|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | **7** |
|  | TEMA |

Conceptos de diseño

[7.1] Diseño en el contexto de la ingeniería del *software*

[7.2] El proceso de diseño

[7.3] Conceptos de diseño

[7.4] Conceptos de diseño orientado a objetos

[7.5] El modelo del diseño

[7.6] Referencias

Esquema

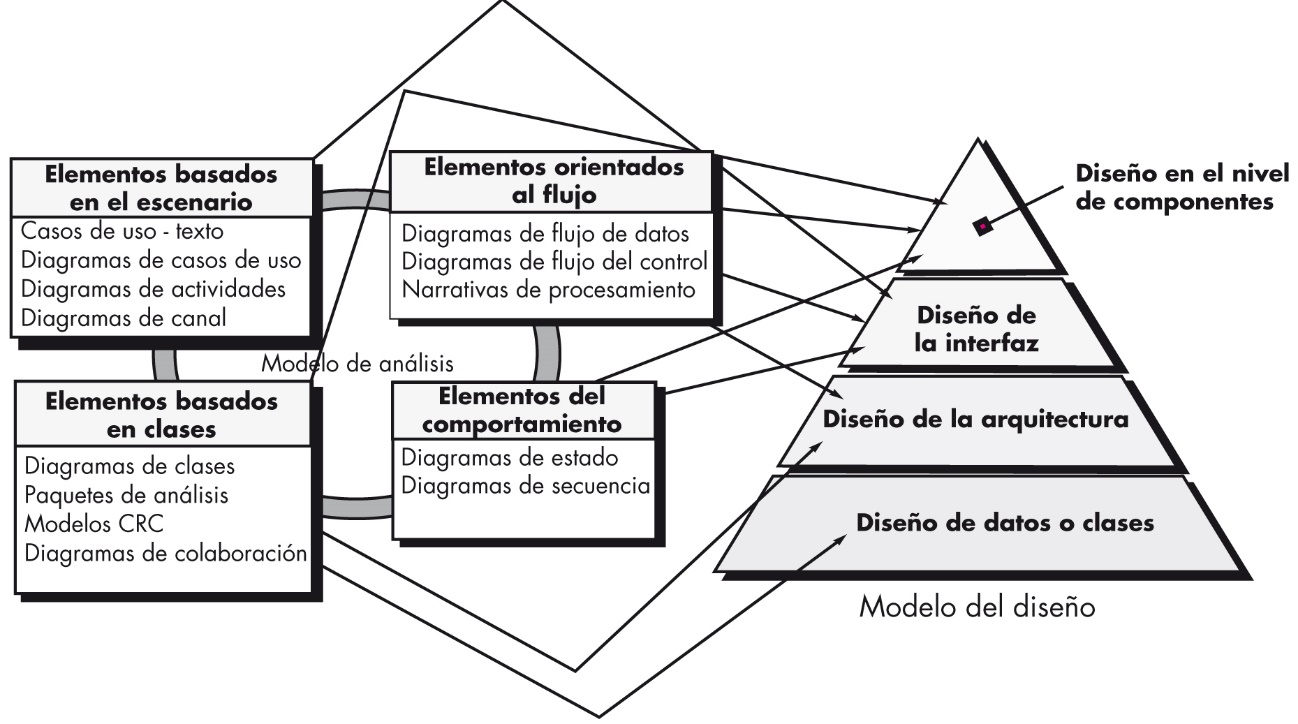


Ideas clave

7.1. Diseño en el contexto de la ingeniería de *software*

El diseño del *software* comienza tras el **análisis y modelado de requisitos** y precede a la **construcción del *software***.

El diseño del *software* se realiza a cuatro niveles: diseño de datos o clases, diseño de la arquitectura, diseño de la interfaz y diseño de componentes



Fuente: Pressman, R. S. (2010). *Ingeniería del software.   
Un enfoque práctico* (7ª ed.) (p. 185). México: McGraw Hill.

7.2. El proceso de diseño



Todo diseño *software* deberá tener como objetivo los siguientes atributos de calidad: funcionalidad, usabilidad, fiabilidad, rendimiento y mantenibilidad.

7.3. Conceptos de diseño

Conceptos importantes del diseño:

* **Abstracción**: diseño con alto nivel de abstracción será el que enuncia la solución en términos generales y, con bajo nivel de abstracción, la descripción de la solución será más detallada.
* **Arquitectura del *software***: es la estructura de los diferentes módulos del sistema, el modo en que interactúan entre ellos y la estructura de datos que utilizan.
* **Patrón de diseño**: estructura de diseño que resuelve un problema particular del diseño en un contexto específico.
* **División de problemas**: un problema complejo se resuelve con mayor facilidad si subdivide en problemas más pequeños.
* **Modularidad**: división del *software* en componentes identificables por un nombre y que pueden resolverse de forma independiente.
* **Ocultamiento de información**: los módulos deben especificarse y diseñarse de forma que la información contenida (datos y algoritmos) sea inaccesible a los que no la necesiten.
* **Independencia funcional**: cada módulo debe resolver un conjunto específico de requisitos y ofrece una interfaz sencilla para su uso.
* **Refinamiento sucesivo**: consiste en el incrementando del número de detalles de diseño del *software* a medida que se va avanzando en el diseño.
* **Rediseño**: reorganización para la simplificación del diseño de un componente sin modificar su funcionalidad.

7.4. Conceptos de diseño orientado a objetos

Hay que tener presentes conceptos ya presentados de orientación a objetos en los temas de modelado de requisitos: clase, objeto, atributo, método, herencia, polimorfismo, mensaje, etc.

