







# Resumen Primer Parcial

 Materia	 <u>Ingenieria de Software 3</u>
 Fecha	@09/04/2025
 Tema	Resumen

---

## Unidad 1 - Introducción a la Ing de Software

### Que es la ingenieria de software ?

Establecimiento y uso de metodos, herramientas y procedimientos, orientados a obtener software económico que sea fiable y funcione de manera eficiente. Es una disciplina en la ingenieria que concierte a todos los aspectos de la producción de software.

La ingeniería de software concierne solo al desarrollo de sistemas o productos de software.

## Diferencias Ing Software vs Ing Sistemas

Ingenieris de Sistemas: Tiene en cuenta todos los elementos del desarrollo de sistemas basados en cómputo, que incluyen hardware, software, bases de datos, recursos humanos.

Ingeniería de Software: Forma parte del proceso de lo mencionado arriba.

## Que es el software ?

Según **sommerville** : Programas de computo y su documentación asociada

Segun **pressman** : Elemento del sistema de información que es logico, con características considerablemente distintas a las del hardware

## Productos de software

**Genericos:** Producidos por una organización para ser comercializados

**A medida:** Desarrollados bajo pedido a un desarrollador

### Características productos

- Mantenibles: Debe evolucionar
- Confiabilidad: No debe causar daños físicos o económicos en caso de fallo
- Eficiencia: No debe desperdiciar recursos
- Utilización adecuada: Debe contar con una interfaz y documentación adecuada

### Clasificación de productos

- Estructura
  - Funcionales
  - Orientados a objetos, listas o componentes
- Función
  - Programas
  - Librerías
  - Bases de datos

- Web
- Plataforma de computo
  - Embebidos
  - Computo distribuido
  - Sist. de tiempo real

## Que es el Proceso de software ?

Conjunto estructurado de actividades requeridas para desarrollar un sistema de software

- Especificación
- Desarrollo
- Validación
- Evolución

### Proceso Generico

- Especificación
- Diseño
- Codificación
- Prueba
- Instalación
- Mantenimiento

## Modelo de desarrollo de software

Representación formal o simplificada de proceso de software

### Modelos Genericos

- Modelo de Cascada: Separa en distintas fases la especificación y el desarrollo
- Desarrollo evolutivo: La especificación y el desarrollo estan intercalados
- Prototipado: Un modelo sirve de prototipo para la construccion final

- Transformación formal: Modelo matemático del sistema se transforma en implementación
- Basado en reutilización: Ensamblado a partir de componentes existentes

## **Modelo en cascada**

La dificultad de este modelo reside en la dificultad de hacer cambios entre etapas

Problemas: Poca visibilidad en el proceso, Probablemente especificados, Se requiere habilidades especiales

Aplicabilidad: Sistemas pequeños, Módulos de sist grandes, sist de corta vida

## **Modelo prototipado**

**Evolutivo:** Objetivo es trabajar con clientes hasta evolucionar a un sistema final, a partir de una especificación inicial.

**Desechable:** Objetivo es entender los requerimientos del sistema.

## **Modelo evolutivo**

Alto riesgo debido a la necesidad de tecnología avanzada y habilidades del grupo desarrollador

## **Fases modelo en espiral**

- Planteamiento de objetivos
- Identificación y reducción de riesgos
- Desarrollo y validación
- Planeación

Ventajas: Reutilización de componentes, objetivos de calidad, desarrollo con mantenimiento

Problemas: Requiere de experiencia en la identificación de riesgos, requiere refinamiento para uso generalizado

## **Como elegir un modelo ?**

Para sistemas bien comprendidos utilizar modelo cascada.

Con requerimientos estables se utiliza modelos formales

Con especificaciones incompletas se usa el modelo de prototipado

## Explica los Retos de la ing de software

**Legados:** Mantener y tratar con sistemas legagos (viejos) → Explicar cualquier cosa

**Heterogeneidad:** Sistemas que incluyen distintas plataformas de software y hardware

**Entrega:** Existe una presión incremental por una entrega a tiempo de los productos de software

**Formalidad:** Existe una gran demanda de que exista formalidad en el proceso de desarrollo de software

## Unidad 2 - Sistemas sociotecnicos

### Que es un sistema ?

Colección de componentes interrelacionados que trabajan conjuntamente para alcanzar objetivos comunes. Puede incluir software, hardware mecanico, etc.

