

Presentación y normas del curso

Diseño de Sistemas Software
Curso 2023/2024

Óscar Segura Amorós
Carlos Pérez Sancho



Universitat d'Alacant
Universidad de Alicante

Departament de Llenguatges i Sistemes Informàtics
Departamento de Lenguajes y Sistemas Informáticos

Datos generales

- Asignatura obligatoria de 3er curso en el Grado de Ingeniería en Informática y 4º curso en el Doble Grado en Ingeniería Informática y Administración y Dirección de Empresas
- 6 créditos ECTS (3 teoría + 3 prácticas)
- La información general de la asignatura, horarios y bibliografía recomendada están en la [guía docente](#)
- Los materiales de la asignatura están en Moodle
- Los anuncios y tutorías se realizarán a través del Campus Virtual

- **Teoría:** Óscar Segura, Carlos Pérez (coordinador)
- **Prácticas:** Antonio Javier Gallego, Jose Martínez, José Ignacio Abreu, Antonio Ríos, Carlos Pérez
- **Tutorías**
 - **Presenciales:** en el Campus Virtual se pueden consultar los horarios y reservar cita
 - **No presenciales:** a través del Campus Virtual

- Introducción al diseño de software
- Patrones de diseño
- Arquitectura de software
- Patrones arquitecturales
- Patrones GRASP
- Patrones GOF

Objetivos

- Ser capaz de realizar el diseño de un sistema de software a partir de los requisitos capturados en la fase de análisis
- Comprender los principales patrones de diseño y saber reconocerlos y aplicarlos en la práctica
- Conocer las principales arquitecturas de software y los patrones en los que se basan
- Manejar con soltura los diagramas UML más importantes para realizar y comprender diseños de software (diagramas de clases y secuencia)
- Saber evaluar de forma crítica un diseño y sus posibles alternativas, para poder seleccionar la solución más adecuada para un problema dado

Evaluación

- **(50 %) Teoría:** examen final el 14 de junio de 2024
- **(50 %) Prácticas:** tres entregas de proyecto

Semana	Contenidos	% en nota final
26 feb - 1 mar	<ul style="list-style-type: none">• Descripción del proyecto• Diagramas de diseño (v1)• Implementación del diagrama	5%
15 abr - 19 abr	<ul style="list-style-type: none">• Diagramas de diseño (v2)• Aplicación funcional sin autenticación de usuarios	15%
20 may - 24 may	<ul style="list-style-type: none">• Diagramas de diseño (v3)• Aplicación funcional	30%

- Se convalidarán las prácticas que obtuvieron nota igual o superior a 7 en la convocatoria anterior

- Para superar la asignatura se deberán obtener al menos **4 puntos sobre 10** tanto en teoría como en prácticas. La nota final de la asignatura (media de teoría y prácticas) debe ser al menos un 5 sobre 10.
- **La asistencia a prácticas es obligatoria.** En caso de tener más de 3 faltas a clase de prácticas por causa NO justificada o no debidamente acreditada, el alumno suspenderá esa parte.

Evaluación: julio

- Para la evaluación de la asignatura en julio **se guardarán las partes (teoría o prácticas) cuya nota sea al menos un 5 sobre 10 en la convocatoria ordinaria.**
- La parte teórica se recuperará mediante un examen en la fecha oficial (5 de julio de 2024)
- La parte práctica se recuperará mediante la presentación del proyecto de prácticas planteado en la convocatoria de junio.
- En esta convocatoria se evaluará mediante una única entrega, con un peso en la calificación correspondiente al 50% de la nota final.

Los trabajos teórico/prácticos realizados han de ser originales. La detección de **copia o plagio supondrá la calificación de “0”** en la prueba correspondiente. Se informará la dirección de Departamento y de la EPS sobre esta incidencia.

La reiteración en la conducta en ésta u otra asignatura conllevará la notificación al vicerrectorado correspondiente de las faltas cometidas para que estudien el caso y sancionen según la legislación (Ley 2/2022 de Convivencia Universitaria, artículo 11g).

- Se puede consultar el grupo asignado en el Campus Virtual
- NO se puede cambiar de grupo salvo causa justificada (gestión a través de la secretaría de la EPS)
- Comienzo de las prácticas: 29 de enero
- **Se realizará un proyecto en grupos de 4 personas**
- En caso de duda sobre la autoría de las prácticas, se realizará una entrevista a los miembros del grupo por separado para determinar su grado de participación

Recomendaciones

- Asistir a clase y realizar los ejercicios propuestos
- Participar y debatir las soluciones propuestas
- No pensar nunca que existe una solución única, ni que la solución propuesta por el profesor es la mejor
- Consultar al profesorado las soluciones propias en caso de duda
- Tratar de implementar las soluciones teóricas para comprenderlas mejor

Introducción al diseño de software

Diseño de Sistemas Software

- Ingeniería Software
- Diseño Software
 - Fases del diseño
 - Ventajas
 - Paradigmas de diseño
 - Herramientas

¿Qué es el diseño de software?

