



Análisis y Diseño de Sistemas

**Tema: Diseño de Sistemas.
Conceptos Básicos**

Urciuolo A.

UNPSJB – 2013

Diseño de sistemas

En cualquier actividad humana, el diseño:

- Provee una estructura para cualquier artefacto complejo
- Descompone un sistema en partes, asignando responsabilidades para cada parte y se asegura coherencia entre las partes para alcanzar juntas un objetivo común
- Algunos principios son generales (cómo descomponer un sistema en partes, propiedades de las partes, etc.) y otros son específicos del dominio.

Diseño de sistemas

El diseño de SW:

- Es un proceso iterativo mediante el cual los requerimientos se traducen en un “plano” para construir el SW (Pressman)
- Debe representar todos los elementos del Modelo del Análisis
- Constituye una guía para la generación de código
- Proporciona una imagen completa del SW (datos, comportamiento, interfaces,...)

Diseño de sistemas software

El diseño de SW:

- *Evaluación de las distintas soluciones alternativas y la especificación de una solución detallada de tipo informático.*
 - Se debe decidir a partir de las distintas alternativas.
 - La elección la hacen propietarios del sistema y equipo técnico

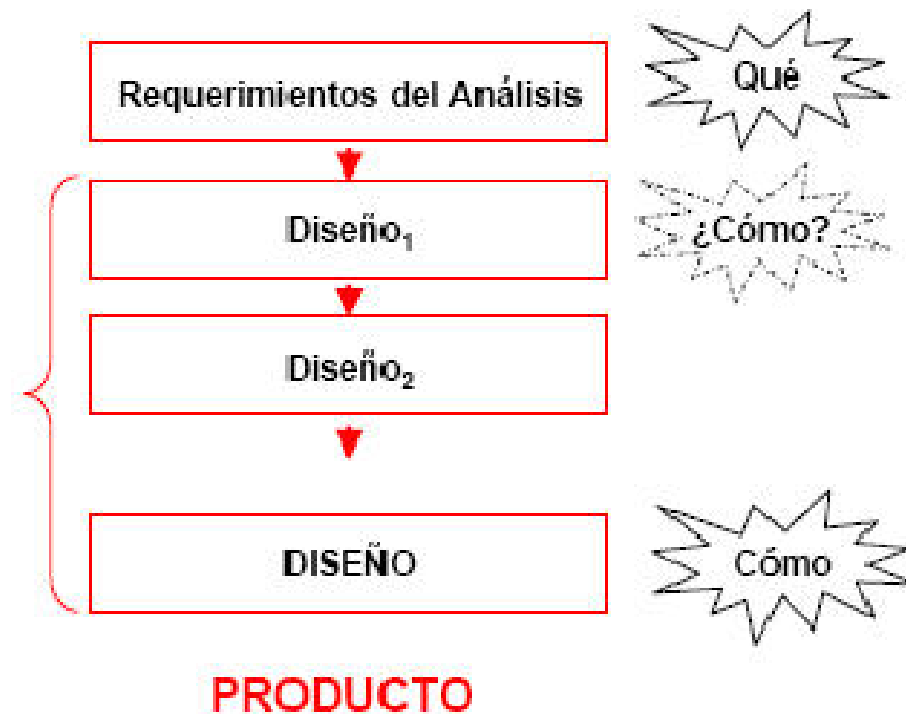
Diseño en el PDSW

- ***Diseño*** es una fase del proceso del desarrollo de software que transforma los requerimientos del sistema a través de etapas intermedias en un producto completo
- ***El Diseño del Software*** es la salida del proceso que
 - Muestra la descomposición del Sistema en partes (“módulos”)
 - Describe lo que debe hacer cada parte y las relaciones entre partes
 - Asegura que las partes “encajan” para alcanzar la meta global.
- “Diseño” refiere tanto a la actividad como a los resultados de la actividad.

Diseño en el PDSW

- El Diseño es una actividad iterativa
 - Comienza en un punto inicial
 - Evoluciona e itera agregando características/capacidades
 - Alcanza un estado de equilibrio basado en:
 - Metas del diseño
 - Interacción con el usuario

Diseño en el PDSW



Actividad de Diseño de software.

- Contexto de aplicación del Diseño de Software:
 - Hace de puente entre los requerimientos y la implementación
 - Le brinda una estructura al artefacto de software
- Actividad del diseño
 - Se define la descomposición del sistema en módulos
 - Produce un documento de Diseño de software
 - Describe la Descomposición del sistema en módulos y los diferentes modelos del diseño.
 - A menudo se produce primero la Arquitectura del software (Diseño de alto nivel).

Arquitectura de software

- Diseño de alto nivel: Arquitectura de software
 - Organización global del sistema a definir
 - Descripción de:
 - Componentes principales del sistema
 - Relaciones entre dichos componentes
 - Justificación para la descomposición en componentes
 - Restricciones a respetar para cualquier diseño de los componentes.
 - Guía el desarrollo del diseño

Diseño de Software. Modelos

- El objetivo del diseño consiste en llegar a un producto final con calidad
- Salidas: el diseño del sistema que incluye
 - Diseño de la *arquitectura*
 - Diseño *detallado* (diseño de módulos, funciones)
 - Diseño de las *estructuras de datos*
 - Diseño de las *interfaces*

Modelos de Diseño

- **Diseño Arquitectural:** se identifican los *subsistemas* que forman el sistema. Se define la relación entre los mismos, los patrones de diseño y las restricciones que afectan a la manera en que se pueden aplicar los patrones.
- **Diseño de Datos:** transforma el modelo del dominio de información del análisis en las *estructuras de datos* que se necesitan para implementar el software.

Modelos de diseño

- **Diseño de Interfaz:** describe *la manera de comunicarse* con el software, con sistemas que interoperan dentro de él y con las personas que lo usan.
- **Diseño a nivel de componente (procedimientos, objetos, etc.):** se asignan servicios a los distintos componentes y se diseñan sus interfaces.

Principios del Diseño de SW

- Se deben tener en cuenta enfoques alternativos
- Debe ser rastreable hasta los requerimientos
- Uso de patrones (No inventar lo que ya está inventado)
- Minimizar la distancia entre el problema y el SW
- Debe presentar uniformidad e integración
- Debe estructurarse para admitir cambios
- Diseñar no es escribir código
- Debe evaluarse mientras se va creando
- Debe revisarse para evitar errores conceptuales

Descomposición modular

- El Software se elabora y refina incrementalmente hasta el nivel de detalle suficiente para soportar la implementación
 - El Sistema se separa en Módulos (Componentes, Packages, Clases, etc.)
- **Módulo:** es un componente bien definido de un sistema de software. Es un proveedor de un recurso o servicio computacional
- Interesa definir las relaciones entre los módulos:
 - Para asignar tareas
 - Para controlar el desarrollo

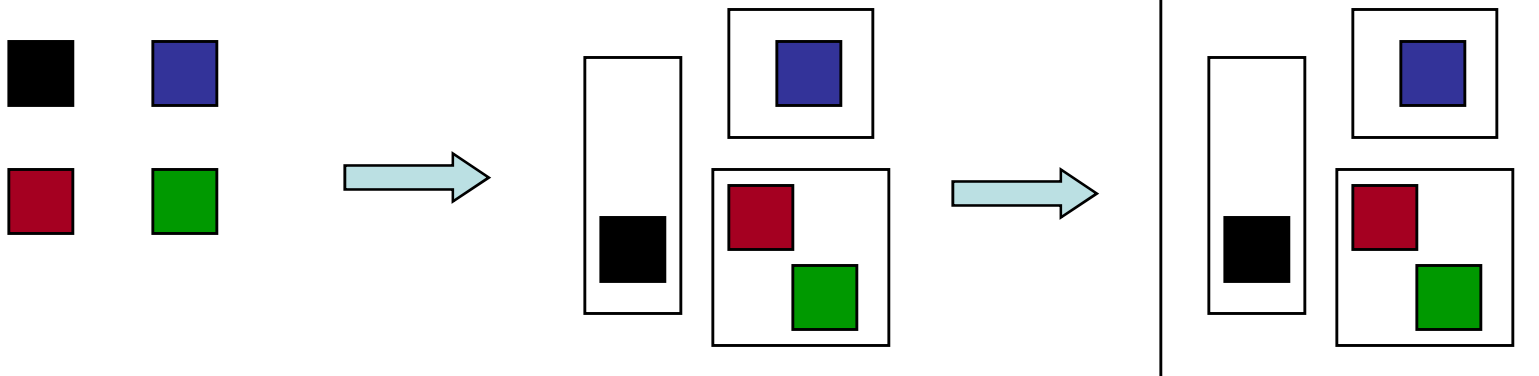
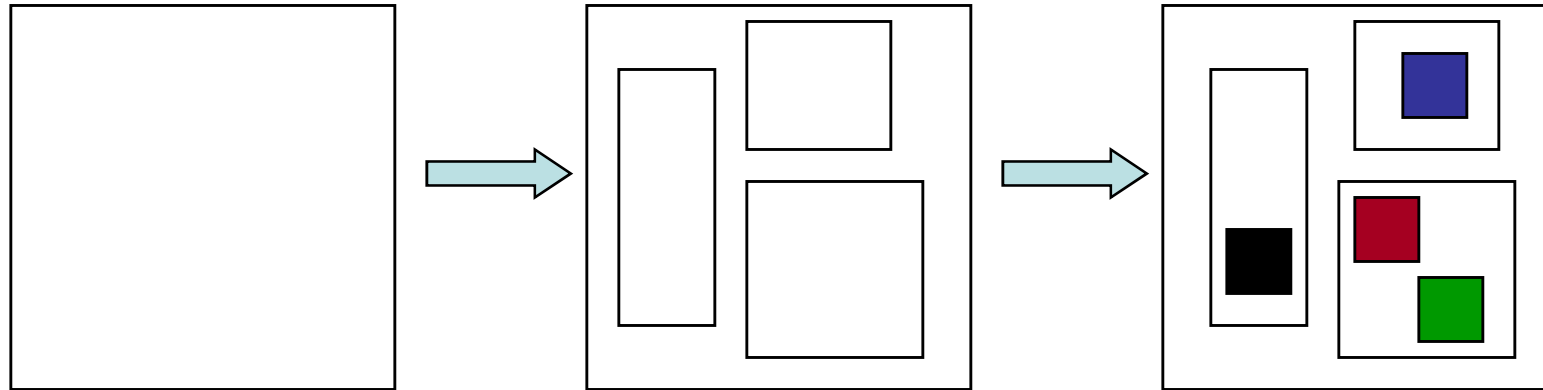
Subsistemas y módulos

- Un **Subsistema** es un sistema en sí mismo que no depende de otros. Se componen de módulos y tienen interfaces que usan para comunicarse con otros subsistemas.
- Un **Módulo** es normalmente un componente de un subsistema que proporciona uno o más servicios a otros módulos. No se suele considerar como un sistema independiente. Lo conforma componentes del sistema más simples.

Descomposición modular

- Descomposición **Top-Down**
 - Grandes Módulos se descomponen en módulos más pequeños
 - Los módulos más pequeños implementan los mayores
 - Los módulos mayores están compuestos de los más pequeños
- Descomposición **Bottom-Up**
 - Se definen primero los módulos de menor nivel
 - Estos módulos se combinan formando módulos mayores
- TP y BU alcanzan el mismo resultado

Descomposición modular



Estrategias de descomposición

- **Descomposición Orientada a Objetos:** Se descompone el sistema en un conjunto de objetos que se comunican.
- **Descomposición orientada a flujos de funciones:** Se descompone un sistema en módulos funcionales que aceptan datos y los transforman en flujos de salida.
- **Descomposición en componentes:** Unidades independientes de sw con interfaces bien definidas que se ensamblan.

Diseño Modular efectivo

- Un diseño modular reduce la complejidad, facilita los cambios y resulta en una implementación más sencilla
- **Independencia funcional:** Módulos independientes son más fáciles de programar, mantener y probar. Se mide mediante dos criterios:
 - **Cohesión**
 - **Acoplamiento**

Cohesión y acoplamiento

- Un módulo tiene **máxima cohesión** si todos sus elementos están fuertemente relacionados. Cooperan para lograr un objetivo común.
- La cohesión es una propiedad interna del módulo.
- **El acoplamiento** mide la interdependencia entre módulos
- Un buen diseño debe:
 - **maximizar la cohesión y**
 - **minimizar el acoplamiento**

Metas importantes de Diseño

- Meta 1: Diseño para el cambio
- Meta 2: Familias de productos
- Ver en Libro: “Fundamentals of Software Engineering”. Ghezzi, 2003 Cap. Design and software architecture

Tipos de Cambios frecuentes

- Algoritmos
- Cambios en la representación de datos
- Cambios en la máquina abstracta subyacente
- Cambios de dispositivos periféricos
- Cambios del entorno social
- Cambios debidos al proceso de desarrollo
- Diferentes versiones del mismo sistema

Decisiones de Diseño

- Organizar el sistema en subsistemas (*Diseño arquitectónico*).
- Identificar la concurrencia.
- Asignar subsistemas a procesadores.
- Elegir enfoque para implementación de almacenamientos.
- Administrar acceso a recursos generales.
- Elegir implementación de control.
- Manejar condiciones límites.
- Setear balance de prioridades.

Decisiones de Diseño: Subsistemas

- El primer paso de diseño es *dividir el sistema en un número pequeño de componentes*.
- Cada componente principal es un *subsistema*.
- Cada *subsistema engloba aspectos del sistema que comparten propiedades comunes*:
 - Funcionalidad, ubicación física, ejecución en el mismo hardware.
- Por ej. en AOO un subsistema es un paquete de clases, asociaciones, operaciones, eventos y restricciones interrelacionados *con una interface razonable y pequeña*.

Decisiones de Diseño: Subsistemas

- Un subsistema se identifica por los *servicios* que provee: grupo de funciones relacionadas que comparten un propósito común.
- Cada subsistema tiene interfaces bien definidas.
- En menor nivel de abstracción se definen *módulos*.
- La relación entre dos subsistemas puede ser:
 - **Cliente-servidor**
 - **Par a par** (peer to peer)

Decisiones de Diseño: Subsistemas

- **Cliente-Servidor:** el cliente invoca al servidor para que realice un servicio y este responde con un resultado.
 - *El cliente conoce las interfaces del servidor, no a la inversa.*
- **Par a Par:** cada subsistema puede invocar a los demás.
 - *Cada subsistema debe conocer las interfaces de los demás.*

Decisiones de Diseño: Concurrency

- Identificar el procesador dónde serán implementados los objetos.
- Identificar objetos concurrentes.
- Identificar objetos mutuamente excluyentes y agruparlos en threads o tareas.

Bibliografía

- ✓ "Fundamentals of Software Engineering". Carlo Ghezzi.
- ✓ "Ingeniería del Software" – Un enfoque Práctico"
R.Pressman.
- ✓ "Software Engineering" . Ian Sommerville.