Fases del desarrollo de software

Análisis
Diseño
Construcción
Pruebas
Instalación
Mantenimiento

1

Análisis preliminar

- La solicitud para recibir ayuda de un sistema de información inicia con la petición de una persona (administrador, empleado, especialista…)
- Se inicia una "investigación preliminar" para poder "Aprobar la solicitud"

2

Investigación preliminar

- **Aclaración de la solicitud**
  - Formulación clara y precisa
  - Planteamiento claro
- **Estudio de Factibilidad**
  - **TECNICA**
    - ¿Se puede realizar con la tecnología actual?
    - Si no, ¿Se puede adquirir o desarrollar?
  - **ECONOMICA**
    - ¿Los beneficios superan los costos?
    - ¿Los costos de No-Desarrollo obligan a aceptar el proyecto?
  - **OPERACIONAL**
    - ¿Si se desarrolla el sistema, se usará?
    - ¿Habrá disminución de beneficios por la Resistencia a usarlo?

3

Análisis Detallado

- Una vez "Aprobado" el proyecto, se procede a reunir información detallada.

Se usan:

Técnicas de recolección de información

Para :

Determinar los requerimientos detallados

4

Técnicas de recolección de información

- **Entrevistas**
  - A usuarios (o futuros usuarios) del sistema.
  - Se CONVERSA, no se INTERROGA.
  - Obtener opiniones, políticas, descripciones subjetivas.
  - Pueden ser Estructuradas o No Estructuradas.
- **Cuestionarios**
  - Util para reunir información de un grupo grande de personas.
  - Se usan formatos estandarizados.
  - Anónimos

5

Técnicas de recolección de información

- **Revisión de registros**
  - Examinar la información asentada en registros, archivos y documentos pasados.
  - Manuales de políticas, reglamentos, procedimientos estándares, etc.
- **Observación**
  - Se observa en forma directa al usuario y al proceso actual, buscando datos específicos.

6

Determinación de los requerimientos

- Deben quedar perfectamente definidas las respuestas a las siguientes preguntas:
  - ¿Qué es lo que se hace?
  - ¿Cómo se hace?
  - ¿Con qué frecuencia se presenta?
  - ¿Qué tan grande es el volúmen de transacciones o decisiones?
  - ¿Con qué grado de eficiencia se realiza?
  - ¿Existe algún problema? ¿Es serio? ¿Qué lo causa?

7

Diseño

- Produce los detalles que establecen la forma en la que el sistema cumplirá con los requerimientos identificados en el análisis.

8

En el diseño se especifican...

- Datos de entrada
- Datos de salida
- Datos calculados
- Datos almacenados o a almacenar
- Procedimientos detallados de cálculo
- Estructuras de archivos
- Dispositivos de almacenamiento
- En el diseño se producen las especificaciones de software completas y perfectamente delineadas para la construcción del software.

9

Construcción

- Los programadores pasan las especificaciones del diseño a un lenguaje de programación.
- Puede usarse:
  - Enfoque Estructurado
  - Enfoque Orientado a Objetos
- Dependiendo de los requerimientos particulares y la plataforma.

10

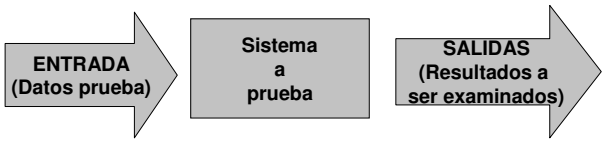
Construcción

- Los programadores también son responsables de:
  - Implementar los archivos y bases de datos
  - Documentar los programas
  - Explicar porqué los procedimientos se codifican de esa manera.

11

Pruebas

- Durante la fase de prueba el sistema se emplea de forma experimental para asegurarse de que el software funciona de acuerdo con las especificaciones.
- Varios usuarios pueden usar el sistema para observar si tratan de emplearlo en formas no previstas.



12

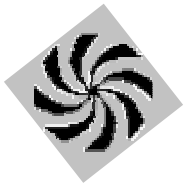
Tipos de pruebas

- **Prueba modular, unitaria o de Componentes.-** Se prueba cada módulo aislado del resto.
- **Prueba de integración.-** Se prueba la inter-relación entre los módulos.
- **Prueba del sistema.-** Comprueba que el sistema satisface los requisitos del usuario.
- **Prueba de aceptación.-** Se realiza ya que el sistema se implantó. Demuestra al usuario que el sistema satisface sus necesidades.
- **Prueba de regresión*.-** Comprueba que toda nueva versión de un software es de no menos calidad que la versión anterior.

13

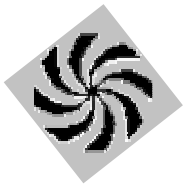
Reflexiones sobre las pruebas

- Un buen caso de prueba es aquel que tiene una probabilidad muy alta de descubrir un nuevo error.
- Una prueba tiene éxito si descubre un nuevo error.
- Una prueba NO asegura la ausencia de defectos. SOLAMENTE puede demostar que existen errores en el software.
- "El 80% de los errores está en el 20% de los módulos". Hay que identificar esos módulos y probarlos muy bien.



