COM350

Estilos Arquitectónicos

Contenido

Estilos arquitectónicos

- Un estilo arquitectónico es una descripción de los tipos de componente y conector y un patrón de control de tiempo de ejecución y / o transferencia de datos.
- Ejemplos :
  - Programa principal con subrutinas
  - abstracción de datos
  - invocación implícita
  - tubos y filtros
  - repositorio (pizarra)
  - capas de abstracción

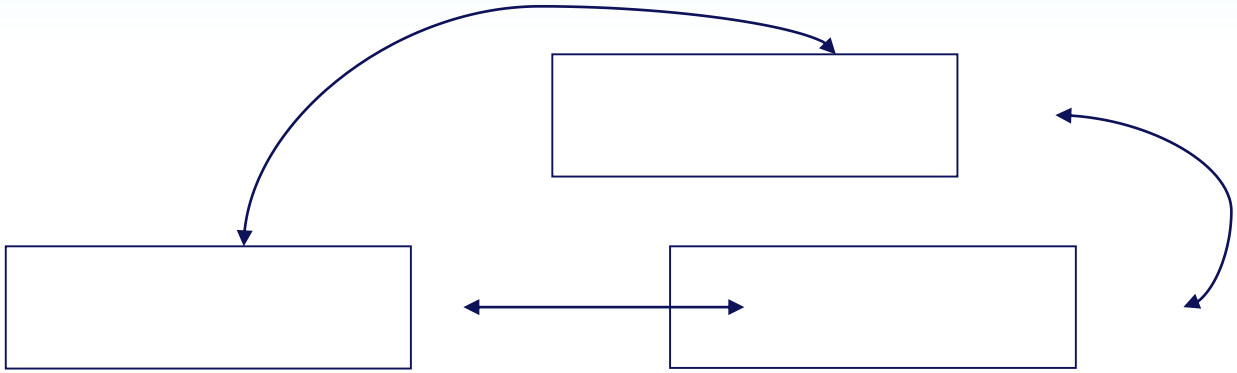
Patrones de Alexanders

- Hay evidencia de abundancia para mostrar que los edificios altos enloquecen a las personas.
- Los edificios altos no tienen una ventaja genuina, excepto en ganancias especulativas. No son más baratos, no ayudan a crear espacios abiertos, dificultan la vida de los niños, destruyen los espacios abiertos cerca de ellos. Pero, aparte de esto, la evidencia empírica muestra que en realidad pueden dañar las mentes y los sentimientos de las personas.
- En cualquier área urbana, mantenga la mayoría de los edificios cuatro pisos o menos. Es posible que ciertos edificios superen este límite, pero nunca deben ser edificios para la vivienda humana.

Sabor general de un patrón

- SI te encuentras en <contexto>, por ejemplo <ejemplos>, con <problema>
- ENTONCES para algunos <motivos>, aplique <patrón> para construir una solución que conduzca a un <nuevo contexto> y <otros patrones>

Componentes y Conectores



- Los componentes están conectados por conectores.
- Son los bloques con los que se puede describir una arquitectura.
- Ninguna notación estándar ha surgido todavía.

Tipos de componentes

- computacional: hace un cálculo de algún tipo. P.ej. función, filtro.
- memoria: mantiene una colección de datos persistentes. P.ej. base de datos, sistema de archivos, tabla de símbolos.
- manager: contiene operaciones state +. El estado se conserva entre invocaciones de operaciones. P.ej. adt, servidor.
- controlador: rige la secuencia de tiempo de los eventos. P.ej. módulo de control, programador.

Tipos de conectores

- llamada a procedimiento (incluido RPC)
- flujo de datos (por ejemplo, tuberías)
- invocación implícita
- paso de mensajes datos compartidos (por ejemplo, pizarra o base de datos compartida)
- instanciación

Marco para describir estilos arquitectónicos

- problema: tipo de problema que aborda el estilo. Las características de los requisitos guían al diseñador en su elección para un estilo particular.
- contexto: características del entorno que limitan al diseñador, las preguntas se imponen por el estilo.
- solución: en términos de componentes y conectores (opción no independiente), y estructura de control (orden de ejecución de los componentes)
- variantes ejemplos

Estilo de programa principal con subrutinas

problema: jerarquía de funciones; resultado de descomposición funcional, hilo único de control

contexto: lenguaje con procedimientos anidados

solucion:

- modelo de sistema: módulos en una jerarquía, pueden ser débiles o fuertes, argumentos de acoplamiento / cohesión
- componentes: módulos con datos locales, así como datos globales
- conectores: llamada de procedimiento
- estructura de control: hilo único, control centralizado: el programa principal tira de las cuerdas
- variantes: OO versus no-OO

Estilo de datos abstractos

problema: identificar y proteger los cuerpos de información relacionados. Representaciones de datos susceptibles de cambiar.

context: OO-métodos que guían el diseño, OO-languages que proporcionan el concepto de clase

solucion:

- modelo de sistema: el componente tiene sus propios datos locales (= secreto que oculta)
- componentes: gerentes (servidores, objetos, adt)
- conectores: llamada de procedimiento (mensaje)
- estructura de control: hilo único, por lo general; el control esta descentralizado

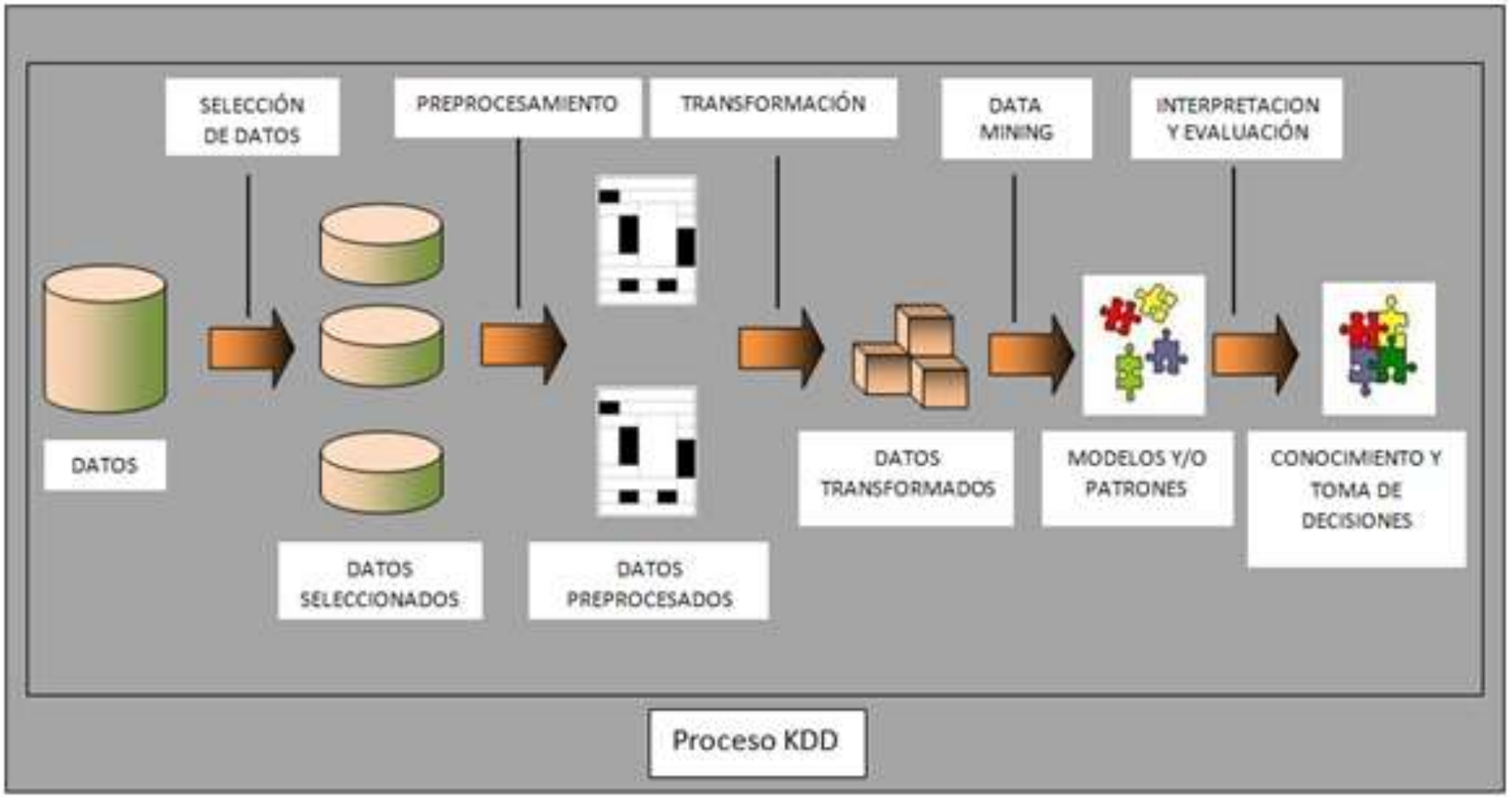
variantes: causado por las instalaciones de lenguaje

Estilo de invocación implícita

- problema: una colección de componentes débilmente acoplada. Útil para aplicaciones que deben ser reconfigurables.
- contexto: equiere controlador de eventos, a través del sistema operativo o el lenguaje.
- solucion:
  - system model: independent, reactive processes, invoked when an event is raised
  - components: processes that signal events and react to events
  - connectors: automatic invocation
  - control structure: decentralized control. Components do not know who is going to react.
- variantes: Marcos de integración de herramientas e lenguajes con características especiales..

