Preguntas Presentacion 7 (Fabelio)

Menciona los tres pasos para la definición de la arquitectura:

1. **Definir requerimientos de arquitectura**
2. **Diseño de la arquitectura**
3. **Validación**

Describa los niveles de prioridad de los requerimientos de arquitectura:

1. **Alta: dirigen el diseño de la arquitectura**
2. **Media: soportado eventualmente**
3. **Baja: Seria bueno tenerlo**

Son creados para probar algunos aspectos de diseño de alto riesgo y que no hayan sido entendidos completamente:

Prototipos

Menciona los propósitos del uso de prototipos:

1. **Prueba de concepto: ¿Puede la arquitectura ser construida y satisfacer los requerimientos?**
2. **Prueba de tecnología: ¿Se comporta como se esperaba?**

Especifican las respuestas del software a determinados estímulos:

Requerimientos de calidad

Decisión de diseño que afectan el control de una respuesta a un atributo de calidad:

Tácticas

Es un mecanismo arquitectónico (sinónimo)

Tácticas

Una estrategia arquitectónica es un plan de diseño? (F/V)

FALSO. Una estrategia arquitectónica es un conjunto de tácticas.

Mencione los fundamentos para documentar la arquitectura

1. **Para que otros puedan entender y evaluar el diseño.**
2. **Entender el diseño después de un largo tiempo**

3. **Para que otros miembros del proyecto puedan aprender de la arquitectura**
4. **Analizar el diseño para mejorar el desempeño o generar métricas**

Cuales son los problemas al documentar la arquitectura?

1. **No existen estándares unificados**
2. **Documentar una arquitectura muy compleja consume mucho tiempo**
3. **Una arquitectura tiene muchas vistas, documentarlas consume tiempo y dinero.**
4. **Mantener la documentación consume recursos de tiempo y calendario.**

Cubre aspectos tanto de comportamiento como de estructura en un sistema de software

Notacion UML 2.0

Preguntas Presentacion 8 (Peter)

Técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software.
Proporciona uno o más escenarios de interacción del sistema con el usuario u otro sistema
CASOS DE USO

Persona, Organización o sistema externo que juega un rol en una interacción con el sistema
Actor

Existe siempre que un actor esté involucrado con una interacción descrita por un caso de uso
ASOCIACIONES

Representa la vista estatica de un software
DIAGRAMA DE CLASES

Elementos que describen cosas
CLASIFICADORES

Identificacion de cada concepto del mundo real
LA CLASE EN UN DIAGRAMA

Relacion entre un todo y sus Partes
AGREGACION (con rombo vacio)

Las partes solo existen asociadas al compuesto(accesan a través de el)
COMPOSICION(con rombo lleno)

Relación entre cliente y un servidor(equivalencia a una relación de uso)
DEPENDENCIA

Una clase que no se puede instanciar, se utiliza para definir subclases, esta no tiene implementación de sus métodos
CLASE ABSTRACTA

Se utiliza cuando queremos utilizar polimorfismo y definir abstracción de diferentes objetos de diferentes tipos
CLASE ABSTRACTA

Es una clase completamente abstracta(una clase sin implementación)
INTERFACES

Las responsabilidades que se definen durante el diseño de clases

- **CREAR OBJETOS,**
- **REALIZAR CÁLCULOS,**
- **INICIAR ACCIONES DE OBJETOS,**
- **CONTROLAR Y COORDINAR ACTIVIDADES EN OTROS OBJETOS**

Por qué estan influenciadas las responsabilidades en la traducción de clases y metodos

- **Granularidad de la responsabilidad**
- **Los metodos cumplen las responsabilidades, estas no son metodos**
- **Son implementadas usando metodos que funcionan solos**

El acoplamiento es una medida de:

- **Que tan fuertemente esta conectado un elemento**
- **Tiene el conocimiento de**
- **Depende de**

Método para medir que tan fuertemente relacionadas y enfocadas son las responsabilidades
ALTA COHESION

Un elemento con baja cohesión se caracteriza por:

- **DIFICIL DE COMPRENDER**
- **DIFICIL DE REUSAR**
- **DIFICIL DE MANTENER**
- **MUY DELICADA; AFECTADA CONSTANTEMENTE POR EL CAMBIO**

- **TIENE RESPONSABILIDADES QUE DEBERÍAN SER DE OTROS OBJETOS**

Preguntas Presentacion 9 (Pablo)

Herramienta de arquitectura para realizar vistas y documentar una aplicación de software, utiliza 5 vistas concurrentes en la que cada vista representa un conjunto específico de problemas?

