Comenzado el	Tuesday, 14 de September de 2021, 09:32
Estado	Finalizado
Finalizado en	Tuesday, 14 de September de 2021, 11:23
Tiempo empleado	1 hora 51 minutos
Calificación	Sin calificar aún

Pregunta **1**

Finalizado

Puntúa como 1,00

Enuncie los cinco desafíos de la Ingeniería del Software mencionados en la materia.

El problema de producir software para satisfacer las necesidades del cliente/usuario guía el enfoque usado en IS, pero hay otros factores que tienen impacto en la elección del
enfoque: Escala, Calidad, Productividad, Consistencia, Cambios,...
Escala: La IS debe considerar la escala del sistema a desarrollar. Los métodos utilizados para desarrollar pequeños problemas no siempre escalan a grandes problemas.
La escalabilidad es hacia arriba o hacia abajo.
Hay dos claras dimensiones a considerar, Métodos de ingeniería y Administración de proyectos. En un sistema pequeño ambos pueden ser informales/ad-hoc, pero en un sistema grande ambos deben ser formalizados.
Productividad: el costo del software es la mano de obra, medido en persona/mes(PM). La productividad se mide por KLOC/PM.
Consistencia: sucesiva producción de sistemas de alta calidad y productividad.
Cambio: los métodos tienen que ser fácilmente adaptables al cambio.
Calidad: Desarrollar software de alta calidad es un objetivo fundamental. La calidad del software es difícil de definir (contrariamente al costo y al tiempo). Aunque no hay una definición fija, la calidad se basa en distintos aspectos:
- Funcionalidad: cumplimiento de las necesidades establecidas.
- Eficiencia: desempeño apropiado relativo a la cantidad de recursos usados.
- Mantenibilidad: capacidad de ser modificado para mejoras y correcciones.
- Portabilidad: adaptable a distintos entornos sin aplicar acciones de más.
- Usabilidad: capacidad de ser comprendido, aprendido y usado.
- Confiabilidad: realización de lo requerido dentro de las condiciones y los tiempos establecidos.

Comentario:
10

Pregunta **2**

Finalizado

Puntúa como 1,00

Enumere las características que debería cumplir una buena SRS.

Correcta: Cada requerimiento representa precisamente alguna característica deseada por el cliente en el sistema final.
Completa: - Todas las características deseadas están descritas.
- La característica más difícil de lograr (para conseguirla uno debe detectar las ausencias en la especificación).
- Corrección y completitud están fuertemente relacionadas.
No ambigua: - Cada requerimiento tiene exactamente un significado (no se superpone con otros). •Si es ambigua los errores se colocarán fácilmente:
- Particular atención si se usa lenguaje natural.
- Los lenguajes formales ayudan a "desambiguar"
Consistente: Ningún requerimiento contradice a otro. Ej.: conflictos lógicos, temporales, de dependencias.
Verificable: - Si cada requerimiento es verificable, i.e. si existe algún proceso efectivo que pueda verificar que el software final satisface el requerimiento.
- La ambigüedad es esencial para la verificabilidad.
- Como la verificación es usualmente hecha a través de revisiones, la SRS debe ser comprensible (al menos por el desarrollador, el usuario y el cliente).
Rastreable (Traceable): - Se debe poder determinar el origen de cada requerimiento y cómo éste se relaciona a los elementos del software.
- Hacia adelante: dado un requerimiento se debe poder detectar en qué elementos de diseño o código tiene impacto.
- Hacia atrás: dado un elemento de diseño o código se debe poder rastrear que requerimientos está atendiendo.
Modificable: - Si la estructura y estilo de la SRS es tal que permite incorporar cambios fácilmente preservando completitud y consistencia.
- La redundancia es un gran estorbo para la modificabilidad (puede resultar en inconsistencia).

Ordenada en aspectos de importancia y estabilidad: - Los requerimientos pueden ser críticos, importantes pero no críticos, deseables pero no importantes.
- Algunos requerimientos son esenciales y difícilmente cambian con el tiempo.
Otros son propensos a cambiar. => Se necesita definir un orden de prioridades en la construcción para reducir riesgos debido a cambios de requerimientos.

