Unidad 5

Diseño

Ingeniería del Software

Universidad Autónoma de Madrid

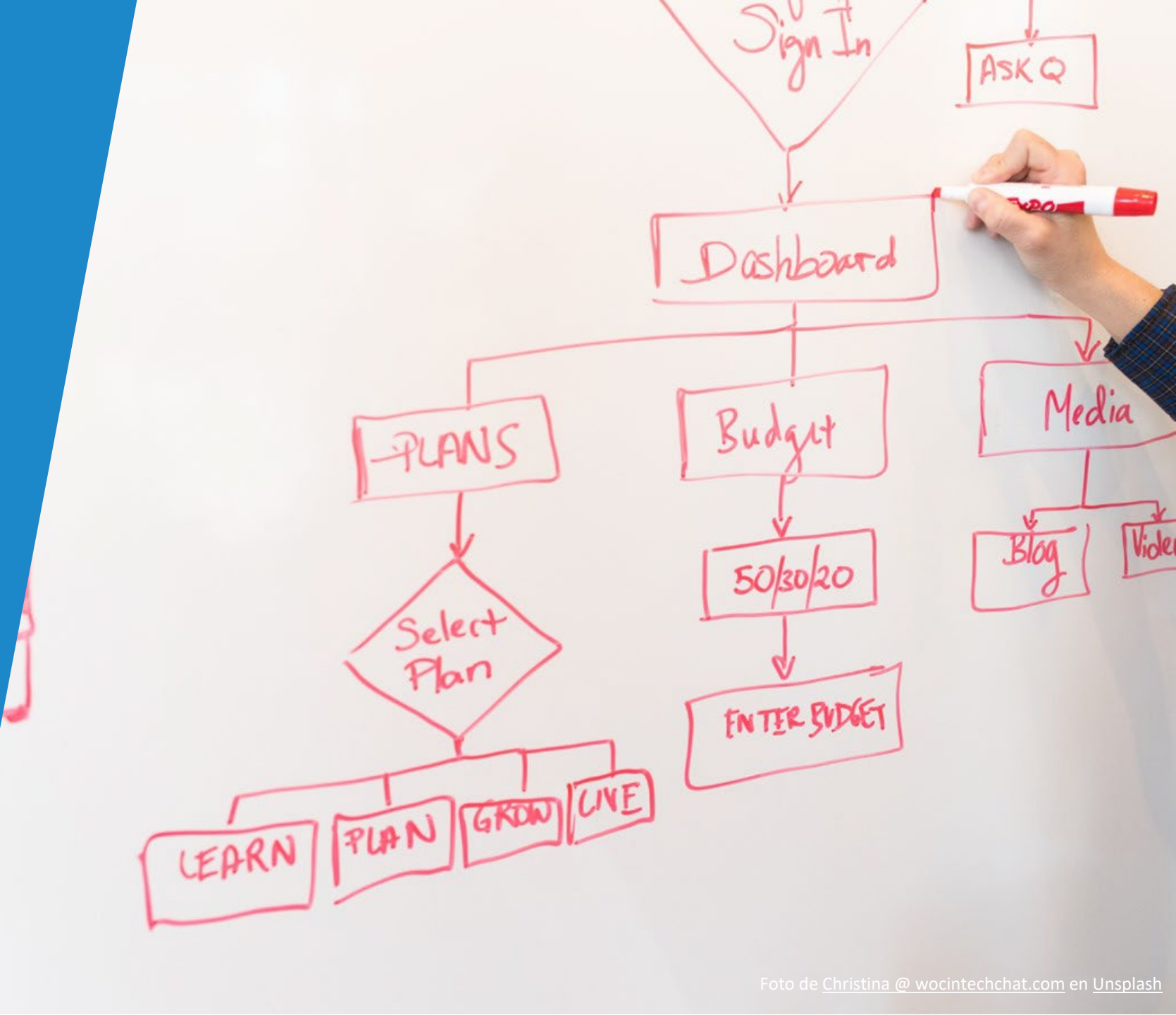
Escuela Politécnica Superior

Contenido

- Introducción
- Diseño estructurado y Diseño orientado a objetos con UML
- Métricas de diseño
- Guías de un buen diseño
- Principales errores en el diseño
- Documentación final del diseño