### Categorias

**Sist. tecnicos basados en computadoras:** No incluyen hardware y software, los operadoras y los procesos operacionales no son considerados como parte del sistema. El sist no es consciente de su finalidad

**Sist socio-tecnicos:** Sistemas que incluyen sistemas tecnicos pero tambien procesos operacionales y gente que utiliza e interactua con el sistema tecnico. Gobernados por politicas y reglas organizacionales

### Caracteristicas de sistemas sociotecnicos

- Propiedades emergentes: Propiedades del sistemas como un todo que dependen de los componentes del sistema y sus relaciones. Consecuencia de las relaciones entre los componentes del sistema.

- Volumen
- Confiabilidad
- Protección
- Reparabilidad
- Usabilidad

**Prop. Emergentes Funcionales:** Aparecen cuando todas las partes de un sistema trabajan juntas para alcanzar el mismo objetivo. Ej: Una bicicleta para transporte

**Prop. Emergentes NO Funcionales:** Se relacionan con el comportamiento del sistema en su ambiente operacional. Son criticos para sistemas basados en computadoras.

- No determinísticos: No siempre producen la misma salida con la misma entrada
- Relaciones complejas con objetivos organizacionales: El sistema apoya con los objetivos organizacionales.

## Explica la ingeniería de sistemas ?

Especificar, diseñar, implementar, validar y mantener sistemas socio-técnicos.

Considerar los servicios que provee el sistema, limitaciones en su construcción y operación y las maneras en las cuales es utilizado.

## Definición de requerimientos de sistema

- Requerimientos funcionales abstractos
- Propiedades del sistema
- Características que no debe mostrar el sistema

## Objetivos del sistema

- **Objetivos funcionales:** Ej: Proveer un sistema de alarma contra fuego e intrusos
- **Objetivos organizacionales:** Ej: Asegurar el normal funcionamiento del trabajo llevado en el edificio

## Proceso de diseño de sistema

- Dividir requerimientos
- Identificar subsistemas
- Asignar requerimientos a los subsistemas
- Especificar funcionalidades de subsistemas
- Definir las interfaces del sub sistema

## Procesos organizacionales

Son aquellos involucrados en utilizar el sistema para el uso que fue definido.

Deberían ser diseñados para ser flexibles y no deberían forzar operaciones que se realizan de una manera en particular.

## Sistemas heredados

Los sistemas heredados es un sistema viejo que continua proveyendo servicios esenciales. Incluyen procesos de negocios, software de aplicación, software de soporte y hardware del sistema

## Unidad 3 - Pruebas de software

### Que son las Pruebas de software ? (TAA) - Que muestran las pruebas ?

Probar es el proceso de ejercitar un sistemas con el objetivo especifico de encontrar errores antes de entregarlo al usuario final

### Características de la facilidad de prueba

**Operabilidad:** Opera limpiamente. (Funciona bien el sistema)

**Observabilidad:** Los resultados de cada prueba son observados en todo momento

**Controlabilidad:** Si se puede controlar el software mejor

**Capacidad de descomposición:** Pruebas realizadas de manera modular

**Simplicidad:** Reducir arquitecturas complejas y logicas

# Proceso de prueba de software

## Pruebas de componentes

Pruebas de programas individuales componentes. Las pruebas se derivan de la experiencia del desarrollador.

## Pruebas del sistema

Pruebas de grupos de componentes integrados para crear un sistema o un sub-sistema

Las pruebas se basan en una especificación del sistema

## Quien prueba el software ?

Desarrollador de software → Pruebas de componentes

Equipo de pruebas independiente → Pruebas de sistema

## Objetivos del proceso de pruebas (TAA)

**Validación:** Demostrar al desarrollador y al cliente del sistema que el software cumpla sus requerimientos. Una prueba exitosa muestra que el sistema opera de la manera en que debe.

