Resumen Examen Ing. De Software 2019

# Intro a Ing. De Software

**Objetivo:**

Brindar los conceptos teóricos y prácticos que permiten comprender y ejecutar los distintos procesos que involucran el desarrollo del software.

## Definición de software (pressman):

1. Instrucciones programas de computadoras que cuando se ejecutan Proporcionan la función y rendimiento deseados.
2. Estructura de datos que permiten a los programas manipular adecuadamente la información.
3. Documentos describen la operación y uso de los programas.

## Definición de ingeniería de software:

1. La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; es decir la aplicación de ingeniería al software
2. Disciplina tecnológica y gerencial abocada a la producción y mantenimiento sistemático de productos de software que sean desarrollados y modificados en tiempo y bajo estimaciones de costos
3. Aplicación disciplinada de principios de ingeniería, ciencia y matemática, métodos y herramientas a la producción económica de software. Busca aplicar ciertas prácticas a la producción de software con el objetivo de lograr la calidad.

**Definición de calidad:**

1. Calidad es el conjunto de características de un producto que satisfacen las necesidades de los clientes y en consecuencia hacen satisfactorio dl producto
2. Calidad consiste en no tener deficiencias en el producto o en el proceso

**Proceso de software:**

1. Define las actividades practica técnicas roles herramientas necesarias para transformar los requerimientos del usuario en un producto de software

**Estos se dividen en dos grandes grupos:**

1. Procesos de ingeniería: Comprenden actividades inherentes a la construcción del producto
2. Procesos de apoyo: Comprenden actividades inherentes a la gestión de los procesos de construcción del producto

# Áreas de la ingeniería de software

1. **Ingeniería de requerimientos de software:** Definición o identificación de los servicios o capacidades que debe proveer el software para satisfacer las necesidades de los usuarios. Definición del problema a resolver.
2. **Diseño de software:** 
   1. **Diseño arquitectónico:**
      1. Determinar la estructura general del sistema.
      2. Asignar los requerimientos del software a los subsistemas.
      3. Definir interfaces de los subsistemas.
   2. **Diseño detallado:**
      1. Descomponer el sistema en clases o módulos.
      2. Seleccionar mecanismos y algoritmos.
      3. Asignar responsabilidades a módulos.
3. **Construcción del software:** Codificar, validar y probar unitariamente el software.
4. **Pruebas del software:** Integración de los distintos componentes del software. Verificación dinámica del software contra la especificación del comportamiento esperado. Existen distintos niveles de prueba y distintos objetivos de las pruebas.
5. **Mantenimiento del software:** Modificar el software una vez entregado para corregir fallas o incorporar a cambios.
6. **Gestión de proyectos de software:** Planificación y seguimiento de las actividades necesarias para la construcción y mantenimiento del software.
7. **Gestión de la configuración del software:** Identificación y control del software y sus componentes durante su construcción, mantenimiento y uso.
8. **Calidad del software:** Asegurar la calidad del producto y de los procesos durante el ciclo de vida del proyecto.
9. **Procesos de ingeniería del software:** Definición, implementación y mejora del proceso de software.
10. **Herramientas y métodos:** Ambientes de desarrollo para software y métodos para la realización de las distintas actividades del desarrollo del software.

# SCM (Gestión de la Configuración)

El arte de identificar, organizar y controlar modificaciones al software que esta siendo construido por un equipo de programadores, con el objetivo de maximizar la productividad, minimizando los errores.

* SCM es una actividad de protección, aplicada durante todo el ciclo de vida.
* Identifica, controla, audita e informa de las modificaciones que se van dando en el desarrollo del software
* Cualquier elemento producido durante el proceso forma a parte de la configuración del software.
* Los elementos se organizan en un repositorio de forma tal que sea posible controlar los cambios.
* Los objetos generados organizaron versiones y variaciones, mediante la gestión de versiones.
* Se aplican mecanismos de control de acceso y sincronización.
* La auditoría de SCM es una actividad SQA que ayuda a asegurar que se mantiene la calidad durante la realización de cambios.
* Los informes proporcionan información sobre los cambios a aquellos que deben estar informados.