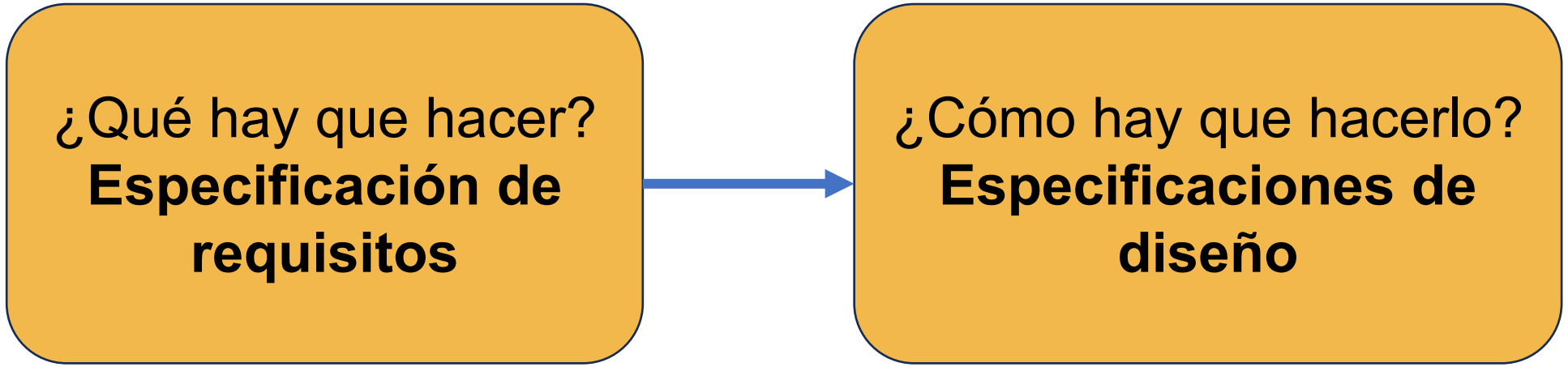
Introducción

- Transición de análisis a diseño
- Definición y objetivos
- Niveles de diseño
- Tareas
- Principios básicos de diseño
- Métodos de diseño



