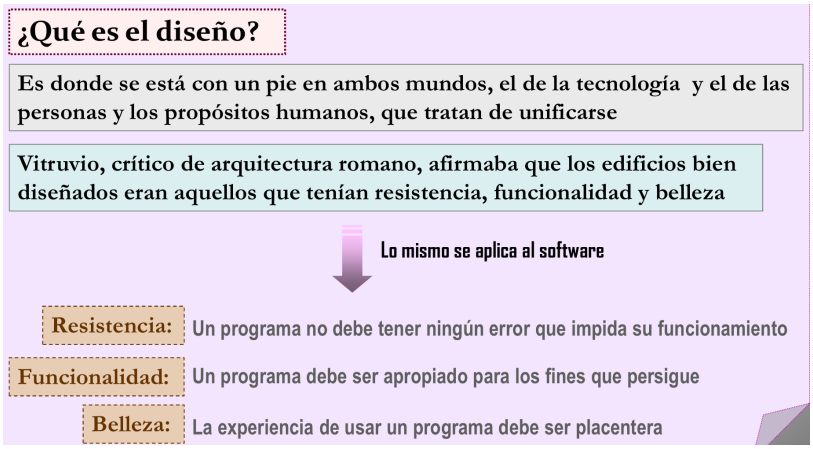
FUNDAMENTOS DE INGENIERÍA DEL SOFTWARE
Soledad Ruiz Gómez 3ºDGIIM

TEMA 3: Diseño e implementación

TEMA 3.1: Fundamentos del diseño de software

A partir de lo que hemos obtenido en los temas 1 y 2, vamos a diseñar cómo funcionan las operaciones.



1. DEFINICIÓN Y CARACTERÍSTICAS

El diseño es el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física.

El diseño de software es el proceso de aplicar distintas técnicas y principios de diseño con el propósito de traducir el modelo de análisis a una representación del software (modelo de diseño) que pueda codificarse.

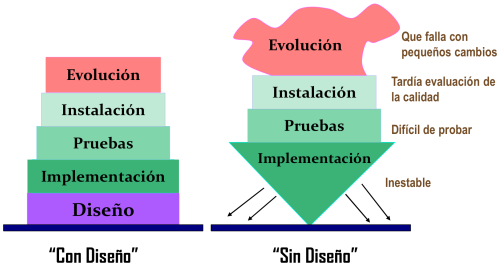
El diseño se puede definir como (1) el proceso para definir la arquitectura, los componentes, las interfaces y otras características de un sistema o un componente y (2) el resultado de este proceso.

- Un componente es una parte funcional de un sistema que oculta su implementación proporcionando su realización a través de un conjunto de interfaces.
- Una interfaz describe la frontera de comunicación entre dos entidades software, definiendo explícitamente el modo en que un componente interacciona con otros.

El proceso de diseño se descompone en dos subprocesos:

- Diseño arquitectónico: se describe cómo descomponer el sistema y organizarlo en los diferentes componentes (arquitectura SW)
- Diseño detallado: se describe el comportamiento específico de cada uno de los componentes SW

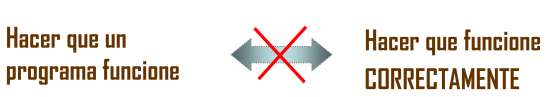
El diseño es una tarea fundamental y crítica:



Algunas de sus características son:

- El diseño implica una propuesta de solución al problema especificado en el análisis
- Es una actividad creativa que se apoya en la experiencia del diseñador
- Está apoyado por principios ("axiomas"), técnicas, herramientas, ...
- Es un proceso clave para la calidad del producto SW
- Es la base para el resto de etapas del desarrollo
- Es un proceso de refinamiento
- El diseño va a garantizar que un programa funcione correctamente

Recordar que la validación es ver que funciona correctamente, y la verificación ver que funcione.



2. PRINCIPIOS DEL DISEÑO

Ayudan a responder a las siguientes preguntas:
- ¿Qué criterios se usan para dividir el software en sus componentes individuales?
- ¿Cómo se extraen los detalles de una función o estructura de datos a partir de la representación conceptual del software?
- ¿Cuáles son los criterios que definen la calidad técnica de un diseño software?

Estudiamos los distintos principios a continuación:

2.1. ABSTRACCIÓN

Mecanismo que permite determinar qué es relevante y qué no lo es en un nivel de detalle determinado, ayudando a obtener la modularidad adecuada para ese nivel de detalle.