R. Modelo 4 + 1 Vistas

Los arquitectos lo utlizan para entender y documentar las muchas capas que una aplicación tiene, de manera sistemática y estandarizada?

R. Modelo 4 + 1 Vistas

Cuales son las 5 vistas del modelo 4 + 1 vistas?

R. Vista Logica
Vista de Procesos
Vista Fisica
Vista de Despliegue
Vista de Casos de Uso

Es la vista que soporta los requerimientos de comportamiento y muestra como el sistema esta distribuido en un conjunto de abstracciones?

R. Vista Lógica

De que forma se representa se representa la vista lógica?

R. Diagramas de clase, colaboración y secuencia

Es la vista utlizada para describir los modulos del sistema, para estudiar en donde se van los archivos en el sistema y ambiente de desarrollo?

R. Vista de Desarrollo

La vista de desarrollo es utilizada para:

R. Asignar tareas de implementacion
Evaluar la cantidad de codigo a ser modificada
Razonar la reutilizacion
Considerar estrategias de entregas

Es la vista que permite describir y estudiar los procesos del sistema y como estos se comunican, util cuando se tienen muliples hilos en el software?

R. Vista de Proceso

Es la vista que ilustra la descomposicion de los procesos del sistema, incluyendo el mapeo de clases y subsitemas e hilos.

R. Vista de Proceso

Es la vista que nos permite identificar problemas de concurrencia, tiempo de respuesta, deadlocks, troughput, tolerancia a fallas y escalabilidad?

R. Vista de Proceso

Es la vista que provee las bases para entender la distribución física del sistema en un conjunto de nodos del sistema usualmente ilustrada por medio del diagrama de deployment?

R. Vista fisica

Es la vista que representa como la aplicación es instalada y como se ejecuta en una red de computadoras, toma en cuenta requrimientos no funcionales.

R. Vista Fisica

Es la vista que describe la organizacion estatica del software en su ambiente de desarrollo?

R. Vista de Despliegue

Es la vista que representa los escenarios e intercepta las otras cuatro vistas?

R. Vista de Casos de uso

Preguntas Presentación 10 (Luis S.)

Agrupa datos encapsulados y procedimientos para representar una entidad?
R/ Un Objeto

Guía a los programadores a pensar en términos de objetos, en vez de procedimientos, cuando planifican su código?
R/ Diseño orientado a objetos.

Es la disciplina que define los objetos y sus interacciones para resolver un problema de negocio que fue identificado y documentado durante un análisis siempre orientado a objetos?
R/ Diseño orientado a objetos.

Son las formas de interactuar con el objeto, también se definen en esta etapa y este se caracteriza por la interacción de esos objetos?
R/ La interfaz del objeto.

Es un acrónimo mnemónico introducido por Robert C. Martin a comienzos de los años 2000?
R/ SOLID

Representa cinco principios básicos de la programación orientada a objetos y el diseño?
R/ SOLID

Son guías que pueden ser aplicadas en el desarrollo de software para eliminar código ineficaz o sucio, provocando que el programador tenga que refactorizar el código fuente hasta que sea legible y extensible?
R/ SOLID

Que significa SRP?
R/ Single Responsability Principle

Cada clase tiene una única responsabilidad, y que esa responsabilidad debería ser completamente encapsulada por la clase. Todos sus servicios deben ser estrechamente alineados con la responsabilidad asignada.
R/ SRP