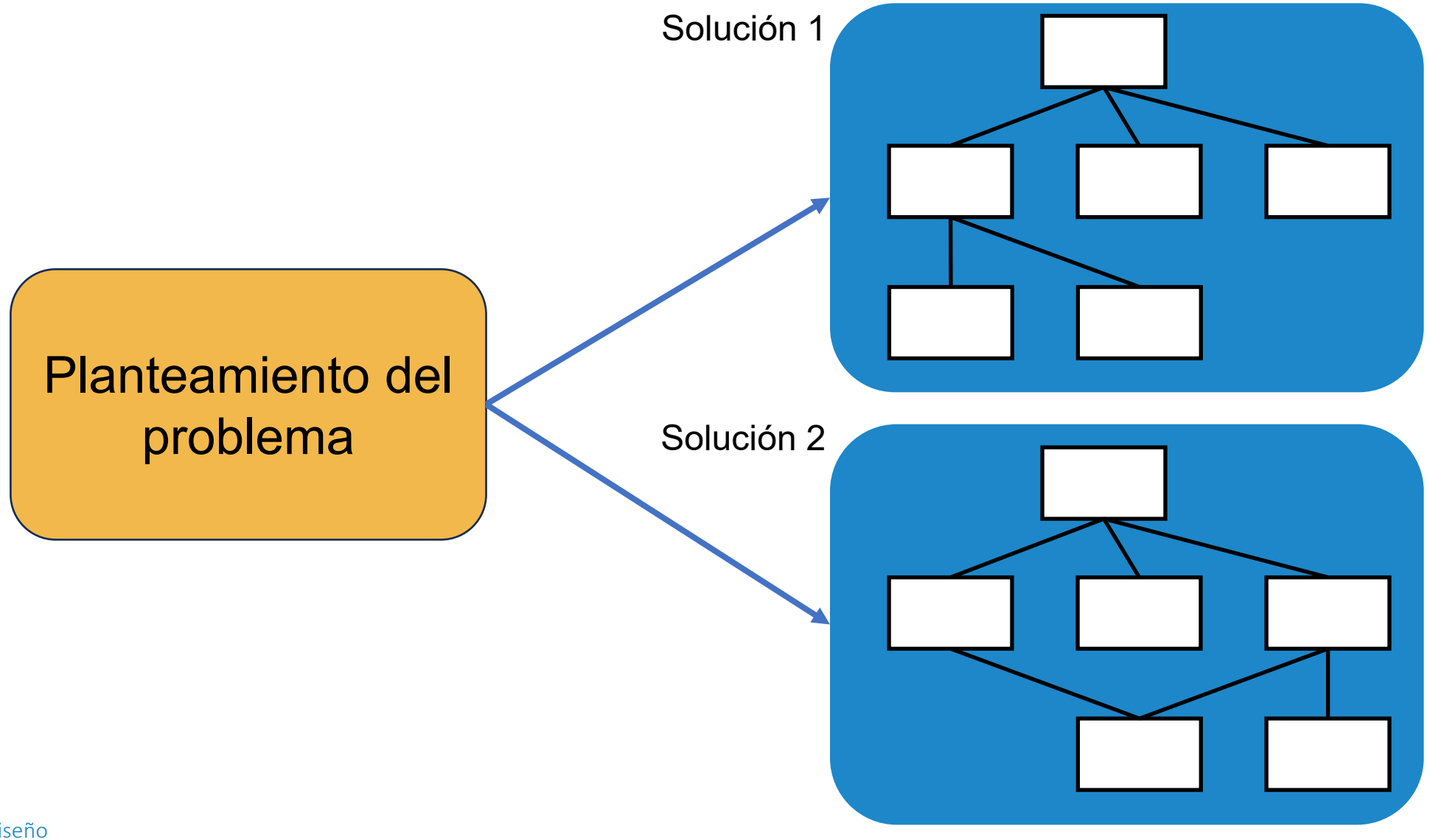
Transición de análisis a diseño

- Cambiar la atención del qué al cómo.



- Transformación de la definición del problema a la solución software.

Diseño



Diseño: Definición

- Es el proceso de definición de la arquitectura, componentes, módulos, interfaces, procedimientos de prueba y datos de un sistema software para satisfacer unos requisitos especificados.

Niveles de diseño

Diseño de la arquitectura

- Proceso de definición de la colección de componentes del sistema y sus interfaces.
- Objeto: determinar el marco de referencia que guiará la construcción del sistema.

Diseño detallado

- Proceso de descripción detallada de la lógica de cada uno de los módulos, de las estructuras de datos que utilizan y de los flujos de control.
- Objeto: describir el sistema con el grado de detalle suficiente y necesario para su posterior implementación.

Tareas

Diseño de la arquitectura:

- Definir los criterios de descomposición
- Descomponer el sistema en módulos
- Determinar las estructuras de datos