## Repositorio

* Es una estructura de directorios en disco, donde se almacenan los elementos componentes de software producidos a lo largo de todo el proyecto.
* Tres áreas básicas:

1. Desarrollo o trabajo.
2. Aprobación o control de calidad.
3. Consumo.

## Control de versiones

* Permitir especificar configuraciones alternativas del software mediante la selección de las versiones adecuadas de los componentes.
* Esto se puede gestionar asociando atributos a cada versión del software y permitiendo luego especificar y construir una configuración describiendo el conjunto de atributos deseados.

## Herramientas

El trabajo de gestión de la configuración puede apoyarse en herramientas, que por ejemplo hagan el control de versiones y control de cambios en forma automática.

1. Implementar un repositorio centralizado y de publicarlo para equipos distribuidos.
2. Controlar el flujo de trabajo definido en el circuito de cambios.
3. Asignar permisos y estados.

# Requerimientos (Tres Niveles)

**Requerimientos de Negocio:**

Representan los objetivos de alto nivel que la organización o cliente requiere. Incluyen los objetivos del negocio, la visión del producto y el alcance del proyecto. Los requerimientos que se presentan en este nivel son los más importantes a cumplir. También debe tomarse en cuenta que cubren aspectos muy generales acerca del funcionamiento real del producto requerido.

**Requerimientos del Usuario:**

Los requerimientos del usuario especifican las tareas que los usuarios esperan poder realizar con el sistema, así como también los atributos o características que el sistema deberá mostrar para satisfacer las necesidades del usuario. Los usuarios serán los que trabajen directamente con el programa. Es necesario que sus necesidades sean cubiertas teniendo como prioridad los requerimientos del negocio.

**Requerimientos de Sistema:**

Definición detallada de las funciones, servicios y restricciones operativas del sistema

1. **Requerimientos Funcionales:**

Los requerimientos funcionales describen el comportamiento o funcionalidades que el producto debe mostrar ante cada condición específica. El analista deriva y evalúa la funcionalidad que el sistema debe brindar al usuario final. Algunas veces estos requerimientos son llamados de comportamiento. En este aspecto los requerimientos manifiestan explícitamente las funcionalidades que deberá tener el producto final y deberán ser cubiertos por el desarrollador.

1. **Requerimientos no funcionales:**

Los requerimientos no funcionales describen las propiedades o restricciones sobre el comportamiento del sistema. En muchas ocasiones terminan siendo cosas muy simples como aspecto y diseño del producto final. Pero otras veces terminan siendo de alta demanda para el desarrollador. Puesto que también pueden referirse a tiempos de ejecución, trabajo con determinado hardware, etc. En este punto los requerimientos no funcionales proveen de aspectos que, aunque no tenga funcionalidades explícitas como en el caso anterior. Su cumplimiento es de vital importancia como el de cualquier otro requerimiento.

## Características de requerimientos:

* **No ambiguo:** El texto debe ser claro, preciso y tener una única interpretación posible.
* **Conciso:** Debe redactarse en un lenguaje comprensible por los inversores en lugar de uno de tipo técnico y especializado, aunque aun así debe referenciar los aspectos importantes.
* **Consistente:** Ningún requisito debe entrar en conflicto con otro requisito diferente, ni con parte de otro. Asimismo, el lenguaje empleado entre los distintos requisitos debe ser consistente también.
* **Completo:** Los requisitos deben contener en sí mismos toda la información necesaria, y no remitir a otras fuentes externas que los expliquen con más detalle.
* **Alcanzable:** Un requisito debe ser un objetivo realista, posible de ser alcanzado con el dinero, el tiempo y los recursos disponibles.
* **Verificable:** Se debe poder verificar con absoluta certeza, si el requisito fue satisfecho o no. Esta verificación puede lograrse mediante inspección, análisis, demostración o testeo.

# Gestión de proyectos

**Proyecto:** Emprendimiento temporario que debe crear un producto o servicio únicos.

* Temporario: tiene un comienzo y un fin.
* Único: se diferencia claramente de otros.

Naturaleza Temporaria y Única del Proyecto

* El proyecto tiene un final: o cumple con sus objetivos, o se cancela ante la certeza de que no logrará cumplirlos. No depende de la duración.
* Objetivo del proyecto: crear un producto o servicio que antes no existía. Al ser único, introduce el concepto de INCERTIDUMBRE.