**Defectos:** Descubrir fallas o defectos en el software donde su comportamiento es incorrecto. Una prueba exitosa es aquella que hace que el sistema se comporte de manera incorrecta

## Tipos de pruebas

Involucra la integración de componentes para crear un sistema

Es un incremento a ser entregado al cliente

Cuenta con dos fases:

**Pruebas de integración:** El equipo de pruebas tiene acceso al código fuente. Se comprueba como están integrados los componentes. Involucra construir un sistema desde sus componentes y probarlo para problemas que surgen de la interacción de los componentes.

**Pruebas de entrega:** El equipo de pruebas chequea el sistema completo a ser entregado como una caja negra. El proceso de probar una versión de un sistema que será entregada a los clientes.



**Pruebas de rendimiento:** (TAA) Incluyen la planificación de una serie de pruebas donde la carga se incrementa establemente hasta que el rendimiento del sistema se torna inaceptable

**Pruebas de estrés:** Ejercitan al sistema mas allá de su máxima carga diseñada. Son particularmente relevantes para sistemas distribuidos.

**Pruebas de componentes:** Proceso de probar componentes individuales por separado. Es un proceso de pruebas por defecto

**Pruebas de clases de objeto:** Probar todas las operaciones asociadas a un objeto. Ejercitar el objeto en todos sus estados posibles.

**Pruebas de interfases:** Su objetivo es detectar fallas debido a errores de interfase o supuestos invalidos acerca de las interfases

**Pruebas de camino:** Asegurar que el grupo de casos de prueba es tal que cada camino en el programa se ejecuta por lo menos una vez.

## Guia para realizar entregas

- Elegir entradas que obliguen al sistema a generar todos los mensajes de error
- Diseñar entradas que causen sobrecarga al sistema
- Repetir la misma prueba o serie de prueba varias veces
- Obligar a que se generen salidas inválidas

## Diseño de casos de pruebas

- Pruebas basadas en requerimiento:  
Son una tecnica de pruebas de validación donde se considera cada requerimiento y se deriva un grupo de pruebas para ese requerimiento.
- Pruebas de partición  
Los datos de entrada y salida de resultados se pueden agrupar en diferentes clases donde todos los miembros de una clase estan relacionados  
Los casos de pueba se debe elegir para cada particion
- Pruebas estructurales o caja blanca:

Casos de prueba de acuerdo a la estructura del programa. El objetivo es ejercitar todos los comandos del programa

## Unidad 4 - Estimación de proyectos de software

### Que es una estimación de software ? (TAA)

Una estimación de software es una predicción de cuánto tiempo durara o costara su desarrollo y mantenimiento

Si es estimación de esfuerzo se habla de horas-hombre, si es una estimación de costo se habla de manera monetaria.

### Las estimaciones se utilizan para ?

- Desarrollar planes de proyectos
- Realizar analisis de inversion
- Elaborar presupuestos
- Fijacion de precios
- Elaborar planificaciones de iteración

### Pasos para crear una buena planificación

- Estimar el tamaño del producto (numero de lineas de codigo o puntos de funcion)
- Estimar el esfuerzo (persona-mes)
- Estimar la planificacion (meses)

### Estimar tamaño de un proyecto

- Utilizar un enfoque algoritmico
- Utilizar un software de estimación
- Haber trabajado en proyectos similares

### Que es un punto de función ? (TAA)

Es una medida sintética del tamaño del programa, se suele utilizar en los primeros estados del proyecto. Son mas faciles de determinar teniendo la cantidad de lineas de codigo.

**Se basan en:**

- Entradas: Pantallas, formularios, cuadros de dialogo, controles o mensajes.
- Salidas: Pantalla, informes, graficos.
- Consultas: Combinaciones de entrada/salida.
- Archivos logicos internos: Grupos logicos de datos de usuarios finales.
- Archivos de interfaz externos: Archivos controlados por otro programa.



LDC = Lineas de codigo

## COCOMO 2

Es un modelo de estimación de costo utilizable en la evaluación del esfuerzo estimado para un proyecto.

**Entradas**

Tamaño del programa en KDLC.

Atributos del producto: Ambiente en cual opera el programa

Atributos de la plataforma: Limitaciones que afecta el esfuerzo del desarrollo debido al hardware y software.

Atributos del personal: Habilidad del personal asignado al proyecto

Atributos del proyecto: Restricciones bajo las cuales opera el desarrollo del proyecto.