Estilo de tubos y filtros

- problema: Transformaciones independientes y secuenciales en datos ordenados. Por lo general, incrementales, tuberías de Ascii.
- contexto: serie de transformaciones incrementales. Las funciones OS transfieren datos entre procesos. Manejo de errores difícilsolucion:
  - modelo de sistema: flujo continuo de datos; los componentes transforman los datos de forma incremental
  - componentes: filtros para procesamiento local
  - conectores: flujos de datos (generalmente ASCII simple)
  - estructura de control: flujo de datos entre componentes; componente tiene flujo propio
- varianets:Desde filtros puros con poco estado interno hasta procesos por lotes


SELECCIÓN DE DATOS
PREPROCESAMIENTO
TRANSFORMACIÓN
DATA MINING
INTERPRETACION Y EVALUACIÓN
DATOS
DATOS SELECCIONADOS
DATOS PREPROCESADOS
DATOS TRANSFORMADOS
MODELOS Y/O PATRONES
CONOCIMIENTO Y TOMA DE DECISIONES
Proceso KDD

Estilo de repositorio

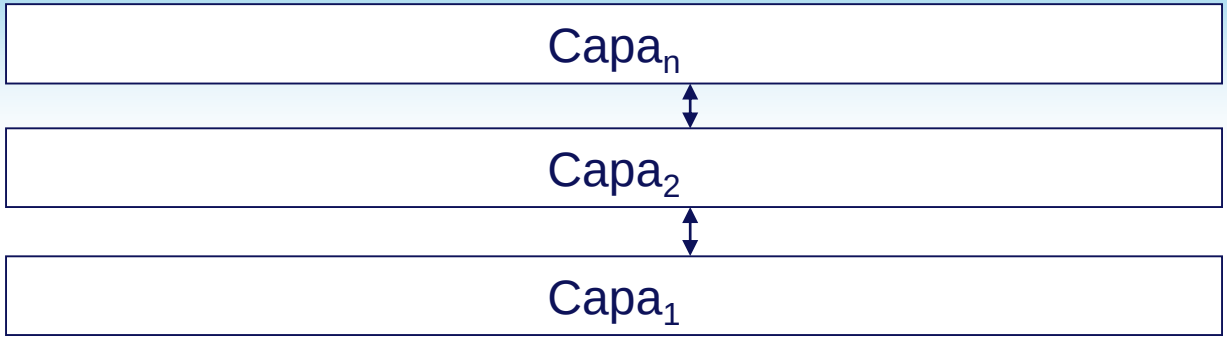
problema: administre información ricamente estructurada, para ser manipulada de muchas maneras diferentes. Los datos son de larga vida.

contexto: datos compartidos sobre los que deben actuar varios clientes

solucion:

- modelo de sistema: cuerpo centralizado de información. Elementos computacionales independientes.
- componentes: una memoria, muchos computacionales
- conectores: acceso directo o llamada de procedimiento
- estructura de control: varía, puede depender de la entrada o el estado de computación
- variantes:
  - sistemas de bases de datos tradicionales, compiladores, sistemas de pizarra

Estilo de capas



problema:

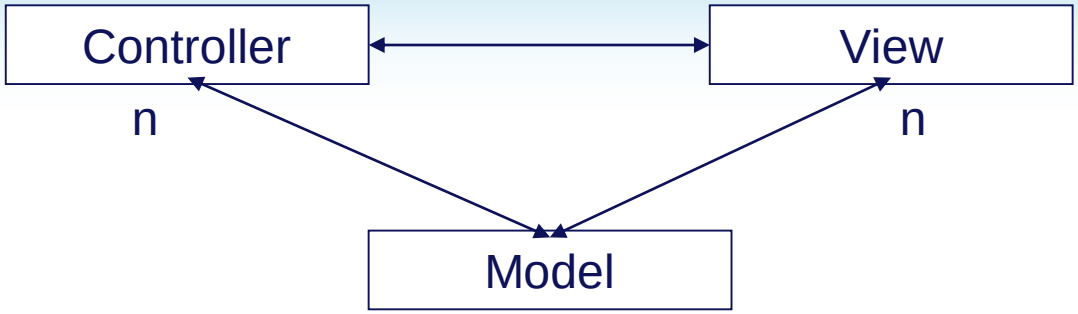
clases de servicios distintas y jerárquicas. "Círculos concéntricos" de funcionalidad

contexto: un sistema grande que requiere descomposición (por ejemplo, máquinas virtuales, modelo OSI)

solutcion:

- modelo de sistema: jerarquía de capas, a menudo visibilidad limitada
- componentes: colecciones de procedimientos (módulo)
- conectores: llamadas de procedimiento (limitadas)
- estructura de control: hilos simples o múltiples
- variantes: capas relajadas

Estilo Modelo-Vista-Controlador (MVC)



- problema: la separación de UI de la aplicación es deseable debido a las adaptaciones esperadas de UI

contexto: aplicaciones interactivas con una interfaz de usuario flexible

solucion:

- modelo de sistema: UI (Vista y Componente (s) del Controlador) está desacoplado de la aplicación (componente del Modelo)
- componentes: colecciones de procedimientos (módulo)
- conectores: llamadas de procedimiento
- estructura de control: solo hilo varia
- Variantes: Vista de Documento