Actores principales

* Cliente: el que aprueba nuestro trabajo (paga nuestras facturas)
* Usuario: el que va a usar el producto
* Contraparte: quien va a interactuar con nosotros durante el proyecto
* Auditores: verifican que el proyecto se esté desarrollando según lo previsto
* Subcontratistas: socios o no en el emprendimiento

¿Qué es la gestión de proyectos?

Aplicación de conocimientos, habilidades, herramientas y técnicas a las actividades de un proyecto de modo de alcanzar o sobrepasar las necesidades y expectativas de el o los interesados (stakeholders - accionistas).

Parámetros Clave de un Proyecto

* Tiempo, Alcance y Costo.
* Se puede optimizar como máximo 2 de los parámetros.
* La gestión de proyectos es el arte de determinar que par de ellos se optimizará.

**Diagrama de Gant:** El diagrama de Gantt es una herramienta para planificar y programar tareas a lo largo de un período determinado. Gracias a una fácil y cómoda visualización de las acciones previstas, permite realizar el seguimiento y control del progreso de cada una de las etapas de un proyecto y, además, reproduce gráficamente las tareas, su duración y secuencia, además del calendario general del proyecto.

### Duración y Esfuerzo

**Duración:** Es el tiempo de trabajo (sin incluir días festivos u otros periodos de no trabajo) que se requieren para completar una actividad u otro elemento del proyecto. Se expresa generalmente días, semanas, meses etc.

**Esfuerzo:** Es el número de unidades de trabajo requeridas para completar una actividad u otro elemento de proyecto. Usualmente se expresa en horas de staff u horas hombre, días de staff, o semanas de staff. No se debe confundir con duración.

# Pruebas

Es el proceso de ejecutar el software con el objetivo de encontrar defectos.

* Una prueba tiene éxito si descubre errores.
* Un buen caso de prueba es aquel con alta probabilidad de descubrir un error no encontrado hasta el momento.

Un programa hace lo que tiene que hacer y un programa no hace lo que no tiene que hacer.

**Error humano:** Acción humana que produce un resultado incorrecto.

**Defecto / Falta [IEEE]:** Anomalía del producto.

**Fallo [IEEE]:** Desviación del componente o sistema de su comportamiento, servicio o resultado esperado.

**Verificar**

1. ¿Estamos construyendo correctamente el producto?
2. Consiste en mostrar que el software cumple con las especificaciones.

**Validar**

1. ¿Estamos construyendo el producto correcto?
2. Consiste en mostrar que el software cumple con las expectativas de cliente.

# Niveles de prueba

### Prueba unitaria

1. Verifica el funcionamiento de programas o módulos en forma aislada.
2. Generalmente la realiza el propio desarrollador.
3. Se desarrollan programas conductores (aceptan los datos del caso de prueba, se los pasan al módulo e imprimen el resultado) y/o programas de resguardo (para reemplazar módulos subordinados)

### Prueba de integración

1. Verifica el funcionamiento resultante de la interacción de un conjunto de componentes integrados.
2. Generalmente implica una estrategia de integración.

### Prueba de sistema

1. Se preocupa de verificar (y/o validar) el funcionamiento de todo el sistema.
2. Compara el sistema contra los requerimientos funcionales y no funcionales.

### Prueba de aceptación

1. Es una prueba del sistema con el objetivo de validarlo por parte del usuario.
2. Contrasta la funcionalidad del sistema con los requerimientos de los usuarios.

# Tipos de prueba

### Pruebas alfa y beta

* Son pruebas realizadas antes de la liberación del sistema, por el usuario final.
* **Prueba alfa:** en el lugar de desarrollo o un entorno controlado.
* **Prueba beta:** por el propio usuario en su ambiente operativo.
* Se realizan con un grupo seleccionando de usuarios.
* Generalmente se realizan sobre un número finito de casos de prueba seleccionados al inicio del proyecto.
* Prueba de regresión
* La prueba de sistema (o subsistema) se orienta a verificar que no se han introducido defectos cuando se modifica el software.
* Volver a ejecutar un subconjunto de pruebas para asegurar que los cambios (debidos a pruebas u otros motivos) no han propagado efectos colaterales no deseados
* Manuales o utilizando herramientas automáticas

3 clases de casos de prueba:

* + Muestra representativa de pruebas que ejercite todas las funciones
  + Pruebas centradas en funciones que probablemente se vean afectadas por el cambio
  + Pruebas centradas en los componentes que cambiaron

# Estimación

Predecir valores de entidades y sus atributos que sean relevantes para el proyecto.

