Resumen Control 2

Patrones de Arquitectura

Solución de Diseño a una Necesidad

Un patrón de arquitectura proporciona una solución de diseño reutilizable y probada para abordar una necesidad específica en el desarrollo de software.

Características

- **Esquema genérico:** Los patrones ofrecen una solución abstracta y generalizable que puede ser aplicada en diversos contextos.
- **Probado:** Han sido validados en múltiples proyectos y escenarios, demostrando su eficacia.
- **Recurrente:** Se utilizan repetidamente para resolver problemas similares.

Especificación

- **Componentes:** Definición de las partes individuales que componen la solución.
- **Responsabilidades:** Asignación clara de las funciones y responsabilidades de cada componente.
- **Relaciones:** Descripción de cómo interactúan los componentes entre sí.

Descripción de un Patrón

Nombre del Patrón

Cada patrón tiene un nombre específico que lo identifica y describe su propósito.

Contexto

- **Situación que origina la necesidad:** Identificación del problema o situación que crea la demanda del patrón.
- **Ámbito de la necesidad:** Contexto general donde surge la necesidad del patrón.
- **Descripción de las situaciones que la originan:**
  - **Descripción general:** Visión amplia del contexto y el problema.
  - **Descripción detallada:** Análisis más profundo y específico de las circunstancias que requieren la solución.

Requerimiento

- **Descripción genérica de la necesidad:** Definición amplia del problema a resolver.
- **Define lo que se debe resolver:** Establecimiento de objetivos y resultados esperados.
- **Fuerzas presentes en el contexto:**
  - **Propiedades:** Características inherentes al problema.
  - **Requisitos:** Necesidades que deben ser satisfechas.
  - **Restricciones:** Limitaciones que deben ser consideradas.

Solución

- **Esquema de solución de la necesidad:** Propuesta de solución para abordar el problema identificado.
- **Balance de fuerzas:** Equilibrio entre las distintas fuerzas presentes en el contexto.
- **Estructura:**
  - **Componentes:** Elementos de la solución.
  - **Relaciones:** Interacciones entre los componentes.
- **Comportamiento:**
  - **Organización de componentes:** Cómo se estructuran y funcionan los componentes en conjunto.
- **Priorización de fuerzas:** Ordenación de las fuerzas según su importancia y impacto en la solución.

Clasificación de patrones

1. Patrones simples

A. Capas (Layers)

- **Estructura:** Aplicaciones descompuestas en tareas con diferentes niveles de abstracción.
- **Contexto:** Sistemas estructurados con diversos niveles de acción.
- **Requerimiento:** organización inadecuada genera problemas de escalabilidad y mantenibilidad
- **Solución:** Estructuración en esquema multi-capa.
  - **Características:**
    - La capa K se relaciona solamente con la capa K-1.
    - No hay otras dependencias entre capas.
    - Cada capa puede estar integrada por distintos componentes.
    - Los componentes pueden interactuar entre sí, pero quedan acoplados.
    - Cada capa expone una interfaz con los servicios que provee.
    - El comportamiento puede ser top-down o bottom-up.
  - **Implementación:**
    - Determinar el número de capas según el nivel de abstracción requerido.
    - Asignar responsabilidades a cada capa.
    - Especificar los servicios ofrecidos por cada capa.
    - Definir la estructura de cada capa.
    - Especificar la interfaz de cada capa.
    - Especificar el método de comunicación intercapas.
    - Definir el esquema para el manejo de errores.
  - **Análisis:**
    - **Ventajas:**
      - Componentes estandarizados.
      - Cambios afectan el nivel local.
      - Reutilización de capas/componentes.
    - **Desventajas:**
      - Cambios afectan en cascada.
      - Ineficiencia.
      - Complejo de definir.

B. Tubos y filtros (Pipes and Filters)

- **Estructura:** Aplicaciones en actividades para procesar flujos de datos donde cada actividad es un filtro unido por un tubo a los filtros contiguos.
- **Contexto:** Procesar flujos de datos.
- **Requerimiento:**
  - Descomponer el procesamiento en una serie de actividades (filtros) que transforman datos de entrada en datos de salida.
  - Transformaciones independientes y sin estado.
- **Solución:**
  - **Tubos (pipes):**
    - Conecta origen de datos con un filtro, filtro con filtro, y filtro con salida de datos.
    - Esquema de procesamiento FIFO.
  - **Filtros (filters):**
    - Aplica procesos de transformación de datos de entrada en datos de salida.
    - Filtros independientes sin estado compartido y desconocimiento de otros filtros.
  - **Implementación:**
    - Dividir el sistema en una secuencia de procesos ordenados e independientes.
    - Definir el formato de los datos transmitidos por los tubos.
    - Especificar el procesamiento de cada filtro.
    - Construir los filtros.
    - Definir el esquema para el manejo de errores.
  - **Análisis:**
    - **Ventajas:**
      - Arquitectura flexible.
      - No requiere de archivos intermedios.
      - Filtros reutilizables.
      - Procesamiento paralelo.
      - Construcción independiente.
    - **Desventajas:**
      - Información no compartida.
      - Conversión de datos (ineficiencia).
      - Errores pueden afectar el flujo de procesamiento.

C. Pizarrón

- Patrón útil cuando
  - No hay una solución completa y específica para un problema.
  - Participan varios sistemas que aportan su conocimiento.
  - Ejemplos: inteligencia artificial, reconocimiento de imagenes, toma de decisiones.
- **Contexto:** Dominio en el que no hay una solución completa y específica para un problema.
- **Problema:**
  - Conocimiento parcial de la solución.
  - Cada solución requiere diferentes paradigmas.
  - El problema abarca muchas especialidades.
  - No es factible una solución completa.
  - Módulos aportan parcialmente a la solución.
- **Solución:**
  - Conjunto de sistemas independientes trabajando colaborativamente.
  - Datos compartidos en un repositorio centralizado.
  - Control centralizado que coordina la ejecución de los sistemas.
  - Sistemas especializados independientes leen y escriben en el pizarrón.
  - Monitoreo centralizado del estado del sistema.
  - Decisión centralizada de las acciones a seguir basada en el progreso alcanzado.

D. Repositorio

---

2. Sistemas interactivos

A. Modelo Vista Controlador (MVC)

- **Qué es**
  - El sistema se divide en tres partes: Modelo, Vista, y Controlador.
  - Modelo: Datos y funcionalidad esencial.
  - Vista: Comunicación con el usuario.
  - Controlador: Controla cambios al modelo.
  - Interfaz de usuario: Vista + Controlador.
  - Lógica del negocio: Controlador + Modelo.

        - Controlador desacopla la vista del modelo.
- **Contexto:** Sistemas interactivos con interfaz flexible.
- **Requerimiento:**
    - Interfaz con diferente representación.
        - Texto, gráficos, listas, iconos
    - Paradigmas de ingreso diversos.
        - Digitación: cajas de texto
        - Selección: listas desplegables, iconos
        - Ingreso mixto
    - Interfaz cambiante: mejora, evolución.
    - Facilidad de modificación de la interfaz.
    - Funcionalidad nueva implica modifica interfaz.
    - Presentación de la información en multi-formato.
- **Solución:**
    - Tres componentes:
        - **Comunicación (Vista):** Envía requerimientos del usuario y recibe datos del modelo.
        - **Administración (Controlador):** Define el comportamiento del sistema y solicita servicios al modelo.
        - **Procesamiento (Modelo):** Provee la funcionalidad requerida por las vistas.
    - **Implementación**
        - Separa la funcionalidad de la interacción del usuario
        - Diseñar e implementar el modelo, las vistas, los controladores, y las relaciones entre vistas y controladores.
    - **Análisis:**
        - **Ventajas:**
            - Modelo soporta múltiples vistas.
            - Flexible, mantenible, adaptable.
            - Frameworks implementan MVC.
        - **Desventajas:**
            - Modelo acoplado con vistas y controladores.
            - Vistas sin acceso a los datos (ineficiencia).
            - Complejidad.

B. Presentación Abstracción Control (PAC)

- **Estructura:** Jerárquica de agentes cooperativos, cada uno responsable de una parte de la funcionalidad.
- **Contexto:** Sistemas interactivos desarrollados utilizando agentes.
- **Requerimiento:**
  - Estructurar un sistema interactivo mediante agentes funcionando de forma integrada.
  - Generar interfaces flexibles de usuario.
  - Separar la presentación de la funcionalidad.
- **Solución**
  -
    - Definir estructura jerárquica de tres niveles de agentes.
      - Alto nivel: Funcionalidad central del sistema.
      - Bajo nivel: Manejo de interfaces específicas de usuarios.
      - Intermedios: Relacionan agentes de bajo nivel.
  - Cada agente está compuesto por:
    - **Presentación:** Aspecto visible del agente.
    - **Abstracción:** Modelo de datos interno y operaciones sobre ellos.
    - **Control:** Conexión entre presentación y abstracción, y comunicación con otros agentes.
  - **Implementación:**
    - Definir la funcionalidad central del sistema.
    - Estructurar la jerarquía de agentes.
    - Definir e implementar cada agente
      - Funcionalidad
      - Interfaz
      - Modelo de datos
      - Mecanismo de control
  - **Análisis:**
    - **Ventajas:**
      - Asigna responsabilidades específicas.
      - Funcionamiento independiente.
      - Soporta multitarea.
    - **Desventajas:**
      - Sistema complejo.
      - Baja eficiencia.
      - Complejo mecanismo de control.

3. Patrones adaptables

4. Sistemas distribuidos