**Que es EMI ?**

Es el i-ésimo facor de ajuste

**Que es PMTotal (Esfuerzo total)**

Es el producto del esfuerzo nomila y el EAF

## Salidas

La salida es el nivel de esfuerzo en meses-hombre para el proyecto siendo estimado, y el tiempo en meses.

Se puede convertir a valor monetario si se tiene el costo mes-hombre.

La distribución por fases es una de sus salidas

## Que es PM nominal ? (TAA)

Es el esfuerzo nominal requerido en meses-hombre

$$PM_{nominal} = A \times size^B$$

**Size:** Tamaño estimado del software en miles de línea de código **KDLC**

## Valoración del factor escalar B

Se calcula apartir de la sumatoria de los aportes de distintas variables escalares.

$$B = 0.91 + 0.01 \times \sum (W_i)$$

## Que es PM ajustado (TAA)

Es el pructo del PMnominal por el multiplicador de esfuerzo.

Representa las características del proyecto y expresa el impacto en el desarrollo total del producto de software

## Duración en meses para completar un proyecto

$$Duración = 3,67 \cdot (PM_{ajustado})^{[0.28+0.2 \cdot (B-0.91)]}$$

## Estimar esfuerzo en COCOMO 2

EJEMPLO: UFP = 26 (UFP = Puntos de función sin ajustar)

**A:** como por defecto en **2.94**

**Size:** producto de los puntos de función sin ajustar por un factor de conversión, que depende del lenguaje Ej: **Java 53 LDC/UFP**

Size =  $53 \times 26 = 1378$  LDC

**B:** Se calcula ponderando las variables escalares

Legalmente despues ya es mucho texto hay que tener la tabla de Wi y demas...  
(no creo que salga)

## Unidad 5 - Planificación de proyectos de software

### Causas de los retrasos en proyectos

- Fecha limite poco realista
- Cambio de requisitos del cliente
- Dificultades tecnicas o humanas
- Falta de comunicación del equipo
- Falta de reconocimiento por parte de la gestion del proyecto

### Problemas con la presion excesiva en la planificacion

- Calidad: Los errores de codigo son causa de la tension.
- Azar: Los directivos y desarrolladores sienten la tentación de apostar al azar en vez de correr riesgos calculados
- Motivación: La planificación cruza el umbral de la credibilidad y la motivación decae rapido.
- Creatividad: La motivacion externa excesiva reduce la motivacion interna lo cual reduce la creatividad
- Cambio de personal: Tienden a causar cambio voluntario excesivo de personal
- Relacion entre desarrolladores y directivos: La presion en la planificación aumenta las diferencias entre desarrolladores y directivos

### Principios basicos de la planificacion temporal

- Compartimentacion

- Interdependencia
- Asignación de tiempo
- Validación de esfuerzo
- Responsabilidades, Resultados e Hitos Definidos

## **Agregar mas gente retrasa el desarrollo ? (TAA)**

Desgraciadamente, añadir gente tarde a un proyecto tiene a menudo un efecto negativo, provocando aún más retraso. El personal agregado debe aprenderse el sistema y la gente que les enseña es la misma que estaba trabajando. Mientras están enseñando no se trabaja, y el proyecto se retrasa todavía más.

### **Texto del ppt:**

Considere cuatro ingenieros del software, cada uno capaz de producir 5000 LDC/año cuando trabajan en proyectos individuales. Cuando se pone a estos cuatro ingenieros en un proyecto de equipo, son posibles seis vías de comunicación potenciales. Cada vía de comunicación requiere un tiempo que podría de otra manera emplearse en desarrollar software. Asumiremos que la productividad del equipo (medida en LDC) se verá reducida en 250 LDC/año por cada vía, debido al gasto indirecto asociado con la comunicación. Por tanto, la productividad del equipo es  $20.000 - (250 \times 6) = 18.500$  LDC/año, un 7,5 por ciento menos de lo que podríamos esperar.

### **Aca habla sobre agregar el integrante**

El proyecto de un año en el que está trabajando el equipo anterior se retrasa ya dos meses de la fecha de entrega se agregan dos personas adicionales al equipo. El número de vías de comunicación se dispara a 15.