Estimamos cuando no podemos medir:

* La entidad todavía no existe
* La entidad no es accesible
* La medición es muy cara o peligrosa

### ¿Por qué estimar?

Sin una estimación precisa, no se puede gestionar los recursos en forma efectiva.

### ¿Qué estimar?

1. Tamaño
2. Esfuerzo
3. Tiempo (Cronograma)
4. Recursos
5. Defectos

### ¿Cómo estimar?

* Medir: La medición como base para los modelos predictivos.
* Calcular: Utilizar modelos para calcular los atributos derivados de otras mediciones
* Juicio experto: Si no existen modelos ni la posibilidad de medir directamente utilizar el juicio del experto

### Elementos que influyen en la estimación de un proyecto de software

1. Tamaño del software
2. Tipo de proyecto
3. Experiencia y habilidades del equipo
4. Lenguajes de programación
5. Proceso utilizado

## Estimación por juicio experto

Utilización de rangos y PERT para el caso esperado

Esperado = (Mejor + Probable\*4 + Peor) / 6

## Estimación por analogía

* También conocida como estimación top-Down.
* Considera todo el proyecto con otros de similares características o se usa el juicio de un experto.
* Poco costosa.
* Menos precisa.

Ej.: A partir de un proyecto similar se estima que el nuevo es un 20% más grande.

Anterior 18 meses-hombre, entonces el nuevo se estima en 22 meses-hombre

## Estimación por descomposición

* Se descompone el proyecto en unidades de trabajo (funciones, módulos o actividades) que son fáciles de estimar.
* Más precisa que la anterior.
* Más costosa que la anterior.

# Ciclo de vida

* Es una estructura aplicada al desarrollo de un producto de software.
* Es una secuencia estructurada y bien definida de las etapas en Ingeniería de software para desarrollar el producto sofware deseado.

En Cascada: Un ejemplo de una metodología de desarrollo en cascada es:

1. Análisis de requisitos.
2. Diseño del sistema.
3. Diseño del programa.
4. Codificación.
5. Pruebas.
6. Implementación o verificación del programa.
7. Mantenimiento.

De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costos del desarrollo. La palabra cascada sugiere, mediante la metáfora de la fuerza de la gravedad, el esfuerzo necesario para introducir un cambio en las fases más avanzadas de un proyecto.

Espiral: Para cada ciclo habrá cuatro actividades:

1. Determinar objetivos
2. Análisis del riesgo
3. Desarrollar y probar
4. Planificación

Incremental: La idea principal detrás de mejoramiento iterativo es desarrollar un sistema de programas de manera incremental, permitiéndole al desarrollador sacar ventaja de lo que se ha aprendido a lo largo del desarrollo anterior, incrementando, versiones entregables del sistema. El aprendizaje viene de dos vertientes: el desarrollo del sistema, y su uso (mientras sea posible).

Sping: Un ejemplo de la metodología de spring es:

1. El Product Owner redacta las User Stories y las sitúa en el Product Backlog.
2. A continuación, el Product Owner prioriza estas User Stories y ordena el Product Backlog en consecuencia.
3. El equipo Scrum se junta en la reunión de planificación del Sprint, con el objetivo de establecer la lista de las User Stories que se tratarán durante el Sprint. Esto forma el Sprint Backlog y a continuación se descomponen en tareas por el equipo de desarrollo.
4. Entonces el Sprint puede comenzar con una iteración de 2, 3 o 4 semanas.
5. El equipo se reúne diariamente para realizar la Melé diaria.
6. Como consecuencia del Sprint, obtenemos un producto potencialmente entregable que forma parte de una demostración durante la revisión del Sprint.
7. El ciclo termina con la retrospectiva del Sprint.

Y a continuación, solo hay que repetir todo de nuevo.