- Diseñar las interfaces
- Definir los flujos de control

Diseño detallado:

- Descripción detallada de los módulos: estructuras y algoritmos
- Descripción detallada de las estructuras físicas de datos
- Descripción de los procedimientos de acceso a las estructuras físicas de datos
- Descripción detallada de las interfaces

Revisión

¿El plano de una casa sería una tarea de diseño de alto nivel o detallado?

¿El color de una puerta de la casa se debe considerar en el diseño de alto nivel?

Principios básicos (I)

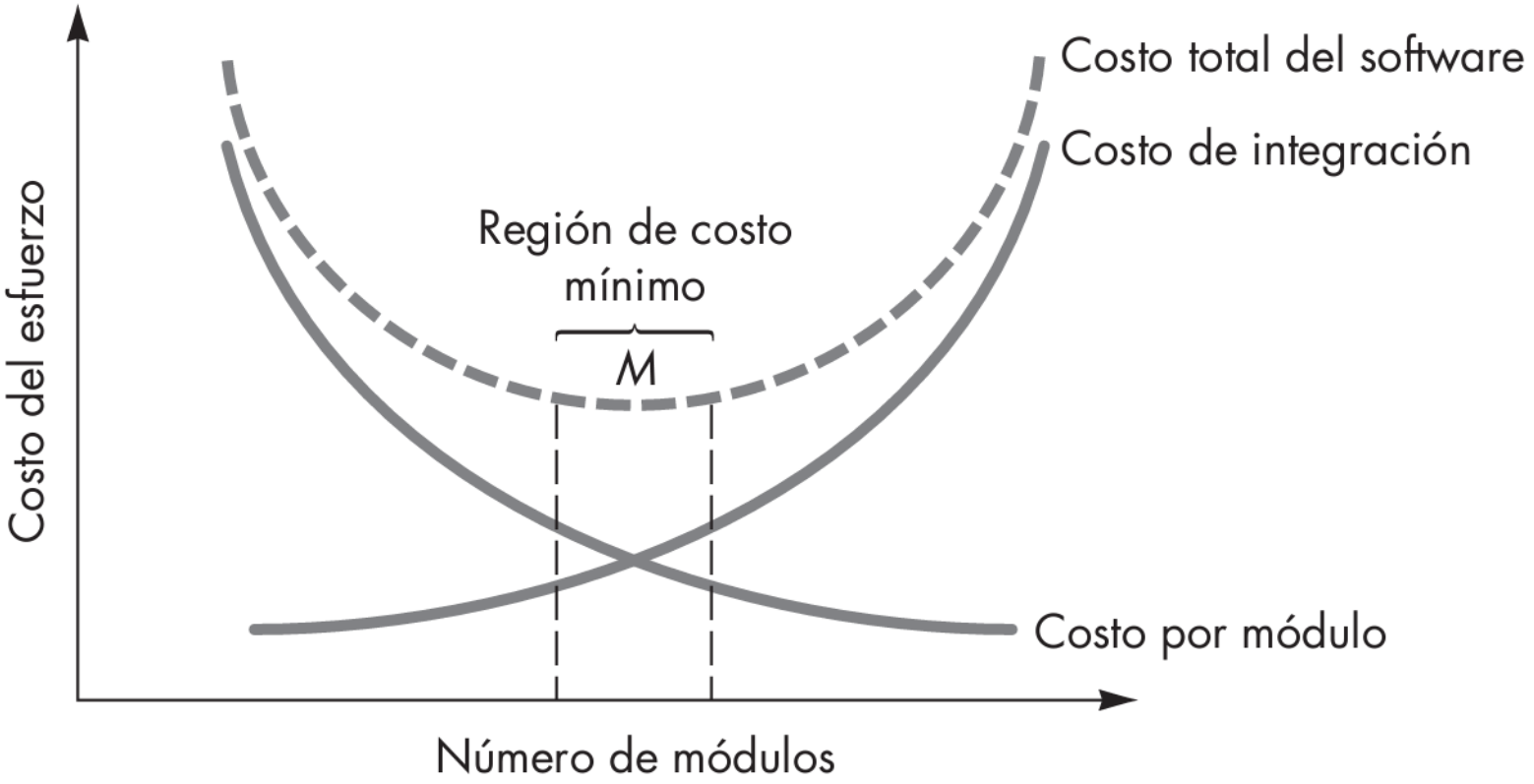
- **Abstracción**: Manejo de conceptos generales y no de instancias particulares.
- **Refinamiento**: Seguir una estrategia de diseño descendente.
- **Modularidad**: División del software en unidades con entidad propia tales como funciones o subrutinas.
- **Ocultación de la información**: Los detalles internos de cada módulo han de ser transparentes a los demás. Cada módulo debe formarse por especificaciones de diseño que sean independientes del resto de los módulos.