14

Reflexiones sobre las pruebas



- Las pruebas deben planificarse mucho antes de que empiecen.
- En una prueba es mejor empezar por lo pequeño y progresar hacia lo grande.
- Son mas efectivas las pruebas dirigidas por un equipo independiente.
- TIP: Si se definen casos de prueba con la mayor probabilidad de encontrar el mayor numero de errores, se invierte MENOS cantidad de esfuerzo y tiempo.
- Se recomienda diseñar casos prueba usando Pruebas de caja blanca y caja negra.

15

Cómo hacer las pruebas

1. Planear la prueba

(Objetivo, Características a probar, Métodos a utilizar, recursos, tiempos,responsabilidades).

2. Diseñar la prueba

(Instrucciones detalladas acerca de como llevar a cabo la prueba y los criterios para determinar si se pasa la prueba).

3. Determinar los casos de prueba

(Entradas y salidas esperadas)

16

Cómo hacer las pruebas [continuación]

4. Planificar el proceso de prueba
(Secuencia de ejecución de cada caso. Determinar cuando Termina cada caso)

5. Ejecución de la prueba / Registro de resultados.

6. Análisis y evaluación de la prueba.

17

Instalación

- La instalación es el proceso de:
  - Instalar nuevo equipo
  - Entrenar a los usuarios
  - Instalar la aplicación
  - Construir los archivos de datos necesarios para utilizarla.

18

Instalación

- Es importante contar con los siguientes recursos actualizados:
  - **Manual del usuario**
    - Indica cómo usar la aplicación
  - **Función de ayuda**
    - Integrada en el programa
    - Que ayude a resolver problemas cotidianos
    - Que sea práctica y entendible
    - Puede ser interactiva o en formato HTML

19

Mantenimiento

- Es efectuar “cambios” o ajustes” a la aplicación desarrollada, a los manuales o a la ayuda integrada.
- **NO** necesariamente son para corregir errores.
- Se puede distinguir entre:
  - Mantenimiento correctivo
  - Mantenimiento adaptativo
  - Mantenimiento perfectivo

20

Tipos de Mantenimiento

- Correctivo.-
  - Arregla algún error (BUG) encontrado en el sistema.
- Adaptativo.-
  - Modifica el software para que interaccione adecuadamente con su entorno cambiante.
- Perfectivo.-
  - Mejora la manera de realizar funciones ya implementadas en el sistema.

21

Ciclos de vida de desarrollo de software

En cascada
En espiral

22

Ciclo de Vida de desarrollo del Software

- Es la serie de etapas de un producto de software.
- Un modelo de ciclo de vida define el estado de estas etapas.
- Existen varios “modelos” de ciclo de vida. Algunos ejemplos son:
  - Cascada (*)
  - Entregas incrementales
  - Prototipos
  - Espiral (*)

Modelo
= Paradigma
= Esquema

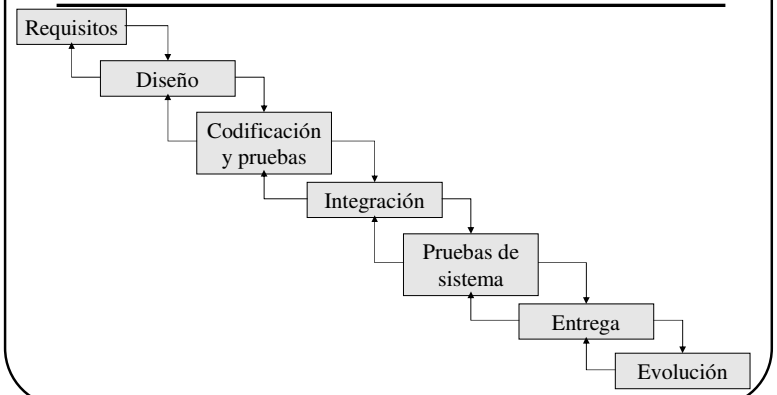
23

Un modelo de Ciclo de vida...

- Es una vista de las actividades que ocurren durante el desarrollo de software.
- Determina el orden de las etapas involucradas y los criterios de transición entre estas etapas.
- Describe las fases principales de desarrollo de software.
- Ayuda a administrar el progreso del desarrollo.
- Provee un espacio de trabajo para la definición de un detallado proceso de desarrollo de software.

24

Modelo en cascada



25

Modelo en cascada

- Sirve como base a los demás modelos
- Surge a finales de los 70.
- Las etapas se realizan linealmente
- Es rígido y restrictivo.

El desarrollo de software puede ser a través de una secuencia simple de fases. Cada fase tiene un conjunto de metas bien definidas. Las actividades dentro de una fase contribuyen a la satisfacción de metas de esa fase.

26

Principios del Modelo en Cascada

- Planear un proyecto antes de iniciarlo.
- Definir el comportamiento externo deseado del sistema antes de diseñar su arquitectura interna.
- Documentar los resultados de cada actividad.
- Diseñar un sistema antes de codificarlo.
- Probar un sistema después de construirlo.