Comentario:
El enunciado está bien, ojo que la explicaciones no lo están

10

Pregunta **3**

Finalizado

Puntúa como 1,00

Una inmobiliaria maneja información acerca de inmuebles que se alquilan (casa, departamentos, locales) y clientes que pueden ser propietarios de inmuebles o inquilinos. El sistema debe brindar funcionalidades tales como dar de alta una propiedad al sistema, dar de alta un cliente, registrar un inquilino, etc. Una misma persona puede ser propietaria de más de un inmueble en alquiler, pero también puede alquilar más de un inmueble. De un inmueble interesa información tal como los metros cuadrados del mismo, su dirección, servicios, etc.; de un departamento interesa el costo de las expensas, cantidad de pisos del edificio, etc.; de un local: si tiene baño, cocina y teléfono. De los clientes se suele requerir información básica como apellido y nombre, DNI, dirección; de los propietarios: teléfono de contacto (fijo o celular), al igual que los inquilinos. Para el alquiler de inmueble siempre se exige una propiedad de garantía, de la cual se registra el número catastral de la misma (se exige una garantía por cada alquiler).

Sobre este problema, describa un casos de uso, sin descuidar casos excepcionales.

Caso de uso: Registrar inquilino.

Actor primario: Persona

Precondición: La persona no esta registrada

Escenario exitoso principal:

1. La persona selecciona Registrar inquilino.

2. El sistema le devuelve una plantilla para ingresar sus datos

3. La persona completa los campos solicitados para el registro (apellido, nombre, DNI, dirección de los propietarios, teléfono de contacto (fijo o celular)).

4. El sistema le indica que los datos ingresados son válidos, los guarda y le avisa que se a registrado correctamente.

Escenarios excepcionales: a) DNI incorrecto. El sistema le comunica vuelva a ingresar su DNI de manera correcta.

b) Direccion de propietarios incorrecta o incompleta. El sistema le comunica que las direcciones son incorrecta o estan incompletas y que vuelva

a ingresar las direcciones.

c) Telefono de contacto incorrecto. El sistema le comunica que vuelva a ingresar el telefono.

Comentario:
El actor primario debe especificar qué tipo de persona realiza el caso de uso..

10

Pregunta **4**

Finalizado

Puntúa como 1,00

A) ¿Qué es un estilo arquitectónico?

B) Describa exhaustivamente uno de los estilos arquitectónicos vistos en la asignatura.

A)
Un estilo arquitectónico define una familia de arquitecturas que satisface las restricciones de ese estilo. Los estilos proveen ideas para crear arquitecturas de sistemas.
Distintos estilos pueden combinarse para definir una nueva arquitectura.
B)
Tubos y filtros: para sistemas que realizan transformaciones de datos. Las componentes se llaman filtros, que realizan transformaciones y se los pasan a
otro filtro a través de un tubo (conector). Los filtros son independientes, y no tienen información sobre los demás. Un tubo sólo conecta 2 filtros.
Datos compartidos: las componentes pueden ser repositorios (almacenan datos) o usuarios (acceden a los datos del repositorio, realizan cálculos y ponen los resultados otra vez en el repositorio). La comunicación entre los usuarios y los datos se hace solamente a través del repositorio. Los conectores son las lecturas y escrituras.
Cliente-servidor: las componentes son clientes o servidores. Los clientes inician la comunicación, y solo se relacionan con el servidor (no con otros clientes).
Los conectores son las solicitudes y las respuestas. Es un estilo multi-nivel:
- Nivel de cliente: contiene a los clientes.
- Nivel intermedio: contiene las reglas del servicio.
- Nivel de base de datos: donde reside la información.

Publicar-suscribir: las componentes publican eventos o se suscriben a ellos. Cada vez que un evento es publicado, se invoca a sus suscriptores.
Peer-to-peer: cada componente le puede pedir servicios a otra.
Procesos que se comunican: a través de pasaje de mensajes.

Comentario:
Tubos y filtros está incompleto

10

preguntarme comentarios

Pregunta **5**

Finalizado

Puntúa como 1,00

Mencione y describa los principios fundamentales, mencionados en clases, que debe seguir un buen diseño de software

Partición y jerarquía
Se basa en dividir el problema en partes manejables:Cada parte tiene que solucionarse y modificarse separadamente. Estas partes no deben ser completamente independientes, sino que deben comunicarse para solucionar el problema, aunque esto agregue complejidad algunas veces. Se intenta mantener la mayor independencia posible. El particionado determina una jerarquía que se forma de la relación "es parte de". Cuando el costo supera el beneficio, hay que detener el particionado.
Abstracción

La abstracción de un componente describe el comportamiento externo sin dar detalles internos. Es esencial en el particionado del problema.
Abstracción de componentes existentes:
● Representa componentes como cajas negras.
● Oculta detalle
● Útil para comprender sistemas existentes (Importante en mantenimiento)
● Útil para determinar el diseño del sistema existente

Abstracción durante el proceso de diseño:
- Las componentes no existen
- Para decidir cómo interactúan las componentes sólo el comportamiento externo es importante
- Te concentras en un componente a la vez
- Consideras un componente sin preocuparse por las otras
- El diseñador maneja la complejidad
- Transición gradual de lo más abstracto a lo más concreto
- Necesaria para solucionar las partes separadamente

Dos abstracciones:
1) Abstracción funcional
2) Abstracción de datos

Abstracción funcional:
Especifica el comportamiento funcional de un módulo como funciones de entrada/salida.
Se puede especificar usando pre y post condiciones.
Abstracción de datos:
Los datos se tratan como objetos junto a sus operaciones
Las operaciones definidas para un objeto solo se realizan en ese objeto
Los detalles internos permanecen ocultos y solo sus operaciones son visibles

Modularidad

Un sistema se dice modular si consiste de componentes discretas tal que puedan implementarse separadamente y un cambio a una de ellas tenga mínimo impacto sobre las otras.
Provee la abstracción en el SW.
Es el soporte de la estructura jerárquica de los programas
Mejora la claridad del diseño y facilita la implementación
Reduce costo de testing, debugging y mantenimiento
Se usan dos enfoques para diseñar jerarquías de componentes: Top Down y Bottom Up.

Comentario:
10

Pregunta **6**

Finalizado

Puntúa como 1,00

A) ¿Qué es el acoplamiento?

B) ¿Qué se busca de él?

C) ¿Qué tipos de acoplamiento aparece en el diseño orientado a objetos?

A) El acoplamiento es la forma y nivel de interdependencia entre módulos de software; una medida de qué tan cercanamente conectados están dos rutinas o módulos de software; así como el grado de fuerza de la relación entre módulos.

B) Se busca un acoplamiento bajo mediante la minimización del número de interfaces por módulos y la complejidad de cada interfaz.

C)
Acoplamiento de Interacción
Ocurre debido a que métodos de una clase invocan métodos de otra clase, para mantener bajo el acoplamiento sólo debemos pasar datos y no objetos cuando invocamos métodos de otra clase.

Acoplamiento de Componentes
Un objeto A de la clase C está acoplado con una clase C1, si C tiene una variable de tipo C1 o tiene un método que toma un parámetro de C1.

Acoplamiento de Herencia
Se genera debido a las herencias entre clases. El problema se da cuando usamos herencia múltiple y cuando queremos pisar métodos heredados. El escenario más débil se da cuando se hereda de una clase sólo para extenderla. Entonces, a la hora de heredar debemos preguntarnos si estamos agregando funcionamiento.

Comentario:
10

En el acoplamiento de intracción no se deben pasar objetos??

Promedio del parcial = 100%

Nota = 10

◄ Cronograma 2021

Ir a...

Parcial 2 ►