Que significa OCP?
R/ Open/Closed Principle

Las entidades de software (Clases, módulos, funciones, etc.) deben estar abiertas para su extensión, pero cerrados a la modificación?
R/ OCP

Una vez terminada, la implementación de una clase sólo puede ser modificado para corregir errores; características nuevas o modificadas requerirían que se creará una clase diferente?
R/ Meyer's OCP

Redefinido para referirse al uso de interfaces abstractas, donde las implementaciones se puede cambiar y múltiples implementaciones podrían ser creadas y polimórficamente sustituirse entre si? **R/ Polymorphic OCP**

Que significa LSP?
R/ Liskov Substitution Principle

Es una definición particular de una relación de subtipos. Se trata de una semántica más que de una relación sintáctica, tiene la intención de garantizar la interoperabilidad semántica de los tipos en la herencia y los tipos de objeto en particular?
R/ LSP

Que significa ISP?
R/ Interface Segregation Principle

Los clientes no deben ser forzados a depender de métodos que no utilizan?
R/ ISP

Este principio tiene la intención de mantener el sistema lo más desacoplado posible y fácil de refactorizar, modificar y redistribuir?
R/ ISP

Que significa DIP?
R/ Dependency Inversion Principle

Este principio invierte la forma en que algunas personas pueden pensar en el diseño orientado a objetos, que dicta tanto de nivel alto y bajo objetos deben depender de la misma abstracción?
R/ DIP

Que significa DRY?
R/ Don’t Repeat Yourself

Es un principio que está destinado a la reducción repetición de la información de todo tipo, es especialmente útil en las arquitecturas multi-nivel?
R/ DRY

Este principio aplica bastante a los esquemas de bases de datos, planes de prueba, el sistema de construcción, incluso la documentación?
R/ DRY

Cada pieza de información debe tener una única e inequívoca representación y autoridad dentro del sistema?
R/ DRY

Que significa IOC?
R/ Inversion of Control

Describe el diseño en porciones custom-written del programa y recibe el flujo de control de una biblioteca reutilizable y genérica.?
R/ IOC

Preguntas Presentacion 11 (Kevin)

¿Qué es un patrón de diseño?

- Es una **solución** general **reutilizable** a un problema que ocurre **comúnmente** en un contexto determinado en el diseño de software.
- Es una descripción o una **plantilla** para **resolver** un problema que se puede utilizar en **muchas** situaciones diferentes.
- Son el esqueleto de las soluciones a problemas **comunes** en el desarrollo de software

¿Qué no es un patrón de diseño?
No es un diseño o solución final que se puede transformar directamente en código fuente

¿Qué es un patrón de diseño orientado a objetos?
Muestran las **relaciones** e **interacciones** entre clases u objetos, **sin** especificar las clases u objetos que están involucrados. Buscan añadir capas de abstracción.

¿Cuáles son las tres categorías de los patrones?
Patrones de creación (creacionales), Patrones de Estructura (estructurales) y Patrones de Comportamiento;

¿Qué son los patrones creacionales?
Son los que crean objetos y aíslan sus detalles, en lugar de tener que crear instancias directamente objetos.

Mencione los patrones creacionales
Abstract Factory, Builder, Factory Method, Prototype y Singleton

¿Qué es el patrón Singleton?
Es un patrón que **restringe** la creación de instancias de una clase a **un solo** objeto. Es útil cuando se necesita un solo objeto para coordinar las acciones de todo el sistema.

Mencione las características del patrón Singleton
Constructor privado, tiene un método estático que devuelve la única instancia de la clase.

¿Qué es el patrón Factory Method?
Utiliza métodos de fábrica para crear objetos sin especificar la clase exacta de objeto a crear.

Preguntas Presentacion 12 (Lester)

¿Qué son patrones de arquitectura?

Expresan el esquema fundamental de organización para sistemas de software, proveen un conjunto de subsistemas predefinidos, especifican sus responsabilidades, e incluyen reglas y guías para organizar las relaciones entre ellos.

¿Qué patrones representan el nivel más alto en el sistema de patrones?

Los patrones de arquitectura ;)

Cuáles son los diferentes patrones de arquitectura?

- **From mud to Structure:** ayudan a evitar un mar de componentes, dividiendo correctamente una tarea en subtareas.
- **Distributed Systems:** dirige el problema de forma en que los sistemas o componentes se comunican con otros en un entorno distribuido, Actúa como un *middleman* entre el componente cliente y el servidor
- **Interactive Systems:** Modelo vista controlador, presentación-abstracción-control
- **Adaptable System:** Microkernel reflection

Lenguaje de patrones

Un lenguaje de patrones contiene un amplio conjunto de patrones, cuyas combinaciones conforman su gramática.

Un lenguaje de patrones provee la misma expresividad que un lenguaje natural pero enfocado a la construcción.

Sistema de patrones

Colección de patrones , junto con las guías para su implementación, combinación y uso práctico en el desarrollo de software.

- Mantiene unidos a sus patrones,
- describe la forma en que se conectan y cómo se complementan entre ellos.
- Facilita el uso eficaz de ellos en el desarrollo.

Patrones de arquitectura de aplicaciones empresariales

- **Layering:** Patrones para dividir un sistema en capas

- **Organización de la lógica del dominio:** Formas de organizar los objetos del dominio.
- **Mapping to relational databases:** se relaciona con la comunicación entre la lógica del dominio y los repositorios de datos.
- **Presentación web:** Esta categoría incluye una serie de patrones para gestionar la presentación Web.
- **Concurrencia:** Manejo de la concurrencia.
- **Estado de sesión:** Patrón para el manejo de la sesión en los servidores web.
- **Estrategias de distribución:** Distribución de objetos en múltiples emplazamientos.

¿Qué son los antipatrones?

Son soluciones negativas que presentan más problemas que los que solucionan. Son una extensión natural a los patrones de diseño.

es una forma que describe una solución común a un problema que genera decididamente consecuencias negativas

¿Que provee comprenderlos?

El conocimiento para evitarlos o recuperarse de ellos.

¿Qué explica un buen antipatrón?

Porque la solución original parece atractiva, porque se vuelve negativa, como recuperarse de los problemas que genera.

Características de los antipatrones

- Método eficiente para vincular una situación general a una clase de solución específica.
- Proveen experiencia del mundo real para reconocer problemas recurrentes en la industria del software, ofreciendo también la solución.
- Establecen un vocabulario común para reconocer problemas y discutir soluciones
- Soportan la resolución de conflictos utilizando recursos a diferentes niveles.

Categorías de antipatrones

- **Desarrollo de software:** Son problemas a nivel de desarrollo de la aplicación
- **Arquitectura de software:** A nivel de estructura en las aplicaciones y componentes a nivel del sistema y empresa
- **Gestion de proyectos de software:** identifican algunos de los escenarios claves donde estos son destructivos para el proceso de software.

Mencione 8 ejemplos de antipatrones:

1. **Base de datos como comunicador de procesos (database as an IPC):** Usar una base de datos para comunicar procesos en uno o varios ordenadores, cuando la comunicación entre procesos directa es más adecuada.

2. **Clase Gorda**: Dotar a una clase con demasiados atributos y/o métodos, haciéndola responsable de la mayoría de la lógica de negocio.

3. **Gran bola de lodo** (big ball of mud): Construir un sistema sin estructura

definida.

4. **Entrada chapuza (input kludge):** No especificar e implementar el manejo de entradas inválidas.

5. **Llamar a super (callsuper):** Obligar a las subclases a llamar a un método de la superclase que ha sido sobrescrito.

6. **Acoplamiento secuencial (sequential coupling):** Construir una clase que necesita que sus métodos se invoquen en un orden determinado.

7. **Singletonitis**: Abuso de la utilización del patrón singleton.

8. **Objeto todopoderoso (god object**): Concentrar demasiada funcionalidad en una única parte del diseño (clase).