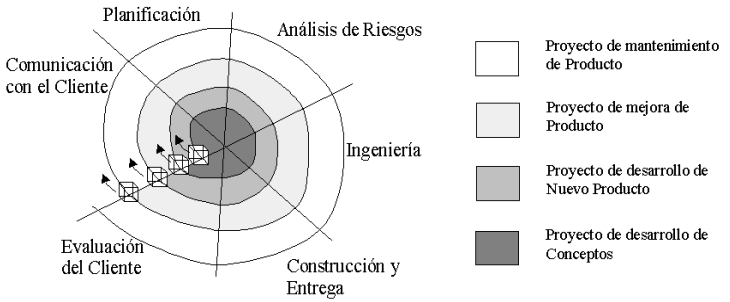
27

Modelo en espiral

- En este modelo, el esfuerzo de desarrollo es iterativo.
- Tan pronto como se completa un esfuerzo de desarrollo, otro comienza.
- Se construyen sucesivas versiones del software cada vez más complejas.
- Cada vuelta de la espiral resuelve un conjunto particular de problemas del cliente.
- Después de cada vuelta se realiza un prototipo.

28

Modelo en espiral



29

Principios del modelo en espiral

- Decidir qué problema se quiere resolver antes de viajar a resolverlo.
- Examinar las múltiples alternativas de acción y elegir una de las más convenientes.
- Evaluar qué se tiene hecho y qué se ha aprendido después de hacer algo.
- Conocer (comprender) los niveles de riesgo que habrá que tolerar.

30

Comparación de Modelos en cascada - Espiral

	Cascada	Espiral
Ha sido usado…	…Mucho	…No tanto
Principal desventaja	Proyectos reales no siempre se ajustan a él.	Difícil convencer al cliente que es controlable. Se requiere analizar riesgos.
Principal Ventaja	Usa sentido común. Es la pauta para los demás modelos.	Se usan prototipos en cada etapa. Añade flexibilidad al modelo de cascada
Requerimientos	Se conocen todos y claramente desde el principio	Se pueden ir adaptando en cada versión.
Primera version del producto	Tarda mucho	Es mas rápida

31

Metodologías para el desarrollo de sistemas

Enfoque funcional
Análisis y Diseño estructurado
Enfoque de datos
Modelo E-R
Jackson System Development
Metodología Warnier - Orr
Enfoque Orientado a Objetos
AOO y DOO
Herramientas Case

32

Análisis Estructurado

- Es un método para el análisis de sistemas manuales o automatizados que conduce al desarrollo de especificaciones para sistemas nuevos o para efectuar modificaciones a los ya existentes.
- Herramientas del Análisis y Diseño Estructurado:
  - Diagrama de flujo de datos
  - Diccionario de datos
  - Especificaciones de Proceso
  - Diagrama de Estructura

33

Diagrama de flujo de datos (DFD)

- Es una herramienta gráfica para describir y analizar el movimiento de datos a través de un sistema, incluyendo procesos, lugares para almacenar datos y retrasos.

	Notación de Yourdon	Notación Gane y Salson
Flujo de datos		
Procesos		
Fuente o destino de datos		
Almacenamiento de datos		

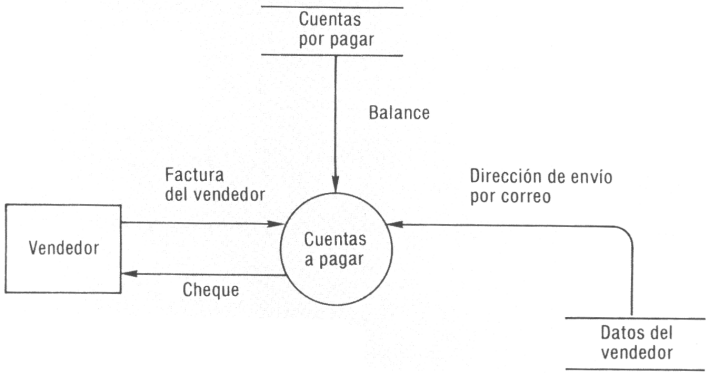
34

Diagrama de Flujo de Datos

- Cada componente tiene un nombre descriptivo.
- Los nombres de los procesos tienen un número para identificación.
- Se realizan varios diagramas con distintos niveles de detalle: de lo general a lo particular.
- El DFD de primer nivel consta solo de un proceso. Representa al sistema en general. Se llama Diagrama de contexto.