### Aca calcula el performance con los nuevos

La productividad de la nueva plantilla es la equivalente a  $840 \times 2 = 1.680$  LDC para los dos

meses restantes antes de la entrega.

La productividad del equipo es ahora  $20.000 + 1.680 - (250 \times 15) = 17.930$  LDC/año

## Tipos de proyecto

- Proyectos de desarrollo del concepto: Inician para explorar un nuevo concepto de negocios
- Proyectos de desarrollo de una nueva aplicación: Se aceptan como encargo de un cliente
- Proyectos de mejoras de aplicaciones: Ocurre cuando un software sufre grandes modificaciones a su funcionamiento
- Proyectos de mantenimiento de aplicaciones: Corrigen/adaptan/amplian un software existente
- Proyectos de reingeniería: Reconstruir un sistema existente (heredado)

## Grados de rigor

- Casual: Aplican todas las actividades estructurales del proceso
- Estructurado: Se aplican la estructura del proceso, actividades estructurales y tareas relativas.
- Estricto: Se aplica el proceso completo para el proyecto con una disciplina que garantice el éxito. Se produce una robusta documentación

## Que es una Red de tareas o red de actividades?

Es una representación grafica del flujo de tareas de un proyecto.

## Tecnica de evaluación y revision de programa (PERT) y metodo de camino critico (CPM)

Son dos metodos de la planificación temporal de un proyecto. Ambas tecnologias son dirigidas por la informacion desarrollada en actividades anteriores de la planificacion del proyecto

## Que es EDT ?

Tarea denominadas estructura de descomposicion del trabajo

# Unidad VI - Desarrollo rapido del software

## Que es un proceso de software ?

Conjunto relacionado de actividades y tareas implicadas en el desarrollo y evolucion de un sistema de software

## Modelos tradicionales

- Cascada: Debe completarse un estado antes de empezar otro. Util para visualizar lo que va a hacer. No refleja la realidad
- Prototipacion: Permite construccion rapida del sistema. Se tiene una vision comun dev-user. Reduce riesgo del desarrollo
- Desarrollo en Fases: Se desarrolla y entrega en partes. Dos ambientes prod y dev
- Espiral: Es de tipo iterativo. En cada iteración se evaluan las diferentes alternativas.
- RUP: Corresponde a un software. Cada ciclo de vida del software abarca 4 meses. La esencia es la iteracion.

## Modelos Agiles (TAA)

Conjunto de estrategias de desarrollo que promueven practicas adaptativas, en lugar de predictivas.

- Adaptive Software Development (ASD): Busca equilibrio entre un ambiente de creatividad y administrar lo que se hace. No se utiliza tanto por que no se especifican detalles de pautas.
- XP: Conjunto de practicas ya conocidas, pero combinadas de manera innovadora. **Planificacion incremental** → Los requerimientos son grabados en historias, se desarrolla sobre esto. **Entregas pequeñas** → Se desarrolla lo que da valor primero. Versiones frecuentes. **Diseño simple** → Solo lo necesario es llevado a cabo. **Refactorizacion** → Se debe reinspeccionar frecuentemente el codigo.



- SCRUM: El resultado se produce de manera incremental. Periodos cortos de trabajo siguiendo un mismo patron (sprint). Sprint → Sprint Planning → Daily → Sprint review → retrospective.

## Tradicional vs Agil

### Tradicional:

- Incumplimiento en entrega
- Reduccion de funcionalidades
- Dificil adaptacion a los cambios
- Mucha documentación

### Agil:

#### **Ventajas:**

- Iteraciones en ciclos cortos.
- Límites de tiempo para cada ciclo.
- Adaptable.
- Orientación hacia las personas.
- Estilo de trabajo en equipo.

#### **Desventajas:**

- Adaptarse puede llevar al caos
- Dificil de mantener interes de los clientes que participan
- Mantener simplicidad es trabajo extra
- Definir contratos puede ser un problema

<b>Metodologías Ágiles</b>	<b>Metodologías Tradicionales</b>
Basadas en heurísticas provenientes de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas normas
Generalmente no existe un contrato tradicional, o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (< 10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos