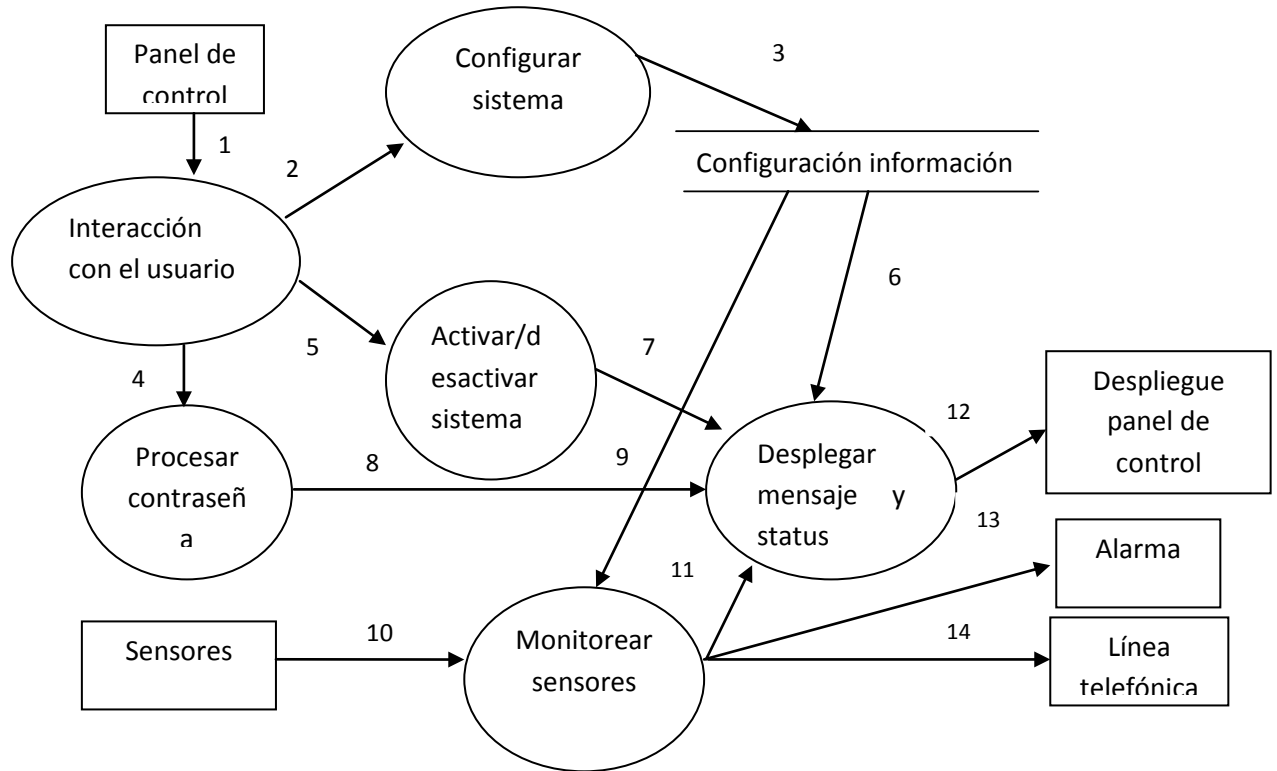
1. El nivel 0 se representa el software/sistema como una sola burbuja
2. La entrada y salida primaria deben establecerse con cuidado
3. La refinación debe comenzar por el aislamiento de procesos, objetos de datos candidatos a ser representados en el siguiente nivel
4. Todas las flechas y burbujas se deben rotular con nombres significativos
5. Se debe mantener la continuidad del flujo de información al cambiar de nivel a nivel
6. La refinación de las burbujas debe hacerse una por una².

Gráfica #7: Ejemplo de un diagrama de flujo de información



1. Comandos y datos de usuario
2. Estatus del sensor
3. Despliegue de información
4. Tipo de alarma
5. Tonos del número telefónico

Gráfica #8: Proceso de flujo del sistema.

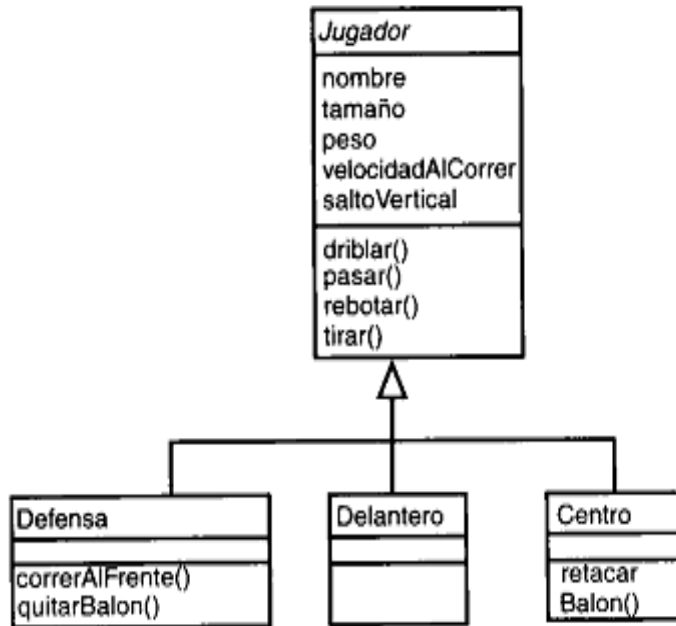


1. Comandos y datos del usuario
2. Configurar solicitud
3. Configuración de datos
4. Contraseña
5. Iniciar detener
6. Configuración de datos
7. Mensaje A/d
8. Mensaje de ID valido
9. Configuración de datos
10. Estatus del sensor
11. Información del sensor
12. Desplegar información
13. Tipo de alarma
14. Tonos del número de teléfono

◆ Modelado basado en clases

Este se refiere a una serie de términos basados en clases de un modelo de análisis como son: clases y objetos, atributos, operaciones, paquetes, modelos y diagramas de colaboración³.

Gráfica #9: Ejemplo de diagrama de clases



Identificación de clases de análisis

Al observar el interior de un lugar físico y que está siendo utilizado por personas, se puede observar e identificar los objetos que están inmersos dentro de él, pero sería diferente observar dentro de un problema de aplicaciones o software cuales son los objetos que podemos hallar dentro.

Al tener un grado de dificultad identificar las clases que pueden ser utilizadas en un proyecto o en un sistema de información, es recomendable reconocer algunas formas de las que se pueden reconocer las clases dentro de las cuales se tiene:

Entidades externas (otros dispositivos, sistemas o personas)

Cosas (reportes, despliegues, letras, señales)

Sucesos o eventos (movimientos de un robot) de acuerdo al programa del sistema

Papeles (gerente, ingeniero, personal de ventas)

Utilidades organizacionales (división, grupo, equipo)

Sitios (piso de manufactura o puerto de carga)

Estructuras (sensores, vehículos, computadoras)

Especificación de atributos

Dentro del modelado de análisis se debe definir cada uno de los atributos que componen la clase ya que por medio de estos se dará la comunicación dentro de todo el modelo para que este sea funcional y maneje o utilice adecuadamente los datos, por ejemplo se tiene la siguiente clase

Gráfica #10: Ejemplo de atributo

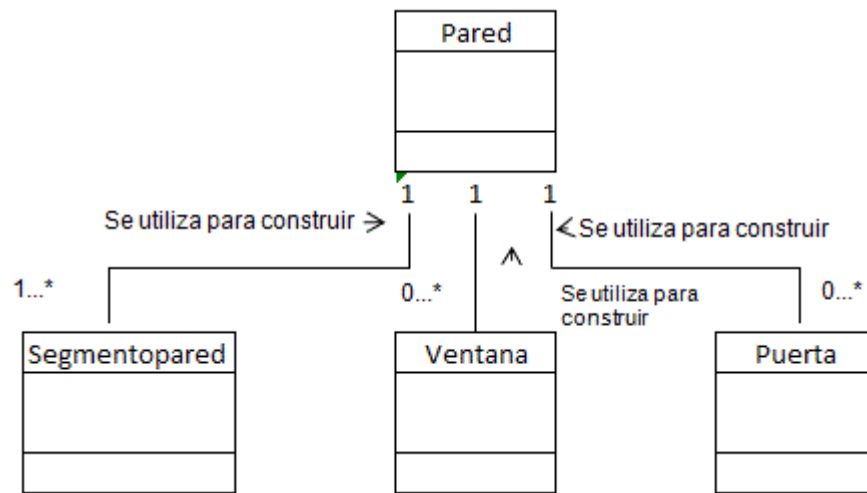
Persona
Identificación Nombre Apellido Dirección Teléfono
Agregar() Guardar() Buscar() Modificar() Cancelar() Llamar()

Asociación y dependencias

Una asociación se puede explicar de una forma adecuada cuando se presenta multiplicidad, en donde un objeto específico se puede construir a partir de uno varios objetos relacionados.

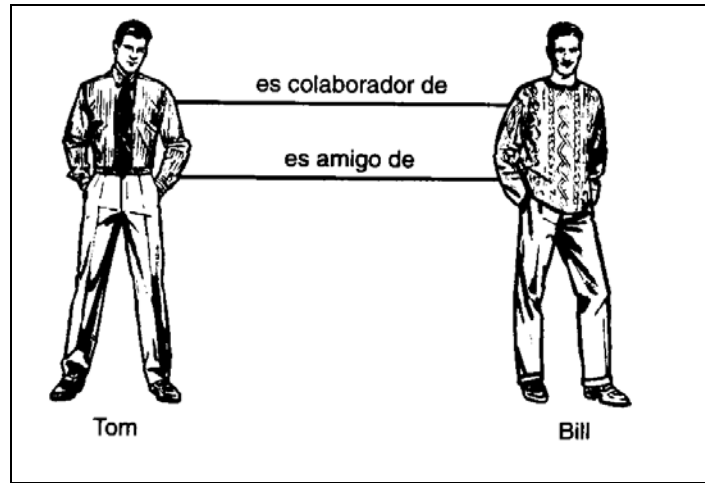
Por ejemplo un objeto llamado pared puede tener inmerso los objetos segmento pared, ventana y puerta, con este conjunto de objetos los cuales pueden ser uno o varios por cada pared hasta conformar la estructura completa.

Gráfica # 11: ejemplo de clases asociadas



Otro ejemplo de asociación se presenta entre dos o más objetos que se relacionen en varias formas

Gráfica # 12.



◆ **Creación de un modelo de comportamiento**

Este modelo da a conocer la manera como responde el sistema con respecto a los eventos externos para lo cual debe tener presente lo siguiente.

1. Identificar claramente los casos de uso
2. Identificar los eventos que indican la forma en que se lleva a cabo la ejecución de un proceso
3. Tener el control de cada caso de uso
4. Construir el diagrama de estado
5. Verificar la exactitud del modelo construido

Para identificar los eventos de un caso se usó, se puede hacer mediante el siguiente ejemplo.

Una persona necesita retirar cierta cantidad de dinero y para ello mediante el teclado introduce la contraseña de cuatro dígitos. Esta clave o contraseña es verificada con respecto al sistema que tiene esta información almacenada. Si la clave es incorrecta, el sistema emite un mensaje del error para que el usuario digite una nueva clave y así poder continuar con el proceso inicial.

Del ejemplo anterior se tiene como objeto persona y el evento sería ingresar la contraseña

◆ **Introducción de Análisis y esquematización de sistemas**

Es el proceso de examinar la situación de una empresa con el propósito de mejorarla con métodos y procedimientos más adecuados.

Panorama del Análisis y Diseño de Sistemas

Lo componen dos partes:

- ◆ **Análisis de sistemas:** es el proceso de clasificación e interpretación de hechos, diagnósticos de problemas y empleo de la información para recomendar mejoras al sistema.
- ◆ **Diseño de sistemas:** es el proceso de planificar, reemplazar o complementar un sistema organizacional existente. Pero antes de realizar esta planeación es necesario comprender en su totalidad el viejo sistema para realizar más eficientemente las operaciones.

(Comunicarse y tratar con las personas es uno de los aspectos muy importantes del trabajo del analista de sistemas)

- ◆ **Finalidad del análisis de sistemas:** está en comprender los detalles de una situación y decidir si es factible o deseable una mejora.

El trabajo del analista de sistemas

Conducir estudios de sistemas para detectar hechos relevantes relacionados con la actividad de la empresa. Su función más importante es reunir información y determinar los requerimientos.

¿Quiénes son los usuarios?

Son las personas que no son especialistas en sistemas de información pero que utilizan el computador para desempeñar su trabajo.

Se agrupan en 4 categorías:

- ◆ **Usuarios primarios:** son aquellos usuarios que interactúan con el sistema. Ejemplo: agentes de reservación de vuelos.
- ◆ **Usuarios directos:** son los que se benefician de los resultados o reportes generados por estos sistemas, pero que no interactúan de manera directa con el hardware o software. Ejemplo: gerentes encargados de las funciones de la organización.

- ◆ **Usuarios gerentes:** tienen responsabilidades administrativas en los sistemas de aplicación.
- ◆ **Usuarios directivos:** toman mayor responsabilidad en el desarrollo del sistema de información.

(El analista especifica que debe hacer el sistema de información, el diseñador establece como alcanzar los objetivos)

CICLO DE VIDA DE UN SISTEMA DE INFORMACIÓN

- ◆ **Investigación preliminar:** se realiza antes de empezar a desarrollar el sistema de información.
- ◆ **Aclaración:** cuando usted define con el usuario a que le va a dar solución, debe quedar una constancia (fecha, responsabilidades (a que se comprometió a darle solución), firma de ambas partes participantes, entre otros.
- ◆ **Factibilidad:** investigar que factibilidad hay de hacer el software (económica, técnica y operativa)
- ◆ **Análisis:** se procede a realizar la recopilación de los requerimientos de una forma planificada y estructurada.
- ◆ **Diseño:** se procede a pasar todo lo que está en maqueta de una forma que el usuario entienda a que se le va a dar solución.
- ◆ **Desarrollo del software:** se procede a realizar el proceso de codificación o realización del programa fuente.
- ◆ **Pruebas:** se verifica que funcione el sistema de información.

Implementación e instalación: se debe entregar el código fuente, BD, manual de usuario, entre otros.

Lista de necesidades que se requieren para construir un S.I.

Para explicar este tema, suponga que está construyendo una base de datos para manejar un sistema de créditos (cartera) de un almacén XY

La lista de necesidades hace referencia a los resultados que debe arrojar el sistema de acuerdo a esta aplicación:

- ◆ Datos de clientes
- ◆ Lista de clientes morosos
- ◆ Datos de las referencias del cliente
- ◆ Listado de clientes más frecuentes
- ◆ Comprobantes de ingresos – egresos
- ◆ Datos de las ventas del mes
- ◆ Listado de ventas de contado
- ◆ Listado de clientes que ganaron sorteos por sus compras
- ◆ Datos de facturación por cliente
- ◆ Datos de los pagos al mes
- ◆ Lista de ventas a crédito, entre otros.

Delimitación o alcance

En este proceso se procede a describir a que puntos de la lista de necesidades se le dará solución, además de incluir la cantidad de formularios de entradas y salidas que se programaran en el desarrollo del producto:

- ◆ Informe de clientes
- ◆ Informe de referencias del cliente
- ◆ Informe de facturación por cliente
- ◆ Lista de clientes morosos
- ◆ Informe de ventas al mes
- ◆ Informe pagos al mes

Formularios de entrada y salida de información:

- ◆ Estudiante
- ◆ Profesor

Se procederá a realizar consulta en un formulario de movimiento que permita visualizar la información de ambos involucrados en el proceso de verificación de resultados.

Definición del medio ambiente del sistema

Se deben describir las áreas que se beneficiaran con la implementación del sistema de información:

Siguiendo secuencialmente con el desarrollo de la base de datos para el sistema de créditos de un almacén XY, tenemos:

Áreas a beneficiar:

- ◆ **Cartera:** se beneficia directamente con el desarrollo del sistema, ya que en esta área se manejan actualmente los procedimientos relacionados con las cuentas por cobrar de la empresa, y es en este departamento donde se instalará el sistema de información.
- ◆ **Ventas:** se beneficiará indirectamente. Actualmente en este departamento se elabora la facturación y el listado de ventas de cada período en forma manual. Con la implementación del sistema de información pasaran a arrojarse por medio del sistema, lo cual optimizará tiempo y recurso humano.
- ◆ **Gerencia:** se beneficiará indirectamente. Este departamento requiere mensualmente para sus estadísticas informes de pagos al mes, los cuales serán arrojados por el sistema.

Gráfica # 13: Diagrama de entradas y salidas



Definición de entradas:

E1: Datos del cliente (cédula, nombre, dirección, teléfono, entre otros...)

Datos de los fiadores (cédula, nombre, dirección, teléfono, entre otros...)

Datos de las ventas (#factura, fecha, valor factura, IVA, cliente, entre otros...)

Datos del pago (#recibo, #factura, fecha, valor cuota, forma pago, entre otros...)

Definición de salidas:

- S1:** Informe pago del mes
Informe ventas del mes
Informe clientes ganadores del sorteo
- S2:** Lista de ventas
Facturación
- S3:** Listado de Procesos de ventas y facturaciones

Estudio de factibilidad

Es la posibilidad que existe para realizar el proyecto. Se analizan 3 casos:

- ◆ **Factibilidad económica:** se deben mostrar los beneficios tangibles e intangibles con los cuales se favorece la empresa por sistematizar sus procesos manuales.
- ◆ **Factibilidad técnica:** se debe realizar un proceso investigativo para averiguar la viabilidad del hardware existente o por comprar y su absoluta compatibilidad con el software a desarrollar.

Factibilidad operativa: hace referencia a la capacitación que se le debe dar al personal de la empresa para que haga un adecuado manejo de la aplicación.

Introducción a las bases de datos

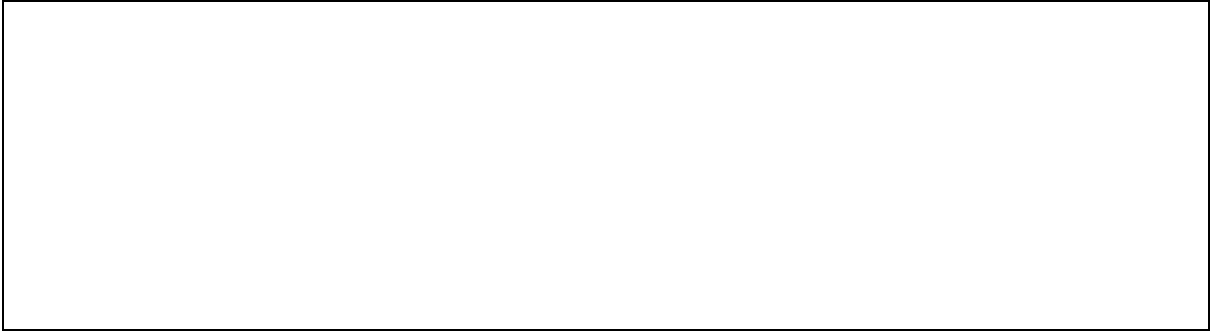
Una base de datos está conformada por un conjunto de campos, registros y tablas relacionadas entre sí y con características comunes.

- ◆ **Campos:** es un conjunto de datos del mismo tipo, también reciben el nombre de atributos, los campos se organizan en columna.
- ◆ **Registros:** también reciben el nombre de tuplas, son un conjunto de campos y se organizan en filas. El número de registros de una tabla equivale a su tamaño.
- ◆ **Tabla:** es un conjunto de campos y registros (filas y columnas), o sea, lo que se conoce como matriz. Una tabla, debe tener un identificador que sea único e irrepetible para cada uno de sus registros, este identificador también recibe el nombre de clave primaria (Primary key). La tabla debe llevar un nombre que debe ser en mayúscula y en singular.

Existen 2 tipos de tablas:

- ◆ **Tablas referenciales:** reciben este nombre porque otras tablas pueden hacer referencia a su información. Se identifican porque tienen clave primaria, campos o atributos.

Gráfica # 14: ejemplo de tabla referencial



- ◆ **Tablas de movimiento o relacional:** se reconocen porque tienen clave primaria, campos o atributos y clave foránea.

Una tabla de relación almacena el detalle o el movimiento de una tabla de referencia. Una clave foránea, permite relacionar una tabla de movimiento con una referencial. (También es conocida como foreign key)

Gráfica # 15

ESTUDIANTE-PROFESOR		OR		
CARNET	CODIGO	DIRECCION	TELEFONO	ELEFONO
001	002	calle 46 sur	2598749	2569874
002	003	ra 20 bb	3136599	3125599
003	002	ra 25 A-50	6825588	6325588

DIAGRAMA RELACIONAL

Gráfica # 16

Un diagrama relacional representa a las diferentes entidades y su respectivo código de relación.

Ejercicios tema Análisis de Sistemas

1. Suponga que se construirá un software para un almacén de cadenas, este producto se direccionará a la mejora del proceso de créditos del negocio (proceso de cartera):
 - a. Realice un cuestionario o encuesta que le permita preguntar lo esencial para comenzar con el proceso de desarrollo del producto.
 - b. Realice una depuración de los requisitos.
 - c. Redacte un informe en donde conste cuales requisitos seleccionó como primordiales y ¿por qué?
2. Teniendo como base el ejercicio anterior, construya los siguientes elementos del modelo de análisis:
 - a. Desarrolle un diagrama de casos de uso para el sistema de crédito.
 - b. Desarrolle un diagrama de actividad que permita validar los requisitos de la etapa de comunicación.
 - c. Realice un diagrama de carril en donde se distribuya las responsabilidades de cada actor participante.
3. De acuerdo a los requerimientos obtenidos en el punto uno (1) identifique lo que es:
 - a. Un objeto.
 - b. Un atributo
 - c. Una relación
 - d. Una asociación
 - e. Un modelo de comportamiento.
4. Construya un diagrama de flujo que le permita observar el comportamiento de la información entrante y saliente del recorrido del sistema
5. Construya un diagrama de clases tomando como base los requerimientos obtenidos el punto uno (1). (Relacione los objetos o clases encontradas)
6. Identifique las tablas o entidades principales y de movimiento que se observan en su sistema de crédito (partiendo de los requerimientos obtenidos) y proceda a:
 - a. Construir el modelo relacional
 - b. Construir el diagrama relacional.

Prueba Final

Realice los siguientes puntos:

1. Mediante un ejemplo práctico explique lo que es el Análisis orientado a objetos
2. Realice un análisis con respecto al tema sobre los casos de uso indicando la ventajas y desventajas
3. Mediante un diagrama de carril, indique la manera como fluye la información para la matrícula de un alumno en una carrera específica.
4. Simule con un ejemplo diferente el cuadro que aparece en la hoja 14, dando una explicación clara sobre el flujo de la información para que el cliente entienda lo que hará el nuevo sistema a implantar.
5. Que es una base de datos y de un ejemplo
6. Que es una tabla y de por lo menos tres ejemplos, teniendo presente
7. Haga un resumen de lo aprendido en esta unidad e indique como lo puede aplicar en la empresa ya sea real o ficticio.

3. DISEÑO DE SOFTWARE

OBJETIVO GENERAL

Plantear diversas estrategias de planificación, supervisión y control de procedimientos necesarios para la realización de un proyecto informático.

OBJETIVO ESPECÍFICOS

- ◆ Reconocer claramente la conceptualización con respecto al modelo de diseño para la construcción de prototipos que se ajusten a la necesidad del cliente
- ◆ Analizar los distintos tipos de datos que se utilizarán para el desarrollo de una aplicación
- ◆ Construir escenarios que ayuden a la interpretación de los requerimientos del cliente
- ◆ Diseñar varios estilos de interfaz del usuario que faciliten el buen manejo de la información
- ◆ Coadyuvar al desarrollo de un producto que garantice la calidad en todas sus fases de realización (obtención de requerimientos, planeación, análisis de requisitos, diseño de sistemas, construcción y despliegue o entrega del producto final).