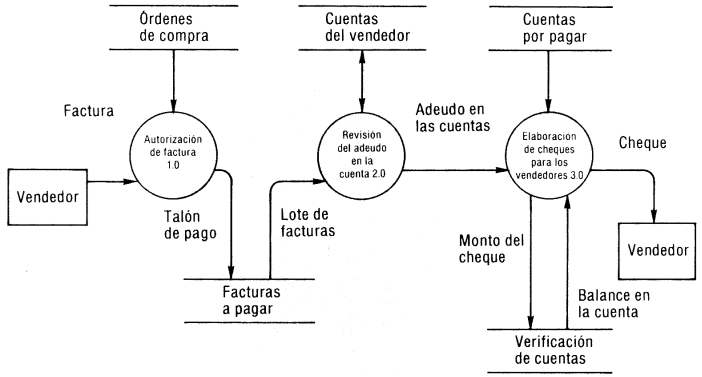
35

Ejemplo: DFD de contexto Sistema de Cuentas por Pagar

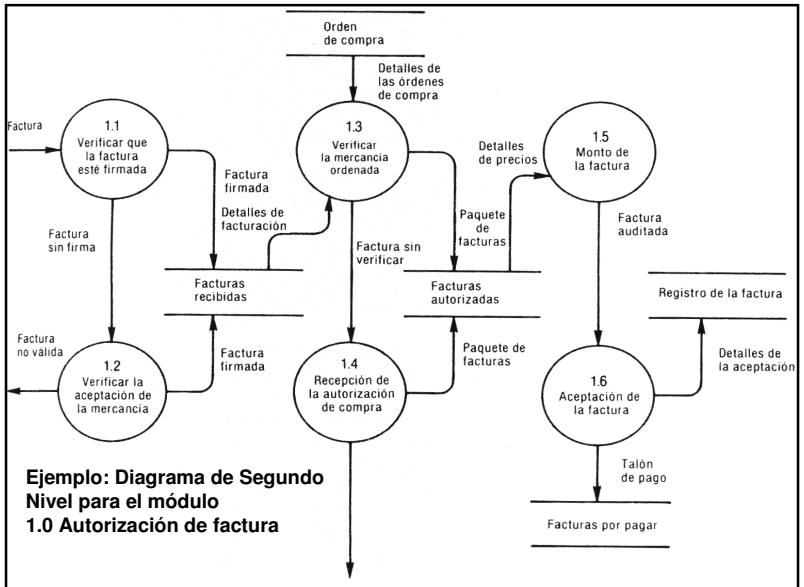


36

Ejemplo: DFD primer nivel Sistema de cuentas por pagar



37



Ejemplo: Diagrama de Segundo Nivel para el módulo 1.0 Autorización de factura

Diccionario de Datos (DD)

- Es un catálogo de los elementos de un sistema.
- En un DD se encuentra la lista de todos los elementos que forman parte del flujo de datos en todo el sistema.
- El DD guarda los detalles y descripciones de estos elementos.

Notación	Significado
=	Está compuesto de
+	Y
( )	Opcional
{ }	Repetición de un componente
[ ]	Una u otra alternativa
**	Comentario

39

Ejemplo del Diccionario de Datos

Detalles de los Articulos = Numero Articulo
+ descripción del artículo
+ costo del artículo
+ número de artículos

Monto de la Factura = {Costo de los artículos}
+ costo de envio
(+ impuestos de venta)

* Esta es una linea de comentarios

40

Especificaciones de proceso (MiniEspecificaciones)

- Es la descripción de lo que sucede en cada "proceso" en el nivel mas bajo de un Diagrama de Flujo de Datos.
- La forma mas utilizada de hacerlo es el lenguaje español estructurado.
- Una frase puede ser una expresión algebraica o una frase imperativa con verbo y sujeto.
- Se recomienda usar pocos verbos:
  - **Pedir** (conseguir, leer, capturar)
  - **Desplegar** (Mostrar, escribir, poner)
  - **Encontrar** (Buscar, localizar)

41

Especificaciones de proceso (MiniEspecificaciones)

- Se utilizan bloques de construcción de programación estructurada, y sentencias SQL, etc.
- Ejemplo:

```
SI Factura está vencida ENTONCES:
   Iniciar proceso de Pago
SI NO,
   Indicar Pago posterior
FIN-SI
```

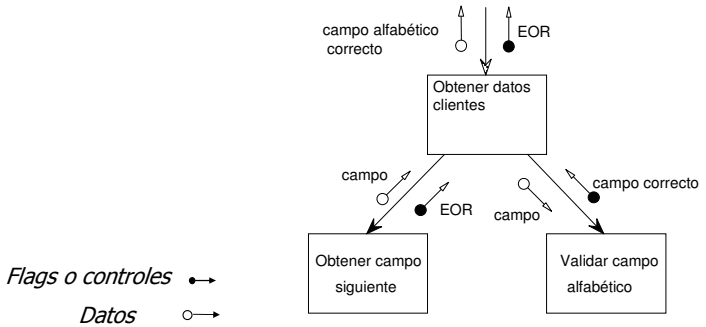
42

Diagrama de Estructura

- Herramienta gráfica utilizada para representar la jerarquía de software.
- Cada rectángulo representa un módulo.
- Las flechas que conectan los rectángulos representan llamadas de módulos.
- El diagrama también muestra parámetros de entrada que se le dan a cada módulo invocado y parámetros de salida devueltos por cada módulo cuando termina su tarea y devuelve el control al que lo llama.

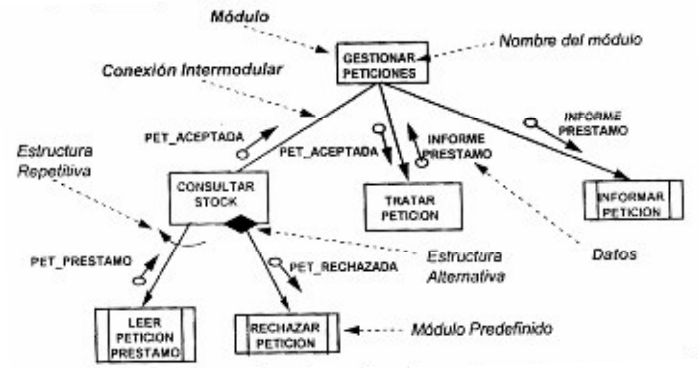
43

Ejemplo de Diagrama de Estructura



44

Otro Ejemplo de Diagrama de Estructura



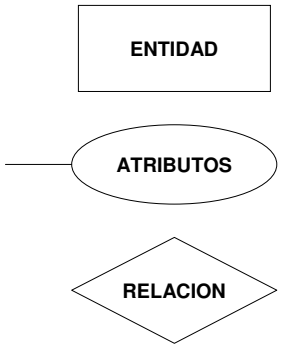
45

Modelo Entidad - Relación

- "Modelo de datos basado en una percepción mundo real que consiste en un conjunto de objetos básicos llamados Entidades y Relaciones entre estos objetos".
- El modelo E-R, tiene su implementación gráfica en el Diagrama Entidad-Relación.

46

Elementos de un Diagrama E-R



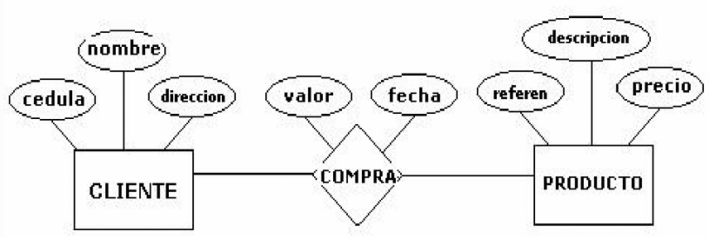
Rectángulo que representa una entidad

Elipse que representa atributos de cada entidad

Rombo que representa conjunto de relaciones

47

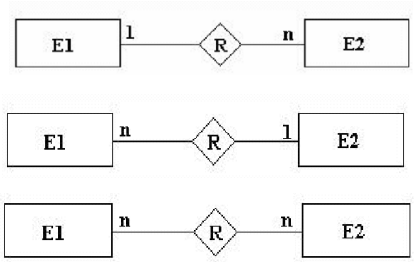
Ejemplo de Diagrama E-R



Los nombres de las entidades se escriben en Mayúsculas y Singular
Los nombres de los atributos en minúsculas y en singular.
Los atributos identificadores se marcan con #
Los atributos obligatorios se marcan con * y los opcionales con o

48

Cardinalidad de las relaciones



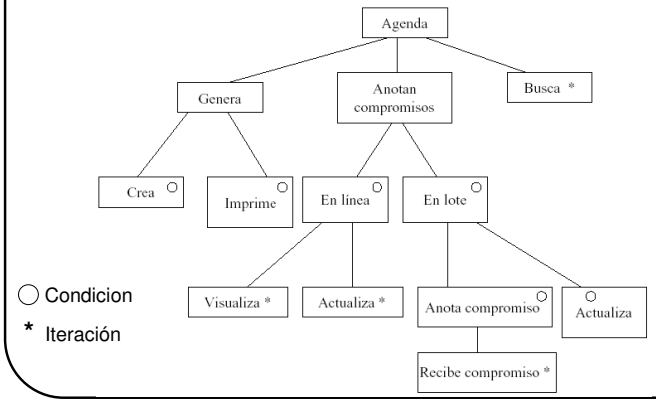
49

Jackson System Development

- Se recomienda para Sistemas basados en eventos.
- Paso Entidad –Acción
  - Se identifican las entidades del mundo real y las acciones que suceden en éste. Ejemplo: Para una agenda de compromisos: entidades serían: secretaria y agenda de compromisos.
- Paso Estructura de Entidad
  - Se utilizan los diagramas de estructura de Jackson para ordenar en el tiempo las acciones que afectan a cada entidad. (Ver diagrama)
- Paso del Modelo Inicial
  - Se define un modelo del mundo real del sistema mediante los diagramas de especificación del sistema (Ver diagrama).

50

Diagrama de estructura Jackson



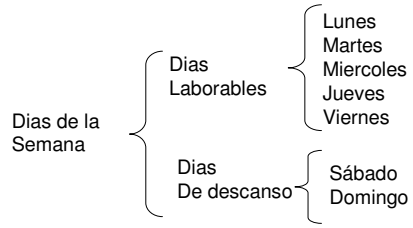
51

Ejemplo Diagrama de especificación del sistema (Jackson)

52

Metodología de Warnier-Orr

- Este diagrama se utiliza para representar gráficamente la estructura jerárquica de un sistema, programa o estructura de datos.
- Utiliza llaves y subconjuntos.



53

Enfoque Orientado a Objetos

- Se basa en el "Modelo de Objetos", el cual describe la estructura de los objetos de un sistema: su identidad, sus relaciones con otros objetos, sus atributos y sus operaciones.
- El Enfoque Orientado a Objetos es el mas adecuado para la mayoría de las aplicaciones.
- Este enfoque implica un modo distinto de pensar acerca de la descomposición, teniendo a los objetos como elemento principal.

54

Elementos del Enfoque Orientado a Objetos

- Objetos
- Clases
- Métodos
- Polimorfismo
- Encapsulamiento
- Herencia
- Mensajes
- Identidad
- Reusabilidad

La característica principal de los Objetos es que contienen datos [Propiedades] Y acciones [Métodos] encapsulados

55

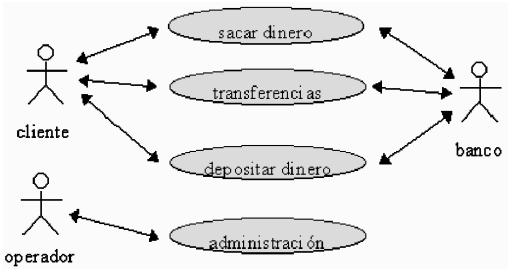
Técnica de Modelado de Objetos

- **Conceptualización**
  - Lista inicial de requisitos. Diagramas de casos de uso
- **Análisis (AOO)**
  - Modelo con elementos del *dominio del problema*
- **Diseño del sistema (DOO)**
  - Decisiones de alto nivel acerca de la arquitectura del sistema.
- **Diseño de objetos**
  - Basado en el análisis pero con detalles de implementación
- **Implementación**
  - En un lenguaje de Programación OO.

56

Diagramas de casos de uso

- Sirven para describir los requisitos iniciales del usuario. Están formados por interacciones entre el sistema y actores.



57

Modelo de Objetos

- Describe la estructura estática de los objetos y sus relaciones. Utiliza diagramas de objetos.

CLASE:

Persona
nombre: Cadena fechanac: Fecha
calcular_edad: Entero

Nombre de la clase

Atributos (Propiedades)

Métodos (Acciones)

OBJETO:

(Persona)

Juan, 23/12/65

58

Ejemplo: Modelo de Objetos

Persona
nombre: Cadena fechanac: Fecha
calcular_edad: Entero matricularse

estudia

Asignatura
nombre: Cadena
imprimir_lista

Indica una relación 1 a muchas

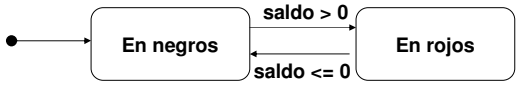
59

Modelo dinámico

- Utiliza diagramas de estados (DE) para modelar el comportamiento común de las clases.
- **ESTADO.-** Valores de los atributos y enlaces que tiene un objeto en un momento determinado.
- La interacción entre objetos se realiza mediante eventos.
- El Modelo dinámico es un conjunto de DEs, uno para cada clase objetos que tenga un comportamiento dinámico no trivial.

60

Ejemplo de Diagrama de Estados (Modelo dinámico)



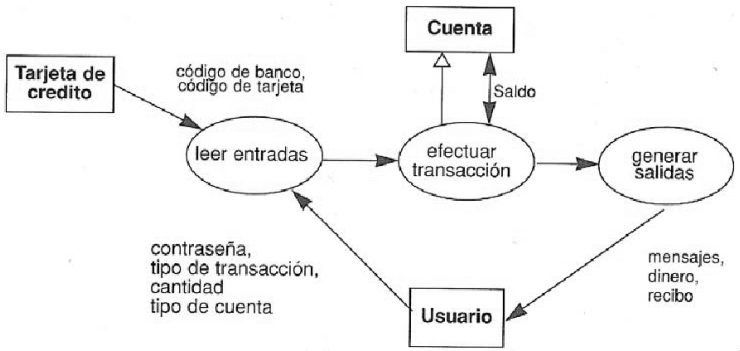
61

Modelo Funcional

- Describe los cálculos que se realizan en un sistema, mostrando como se obtienen los valores de salida a partir de los de entrada.
- Muestra cómo se realizan las operaciones del modelo de datos y las actividades y acciones de los Diagramas de Estados.

62

Ejemplo: Modelo Funcional de alto nivel



63

Diseño Orientado a Objetos

- Se decide la estructura y el diseño global.
- Se realizan decisiones acerca de la arquitectura del sistema.
- Se incrementa la modularidad.
- Se refinan clases y objetos.

64

Herramientas CASE

- Automatizan los aspectos clave del desarrollo de un sistema.
- Las herramientas CASE combinan:
  - Herramientas de automatización del desarrollo de software
  - Metodologías que definen los procesos a automatizar.

CASE
=
Computer Aided Software Engineering
=
Ingeniería de software asistida por computadora

65

Clasificación de las Herramientas CASE según las fases que automatizan

- UPPER CASE
  - Planificación estratégica
  - Requerimientos de desarrollo funcional de planes corporativos.
- MIDDLE CASE
  - Análisis
  - Diseño
- LOWER CASE
  - Generación de código
  - Prueba
  - Implementación

66

Componentes de las Herramientas CASE

- Repositorio (Enciclopedia)
  - Base de datos que contiene toda la información relacionada con la especificación, análisis y diseño del software.
  - Incluye Datos, Procesos, Gráficos, diagramas, reglas, etc.
- Módulos para:
  - Documentación
  - Generación de código
  - Chequeo de errores
  - Gestión del proyecto
  - Control de versiones

67

Administración de Proyectos

Estimación del tiempo y costo de cada actividad.

Planificación temporal de proyectos [Gráficas de GANTT].

68

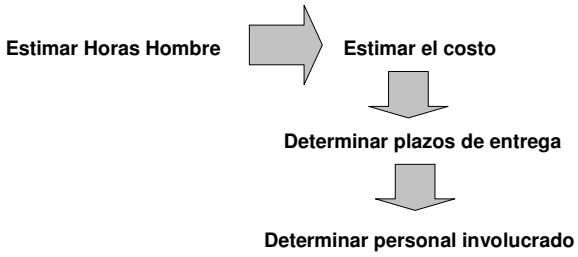
Aspectos a considerar en la estimación de tiempos

- Disponibilidad de recursos
- Complejidad
- Tamaño del proyecto
- Grado de estructuración del proyecto
- Disponibilidad de información histórica
- Considerar escenarios
  - "Mejor caso" y "Peor caso"

69

Estimación de tiempo y costo de cada actividad

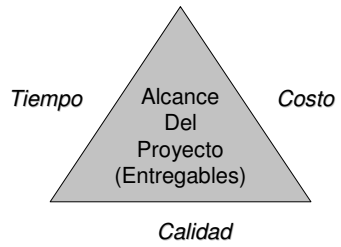
- Costo estimado de un proyecto (A partir de horas hombre)



70

Balance de los elementos básicos de los proyectos

- Es común que los proyectos se planteen siempre con las mismas características: Urgentes, de poco presupuesto y Alta Calidad. Se recomienda negociar y no crear expectativas falsas.
- Hay que tener en mente siempre el siguiente esquema:



71

Gráficas de Gantt

- Gráfica de barras usada para representar tareas o actividades.
- Muestra la ocurrencia de actividades en paralelo o en serie en un determinado periodo de tiempo.
- Cada barra = Una tarea
- Eje horizontal = tiempo (fechas)
- Eje Vertical = (Tareas)

72

Ejemplo Gráfica de Gantt

GRAFICA
Presentación del TIEMPO en el cronograma de actividades

Tiempo / Actividades	FEB.	MAR.	ABR.	MAY.	JUN.
Actividad 1					
Actividad 2					
Actividad 3					
Actividad 4					
Actividad 5					
Actividad 6					
Actividad 7					
Actividad 8					
Actividad 9					
Actividad 10					

73

Aseguramiento de la calidad

Concepto de calidad
Estándares
Especificaciones
Factores de calidad
Métricas de calidad
Revisiones técnicas

74

Concepto de calidad

- "Capacidad del producto software para satisfacer los requisitos establecidos"
- "Suma de todos aquellos aspectos o caracteristicas de un producto o servicio que influyen en su capacidad para satisfacer las necesidades expresadas o implícitas" [ISO 8402]
- "Grado con el cual el cliente o usuario percibe que el software satisface sus expectativas" [IEEE 729-83]

75

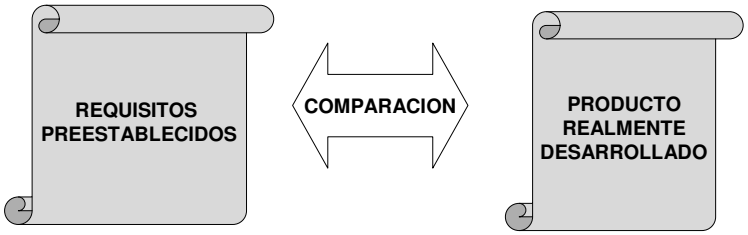
INVOLUCRA TODOS SUS ESTADOS DE EVOLUCION

CALIDAD DEL SOFTWARE

- Análisis
- Diseño
- Código
- …. etc….

76

CALIDAD DEPENDE DE Requisitos que se desea satisfacer



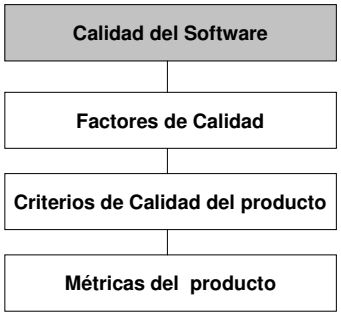
77

Estándares y especificaciones

- Los estándares son convencionalismos que determinan especificaciones mínimas para la calidad de un producto.
- Existen estándares internacionales, que sirven como guía en el proceso de calidad. (Ejemplo ISO)
- Un estándar contiene también procedimientos, etapas, documentación, etc.

78

Definición de Calidad con un Modelo



Punto de Vista del Usuario (Atributos de calidad Externos)

Punto de Vista del Software (Atributos de calidad Internos)

Medidas cuantitativas que indican el grado de calidad que posee un atributo.

79

Modelo de McCall

CORRECCION (¿Hace el software lo que yo quiero?)
FIABILIDAD (¿Lo hace de forma exacta todo el tiempo?)
EFICIENCIA (¿Se ejecutará sobre mi hardware lo mejor posible?)
INTEGRIDAD (¿Es seguro?)
FACILIDAD DE USO (¿Puedo ejecutarlo?)

Modelo de McCall

Punto de vista	Factores
Operación del producto	•Facilidad de uso
	•Integridad
	•Corrección
	•Fiabilidad
	•Eficiencia
Revisión del producto	•Facilidad de Mantenimiento
	•Facilidad de prueba
	•Flexibilidad
Transición del producto	•Facilidad de reutilización
	•Interoperabilidad
	•Portabilidad

81

Ejemplos de Métricas de calidad

Fiabilidad = 1 – (Num de errores / Num lineas de codigo)

Facilidad de Mantenimiento = 1 – 0.1 * (Numero medio de diasHombre por Corrección)

Portabilidad = 1 – (Esfuerzo para portar / esfuerzo para implementar)

Flexibilidad = 1 – 0.05 * (Num medio de diasHombre por cambio)

82

Revisiones Técnicas

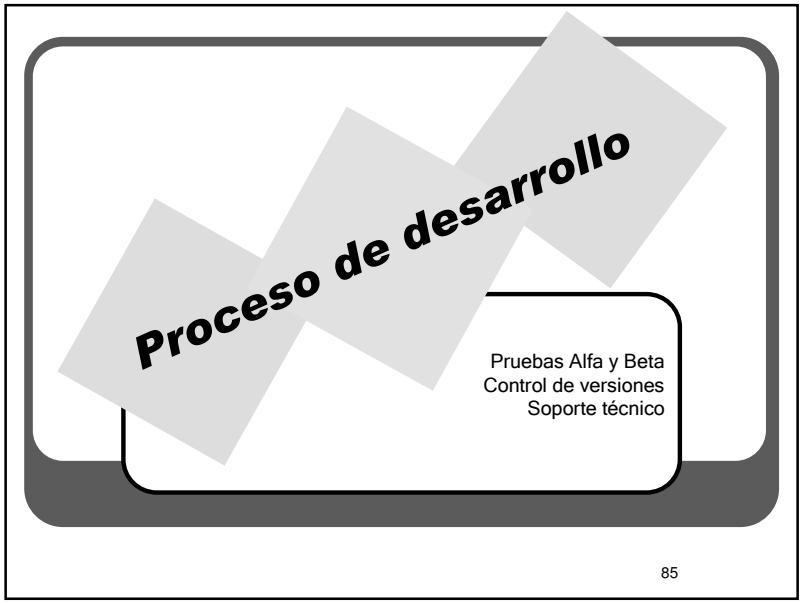
- Son Reuniones formales en las que se presenta el estado actual de los resultados de un proyecto a un usuario, cliente u otro tipo de persona interesada, y se realiza un análisis de los mismos.
- La evaluación técnica no recae sobre las mismas personas involucradas en la producción del software.

83

Tipos de revisiones

- Tipos de Revisiones técnicas
  - Revisión de la especificación de los requisitos.
  - Revisión del diseño.
  - Revisión del código.
  - Revisión de las pruebas.
  - Revisión del manual de usuario.
- Objetivos de las Revisiones de gestión
  - Control de la progresión del proyecto
  - Evaluación de los riesgos asociados al proyecto
  - Evaluación general del producto

84



Pruebas Alfa y Beta

Un programa antes de ser liberado pasa por pruebas que evalúan su factibilidad, funcionamiento y facilidad de uso.

- **PRUEBA ALFA.-**
  - Sirven para probar una versión aún no-fiable de un programa,
  - El programa está disponible a usuarios muy seleccionados.
  - Implica una prueba "local" que no está al alcance de usuarios comunes.
- **PRUEBA BETA.-**
  - Los desarrolladores finalizaron el programa pero buscan ayuda de los usuarios para encontrar fallas.
  - Son pruebas de un programa antes de su lanzamiento comercial.
  - Se realizan después de las ALFA.

86

Control de versiones

- Se requiere una administración formal de los cambios en el software para llevar un orden y seguimiento adecuado.
- El número de la versión es un indicador de la vigencia del programa.
- Muchos sistemas automatizan el control de sus versiones y checan si hay una versión mas nueva y disponible antes de iniciar.

87

Control de Versiones

Ejemplo: Version 3.1.1.

Versión principal

Actualización general. Agrega detalles nuevos

Parche que corrige fallos

88

Soporte Técnico

- Es importante que después de que un producto sea liberado, se proporcione soporte a los usuarios.
- El soporte brinda apoyo acerca de:
  - Problemas que se presenten en el uso del software
  - Situaciones no consideradas en la documentación
  - Procedimientos correctos para realizar tareas con el software.

89