Es una disciplina de la Ingeniería de Software...

¿Qué es la Ingeniería de Software?

- La **ingeniería de software** es la disciplina que estudia el **desarrollo y mantenimiento** de sistemas de software **asequibles, confiables, de calidad** y que satisfagan todos los **requisitos** que los clientes han definido para ellos.

“Software engineering is the discipline of developing and maintaining software systems that behave reliably and efficiently, are affordable to develop and maintain, and satisfy all the requirements that customers have defined for them.”

(IEEE Computing Curricula 2005)

“(Students should be able to) Demonstrate an understanding of and apply appropriate theories, models, and techniques that provide a basis for problem identification and analysis, software design, development, implementation, verification, and documentation.”

(IEEE Software Engineering 2014)

¿Qué es el diseño de software?

- **Diseño de software** es tanto el **proceso de definir** la arquitectura, componentes, interfaces y otras características de un sistema, como el **resultado** de ese proceso
- Permite formar un "plan" de la solución de la aplicación, pudiendo evaluarse y mejorarse antes de generar el código

“Software design is both the process of defining the architecture, components, interfaces and other characteristics of a system, and the result of that process”.

(IEEE Computing Curricula 2005)

- **Análisis, diseño e implementación** son disciplinas centrales de la ingeniería del software
- Están **fuertemente relacionadas**, y las decisiones sobre cómo realizar una de ellas afecta directamente a las demás
- Lenguajes, herramientas, actividades a realizar, etc. deben decidirse al comienzo de un proyecto

- **Diseño explícito:** realizado para planificar o documentar el software desarrollado
- **Diseño implícito:** la estructura que el software tiene realmente, aunque no se haya diseñado formalmente

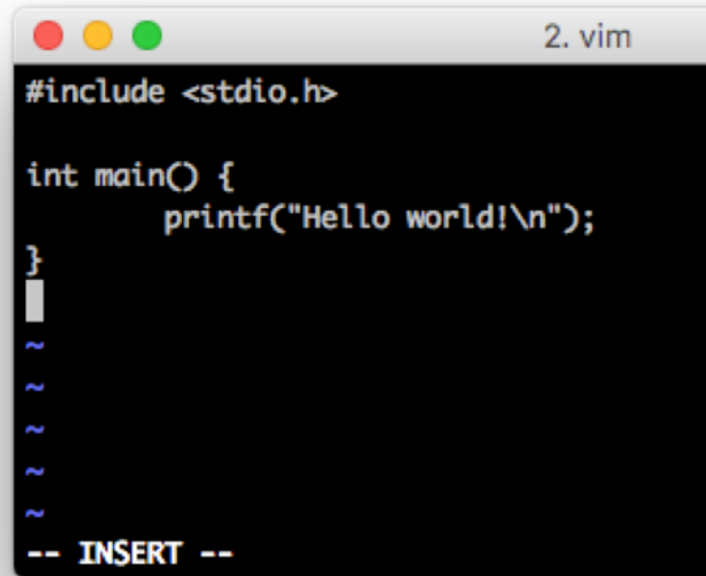
→ **Todo el software tiene un diseño**

Fases del diseño

- **Diseño arquitectural:** estructura de alto nivel, identificación de los componentes principales junto con los requisitos funcionales y no funcionales
- **Diseño detallado:** los componentes se descomponen con un nivel de detalle más fino. Guiado por la arquitectura y los requisitos funcionales

¿Por qué es importante el diseño?

¿Por qué diseñar?



```
2. vim
#include <stdio.h>

int main() {
    printf("Hello world!\n");
}

-- INSERT --
```