Prueba Inicial

A continuación responda los siguientes puntos:

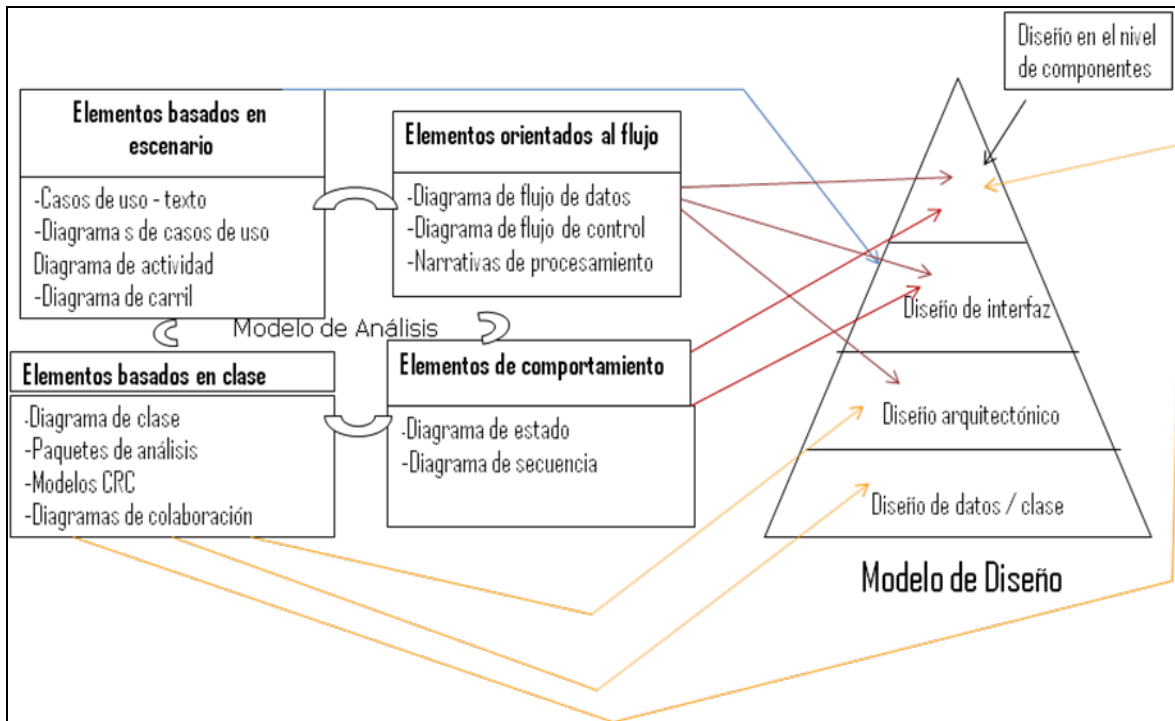
1. Como entiende usted lo que es el proceso y la calidad del software
2. Realice un diseño (simulado) en un aplicativo para observar la calidad requerida en la aplicación, puede hacerlo en cualquier aplicativo del office.
3. Como utilizaría la modularidad para el desarrollo de un software para la empresa el verdugo, dedicada a la compra y venta de productos textiles en donde desea tener la información bien clasificada en los puntos específicos de dicha empresa.
4. ¿Cree que la ocultación de información seria de suma importancia para que las empresas tengan asegurada dicha información? Justifique su respuesta

5. ¿Qué entiende usted por diseño de software?
6. Mencione una diferencia entre el análisis de software y lo que usted cree que es el diseño.
7. ¿Cuál cree usted que es la importancia del diseño de software?

3.1. Ingeniería del diseño

Diseño de sistemas

Gráfica # 17: Paso del modelo de análisis a un modelo de Diseño



◆ Proceso y calidad del diseño

El diseño de software es un proceso similar a los planos de un edificio, en donde se extrae paso a paso todo lo abstracto que hay dentro de él y luego se va organizado todo de una forma que se le vea sentido para alcanzar los objetivos⁴.

- ◆ Para evaluar si un diseño cumple con las expectativas esperadas se deben tener los siguientes criterios

- ◆ El diseño debe implementar todos los requisitos contenidos en el modelo de análisis y debe ajustarse a todos los requisitos del cliente
- ◆ El diseño debe ser una guía legible y comprensible para quienes generen código y quienes realizan pruebas y en consecuencia dan soporte al software.
- ◆ El diseño debe proporcionar una imagen completa del software, dando direcciones a los dominios de datos, funcionales y de comportamiento desde una perspectiva de implementación.
- ◆ Un buen diseño arquitectónico requiere las siguientes directrices:
 - ◆ Buena estructura mediante diseños reconocibles y que se pueda implementar de forma evolutiva
 - ◆ Diseño modular
 - ◆ Debe contener distintas representaciones de los datos
 - ◆ Debe conducir a estructuras de datos apropiadas
 - ◆ Reduzcan la complejidad de las conexiones
 - ◆ Debe hacer referencia a la información recolectada en la parte de análisis
 - ◆ Que permita claridad en lo que se quiere entregar al usuario final.
- ◆ **Conceptos del diseño**

Cuando se tiene un buen diseño se puede estar seguro que los demás procesos que se van a realizar funcionan adecuadamente con respecto a la solución que el cliente necesita para tomar decisiones acordes sin afectar al personal que tiene a su cargo.

Abstracción

Se debe seccionar nuestro código en grupos de código más pequeño que, al unirlos, hacen el trabajo. Un buen ejemplo de abstracción es el cuerpo humano, aunque el cuerpo es una unidad, está dividido en lo que conocemos por sistemas (el sistema respiratorio, el sistema linfático, cardiovascular, etc., etc.). Estos sistemas, a su vez están compuestos por otros más pequeños: los órganos, y así sucesivamente. La abstracción nos permite dividir nuestro programa en distintos objetos que se agrupan para formar cosas más complejas.

La abstracción es la capacidad de separar los elementos (al menos mentalmente) para poder verlos de forma singular. Como cuando describimos el cuerpo humano y decimos cabeza, brazo(s), pierna(s), etc.

Otro ejemplo de abstracción podemos considerar una puerta la cual contiene tipo, dirección, peso, dimensiones, entre otras.

Arquitectura

Se refiere a la estructura de los componentes del programa o módulos y la forma como estos interactúan para alcanzar unos objetivos concretos, tales como los modelos del marco de trabajo que tienen una arquitectura específica que permite orientar al desarrollador para alcanzar los objetivos específicos.

Patrón

Es considerada como una semilla de conocimiento y conserva la estructura que debe llevarse a cabo para lograr metas trazadas desde el inicio del proyecto. Los patrones que se utilizaran en el desarrollo de proyectos pueden ser reutilizados y con esto se disminuye un poco el tiempo y el costo de implantar nuevos patrones para los diferentes proyectos sobre los cuales se desea trabajar.

Modularidad

Se refiere a la división mediante módulo para los proyectos que se están desarrollando, permitiendo esta la interacción de los diferentes grupos de trabajo que están empeñada en solucionar problemas que las empresas y organizaciones presentan.

La manera más sencilla de entender, controlar y supervisar un proyecto es creando subdivisiones o subproyectos que orienten al personal que hará parte del trabajo a no perder el enfoque del logro del objetivo y contribuir a disminuir el tiempo y el esfuerzo utilizados para este.

Es más fácil moldear, guiar y solicitar ayuda para el desarrollo de un producto cuando se subdivide que cuando se quiere llevar todo por completo y con la probabilidad que el margen de error y de incumplimiento sea cada vez mayor.

Ocultación de información

Dentro de la programación el proceso de ocultación tiene que ver con el manejo de la información de manera interdependiente, es decir, que lo que se trabaje en con respecto a cierta información no sea alterada o modificada desde otras aplicaciones o formularios conservando esta la información original y que solo actualice lo que es necesario, incluyendo dentro de esto un proceso de trazabilidad para identificar claramente qué usuario hizo el respectivo ingreso al sistema.

La ocultación ayuda a definir las restricciones que debe tener el software y con ello se garantiza la transparencia de los datos ayudando a tomar decisiones que fortalezcan el crecimiento de la empresa u organización para permanecer en el mercado.

Refinamiento

Se refiere al trabajo detallado de cada uno de los procedimientos para garantizar el buen funcionamiento del proyecto. El refinamiento hace que el diseñador trabaje sobre un enunciado original permitiendo con esto que no se omita ningún detalle y así tener un mejor grado de confiabilidad.

Re fabricación

Es el proceso de cambiar un sistema de software de tal forma que no se altere el comportamiento externo de su código (diseño) y aun así se mejore su estructura interna.

Cuando se aplica la re fabricación se analiza si existen redundancias, algoritmos mal aplicados y procedimientos que no son muy funcionales para hacer la respectiva corrección y mejorarlos y con esto tener los resultados esperados.

Para el manejo de la información es de gran importancia los términos tratados en este capítulo ya que dicha información no puede ser manipulada o vista por cualquier persona sino única y exclusivamente por los implicados directamente porque son la vida de la empresa y organización.

La ocultación, la modularidad, abstracción son términos y/o temas que son complejos de manejar pero que son de vital importancia y sobre los cuales se debe dedicar un buen tiempo para su excelente funcionalidad para que garantice que esa información sea supervisada, controlada y mejorada y así satisfacer las necesidades del cliente.

El modelo de diseño

Para este modelo se hace uno de diagramas uml pero de una forma estructurada y refinada, en donde se explica con detalles las estructuras que están inmersas dentro de este para facilitar un mejor entendimiento, y de acuerdo a este orden en el que se entrega el diseño se va llevando el control en el desarrollo.

Dentro del modelado de diseño se deben tener presente todos los elementos que hacen parte de él, así como para una casa se necesita saber el número de habitaciones, el tamaño, la forma, las ventanas, las puertas y su ubicación, de igual manera se debe estar ubicado y bien estructurado

sobre lo que se va a hacer en la ejecución de un proyecto o en la elaboración de un desarrollo de software.

Dentro del modelado de diseño se deben destacar algunos elementos para garantizar un buen funcionamiento de la interfaz dentro de los cuales están:

Interfaz con el usuario: Es una de las partes fundamentales de la ingeniería del software, ya que es aquí en donde van implícitos todas las necesidades del cliente (distribución, tamaño, tipo y tamaño de letra, orden en el ingreso de la información, entre otros), para tener un control efectivo y que el cliente pueda satisfacer sus necesidades en el manejo de la información.

Interfaces externas: Se debe especificar como va a funcionar el sistema, si bajo red o como usuario independiente, que tipo de red va a utilizar y con esta información clara el desarrollador tomará las medidas necesarias para una planificación acorde a las necesidades del cliente.

Interfaces internas: Esto tiene que ver más con el diseño a nivel de componentes, el manejo adecuado de las clases para que la comunicación entre el sistema y el usuario se más amigable.

Dentro del modelado de diseño también se pueden desatacar los elementos a nivel de componentes, en donde se debe plasmar con detalles todas las partes que intervienen el proyecto por ejemplo para una vivienda se debe tener claro dónde van puestos los emisores y receptores, desagües, tina, closet, pisos, entre otros.

Y dentro de la parte del software se debe tener claridad en el modelo a implementar, las estructuras de los datos, modelos de datos, el proceso algorítmico, diagramas uml, entre otros.

Diseño de software basado en patrones

Los problemas que a diario suelen suceder desde todos los campos deben entenderse con claridad para aplicar un determinado patrón existente que cumpla con las expectativas necesarias de acuerdo a la necesidad.

Cada persona y según su campo de acción utiliza patrones específicos que sirven como base para la implementación de nuevos trabajos o nuevos proyectos, pero teniendo presente la gran variabilidad que se tiene entre uno y el otro y es ahí en donde se deben aplicar una serie de conceptos de ingeniería del software para que lo que se esté llevando a cabo cumpla con las expectativas que tiene el cliente sin alterar su objetivo.

Una de las principales dificultades que existen dentro del software basado en patrones es encontrar uno que se ajusten completamente a las necesidades de la nueva implementación.

Ejercicios tema Ingeniería del Diseño

Conteste las siguientes preguntas:

1. Si un diseño de software no es un programa, ¿entonces, qué es?
2. Describir con argumentos propios lo que es el diseño de software.
3. ¿Cómo se evalúa la calidad de un diseño de software?
4. ¿Qué es un modelo de diseño de software?
5. ¿Cómo se mide el proceso y la calidad del diseño de software?

3.2. Diseño arquitectónico

3.2.1. Arquitectura del software

La arquitectura del software de un sistema de cómputo es la estructura o las estructuras del sistema, que incluyen los diversos componentes de este, sus propiedades externas de ese componente y las relaciones entre estos.

Esta arquitectura de software no es el software operativo. El cual permite al ingeniero realizar las siguientes funciones⁵:

- ◆ Analizar la efectividad del diseño con el cumplimiento de los requisitos establecidos.
- ◆ Se tiene opciones arquitectónicas en el cual se puede efectuar cambios en el diseño.
- ◆ Se reduce los riesgos asociados en la construcción del software.

Importancia de la arquitectura del software

- ◆ Facilita la comunicación entre las partes interesados en el desarrollo del software.
- ◆ Se destaca las decisiones iniciales del diseño de este.
- ◆ Se puede construir un modelo pequeño que se pueda entender y así saber cómo trabajan los componentes.

Roger S. Presman, (Ingeniería del Software, Un Enfoque Práctico, sexta edición) Página 275

Diseño de datos

El diseño de datos cambia a los objetos definidos como parte del modelo de análisis en estructuras globales al nivel de componentes de software, y en ocasiones a una estructura de base de datos aplicativa.