7.5. El modelo del diseño

El modelado del diseño incluye cuatro elementos:

* **Elemento de diseño de la arquitectura**: utiliza información obtenida del dominio de aplicación, del modelo de requisitos y del catálogo de patrones y estilos para obtener una representación estructural completa del *software*.
* **Elementos de diseño de la interfaz**: que modelan las interfaces de la aplicación (internas, externas y de usuario).
* **Elementos de diseño a nivel de componentes**: definen los módulos que componen la arquitectura.
* **Elementos del diseño de despliegue**: que tendrá en cuenta el entorno físico en el que se implantará el sistema.

7.6. Referencias

Pressman, R. (2010). *Ingeniería del software* (7ª ed.). México: McGrawHill.

Lo + recomendado

No dejes de leer…

**Diseño de *software* propietario vs. *software* libre**

Ensayo del *hacker* Eric S. Raymond que compara la construcción de *software* propietario como la de una catedral, en la que es necesario planificar la totalidad del proyecto, diseñar toda la estructura por anticipado, calcular materiales y recursos humanos a utilizar, etc. Esto hace que cambios en fases finales del proyecto son complejos de asumir. Por otro lado, el desarrollo de *software* libre es como un bazar, es algo simple que va aumentando a medida que va teniendo éxito, el personal participa voluntariamente, etc.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:

<http://biblioweb.sindominio.net/telematica/catedral.html>

No dejes de ver…

**Entrevista a Steve Jobs**

Extracto de la entrevista realizada a Steve Jobs en la que explica el modo de trabajo en Apple.



Accede al vídeo desde el aula virtual o a través de la siguiente dirección web:

<http://www.youtube.com/watch?v=NO7QFnGK3qs>

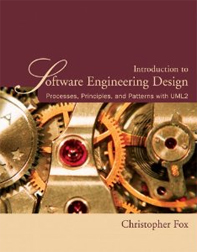
+ Información

A fondo

**Diseño de ingeniería de *software***

Christopher Fox (2006). *Introduction to Software Engineering Design: Processes, Principles and Patterns with UML 2*. Ed. Addison Wesley.

Este libro presenta el diseño de *software* destacando la parte práctica, utilizando para ello técnicas de análisis y diseño orientado a objetos y UML 2. El autor abarca todo el diseño de ingeniería de *software* con un enfoque en los procesos, principios y prácticas utilizadas para diseñar el *software*. El libro ofrece el contenido para aprender a diseñar *software* utilizando una gran cantidad de herramientas pedagógicas y ejercicios prácticos.



**Diseño de *software* de código abierto**

En este artículo de la revista Computer World nos plantea la importancia del buen diseño para garantizar la seguridad en el código abierto.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:

[http://cioperu.pe/articulo/16747/es-seguro-el-*software*-de-codigo-abierto/](http://cioperu.pe/articulo/16747/es-seguro-el-software-de-codigo-abierto/)

Recursos externos

Herramientas de *mockups*

Los *mockups* permiten realizar el diseño informal del aspecto que tendrá una aplicación para presentárselo al cliente o transmitírselo al equipo. Estos diseños informales se realizan rápidamente gracias a las herramientas existentes de este tipo. A continuación se pone el enlace a algunas de ellas.

Accede a las páginas desde el aula virtual o a través de las siguientes direcciones web:

Cacoo: <https://cacoo.com/lang/es/>

Mockflow: <http://www.mockflow.com/>

Moqups: <https://moqups.com/>

Wireframe: <https://wireframe.cc/>

Test

**1.** Indica el nombre del atributo de calidad de diseño que facilita la ampliación y adaptación futura del *software*.

A. Usabilidad.

B. Fiabilidad.

C. Rendimiento.

D. Mantenibilidad.

**2.** Un buen diseño:

A. Debe implementar todos los requisitos del modelo de requisitos.

B. Debe ser una guía legible y comprensible para los programadores.

C. Debe proporcionar una visión completa de los datos, funciones y comportamiento del *software*.

D. Todas son correctas.

**3.** La fase de diseño del *software*:

A. Precede a la fase de mantenimiento.

B. Finaliza con la entrega del código fuente al cliente.

C. Comienza una vez analizado y modelado los requisitos.

D. Puede omitirse si el modelado de requisitos se hizo con alto nivel de detalle.

**4.** ¿Qué nombre se le da a la estructura de diseño que resuelve un problema particular del diseño en un contexto específico?

A. Patrón de diseño.

B. Divide y vencerás.

C. Encapsulamiento.

D. Cohesión.

**5.** ¿Cómo se conoce a la división del *software* en componentes independientes?

A. Concurrencia.

B. Modularidad.

C. Acoplamiento.

D. Atomicidad.

**6.** El rediseño consiste en:

A. Definir el mínimo número de detalles de diseño a alto nivel.

B. Consiste en repetir sucesivamente la fase de diseño hasta que los programadores comprendan cómo tienen que construir el *software*.

C. Simplificación del diseño del *software* sin modificar su funcionalidad.

D. La encapsulación de la información para que solo sea accesible desde el exterior de un componente lo mínimo necesario.

**7.** El modelo de diseño se compone de elementos de diseño de:

A. Datos, arquitectura, interfaz, componentes y despliegue.

B. Código fuente, pruebas y documentación.

C. Código fuente, estructuras de datos y documentación.

D. Código fuente, despliegue, pruebas y documentación.

**8.** El diagrama UML de despliegue se utiliza para definir los elementos de diseño de:

A. Despliegue.

B. Pruebas.

C. Código fuente.

D. Documentación.

**9.** La fase de diseño da respuesta a la pregunta:

A. ¿Qué hay que hacer?

B. ¿Cómo lo hay que hacer?

C. ¿Cuándo lo hay que hacer?

D. ¿Dónde lo hay que hacer?