Los distintos tipos de abstracciones son:
- **Abstracción de datos**: define un objeto compuesto por un conjunto de datos
- **Abstracción de control**: define un sistema de control sin describir información sobre su funcionamiento interno
- **Abstracción procedimental**: se refiere a la secuencia de pasos que conforman un proceso determinado

2.2. DIVISIÓN DE PROBLEMAS Y MODULARIDAD

La división de problemas sugiere que cualquier problema complejo puede manejarse con más facilidad si se subdivide en elementos susceptibles de resolverse u optimizarse de manera independiente (Divide y vencerás). Matemáticamente:

$C(x)$ función que define la complejidad de un problema x

$E(x)$ función que define el esfuerzo de desarrollo de un problema x

Si para dos problemas p_1 y p_2 $C(p_1) > C(p_2)$ se deduce que $E(p_1) > E(p_2)$

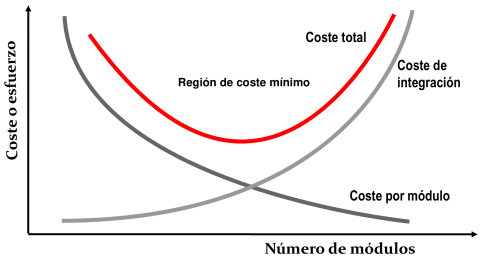
Además se cumple $C(p_1 + p_2) > C(p_1) + C(p_2)$ y que $E(p_1 + p_2) > E(p_1) + E(p_2)$

La modularidad es la manifestación más común de la división de problemas. El software se divide en componentes con nombres distintos y abordables por separado, denominados módulos, que se integran para satisfacer los requisitos del problema.

Ventajas que presenta la modularidad:
- Son más fáciles de entender y de documentar que todo el subsistema o sistema
- Facilitan los cambios

- Reducen la complejidad
- Proporcionan implementaciones más sencillas
- Posibilitan el desarrollo en paralelo
- Permiten la prueba independiente (prueba de unidad)
- Facilitan el encapsulamiento

Hay que escoger el grado adecuado de modularidad:



2.3. OCULTAMIENTO DE INFORMACIÓN

¿Cómo descomponer una solución software para obtener el mejor conjunto de módulos?

Los módulos deben especificarse y diseñarse de forma que la información (algoritmos y datos) contenida en un módulo sea inaccesible para los que no necesiten de ella.

Ventajas del ocultamiento de información:
- Reduce la probabilidad de "efectos colaterales"
- Limita el impacto global de las decisiones de diseño locales
- Enfatiza la comunicación a través de interfaces controladas
- Disminuye el uso de datos globales
- Potencia la modularidad
- Produce software de alta calidad

2.4. INDEPENDENCIA FUNCIONAL

El software debe diseñarse de manera que cada módulo resuelva un subconjunto específico de requisitos y tenga una interfaz sencilla cuando se vea desde otras partes de la estructura del programa.

La independencia se evalúa con dos criterios:

- **Cohesión**: mide lo relacionadas que están las funciones que componen un módulo. Un módulo cohesivo ejecuta una sola tarea, por lo que requiere interactuar poco con otros componentes en otras partes del programa. Idealmente hace una sola cosa.

  La ALTA COHESIÓN proporciona módulos fáciles de entender, reutilizar y mantener.

- **Acoplamiento**: mide el grado de comunicación que hay entre módulos. Indica la interconexión entre módulos en una estructura de software y depende, básicamente, de la complejidad de la interfaz entre módulos.

  El BAJO ACOPLAMIENTO proporciona módulos fáciles de entender y con menos efectos colaterales.

Hay que buscar máxima cohesión y mínimo acoplamiento.

3. HERRAMIENTAS DE DISEÑO

Son instrumentos que ayudan a representar los modelos de diseño de SW. Algunas de las más usuales son:
- Diagramas de UML: de clase, de interacción, de paquetes, de despliegue, ...
- Cartas de estructura (o diagramas de estructura)
- Tablas de decisión (o tablas de transiciones)
- Diagramas de flujo de control (u organigramas estructurados)

- Diagramas de Nassi-Shneiderman (o diagramas NS o de Chaplin)
- Lenguajes de diseño de programas (LDP o pseudocódigo)

4. MÉTODOS DE DISEÑO

Proporcionan las herramientas, técnicas y los pasos a seguir para obtener diseños de forma sistemática.

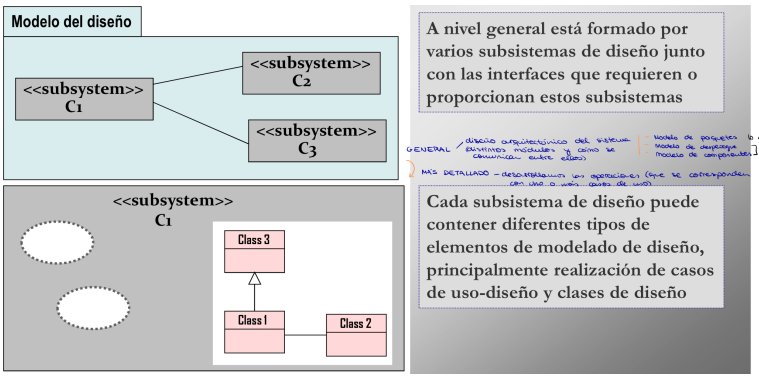
Algunas características son:
- Principios en los que se basa
- Mecanismos de traducción del modelo de análisis al modelo de diseño
- Herramientas que permiten representar los componentes funcionales y estructurales
- Heurísticas que permiten refinar el diseño
- Criterios para evaluar la calidad del diseño

Los principales métodos de diseño son:
- SSD (Diseño Estructurado de Sistemas)
- JSD (Desarrollo de Sistemas Jackson)
- ERA (Entidad-Relación-Atributo)
- OMT (Técnicas de Modelado de Objetos)
- Método Booch (Método de diseño orientado a objetos)
- Métodos orientados a objetos (La gran mayoría usan como herramienta de modelado UML y como proceso de desarrollo el PU)

5. MODELO DEL DISEÑO

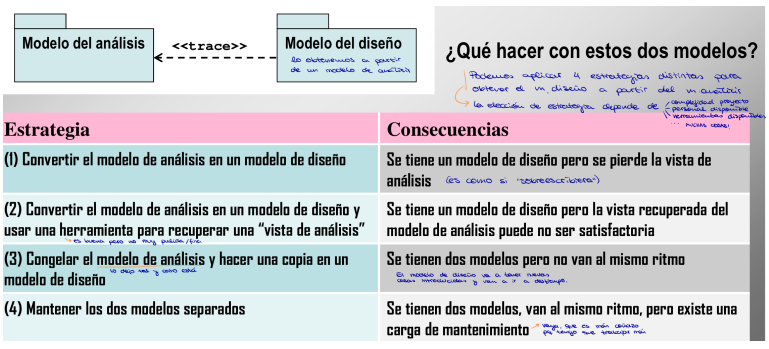
Consta de dos partes:

5.1. CONTENIDO DEL MODELO

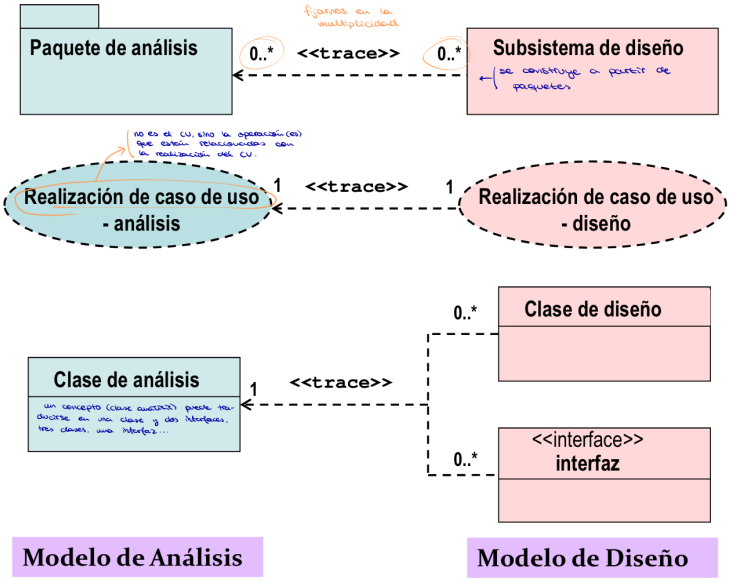


5.2. RELACIÓN CON EL MODELO DE ANÁLISIS

Se puede pensar en el modelo de diseño como una elaboración del modelo de análisis con detalles añadidos y soluciones técnicas específicas.

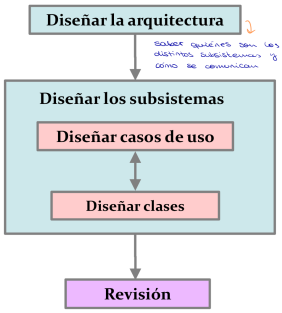


La trazabilidad entre los distintos elementos del modelo de análisis y del de diseño es:



6. TAREAS DEL DISEÑO

Consisten en:



Más detalladamente: