

Introducción al diseño de software

Diseño de Sistemas Software

Curso 2022/2023

Carlos Pérez Sancho
Ana Lavalle López



Universitat d'Alacant
Universidad de Alicante

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

- 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

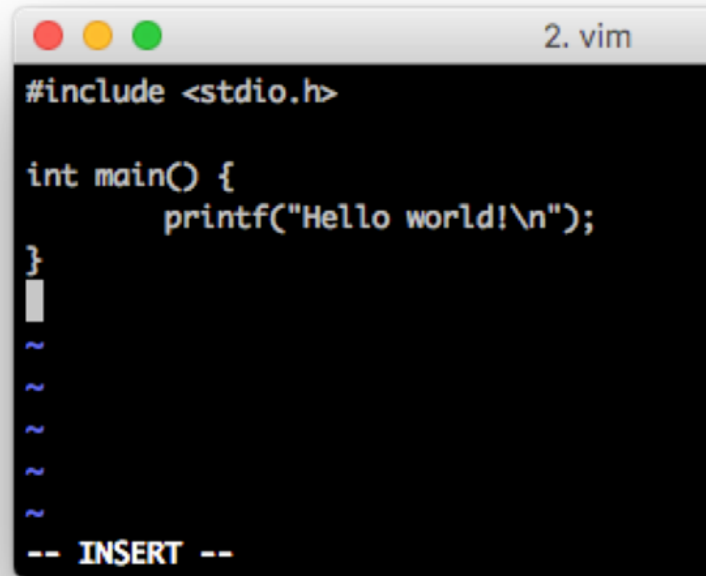
→ **Todo el software tiene un diseño**

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?

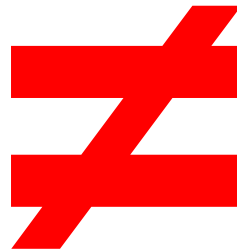
¿Por qué diseñar?



```
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

```
7 ^A^H<89><85>8ÿÿÿH<85>ÀuŮé º p ÿÿH<8d>=<98>Ð $^@10101Éè Ã)^@^H<8d>¼ ÿÿÿH<8d>5<
8 ^@^@A)Æ H<89>B L<89>æ è <80>ó ^D^H<89>ÄAÿİuİ K<8d>^DŁ I<83>|Ä^P^H<8d><9d>@ÿÿÿt
9 ^@^H<8d>=á Á$^@¼ ^A^@^@101Éè <96>"^@^@A<89>Äé a
10 ^@^@<8a>^EÄ!/^@<84>Äu&è ~G^@^@<85>Ät^]ÇC^H^@^@^@<80>=<9d>Ł &^@^AE^Yİ A+ ŌA<83>İ^
11 ^@^@<83>=!^_/^@^@t
12 H<89>B 1öè C^I^@^@<8a>^EYI/^@<84>Ä^0<85>T
13 ^@^@I<89>
14 <8b>shè
15 ^@^@^@è
16 ^@^@^@è
17 ^@^@^@è
18 /^@^At^E
19 ^N/^H<8
20 t^TH<8b>
21 ^@^@^@è
22 H<8d>55~
23 ^@^@^@10
24 ^@^@^@è
25 -&^H<8b>
26 (&^H<8b>
27 Ç^E<90>^
28 Ç^E}^A/^@^A^@^@<80>=w^H/^@^@t^G<80>^M¶ i &^@^Aö^E<86>ø .^@^AtCH<8b>^Euø .^@H<85>
29 ^C^@Ł ^A^@^@^@è (İ^D^H<8d>=})+ .^@¼ ^@^B^@^@è <97>M^@^@H<8d>=$ò .^@¼ <80>^@^@^@è A
30 ^@^@Ł ^B^@^@^@è ½ İ^D^H<8d>5w^H^@^@Ł ^C^@^@^@è -İ^D^H<8d>53^L^@^@Ł ^\^@^@^@è <9b
31 ^C^@^@H<8d>=Yñ. ^@L<8b>5<96>ö. ^@è Qm^@^@H<89>Äè 3^L^@^@H<8d>5<80><97>$^@L<89>+ H<89
32 ^E^Rÿ.^@
33 ^E^Nÿ.^@
34 ^E ÿ.^@^0<85>y^A^@^@H<8d>=â<97>$^@è ^HĖ^D^H<85>Ät^H<89>Ç¼ ^P^@^@^@è İ^0^C^H<89>
```

CATASTROPHIC ERROR

User attempted to use program in the manner
program was meant to be used.

Options:
1) Erase computer
2) Weep

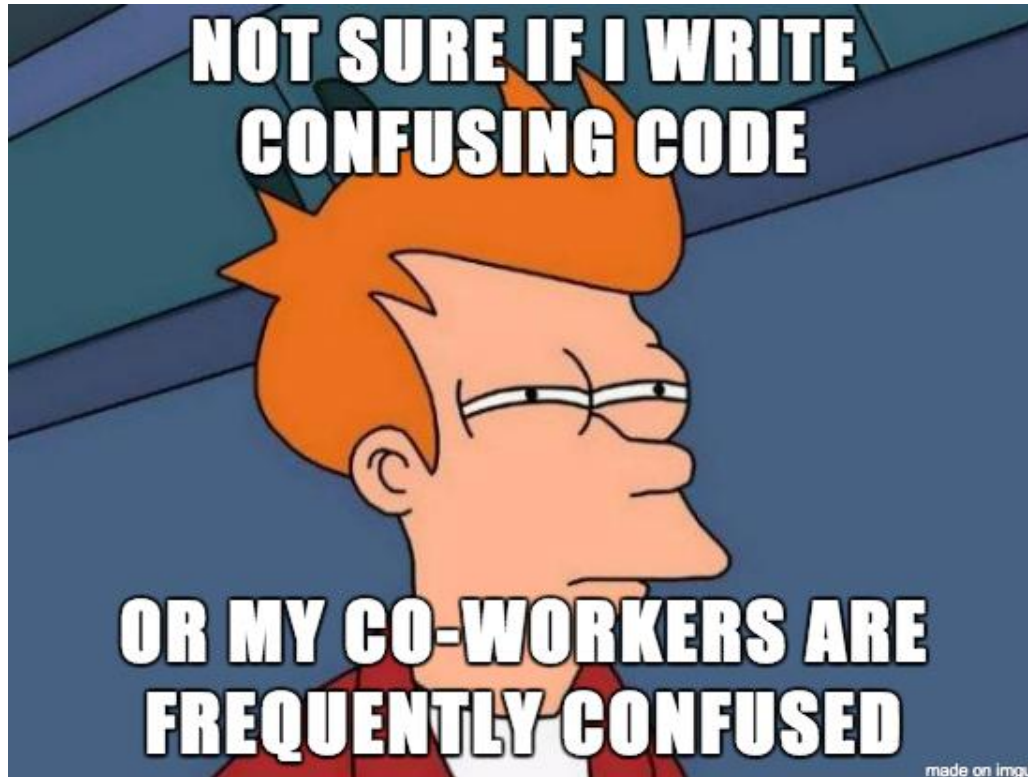
Ventajas de un buen diseño

- 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.