Diseño de datos al nivel arquitectónico

En la comunidad de la tecnología de la información (TI) se ha desarrollado la minería de datos, es también llamada descubrimiento de conocimiento de Base de Datos (DCBD) en el cual por medio de la base de datos grande e independiente se almacena la información de acuerdo al tipo de negocio.

Diseño de datos al nivel de componentes

Se observa por la representación de estructuras de datos a la que se tiene acceso en forma directa mediante uno más de sus componentes.

Principios para la especificación de datos:

1. Los principios del análisis sistemático aplicado a la función y el comportamiento también deben aplicarse a los datos.
2. Se deben identificar todas las estructuras de datos y las operaciones que se realizarán.
3. Se debe establecer un mecanismo para la definición del contenido de cada objeto de datos y las operaciones que se les aplican.
4. Las decisiones del diseño al nivel de datos deben proponerse hasta una de las últimas etapas del proceso de diseño.
5. La representación de una estructura de datos sólo debe conocerse para los módulos que deben usar directamente los datos que contiene tal estructura.
6. Debe desarrollarse una biblioteca de estructuras de datos útiles y también las operaciones que pueden aplicárseles.
7. Un diseño de software y un lenguaje de programación deben dar soporte a la especificación y la realización de los tipos de datos abstractos

Estilos y patrones arquitectónicos

El software se construye para sistemas de cómputo también muestra uno o muchos estilos arquitectónicos. Cada estilo de estos tendrán las siguientes categorías del sistema:

- ◆ Conjunto de componentes que efectúan una función requerida por el sistema.
- ◆ Conjunto de conectores que permiten la comunicación, coordinación y cooperación entre componentes.
- ◆ Restricciones que definen como se integran los componentes para formar un sistema.
- ◆ Modelos sistemáticos que permiten a un diseñador comprender las propiedades del sistema.

Un patrón arquitectónico impone una transformación en el diseño de la arquitectura. Características:

- ◆ Alcance de un patrón es menor.
- ◆ Impone una regla sobre la arquitectura.
- ◆ Los patrones arquitectónicos tienden abarcar aspectos específicos del comportamiento dentro del contexto de la arquitectura.

Para garantizar que un desarrollo de software cumpla con lo esperado por el cliente, se debe tener y entregar un buen diseño que contemple toda aquella información que fue recolectada en el momento de estar realizando la comunicación con el cliente y la adecuada planeación, que estos dos puntos son los pilares para alcanzar el éxito o logro de objetivos en la solución de problemas en el manejo de la información.

El diseñador los prototipos donde agrupan los datos que utilizará el software debe transmitir la suficiente claridad en todos y cada uno de los diseños que está implementando porque esto ayuda a que la programación siga un rumbo hacia la entrega con calidad y a tiempo y no obstaculice los continuos procesos porque esto ocasionaría grandes dificultades a todo el personal que labora y a los clientes.

Diseño arquitectónico

El diseño arquitectónico debe ponerse en contexto el software que se habrá de desarrollar; es decir, el diseño debe definir las entidades externas con las que interactúa el software y la naturaleza de la interacción.

Representación del sistema de contexto

Un diagrama de contexto del sistema cumple con este requisito al representar el flujo de la información dentro y fuera del sistema, la información del usuario y el procesamiento relevante del soporte.

Los sistemas de destino se representan así:

- ◆ Sistemas superordinados: como parte de algún esquema de procesamiento del nivel más elevado.
- ◆ Sistemas Subordinados: que proporcionan a los datos o el procesamiento necesarios para completar la funcionalidad del sistema de destino.
- ◆ Sistemas al nivel de par: Los que interactúan de igual a igual.
- ◆ Actores: entidad que interactúan con el sistema de destino produciendo o consumiendo información necesaria para el procesamiento.

Refinamiento de la arquitectura en componentes

Al refinar la arquitectura del software en componentes, la estructura del sistema empieza a emerger.

¿Cómo se eligen los componentes? En primera instancia se eligen las clases de análisis que estas representan entidades dentro del dominio de la aplicación que deben atenderse dentro de la arquitectura del software. Otra de las fuentes del dominio es la infraestructura.

Ejercicios tema Diseño Arquitectónico

Conteste las siguientes preguntas:

1. ¿Quién o quienes realizan el diseño arquitectónico?
2. ¿Qué es un diseño arquitectónico?
3. ¿Cuáles son los pasos para realizar un diseño arquitectónico?
4. ¿Qué es la arquitectura del software?
5. ¿Por qué es importante la arquitectura del software?

3.3. Evaluación de diseños arquitectónicos alternos

El diseño produce varias opciones arquitectónicas que se evalúan para determinar cuál es la más apropiada respecto al problema que habrá de resolver.

◆ Método de análisis de compensación para la arquitectura

Las siguientes actividades de análisis del diseño se realizan iterativamente:

1. Recopilar escenarios.
2. Deducir requisitos, restricciones y descripción del entorno.
3. Describir los estilos/patrones arquitectónicos que se han elegido para dirigir los escenarios y requisitos.
4. Evaluar los atributos de calidad al considerar cada atributo de manera aislada.
5. Identificar la sensibilidad de los atributos de calidad respecto de varios atributos arquitectónicos para un estilo arquitectónico específico.
6. Analizar las arquitecturas alternas (desarrolladas en el paso 3) empleando el análisis de sensibilidad utilizado en 5 paso.

Complejidad arquitectónica

Esta consiste en considerar las dependencias entre componentes dentro de la arquitectura. Estas dependencias las orienta la información, el flujo de control, o ambas, dentro del sistema.

Zhao sugiere tres clases de dependencias

- ◆ Dependencias compartidas que representan relaciones de dependencia entre consumidores que usan el mismo recurso o los productores que producen para los consumidores.
- ◆ Dependencias de flujo que representan las relaciones de dependencia entre productores y consumidores de recursos.

- ◆ Dependencias restringidas que representan restricciones al flujo relativo de control entre un conjunto de actividades.

- ◆ **Correlación del flujo de datos en una arquitectura de software**

La correlación arquitectónica aplicada a la arquitectura de llamada y retorno (una estructura muy común en muchos tipos de sistemas). Esta técnica de correlación permite que un diseñador derive arquitectura llamada y retorno razonablemente complejas a partir de diagramas de flujo de datos dentro del modelo de análisis.

Flujo de transformación

La información ingresa en el sistema por rutas que transforman los datos externos en una forma interna. Estas rutas se identifican como flujo de entrada. En el núcleo del software ocurre una transición. Los datos entrantes se pasan por un centro de transformación y empiezan a moverse por rutas que ahora los llevan fuera del software. Estas se denominan flujos de salida.

Ejercicios tema Evaluación de Diseños Arquitectónicos

Conteste las siguientes preguntas:

1. ¿Qué es una recopilación de escenarios?
2. ¿Cuándo se presenta la complejidad arquitectónica de un software?
3. ¿Qué es una arquitectura de flujo de datos?
4. ¿Qué se entiende por flujo de transformación del diseño de software?
5. ¿Qué es para usted la arquitectura orientada a objetos?

3.4. Diseño de la interfaz del usuario

3.4.1. Las reglas de oro

Las reglas del diseño de la interfaz del usuario, son las siguientes:

1. Dar control al usuario.

Se definen varios principios de diseño que permiten al usuario mantener el control: Definir los modos de interacción de forma que el usuario no realice acciones innecesarias o indeseables, proporcionar una interacción flexible, incluir las opciones de interrumpir y deshacer la interacción del usuario, depurar la interacción a medida que aumentan los grados de destreza y permitir que se personalice la interacción, oculte al usuario ocasional los elementos técnicos internos, diseñar interacción directa con los objetos que aparecen en la pantalla.

2. Reducir la carga en la memoria del usuario

Los principios de diseño que logran que una interfaz reduzca la carga de memoria que recae en el usuario: reducir la demanda de memoria a corto plazo, definir valores por defecto que tengan significado, definir accesos directos intuitivos, el formato visual de la interfaz debe basarse en una metáfora tomada de la realidad, desglosar la información de manera progresiva.

3. Lograr que la interfaz sea consistente

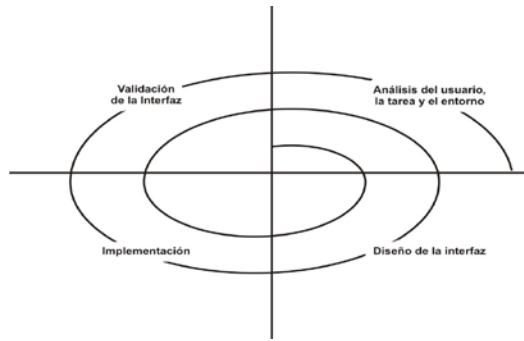
Los principios de diseño que ayudan a construir una interfaz consistente: Permitir que el usuario incluya la tarea actual en un contexto que tenga algún significado; mantener la consistencia en toda una familia de aplicaciones; si modelos interactivos anteriores han generado expectativas en el usuario, no hacer cambios a menos que haya razones inexcusables.

◆ Análisis y diseño de la interfaz de usuario

Cuando se analiza y se diseña una interfaz de usuario entran en juego cuatro modelos diferentes. Un ingeniero del software establece un modelo del usuario; el ingeniero del software crea un modelo de diseño; el usuario final desarrolla una imagen mental que suele denominarse modelo mental del usuario o percepción del sistema, y quienes implementan el sistema crean un modelo de la implementación.

El proceso:

1. Análisis y modelado de usuarios, tareas y entornos
2. Diseño de la interfaz
3. Construcción (implementación) de la interfaz
4. Validación de la interfaz



Gráfica # 18: ejemplo de análisis en interfaz de usuario

Análisis de la interfaz

Un principio clave de todos los modelos de procesos de ingeniería del software reza: es mejor comprender el problema antes de tratar de diseñar una solución. En el caso del diseño de la interfaz de usuario, comprender el problema significa comprender 1) a las personas (los usuarios finales) que interactuarán con el sistema por medio de la interfaz; 2) las tareas que los usuarios finales deben realizar para hacer su trabajo; 3) el contenido que se presenta como parte de la interfaz, y 4) el entorno en que se realizarán estas tareas.

Pasos del diseño de la interfaz

Los pasos son los siguientes:

1. Con base en la información desarrollada durante el análisis de la información, definir los objetos y las acciones de la interfaz (operaciones);
2. Definir eventos (acciones del usuario) que cambiarán el estado de la interfaz;
3. Representar cada estado de la interfaz tal como lo verá el usuario final;
4. Indicar cómo interpreta el usuario el estado del sistema a partir de la interfaz proporcionada mediante la interfaz.

Gráfica # 18: Ejemplos de Interfaz de usuario

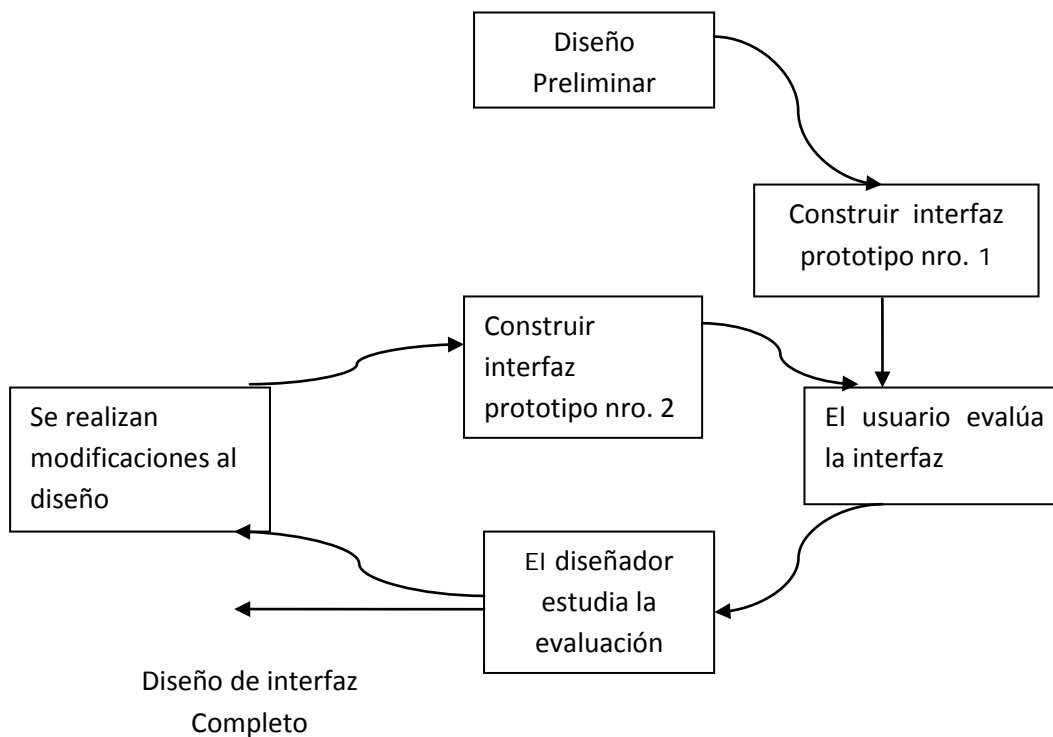




Evaluación del diseño

La evaluación puede abarcar un espectro de grados de formalidad que va desde una prueba de manejo informal, en la cual un usuario proporciona retroalimentación informal, hasta un estudio diseñado formalmente, el cual emplea métodos para la evaluación de cuestionarios que llena una población de usuarios finales.

Gráfica # 19: ciclo de evaluación del diseño de interfaz.



Ejercicios tema Diseño de la Interfaz

Conteste las siguientes preguntas:

1. ¿Qué es un diseño de interfaz de usuario?
2. Describa la mejor y la peor interfaz con la que usted haya trabajado alguna vez y realice una crítica constructiva al respecto teniendo presente los conceptos aprendidos en clase.
3. Realice una serie de tareas de análisis acerca de un software clínico que le permitan construir una interfaz hipotética.
4. ¿Es posible para usted afirmar que una interfaz de usuario es el elemento más importante de un sistema o producto de cómputo? Justifique su respuesta.
5. ¿Qué entiende por ciclo de evaluación del diseño de la interfaz?

3.5. Diseño a nivel de componentes

3.5.1. ¿Qué es un componente?

Un componente es un bloque de construcción modular para el software de cómputo.

Concepto orientado a objetos

Un componente tiene un conjunto de clases que colaboran entre sí.

El proceso de elaboración del diseño se ilustra imaginando que el software se construirá para una imprenta sofisticada. El objetivo general del software es recopilar las necesidades del cliente en el mostrador, cotizar un trabajo de impresión y pasarlo a una planta de producción automatizada.

El concepto convencional

Un componente es un elemento funcional de un programa que incorpora la lógica del procesamiento, las estructuras internas de los datos necesarios para implementar dicha lógica, y una interfaz que permita la invocación del componente y el paso de datos. Este tiene tres papeles importantes: componente de control, componente del dominio y componente de infraestructura.

Diseño de componentes basados en clases

La descripción detallada de los atributos, las operaciones y las interfaces empleados por estas clases representa el detalle de diseño requerido como precursor de la actividad de construcción.

Líneas generales de diseño al nivel de componentes

Un conjunto pragmático de líneas generales de diseño a medida que avanza el diseño al nivel de componentes. Estas líneas generales se aplican a componentes, sus interfaces y las características de dependencia y herencia que impactan el diseño resultante.

Cohesión

La cohesión implica que un componente o una clase sólo encapsula atributos y operaciones relacionadas estrechamente entre sí y con la clase del propio componente.

Acoplamiento

Es una medida cualitativa del grado al que las clases se conectan entre sí. A medida de que las clases (y los componentes) se vuelven más interdependientes, el acoplamiento aumenta. Un objetivo importante en el diseño al nivel de componentes consiste en mantener el acoplamiento lo más bajo posible.

◆ Conducción del diseño al nivel de componentes

El diseñador debe transformar la información del análisis y los modelos arquitectónicos en una representación de diseño que proporcione suficiente detalle para guiar la actividad de construcción (codificación y prueba).

Los siguientes pasos representan un conjunto de tareas típicas para el diseño al nivel de componentes

1. Identificar todas las clases de diseño que corresponden al dominio del problema.
 2. Identificar todas las clases de diseño que corresponden al dominio de infraestructura.
 3. Elaborar los detalles de diseño que no se adquieran como componentes reutilizables.
- 3ª. Especificar los detalles del mensaje cuando las clases o los componentes colaboran.

- 3b. Identificar las interfaces apropiadas para cada componente.
- 3c. Elaborar atributos y definir los tipos y las estructuras de datos necesarios para implementarlos.
- 3d. Describir de manera detallada el flujo de procesamiento dentro de cada operación.
- 4. Describir fuentes de datos persistentes (bases de datos y archivos) e identificar las clases necesarias para manejarlas
- 5. Desarrollar y elaborar representaciones del comportamiento de una clase o un componente.
- 6. Elaborar diagramas de despliegue para proporcionar detalles de la implementación adicional.
- 7. Factorizar todas las representaciones del diseño al nivel de componentes y siempre deben considerarse alternativas.

Lenguaje de restricción de objetos

El lenguaje de restricción de objetos (OCL) complementa al UML al permitir que un ingeniero de software use gramática y sintaxis formales para construir instrucciones sin ambigüedades relacionadas con varios elementos del modelo de diseño. Las instrucciones se construyen en cuatro pasos: 1) un contexto que define la situación limitada en que es válida la instrucción; 2) una propiedad que representan algunas características del concepto; 3) una operación (aritmética, orientada a conjuntos) que manipula o califica una propiedad, y 4) palabra clave (como if, then, else, and, or, not, implies) con que se especifican expresiones condicionales.

Diseño de componentes convencionales

Las construcciones son secuencia, condición y repetición. Secuencia implementa los pasos de procesamiento esenciales en la especificación de cualquier algoritmo. Condición proporciona las funciones para el procesamiento seleccionado con base en algún evento lógico, y repetición permite los bucles. Estas tres construcciones son fundamentales para la programación estructurada, que es una importante técnica de diseño al nivel de componentes.

La complejidad de las métricas indica que el uso de las construcciones estructuras reduce la complejidad del programa y, por tanto, mejora las opciones de lectura, prueba y mantenimiento. El uso de un número limitado de construcciones lógicas también contribuyen a un proceso de comprensión humana que los psicólogos llaman fragmentación.

Ejercicios tema Ingeniería del Diseño

Conteste las siguientes preguntas:

1. ¿Por qué son necesarios los componentes de control en el software convencional y no lo son en el orientado a objetos?
2. ¿Qué es un componente de software?
3. ¿Qué es una fuente de datos persistente?
4. Investigue y desarrolle una lista de categorías típicas para los componentes de infraestructura.
5. ¿Es posible decir que los componentes del dominio del problema nunca deben mostrar acoplamiento externo? Si está de acuerdo, ¿Cuáles tipos de componentes mostrarían este tipo de acoplamiento?
6. Construya un ejemplo de diseño de componentes.

Prueba Final

1. ¿Cómo puede saber que un software tiene un diseño de alta calidad?
2. Como entiende el término abstracción y explíquelo con dos o tres ejemplos prácticos
3. ¿Cuando hablamos de patrón es lo mismo que hablar de patrón, cuales su diferencia?
4. Como ve usted la ocultación de información, lo ve necesario o simplemente un capricho del proveedor, explique su respuesta
5. Que es la re fabricación de software
6. Porque es tan importante un buen diseño de datos
7. Cree usted que mientras más complejo sea el diseño arquitectónico de un software puede garantizar una mejor calidad en sus resultados esperados?

8. Las siguientes actividades de análisis del diseño se realizan iterativamente, realice un análisis con respecto a los siguientes puntos
 - ◆ Deducir requisitos, restricciones y descripción del entorno.
 - ◆ Describir los estilos/patrones arquitectónicos que se han elegido para dirigir los escenarios y requisitos.
 - ◆ Evaluar los atributos de calidad al considerar cada atributo de manera aislada.
9. El flujo de transformación lo podemos comparar con la teoría general de sistemas en donde se tienen entradas, procesos, salidas y retroalimentación?, justifique su respuesta
10. ¿Cree usted que el diseño de componentes basados en clases es lo mismo que un diseño tradicional u orientado a procedimientos?
11. Como entiende usted el acoplamiento dentro del desarrollo de software
12. Existen tres reglas de oro dentro de la parte de desarrollo de software las cuales son: Dar control al usuario, Reducir la carga en la memoria del usuario, Lograr que la interfaz sea consistente, como explicaría estos tres puntos que son fundamentales para tenerlos presente al desarrollar el software.
13. Cree usted ser capaz de evaluar un diseño de un software según lo aprendido hasta el momento?, justifique

4. LAS PRUEBAS Y LA MEDICIÓN

OBJETIVO GENERAL

Inculcar en el estudiante sobre la responsabilidad que tiene cuando se enfrenta a la construcción de un determinado proyecto y a la aplicación idónea de todos los conocimientos adquiridos en la materia y la recopilación de otras.

OBJETIVOS ESPECÍFICOS

- ◆ Comprender la importancia de la ejecución de las pruebas de software durante y después de su construcción.
- ◆ Determinar la importancia de la elaboración de un producto competitivo que sea aceptado en el mercado por su facilidad de uso y facilidad de mantenimiento, teniendo presente las características fundamentales que propicien la creación de la solución a los problemas que a diario se presentan en la empresas u organizaciones.

Prueba Inicial

Responda las siguientes preguntas relacionadas a continuación:

- 1.Cuál es la diferencia entre verificación y validación de software
- 2.¿Qué entiende usted por una prueba de software?
- 3.¿Qué entiende por pruebas de integración?
- 4.¿Qué entiende por prueba de humo?
- 5.¿Qué entiende por pruebas del sistema?
- 6.¿Qué entiende por pruebas de caja negra e indique con dos ejemplos?
- 7.¿Cómo cree usted que se desarrollan las pruebas para un producto de software?
8. Para el desarrollo de un software educativo:

- a) Cómo ejecutaría usted las pruebas.
- b) Sobre que parámetros realizaría las pruebas al producto.
- c) En la implementación de las pruebas, ¿qué es lo primero que tendría en cuenta y por qué?

4.1. Estrategias y la medición

Enfoque para la prueba de software

La prueba es un conjunto de actividades que se planean con anticipación y se realizan de manera sistemática. Por tanto, se debe definir una plantilla para las pruebas del software.

Todas proporcionan al desarrollador del software una plantilla para pruebas y todas tienen las siguientes características genéricas:

Las pruebas que hace el equipo de software debe ser muy efectivas porque con esto se eliminan todos los errores existentes para garantizar la calidad del software donde es dirigido por el desarrollador y aplicada en diferentes momentos

Verificación y validación de las pruebas

Verificación es el conjunto de actividades que aseguran que el software implemente correctamente una función específica.

Validación es un conjunto diferente de actividades que aseguran que el software construido corresponde con los requisitos del cliente.

Organización para las pruebas del software

El desarrollador del software siempre será el responsable de probar las unidades individuales (componentes) del programa y asegurar que cada una realice la función o muestre el comportamiento para el que se diseñó. En muchos casos, el desarrollador también aplica la prueba de integración. Sólo después de que la arquitectura del software esté completa participará un grupo independiente de prueba.

El papel de un grupo independiente de prueba consiste en eliminar los problemas propios de dejar que el constructor pruebe lo que él mismo ha construido.

◆ Estrategia de prueba para arquitecturas convencionales de software

Al principio, la ingeniería del sistema define el papel del software y lleva al análisis de los requisitos de éste, donde se establecen el dominio de la información, la función, el comportamiento, el desempeño, las restricciones y los criterios de validación del software. Al desplazarse hacia el interior de la espiral se llega al diseño y por último, a la codificación.

Es recomendable que dentro de las pruebas se haga un buen seguimiento de la espiral de adentro hacia afuera y de esta manera se asegura que el trabajo del software sea efectivo y de buena calidad.

◆ Estrategia de prueba del software orientado a objetos

La definición de prueba debe ampliarse para incluir técnicas de descubrimiento de errores que se aplican para analizar y diseñar modelos.

También dentro de esta prueba es necesario probar lo pequeño y seguir hacia lo grande, pero teniendo presente que se debe aplicar con claridad cada una de las pruebas independiente del tamaño del proceso que se lleve a cabo.

Además dentro de esta prueba se debe analizar cada una de las clases que están inmersas dentro del software ya que la comunicación entre ellas debe ser muy directa.

Aspectos estratégicos

Deben atenderse los siguientes temas, si se desea implementar con éxito una estrategia de prueba del software:

- ◆ Especificar los requisitos del producto de manera cuantificable mucho antes de que empiecen las pruebas
- ◆ Establecer explícitamente los objetivos de la prueba.
- ◆ Comprender cuáles son los usuarios del software y desarrollar un perfil para cada categoría de usuario.
- ◆ Desarrollar un plan de prueba que destaque la prueba de ciclo rápido.
- ◆ Usar revisiones técnicas formales y efectivas como filtro previo a la prueba.

- ◆ Desarrollar un enfoque de mejora continua para el proceso de prueba.

Resumen de las estrategias de pruebas más conocidas:

Estrategias de prueba para el software convencional

Las pruebas son importantes hacerlas desde el principio y también en medio de la marcha ya que es más fácil detectar las fallas o errores que pueda existir y así se deja para cuando finalice el trabajo será difícil encontrar los mínimos errores que pueden ocasionar grandes pérdidas de información y en un alto porcentaje no se encuentran las fallas al primer intento.

◆ **Prueba de unidad**

Se refiere a la verificación de la unidad mínima del diseño de software y estas están centradas en la lógica aplicada internamente y las estructuras de datos.

Es necesario observar con claridad el flujo de datos de la interfaz para que las pruebas sean de gran éxito.

Se puede utilizar todos los tipos de prueba necesarios para que la calidad del software se vea reflejada en el momento de su ejecución, teniendo presente que un alto porcentaje del total de desarrollo de una aplicación está en las pruebas.

Prueba de Integración

Es una técnica sistemática para construir la arquitectura del software mientras, al mismo tiempo, se aplican las pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar componentes a los que se aplicó una prueba de unidad y construir una estructura de programa que determine el diseño.

Integración descendente. La integración descendente es un enfoque incremental para la construcción de la arquitectura del software. Los módulos se integran al descender por la jerarquía de control, empezando con el módulo de control principal.

Integración ascendente. Debido a que los componentes se integran de abajo hacia arriba, siempre está disponible el procesamiento requerido para los componentes subordinados a un determinado nivel y se elimina la necesidad de resguardos.

Prueba de regresión. Se vuelven a ejecutar otra vez el subconjunto de pruebas para descubrir que los cambios no han afectado el sistema en su totalidad o parcialmente.

Prueba de humo. Es una prueba que se realiza mientras se está construyendo el software, para que el equipo de software evalúe el ritmo que se está llevando a cabo. Con esta prueba de humo se minimiza el riesgo, se mejora la calidad del producto final, se simplifica el diagnóstico y la corrección de errores y el progreso es más fácil de evaluar.

Estrategias para probar el software orientado a objetos

El objetivo de probar, para definirlo de manera simple, es encontrar la mayor cantidad de errores aplicando una cantidad manejable de esfuerzo en un periodo realista.

Prueba de unidad en el contexto orientado a objetos

Una clase encapsulada suele ser el eje de las pruebas de unidad. Sin embargo, las unidades más pequeñas son las operaciones dentro de la clase. Debido a que una clase puede contener varias operaciones diferentes y a que una operación determinada puede existir como parte de varias clases diferentes, deben cambiar las tácticas aplicadas para la prueba de unidad.

Prueba de integración en el contexto orientado a objetos

Hay dos estrategias diferentes para la prueba de integración de los sistemas orientados a objetos. La primera, prueba basada en subprocesos, integra el conjunto de clases requerido para responder a una entrada o un evento del sistema. La prueba de regresión se aplica para asegurar que no se presenten efectos colaterales. El segundo enfoque de integración, la prueba basada en el uso, empieza la construcción del sistema con la prueba de esas clases que usan muy pocas clases de servidor (o ninguna).

◆ Pruebas de validación

Esta comienza cuando la prueba de integración termina. Aquí desaparece la distinción entre el software convencional y el orientado a objetos

La validación se define muchas formas, pero una definición simple es que se alcanza cuando el software funciona de tal manera que satisface las expectativas razonables del cliente.

Prueba del sistema

Uno de los principales problemas que suceden aquí, es apuntar con el dedo, cuando se culpa a otro por el error cometido. Para evitar este inconveniente es necesario diseñar rutas de manejo de errores, aplicar pruebas que simulen datos incorrectos, registrar los resultados de las pruebas como evidencia y participar activamente en la planeación y el diseño de pruebas.

Prueba de recuperación

Aquí se busca que el software falle de alguna manera y así poder lograr que la recuperación sea apropiado, en caso de tener errores se hace la evaluación para la solución en el menor tiempo posible.

Prueba de seguridad

La prueba de seguridad comprueba que los mecanismos de protección integrados en el sistema realmente lo protejan de irrupciones inapropiadas.

Prueba de resistencia

Aquí se lucha mucho por encontrar un error en donde el usuario que lleva a cabo las pruebas se interroga hasta dónde puede llegar antes de encontrar un error.

Prueba de desempeño

La prueba de desempeño está diseñada para probar el desempeño del software en tiempo de ejecución de un sistema integrado. Esta prueba se hace hasta que se logre asegurar la integración de todos los elementos y se tenga un buen desempeño del sistema.

El proceso de depuración

La depuración ocurre como consecuencia de una prueba realizada con éxito. Es decir, cuando un caso de prueba descubre un error, la depuración es la acción que lo elimina.

Dentro de este proceso el síntoma y la causa pueden estar separados o el síntoma lo cause algún error o el síntoma finalice el corregir un error o puede ser un error humano, entre otros.

Ejercicios tema Estrategias Y la Medición

Responda las siguientes preguntas:

1. Con palabras propias describa la diferencia entre verificación y validación.
2. Elabore una lista de algunos problemas que pudieran estar asociados con la creación de un grupo independiente de pruebas para el sistema de crédito Planteado en los ejercicios propuestos de la etapa de análisis.
3. Como afecta la calendarización la prueba de integración.
4. Desarrollar una estrategia de prueba completa para el sistema de crédito propuesto en la etapa de análisis.
5. Quien cree usted que debe aplicar la prueba de validación: el desarrollador o el Usuario de software y ¿porque?
6. Investigue que es una prueba de bucles y de ejemplo direccionándolo el sistema de Crédito.
7. Realice un análisis de lo que permite hacer un método gráfico de prueba. (Entregue el documento organizado a su docente)

4.2. Técnicas de prueba de software

4.2.1. Técnicas de prueba de software

Características que propician la creación de software

- ◆ **Operatividad** Cuanto mejor funcione, con mayor eficiencia podrá probarse.
- ◆ **Observabilidad.** Lo que se ve es lo que se pruebe.

- ◆ **Controlabilidad.** Cuando mejor se controle el software, mejor se automatizarán y mejorarán las pruebas.
- ◆ **Simplicidad** Cuanto menos haya que probar, más rápido se hará.
- ◆ **Estabilidad** Cuanto menos cambios haya, menores alteraciones habrá en la prueba.
- ◆ **Facilidad de comprensión** Cuanto mayor información se tenga, con mayor inteligencia se aplicará la prueba.

Características de una buena prueba

Tiene elevada probabilidad de encontrar error, no es redundante, debe ser la mejor y debe ser muy simple

Pruebas de la estructura de control

Prueba de condición: Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en un módulo del programa.

Prueba del flujo de datos: El método de prueba de flujo de datos selecciona rutas de prueba en un programa de acuerdo con las ubicaciones de las definiciones y los usos de las variables en el programa.

Prueba de caja negra: Permiten al ingeniero de software derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa.

Las pruebas de caja negra tratan de encontrar errores en las siguientes categorías:

- 1) Funciones incorrectas o faltantes,
- 2) errores de interfaz,
- 3) errores en estructuras de datos o en acceso a bases de datos externas,
- 4) errores de comportamiento o desempeño, y 5) errores de inicialización y término.

Prueba basada en fallas: El objetivo de la prueba basada en fallas dentro del sistema orientado a objetos es diseñar pruebas que tengan una alta probabilidad de descubrir posibles fallas.

◆ Métricas de software

Calidad general

Es el cumplimiento de los requisitos de funcionalidad y desempeño explícitamente establecidos, de los estándares de desarrollo explícitamente documentados y de las características implícitas que se esperan de todo software desarrollado profesionalmente.

Marco conceptual para las métricas del producto

Aunque medida, medición y métrica son términos que suelen utilizarse de manera intercambiable, es importante observar las sutiles diferencias entre ellos. En el contexto de la ingeniería de software una medida proporciona una indicación cuantitativa de la extensión, la cantidad, la capacidad o el tamaño de algún atributo de un producto o proceso.

Medición: Es el acto de determinar una medida.

Métrica: como una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo determinado.

Un ingeniero de software recopila medidas y desarrolla métricas para obtener los indicadores. Un **indicador** es una métrica o una combinación de métricas que proporcionan conocimientos acerca del proceso del software, un proyecto de software o el propio producto.

Principios de medición

Antes de introducir una serie de métricas del producto que:

- 1) ayuden los momentos de análisis y diseño,
- 2) ofrezcan una indicación de la complejidad de los diseños procedimentales y el código fuente, y
- 3) faciliten el diseño de pruebas más efectivas, es importante comprender los principios básicos de la medición.

Roche sugiere un proceso de medición al que caracterizan cinco actividades: formulación, recolección, análisis, interpretación y retroalimentación.

Principios:

- ◆ Una métrica debe tener propiedades matemáticas deseables.
- ◆ Cuando una métrica representa una característica de software que aumenta cuando se presentan rasgos o positivos o que disminuye al encontrar rasgos indeseables, el valor de la métrica debe aumentar o disminuir en el mismo sentido.
- ◆ Cada métrica debe validarse empíricamente en una amplia variedad de contextos antes de publicarse o aplicarse en la toma de decisiones.

Las pruebas es un conjunto de actividades que se planean con anticipación y de manera organizada, usando plantillas específicas que permitan detectar las fallas existentes.

Dentro de las pruebas son varias personas las que deben intervenir para garantizar la calidad de dicho desarrollo, cuyas personas pueden ser el mismo desarrollador, terceras personas y los futuros clientes, teniéndose claridad en los puntos a evaluar durante todo el proceso de prueba.

Dentro del planteamiento de desarrollo de software es bueno manejar adecuadamente la espiral o el orden de prioridades para no alterar los procesos que son los que harán cumplir con lo esperado por el cliente.

Dentro de las diferentes pruebas en el software es bueno destacar algunos aspectos tales como observar los requisitos, establecer los objetivos, planear el plan de pruebas, revisión técnica formal y el enfoque de mejora.

Ejercicios tema Técnicas de Pruebas

Responda las siguientes preguntas:

1. ¿Con que objetivo se realiza una prueba de software?
2. ¿Qué es una prueba de software?
3. ¿Qué es una prueba de caja blanca?
4. ¿Qué es una prueba de caja negra?
5. ¿Cuántas clases de prueba de software conoce usted? Relaciónelas.
6. ¿Qué es un patrón de prueba?
7. ¿Qué es una prueba de sistema de tiempo real?
8. ¿Qué es una prueba de interfaz gráfica de usuario?
9. ¿Qué entiende por diseño de caso de prueba de interclase?

Prueba Final

1. Cree la verificación y la validación de software es lo único que se debe hacer para asegurar la calidad del software?, justifique su respuesta
2. Quienes serán los responsables de realizar las pruebas del software y harían dentro de ellas o como plantearía usted los aspectos a tener en cuenta para dicha prueba?
3. Consulte en internet sobre las pruebas de caja negra y realice un ensayo y entregárselo al docente con respecto al tema.
4. Explique cada una de las Características que propician la creación de software.
5. Haga una clara diferencia entre la medición y la métrica

5. MODELOS OPERATIVOS WEB

OBJETIVO GENERAL

Inducir a que el estudiante experimente individual y colectivamente a través de la realización de diversos laboratorios la funcionalidad y puesta en marcha del desarrollo de un proyecto de excelente calidad.

OBJETIVOS ESPECÍFICOS

- ◆ Identificar la importancia del desarrollo e implementación del software a través de la web, clasificando los diferentes atributos y métodos utilizados que contribuyan en una solución que garantice confiabilidad y seguridad en la administración de su información.

Prueba Inicial

1. ¿Cree usted que los modelos operativos web, son la solución a todos los problemas empresariales actuales y a futuro?, Justifique su respuesta
2. ¿Cree usted que para aplicar el modelo web es necesario tener presente el marco de trabajo de desarrollo de software?, justifique su respuesta
3. ¿Cuál sería el cronograma de actividades que usted le plantearía a un cliente para realizarle un desarrollo de software tipo web?
4. ¿Cómo cree usted que debía ser la motivación que debe tener una persona para elaborar un desarrollo web?, justifique su respuesta
5. Si usted fuera el cliente x que necesita un desarrollo web, ¿cuál de la siguiente opción elegiría y por qué?: Subcontrataría un tercero o empresa con alto grado de confiabilidad y con experiencia o contrataría a la persona que le está llevando a cabo el análisis del sistema actual que también tiene experiencia y cuyo costo puede aumentar un poco
6. ¿Cómo y para qué probaría usted un aplicativo web?

5.1. Desarrollo Web

5.1.1. Ingeniería web

La ingeniería Web aplica a los sólidos principios científicos, de ingeniería y de administración, y enfoques disciplinados y sistemáticos para el desarrollo, despliegue y mantenimiento exitosos de sistemas y aplicaciones basados en Web de alta calidad.

Atributos de los sistemas y aplicaciones basados en Web

En la actualidad, LAS WebApps han evolucionado en sofisticadas herramientas de computación que no sólo proporcionan función por sí mismas al usuario final, sino que también se han integrado con bases de datos corporativas y aplicaciones de negocios.

En la mayoría de WebApps se encuentran los siguientes atributos: Intensidad de red, concurrencia, carga impredecible, desempeño, disponibilidad, gobernada por los datos, sensibilidad al contenido, evolución continua, inmediatez, seguridad y estética.

Pensar en el desarrollo de un aplicativo web sin tener claridad en los atributos anteriores en trabajar sin un enfoque bien definido y esto hace que tanto el desarrollador como el cliente pierdan el tiempo ya que dentro de la planificación que se hace se definen con claridad lo que el aplicativo va a llevar y así garantizar la calidad y funcionalidad de dicho trabajo.

No basta con que el trabajo quede con excelente presentación, es fundamental que la funcionalidad esté acorde a las necesidades del cliente y de los usuarios que harán uso de ella.

Estratos de la ingeniería WepApp

Los procesos, métodos y tecnologías (herramientas) proporcionan un enfoque en estratos de la IWeb que es conceptualmente idéntico a los estratos de la ingeniería del software tales como procesos, métodos y herramientas y tecnología

- ◆ **Proceso:** El problema debe analizarse muy minuciosamente para garantizar la eficiencia y eficacia del producto final, debe diseñarse varios estilos para tomar la decisión por el mejor, debe guiarse de una manera incremental y organizar el enfoque de prueba que va a ser utilizada.

- ◆ **Métodos:** Son las labores técnicas que ayudan a interpretar la información para el aplicativo web teniendo presente los métodos de comunicación, análisis de requisitos, métodos de diseño y métodos de prueba
- ◆ **Herramientas y tecnología:** En esta parte se debe tener presente que las nuevas exigencias del medio y los avances tecnológicos y tanto las nuevas herramientas y lenguajes de programación, han puesto al desarrollador a hacer uso de ellas y realizar trabajos de acuerdo a estándares.

El proceso de Ingeniería Web

El desarrollador de este tipo de aplicativos debe seleccionar el modelo de proceso ágil que le permita alcanzar el objetivo de una forma rápida para continuar con el o los siguientes aplicativos, en caso de que la aplicación a desarrollar sea muy grande se debe seleccionar un modelo de desarrollo incremental.

Para el desarrollo de grandes aplicaciones de este estilo se requiere de un buen número de personas con competencias específicas para que cada uno tenga el trabajo a desarrollar y así evitar pérdida de tiempo que luego se traduce en dinero que no se puede recuperar.

Dentro de estos integrantes se puede contar con analistas, diseñadores, administradores de bases de datos, publicistas, entre otros.

Así como al desarrollar una aplicación en un determinado lenguaje de programación en donde se le debe aplicar el marco de trabajo de un proyecto, de igual manera con un aplicativo web se deben aplicar las misma subdivisión del marco tales como son comunicación, planeación, modelado, construcción y despliegue, los cuales debe ser llevados paso a paso para el alcance de los objetivos.

Mejores prácticas en Ingeniería Web

Estos son un conjunto fundamental de mejores prácticas:

1. Tomar tiempo para atender las necesidades del negocio y los objetivos del producto, incluso si los detalles de la Web App son vagos.
2. Describir cómo interactúan los usuarios con la WebApp aplicando un enfoque basado en escenarios.
3. Desarrollar un plan del proyecto, incluso si es muy breve.

4. Utilizar algún tiempo para modelar lo que se construirá.
5. Revisar la consistencia y calidad de los modelos.
6. Utilizar herramientas y tecnologías que permitan construir el sistema con tantos componentes reutilizables como sea posible.
7. No apoyarse en usuarios anteriores para depurar la WebApp; diseñense pruebas amplias y ejecútense antes de liberar el sistema.

Formulación de sistemas basado en Web

Para un buen desarrollo de aplicaciones basadas en web se debe tener claridad en la secuencia las cuales comienza con el reconocimiento de las necesidades del negocio, los objetivos que tiene la web y las funciones que van a ser implementadas en dicho desarrollo

Es de suma importancia dentro del desarrollo de aplicaciones web, preguntar dónde se inicia con el análisis de requisitos y donde culmina la formulación del problema

Dentro del desarrollo de aplicaciones web puede surgir varios interrogantes que hacen que exista o puede definirse el objetivo principal del trabajo que se va a desarrollar tales como

1. ¿Cuál es la motivación?
2. ¿Cuáles son los objetivos?
3. ¿Quién usará la aplicación?

En esta parte no se desarrolló a nivel general todos los objetivos ni la aplicación sino que se enuncia a nivel general lo que va a contener.

Dentro de todo este proceso se debe detectar las metas informativas y las metas aplicables utilizadas dentro de la aplicación a implementar.

Luego de detectar las metas a utilizar se debe desarrollar un perfil del usuario para simular los usuarios que van a utilizar la aplicación y la seguridad que va a tener.

Para recopilar la información que necesita para el desarrollo de la aplicación es necesario tener una buena comunicación con el usuario de tal manera que exprese y dé a conocer lo que necesita para lograr el objetivo esperado al finalizar el trabajo.

◆ El equipo de Ingeniería Web

Se debe conformar equipos idóneos con excelente talento y que trabajen conjuntamente ya que para este tipo de desarrollos el cliente necesita resultados en el menor tiempo posible y aplicando la tecnología más adecuada y actualizada, dado a que esta tecnología cambia permanentemente, la empresa no puede estar invirtiendo grandes sumas de dinero por el plazo exagerado que algunos desarrolladores requieren para terminar un aplicativo.

Los actores

Los equipos que se conforman para la solución de aplicaciones web, pueden ser en la misma forma como lo hacen en el desarrollo tradicional, pero deben ser personas con grandes habilidades en el manejo adecuado de herramientas que servirán como base para una excelente presentación y calidad de aplicativo web.

◆ Conflictos de gestión de proyecto para ingeniería Web

Luego de definir cómo va a quedar el aplicativo web, la empresa debe elegir la manera cómo va a ser desarrollada la aplicación, la cual puede ser por: subcontratar una empresa que tenga experiencia y que se comprometa con la entrega con una excelente calidad, al menor costo y al menor tiempo posible o contratar una personas por la empresa con alto nivel de conocimiento en este desarrollo y tenerlo dentro de la nómina de la empresa como un empleado interno.

El modelo de análisis de WebApp

El modelado de análisis para una WebApp se basa en la información que contienen los casos de uso desarrollados para la aplicación. Existen varias actividades dentro del análisis que hacen que las aplicaciones web proporcionen una funcionalidad adecuada a las necesidades del cliente, dentro de las cuales se tienen:

1. Análisis de contenido: Este incluye texto, gráficas e imágenes, datos de video y audio.
2. Análisis de interacción: describe cómo interactúa el usuario con la web.
3. Análisis de funciones: define las operaciones que se aplicarán al contenido de la WebApp.
4. Análisis de configuración: describe el ambiente y la infraestructura en la que reside la WebAPP.

Análisis Relación – Navegación

Proporciona los pasos necesarios para descubrir la relación entre los elementos existentes para la creación del modelo.

El análisis de relación se organiza en cinco pasos:

Análisis de participantes (categorías de usuario)

Análisis de elementos (objetos de contenido y elementos funcionales)

Análisis de relaciones (relación entre las aplicaciones web)

Análisis de navegación (como acceden los usuarios a los elementos individuales o grupales)

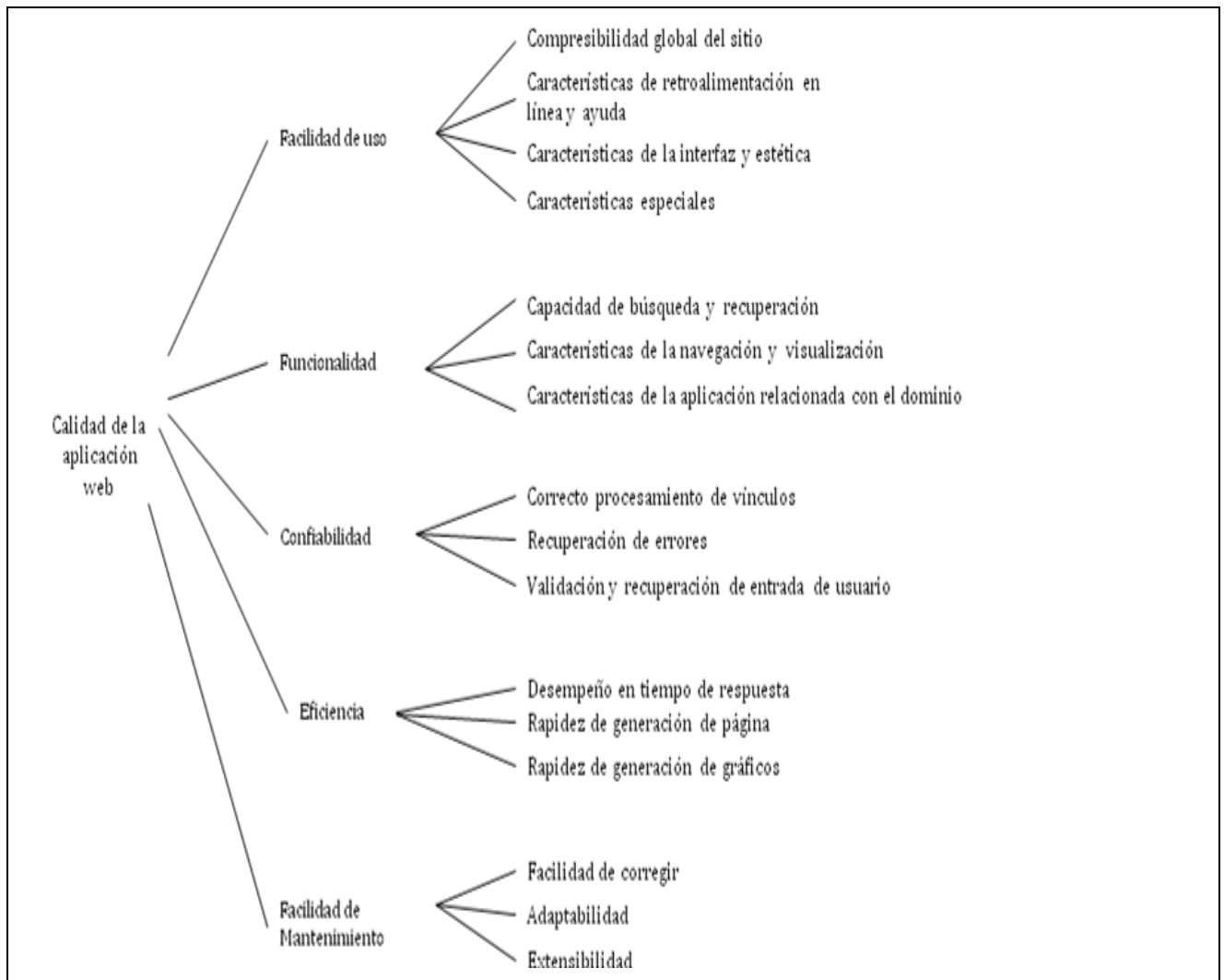
Análisis de evaluación (costo-beneficio).

Modelado de diseño para aplicaciones web

El modelo de diseño, sin importar su forma, debe contener suficiente información para reflejar cómo habrán de traducirse los requisitos de los participantes en contenido y código ejecutable y dependiendo de esta claridad se pueda ejecutar sin ningún inconveniente la aplicación.

Es recomendable que para que una página satisfaga las necesidades del cliente debe contener los siguientes:

Facilidad de uso, funcionalidad, confiabilidad, eficiencia y facilidad de mantenimiento.



Gráfica # 20: Modelo de diseño para aplicaciones web de alta calidad.

Metas de diseño:

Simplicidad: Realizar la aplicación lo más simple posible, pero que no se olvide de la calidad

Consistencia: Se deben conservar los mismos tonos de color de diseño y fuente para todas las páginas dependientes de la principal.

Identidad: Que la página diseñada tenga que ver directamente con el producto o servicio que se desea implementar.

Robustez: Que contenga los contenidos de forma completa para que el cliente o usuario alcance el objetivo que busca

Navegabilidad: Que la navegación por la página sea clara y sencilla, que no tenga que hacer uso de grandes manuales para interpretarla

Apariencia visual: Que los diseños y colores estén bien combinados, dado a que por la vista se determina si la página será atractiva o no.

Compatibilidad: Que la aplicación puede utilizarse bajo diferentes ambientes y que la información y presentación no se distorsione.

¿Cómo probar aplicaciones web?

Las pruebas de aplicaciones web es de gran importancia antes de entregarla a al usuario final o antes de subirla al lugar donde se alojará, ya que desde aquí se observa la calidad que va a tener la página cuando los usuarios inicien su navegación.

Dentro de las pruebas realizadas a las aplicaciones web antes de ser entregadas, deben analizar el contenido, la estructura, la facilidad de uso, el desempeño, la compactibilidad, la interoperabilidad y la seguridad.

No puede obviarse el más mínimo error que se presente ya que el público objetivo que son los usuarios externos darán buen o mal testimonio del acceso a la página y esto hará que continúen navegando o utilizando adecuadamente la web.

Errores dentro de un ambiente WebApp

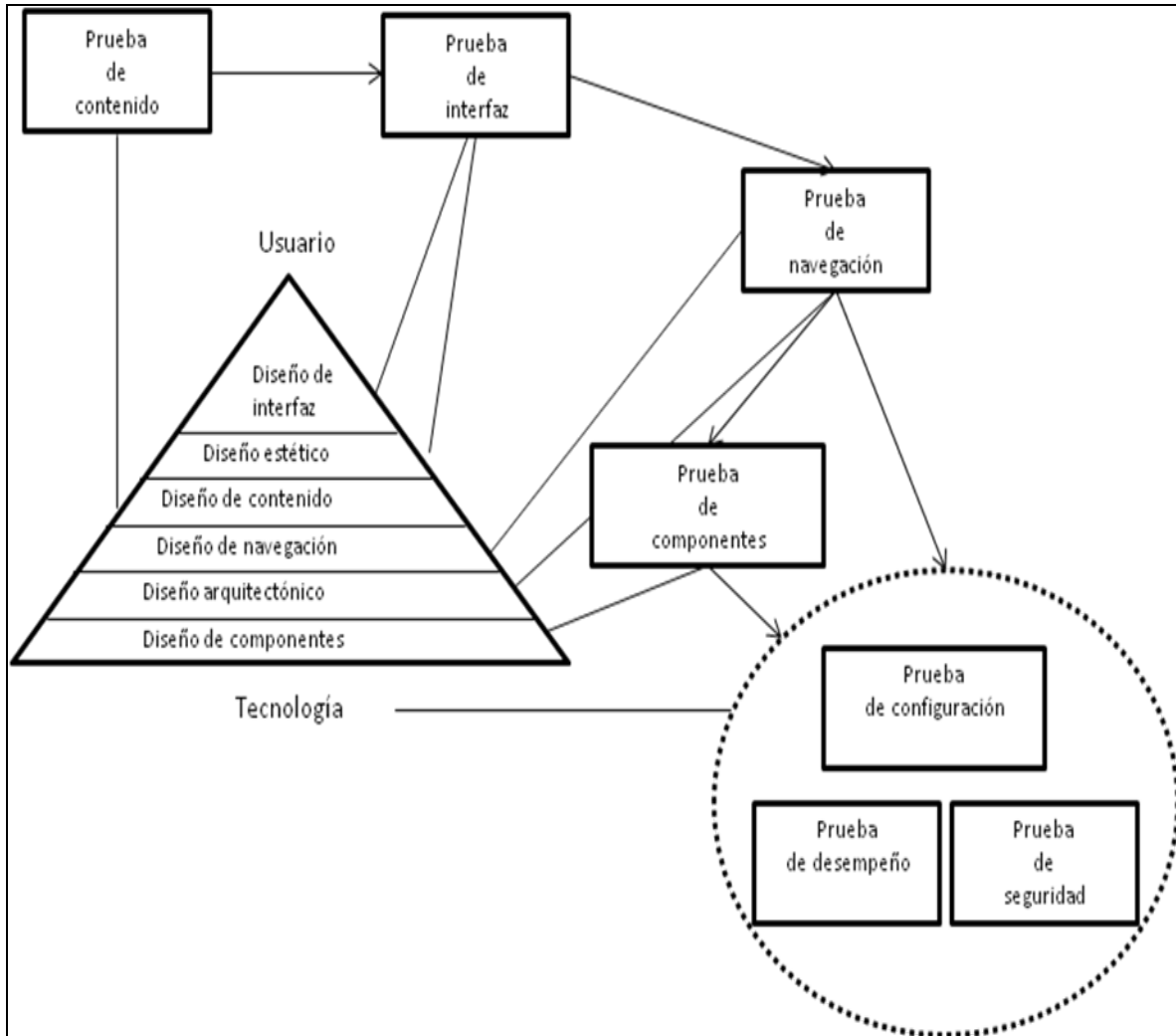
Se debe tener presente que aquellos mínimos errores que para el programador no son errores, para el usuario final es el más grave error que puede existir y por lo tanto no continuará con el proceso ya que no ofrece garantías en trabajo que está realizando.

El desarrollador de páginas web debe tener presente que no se sabe sobre cual plataforma y sistema operativo ejecutará la aplicación, por lo tanto es recomendable que realice aplicaciones genéricas que puedan ser ejecutadas en cualquier entorno y así poder prestar un buen servicio o comercializar un determinado producto para satisfacer a un público objetivo que requiere de algo haciendo uso de las tecnologías que ofrece nuestro medio.

El proceso de prueba: un panorama

Los procesos de prueba para ingeniería Web comienzan con pruebas que ejercitan el contenido y la funcionalidad de la interfaz que es inmediatamente visible para los usuarios finales.

Gráfica #21: El proceso de prueba de una aplicación WEB



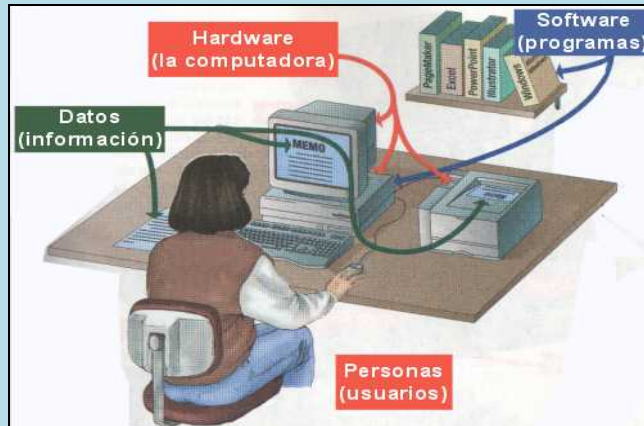
Esta es la manera de realizar las diferentes pruebas de las aplicaciones web.

- ◆ **La prueba de contenido** (y las revisiones) intentan descubrir en el contenido.
- ◆ **La prueba de interfaz** ejercita los mecanismos de interacción y valida los aspectos estéticos de la interfaz del usuario.
- ◆ **La prueba de componentes** ejercita el contenido y las unidades funcionales.

- ◆ **Las prueba de configuración** intentan descubrir los errores que son específicos respecto de un cliente o ambiente de servidor particulares.
- ◆ **La prueba de seguridad** incorpora una serie de pruebas diseñadas para explotar las vulnerabilidades en la WebApp y su ambiente.
- ◆ **La prueba de desempeño** abarca una serie de pruebas diseñadas para valorar 1) aumenta el tráfico de usuarios la respuesta en tiempo y confiabilidad, 2) que componentes son responsables de la degradación del desempeño y que características provocan la degradación, 3) como la degradación impacta en objetivos y en los requisitos.

Ejercicios tema Desarrollo Web

Realice los siguientes ejercicios:



1. Teniendo presente el anterior esquema, realice un análisis de la importancia que tienen los datos – información – conocimiento en un proceso de desarrollo de aplicaciones web. (Tenga presente que los usuarios cumplen un papel fundamental en este proceso y que el hardware y software hacen parte de un sistema de información)
2. Construya una pequeña aplicación web acerca de un sistema de inventarios. Tenga presente la organización de la interfaz.
3. Con palabras propias, exponga cómo se “analiza” la información recopilada durante la comunicación con el cliente y cuál es el resultado de esta Actividad.
4. Describa 3 diferencias entre el desarrollo de software convencional y Desarrollo web.
5. Elabore una lista de riesgos proactivos que puedan afectar un desarrollo de software de aplicaciones web.
6. Proponga un ejemplo del desarrollo de un producto XY, utilizando la ingeniería Web.

Prueba Final

1. Realice mediante un excelente diseño, un formulario de presentación de un ambiente web, teniendo claridad en los conceptos antes mencionados.
2. Plante un nuevo cronograma de actividades para presentarse a un cliente una solución web cuya razón social es la compra y venta de electrodomésticos en la avenida Colombia la cual está muy preocupada porque no tiene la información al día y desea tener la posibilidad de distribuir sus productos vía internet, el cual cuenta con los recursos necesarios tanto para la construcción de la aplicación como para implementación de la tecnología.
3. Construya una interfaz acerca de una aplicación web que proporcione facilidad de uso al usuario final o cliente.

5.2. Pistas de Aprendizaje

Tener en cuenta: Que la conceptualización es la base fundamental para desarrollar aplicaciones de alta calidad, aunque para muchos lo más importante es la práctica

Tener en cuenta: Que si el análisis de requisitos tiene diversas falencias, no dude que el desarrollo de la solución puede sufrir la misma problemática.

Tener en cuenta: Que los datos que serán utilizados en la elaboración del software, deben ser todos aquellos que le ofrezcan valor agregado a la empresa para que tome decisiones en el menor tiempo posible y con el más mínimo margen de error.

Tenga presente: Que los diferentes modelos de casos de uso que pueden ser utilizados en el desarrollo de software, son simplemente una manera de modelar una necesidad y de indicar como está fluyendo la información.

Tener en cuenta: Que en el desarrollo de proyectos no podemos creernos autosuficientes sino que necesitamos de otra persona para que contribuya positivamente en la construcción de la solución.

Tenga presente: Que el ciclo de vida del desarrollo de un proyecto no es solo por llenar papel sino que es el fundamento para alcanzar grandes objetivos durante la etapa de la ejecución.

Tener en cuenta: Que toda la información que se pueda recolectar durante la etapa de análisis de requisitos es fundamental para saber hacia dónde se va con la solución, dentro de los cuales tenemos principalmente los documentos comerciales que utiliza para empresa para el control de la información.

Traer a memoria: Que el estudio de la factibilidad es fundamental para hacer los respectivos requerimientos y saber con qué se cuenta durante todas las etapas del proyecto.

Traer a memoria: Que los diseños que se van a hacer para el software del cliente deben ser de acuerdo a lo que el cliente quiere porque al cliente hay que darle la razón y ese es el gusto de él.

Tenga presente: Que existen muchas páginas web en el medio que no satisfacen las necesidades del cliente y que por lo tanto usted debe de hacer aplicativos con calidad y que ayuden a la empresa a desarrollar mejor su actividad económica pero también tenga presente que busque siempre en que las dos personas ganen.

5.3. Glosario

BD: Base de datos, conjunto de campos, registros y tablas relacionados entre sí para cumplir con un objetivo.

Modelo: Esquema teórico de un comportamiento.

Modelo relacional: Base de datos

Diagrama relacional: Conjunto de tablas conectadas por una clave primaria.

Clave primaria: identificador principal de una tabla.

Clave foránea: Identificador secundario de una tabla o entidad.

Formulario: Pantallazos visuales que permiten ingresar información.

Diseño: Proyecto esquematizado.

Registro: Información bien organizada.

Arquitectura: Representación esquemática del comportamiento de un flujo de información.

Integrada: Respaldado, seguro, protegido.

Menú: Conjunto de actividades visuales ante los ojos del usuario final.

WebAPP: aplicación web

Tecnología: Conjunto de teorías y de técnicas que permiten el aprovechamiento práctico del conocimiento científico.

Herramienta: Instrumento útil para organizar procesos.

5.4. Bibliografía

Fuentes

Libros

Ph.D Roger S. Pressman, University of Connecticut, Ingeniería del Software un Enfoque Práctico, sexta edición, Mc Graw Hill, 2002, 2005.

Alfredo Weitzenfeld, Sur de California (Estados Unidos). Ingeniería del Software Orientado a Objetos con UML, Java e Internet, Thomson, 2004.