GUÍA DE ESTUDIO DEL TEMA 3

Objetivo del Diseño de Software. ¿De qué parte el proceso de diseño y qué se obtiene como resultado de dicho proceso?

Parte del análisis de requisitos y la especificación.
A partir del proceso de diseño se obtiene:

- La arquitectura del sistema software: Descripción de los subsistemas y componentes de un sistema software y las relaciones entre ellos.
- Diseño detallado del sistema software: Diseño de los componentes del Sistema.

Explica qué es un patrón de diseño y los tipos de patrones que hay.

Técnica de diseño de software que da solución a un problema de diseño.
Tipos de patrones:

- Patrón arquitectónico
- Patrón de diseño
- Modismo

Explica el patrón arquitectónico en 3 capas.

Patrón utilizado en sistemas grandes que requieren de una descomposición en grupos de subtareas tales que cada grupo de subtareas está en un nivel determinado de abstracción.

Problemas

Las tareas de alto nivel se basan en las de bajo nivel, y las de alto nivel necesitan de servicios intermedios para ser implementados a causa de su complejidad.
Fuerzas a equilibrar en el patrón:

- Mantenibilidad: Los cambios en el código no deberían propagarse en todo el sistema.
- Reusabilidad: Los componentes deberían poder ser reutilizados y reemplazados por implementaciones alternativas.
- Cohesión: Se deben agrupar responsabilidades similares para favorecer la comprensibilidad y mantenibilidad.
- Portabilidad

Solución

Estructuramos el sistema en 3 capas y las colocamos verticalmente. Los componentes de una misma capa trabajarán en el mismo nivel de abstracción y Los servicios que proporciona la capa j utilizan servicios proporcionados por la capa j-1. Al mismo tiempo, los servicios de la capa j pueden depender de otros servicios en la misma capa.
A continuación se definen las tres capas en orden descendente:

- Capa Presentación: responsable de la interacción con el usuario. Sabe cómo presentar los datos al usuario, pero no qué hay que hacer para dar respuesta a las peticiones del mismo.
  - Percibe las peticiones del usuario
  - Ordena la ejecución de acciones y comunica el resultado de las mismas a los usuarios.
  - Trata la interfaz gráfica (ventanas, menús...)
- Capa de Dominio: sabe cómo satisfacer las peticiones del usuario pero ignora dónde se guardan los datos y cómo se presentan al usuario.
  - Recibe eventos
  - Ejecuta acciones
  - Controla la validez de eventos
  - Obtiene los resultados
  - Camba el estado del dominio
  - Envía respuestas a la capa Presentación
- Gestión de Datos: sabe dónde y cómo están almacenados los datos, pero desconoce

cómo tratarlos, y transforma operaciones conceptuales en operación físicas:
- Permite que el dominio pueda ignorar dónde están los datos.
- Las funciones concretas de esta capa dependen del SGBD/SGF que se use.

Define qué son los patrones GRASP.

Son Patrones de Principios Generales para Asignar Responsabilidades (capa de dominio).

- Se trata de principios fundamentales para guiar la asignación de responsabilidades a objetos.
- Dicha asignación de responsabilidades se hace durante el diseño, al definir las operaciones de cada clase de objetos, y se muestra en los diagramas de interacción mediante mensajes que se envían a las clases de objetos.

Patrones GRASP básicos.

Patrón Experto en Información

- Contexto: Asignación de responsabilidades a objetos.
- Problema: ¿Cuál es el propósito general para asignar responsabilidades? Decidir a qué clase hemos de asignar una responsabilidad concreta.
- Solución: Asignar una responsabilidad a la clase que tiene la información necesaria para realizarla.
- Es necesario tener claramente definidas las responsabilidades a asignar, que vienen siendo las postcondiciones de las operaciones.
- Pueden existir diversos expertos parciales para una responsabilidad que deban colaborar.

Patrón Creador

- Contexto: Asignación de responsabilidades a objetos.
- Problema: ¿Quién ha de tener la responsabilidad de crear una instancia de una clase?
- Solución: Asignar a la clase B la responsabilidad de crear una instancia de una clase A si satisface una de las siguientes condiciones:
  - B es un agregado de A.
  - B contiene objetos de A.
  - B tiene los datos necesarios para inicializar un objeto de A.
  - B usa muchos objetos de A.
  - B guarda instancias de A.
- (Se representa en el diagrama de interacción o de clases mediante la operación create(parámetros)).

Patrón de diseño controlador

- Contexto: Los sistemas software reciben eventos externos, de manera que cuando se recibe un evento en la capa de presentación, algún objeto de la capa de dominio ha de recibir y ejecutar las acciones correspondientes.
- Problema: ¿Qué objeto es el responsable de tratar un evento externo?
- Solución:Asignar esta responsabilidad a un controlador, que será un objeto de una clase (no pertenece a la interfaz de usuario). Posibles controladores:
  - Controlador de Fachada: Clase que representa todo el sistema.
  - Controlador de Caso de uso: Clase que representa una instancia de un caso de uso.

Bajo Acoplamiento

Medida del grado de conexión, conocimiento y dependencia de una clase respecto a otras clases.

- Contexto: Asignación de responsabilidades a objetos.
- Problema: ¿Cómo soportar bajas dependencias, bajo impacto del cambio e

incremento de reutilización?

- Hay un acomplamiento de la clase A a la B si:
  - A tiene un atributo de tipo B o una asociación navegable con B
  - B es un parámetro o el retorno de una operación de A
  - Una operación de A hace referencia a un objeto de B
  - A es una subclase directa o indirecta de B
- Solución: Reducir el acoplamiento para evitar que cambios en una clase tengan efecto sobre otras clases, facilitar la comprensión de las clases y facilitar la reutilización.

Alta Cohesión

Medida del grado de relación y de concentración de las distintas responsabilidades de una clase.

- Contexto: Asignación de responsabilidades a objetos
- Problema: ¿Cómo mantener la complejidad controlable?
- Solución: Asignar responsabilidades de manera que la cohesión permanezca alta. Para que la cohesión permanezca alta, la clase debe tener pocas responsabilidades en un área funcional (no muchas en distintos áreas), pocas operaciones muy relacionadas funcionalmente (no muchas operaciones poco relacionadas), , y colaborar con otras clases para hacer las tareas.
- Las ventajas de una alta cohesión son una comprensión, reutilización y mantenimiento más fáciles.
- Los controladores fachada suelen ser poco cohesivos, por lo que suele ser mejor utilizar un controlador de caso de uso.

Tipos de Diagramas de Interacción.

- Diagramas de Secuencia: Destacan la ordenación temporal de los mensajes
- Diagramas de Colaboración: Destacan la organización estructural de los objetos participantes.

Diagramas de secuencia.

Los diagramas de secuencia representan escenarios de casos de uso e incluyen objetos y su línea de vida, mensajes, información de control (condiciones y marcas de iteración), indicar el objeto devuelto por el mensaje.

(A partir de aquí lo mejor es estudiar directamente de las transparencias, de la página 45 en adelante).