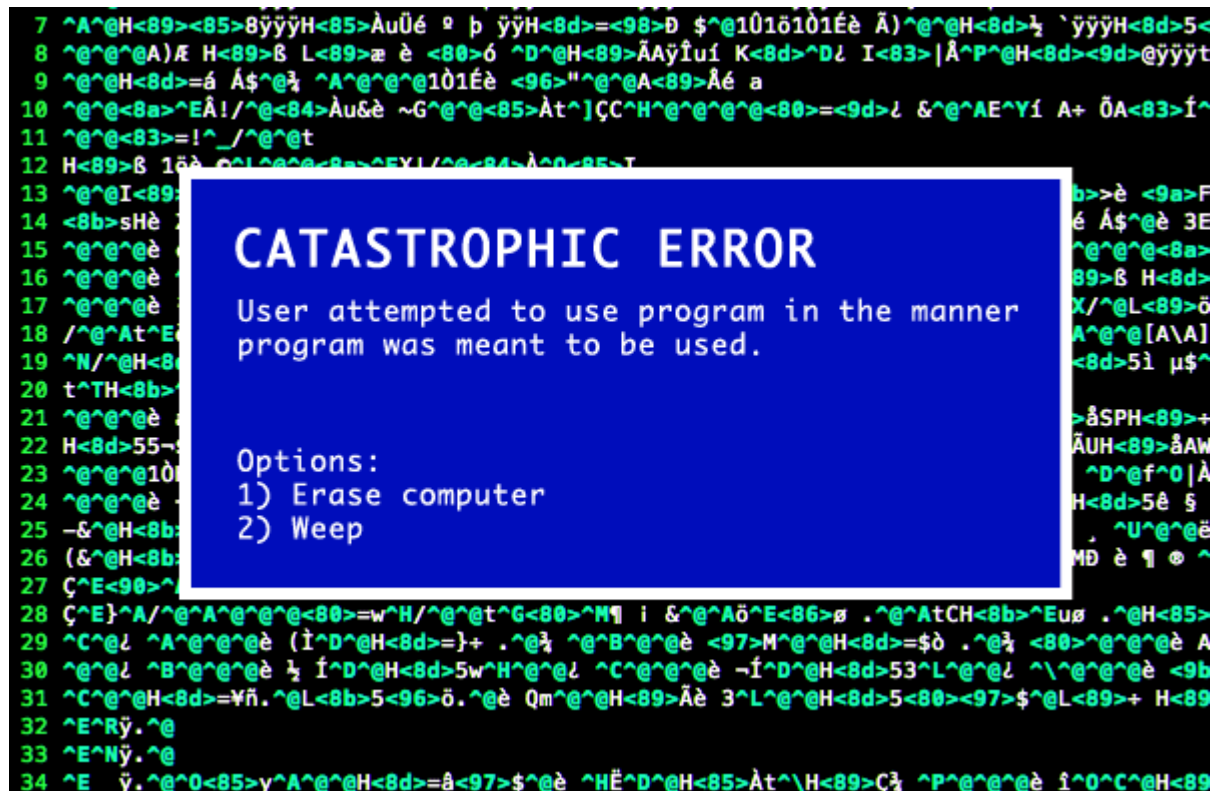
Ventajas de un buen diseño

- Ayuda a trasladar las especificaciones a los programadores de forma no ambigua



Ventajas de un buen diseño

- Ayuda a escribir código de calidad
→ conforme a las especificaciones



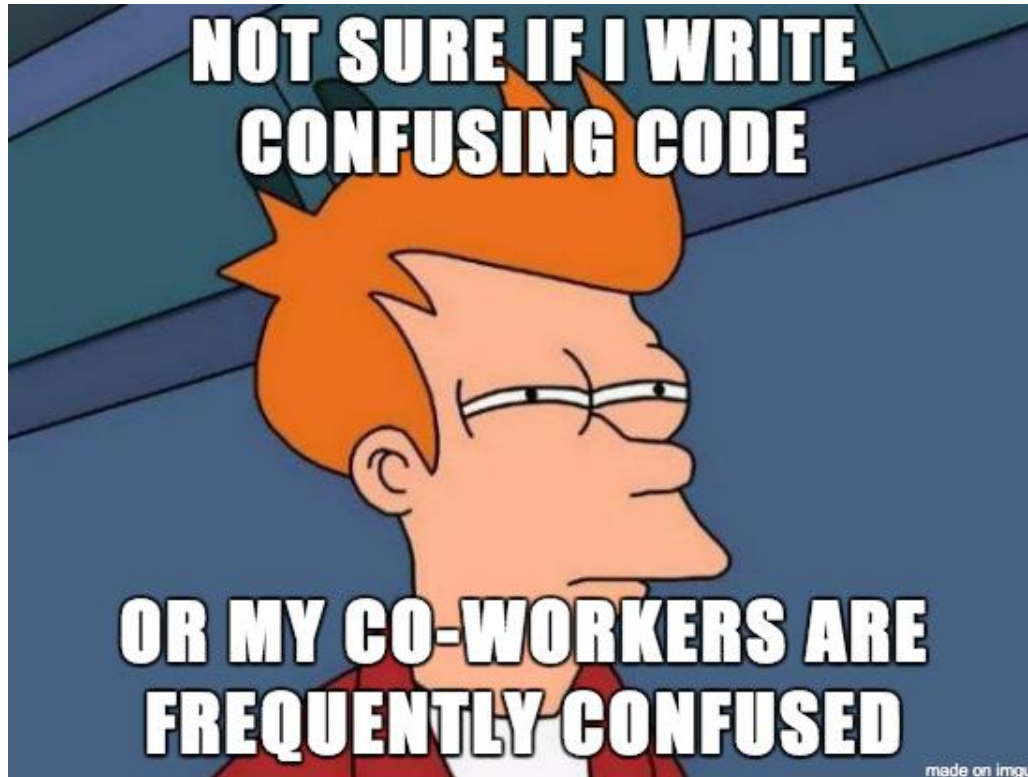
Ventajas de un buen diseño

- Ayuda a escribir código de calidad
→ fácil de mantener y extender



Ventajas de un buen diseño

- Ayuda a escribir código de calidad
→ que los demás puedan comprender



Ventajas de un buen diseño

- Ayuda a trasladar las especificaciones a los programadores de forma no ambigua
- Ayuda a escribir código de calidad
 - ... conforme a las especificaciones
 - ... fácil de mantener y extender
 - ... que los demás puedan comprender

Aumenta las probabilidades de finalizar con éxito el proyecto

Diseñar bien no es garantía de éxito

Sin embargo...

- Realizar un diseño previo no garantiza que el software resultante sea conforme a ese diseño ni a la especificación
- Existen disciplinas y técnicas para garantizar la calidad del software (p.ej. pruebas de código)

¿Cómo y cuánto hay que diseñar?

(Algunos) Paradigmas de diseño

- **Diseño orientado a objetos:** los sistemas de software se componen de objetos con un estado privado y que interactúan entre sí
- **Diseño orientado a funciones:** descompone el sistema en un conjunto de funciones que interactúan y comparten un estado centralizado
- **Diseño de sistemas de tiempo real:** reparto de responsabilidades entre software y hardware para conseguir una respuesta rápida
- **Diseño de interfaces de usuario:** tiene en cuenta las necesidades, experiencia y capacidades de los usuarios

Herramientas de Diseño Orientado a Objetos

- Tarjetas CRC (Clase-Responsabilidad-Colaboración)
 - La técnica consiste en dibujar una tarjeta por cada clase u objeto, y dividirla en tres zonas:
 - Nombre de la **clase**
 - **Responsabilidades** de dicha clase (Objetivos, a alto nivel)
 - **Colaboradores**, otras clases que ayudan a conseguir cumplir a esta con sus responsabilidades

VENTAS	
Ingresar factura de venta Modificar factura Consultar factura Eliminar factura Ingresar cliente Mostar cliente Ingresar producto Mostrar producto Calcular IVA Calcular descuento Calcular total a pagar	Clientes Productos

Herramientas de Diseño Orientado a Objetos

- UML (Unified Modeling Language), diagramas más usados según Sommerville (2010):
 - **Diagramas de actividad:** actividades involucradas en un proceso o en el procesamiento de datos
 - **Diagramas de caso de uso:** interacciones entre el sistema y su entorno
 - **Diagramas de secuencia:** interacciones entre los actores y el sistema y entre componentes del sistema
 - ➡ ◦ **Diagramas de clase:** clases de objetos y sus asociaciones
 - **Diagramas de estados:** cómo reacciona el sistema ante eventos internos y externos



¡ATENCIÓN!

UML no es más que un formato de representación

Usar UML no garantiza que el diseño sea bueno

Formas de usar UML

- Como un medio para **facilitar el debate** sobre un sistema existente o propuesto
- Como una forma de **documentar un sistema** existente
- Como una **descripción detallada** que se puede usar para **desarrollar** una implementación
- Usando herramientas especializadas, los modelos se pueden usar para **generar código automáticamente** (Model-Driven Engineering, MDD)

La metodología impone las reglas

- En metodologías tradicionales hay que generar **numerosos modelos** antes de tocar una sola línea de código
- Las metodologías ágiles recomiendan modelar lo **mínimo** imprescindible, en **grupo** y de manera **informal**
→ los modelos se usan como herramienta de comunicación

“For a three-week timeboxed iteration, spend a few hours or at most one day (with partners) near the start of the iteration at the walls...”

(Larman, 2004, ch. 14.3)

¿Preguntas?

Bibliografía

- [IEEE Computing Curricula 2005](#)
- [IEEE Software Engineering 2014](#)
- Larman, C. (2004). **Chapter 14. In Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd edition.** Addison Wesley Professional.
[Leer O'Reilly Online](#)
- Tsui, F., Karam, O., Bernal, B. (2013). **Chapters 2 & 7. In Essentials of Software Engineering, 3rd edition.** Jones & Bartlett Learning.
[Leer en O'Reilly Online](#)
- Sommerville, I. (2010). **Software Engineering, Ninth edition.** Addison-Wesley.