Principios básicos (II)

- La complejidad puede reducirse descomponiendo el problema en piezas cada vez más pequeñas, siempre que éstas sean relativamente independientes.
- A partir de un momento dado, la complejidad aumenta a causa de la interdependencia de las piezas.

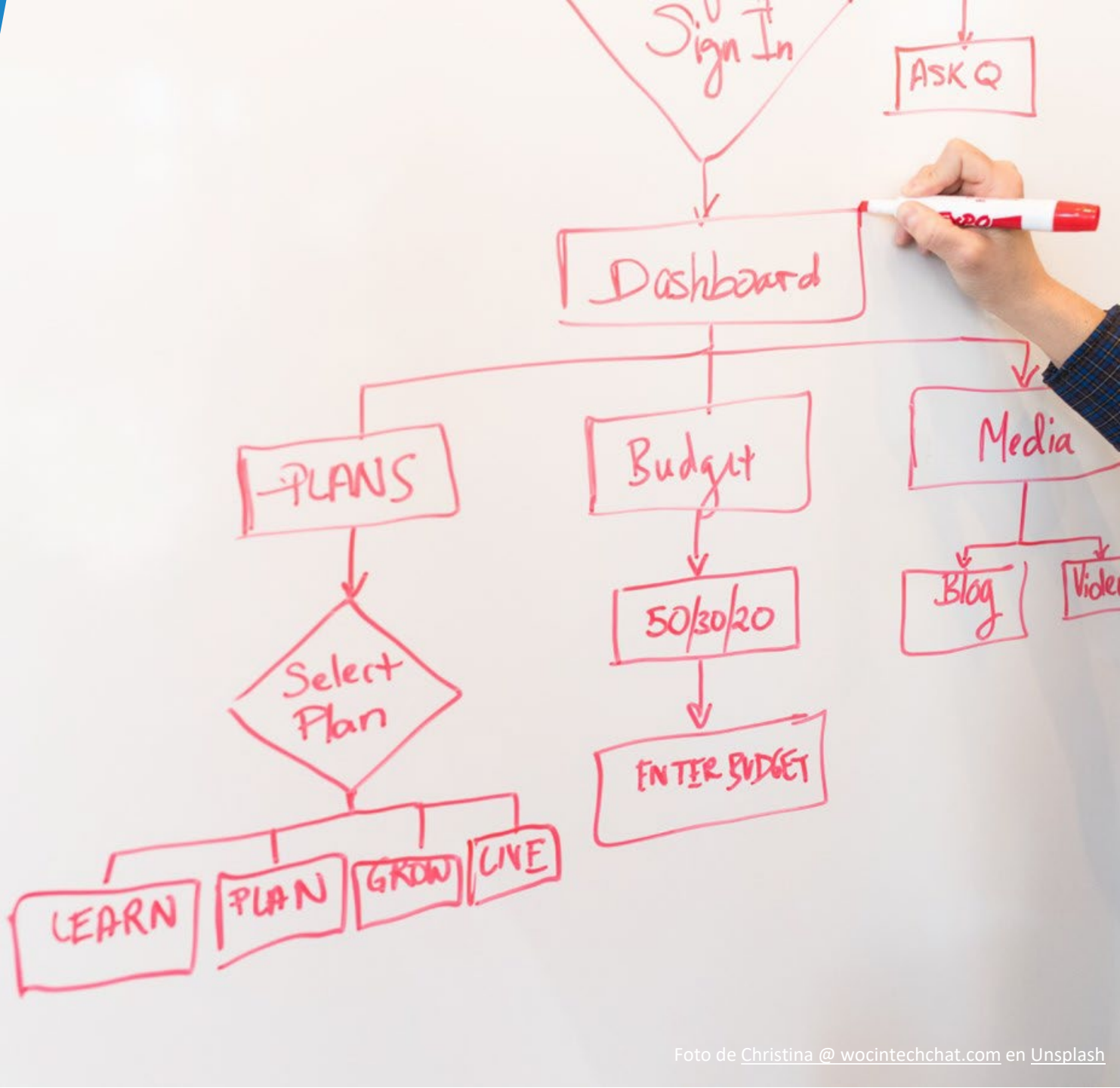
Principios básicos (III)

¿Hasta dónde debe llegar la descomposición de un sistema?



Ref: Pressman

Diseño estructurado y Diseño orientado a objetos



Métodos de diseño

Diseño estructurado.

- Se considera que el sistema es un conjunto de módulos o unidades, cada uno con una función bien definida, organizados de forma jerárquica.

Diseño orientado a objetos.

- Se considera que el sistema es una colección de objetos que interactúan a través de mensajes. Cada objeto tiene su propio estado y conjunto de operaciones asociadas.

Análisis y Diseño Orientado a Objetos con UML

Diagramas Estructurales

- D. de Clases
- D. de Objetos
- D. de Componentes
- D. de Despliegue
- D. de Paquetes

Diagramas de Comportamiento

- D. Casos de Uso
- D. de Secuencia
- D. de Comunicación
- D. de Estados
- D. de Actividad

Análisis y Diseño Orientado a Objetos con UML

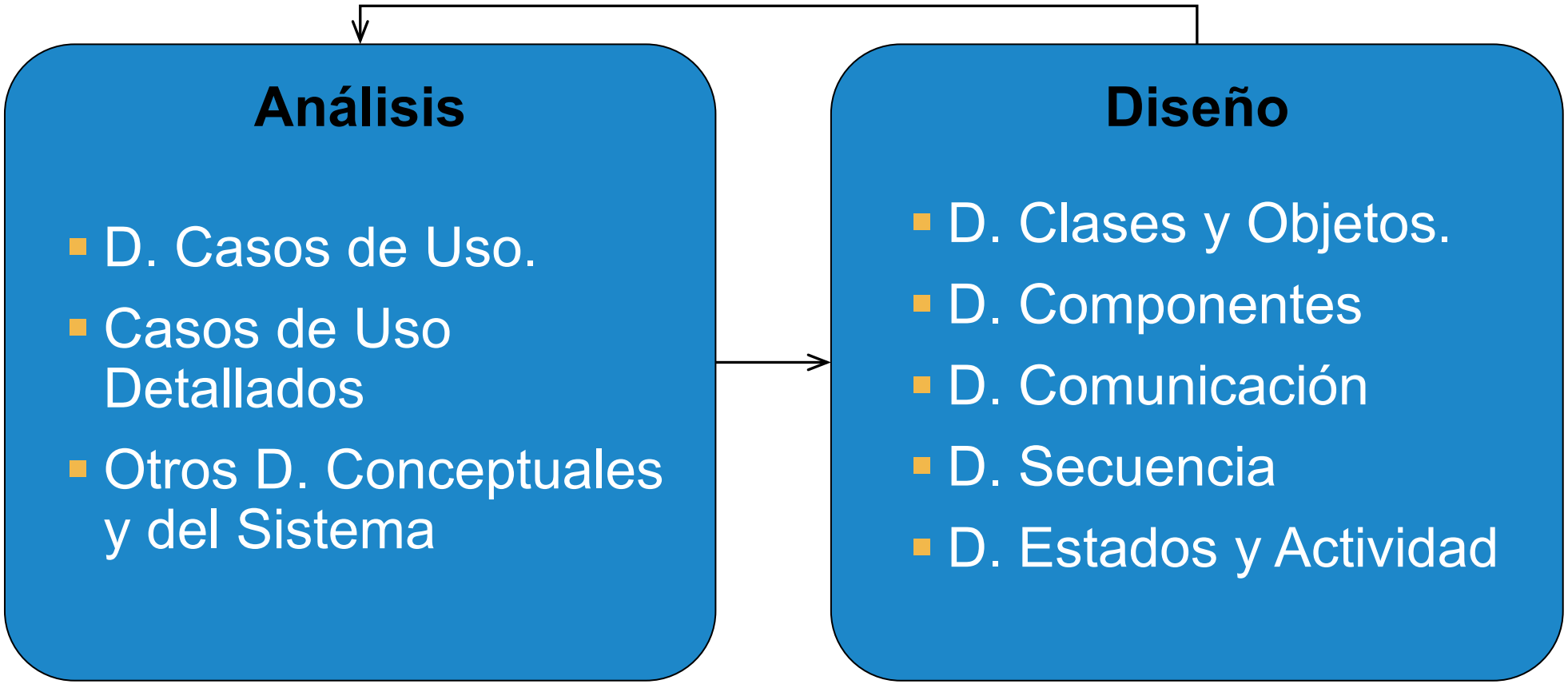


Diagrama de casos de uso

- **Análisis**
- Diagrama de comportamiento
- Actores y funcionalidades

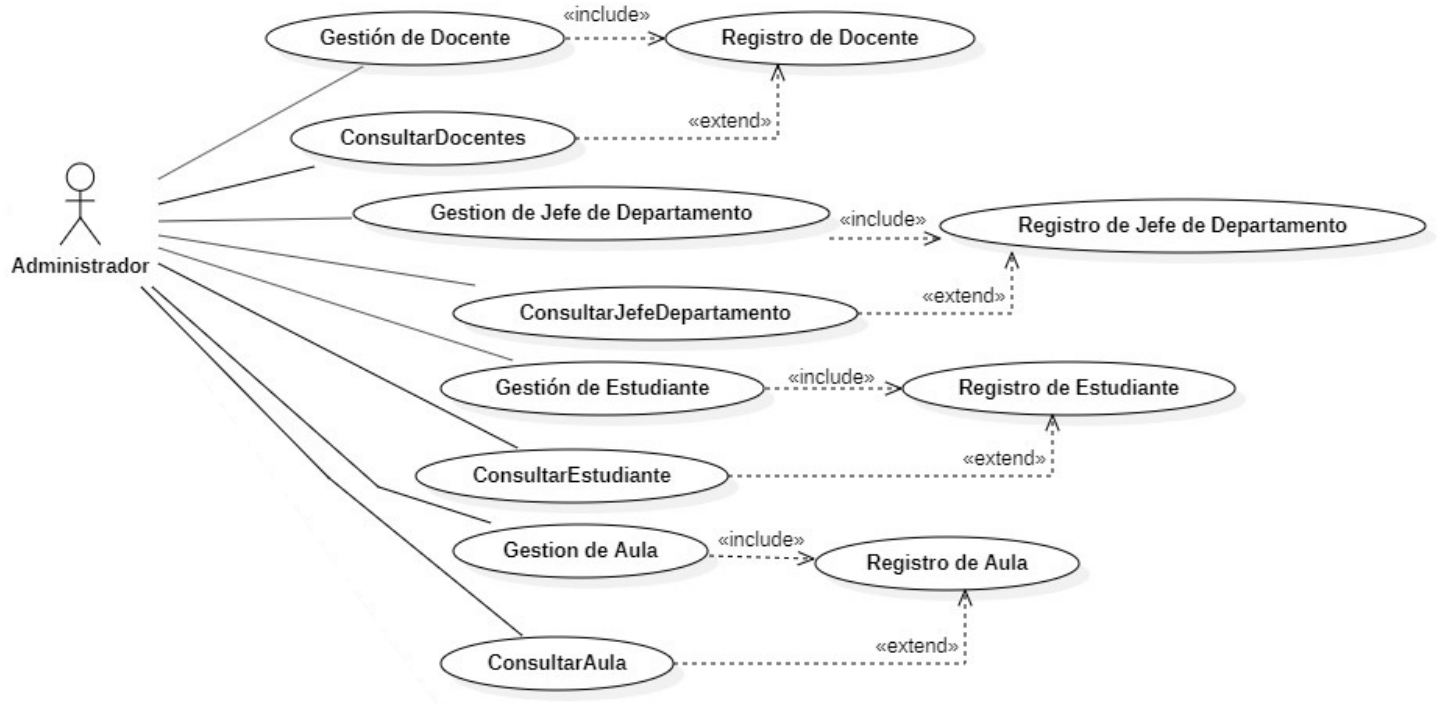


Diagrama de componentes

- Diseño
- Diagrama de estructura
- División del sistema en componentes y dependencias entre ellos
- Componente, Interfaz y Relación de dependencia

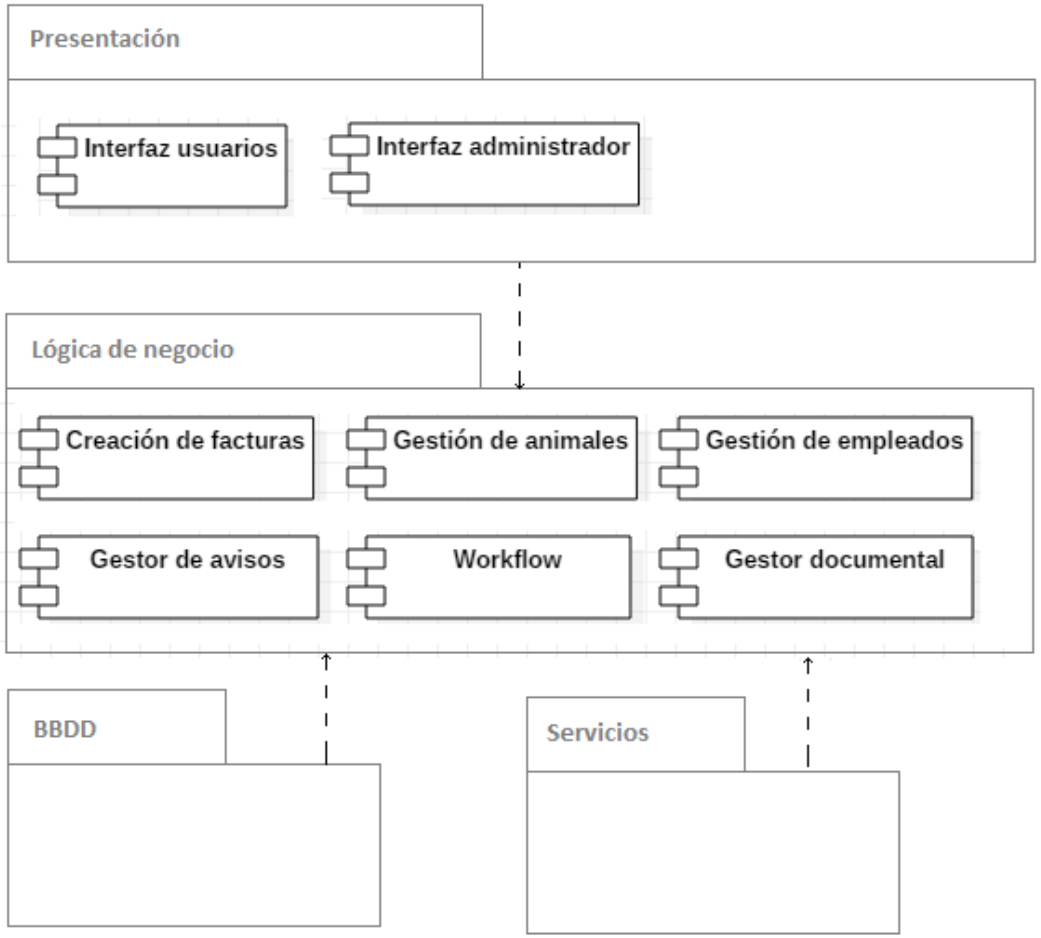


Diagrama de clases

- Diseño
- Diagrama de estructura
- Clases del sistema, atributos, métodos y relaciones entre clases

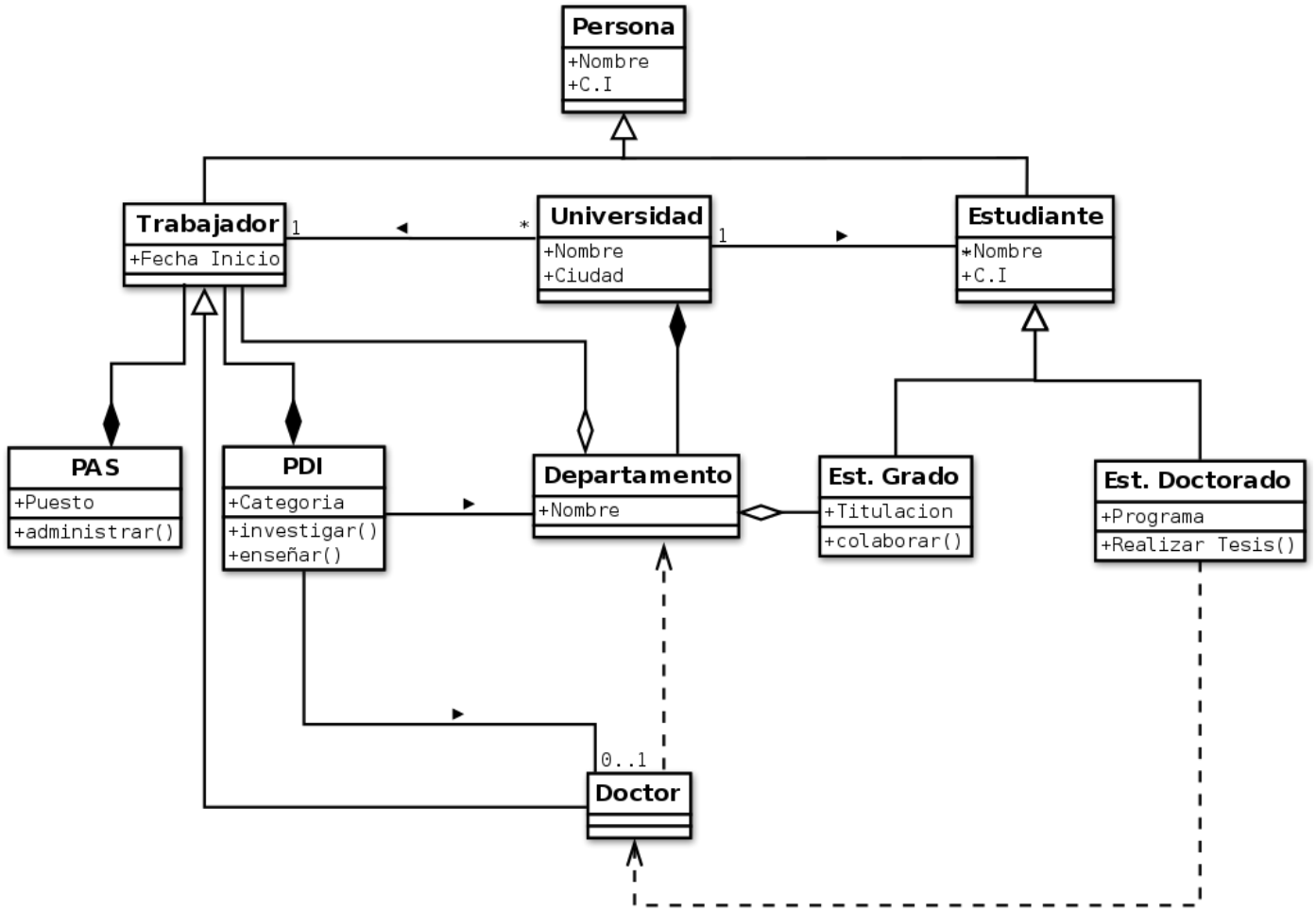


Diagrama de objetos

- Diseño
- Diagrama de estructura
- Instancia del diagrama de clases (UML 1.4.2)
- Estructura del sistema en un instante

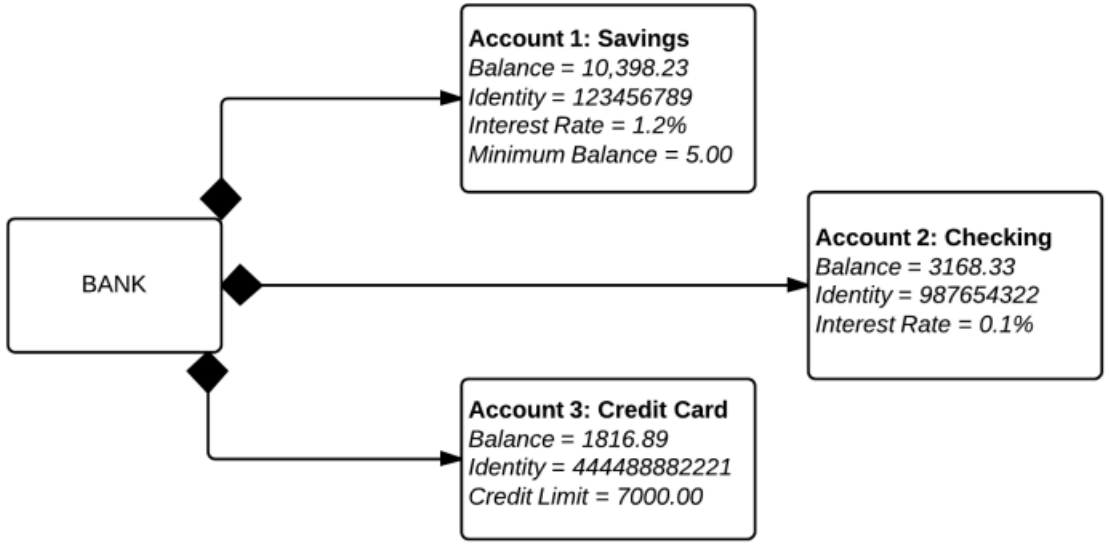


Diagrama de secuencia

- Diseño
- Diagrama de comportamiento
- Interacción entre objetos en un sistema

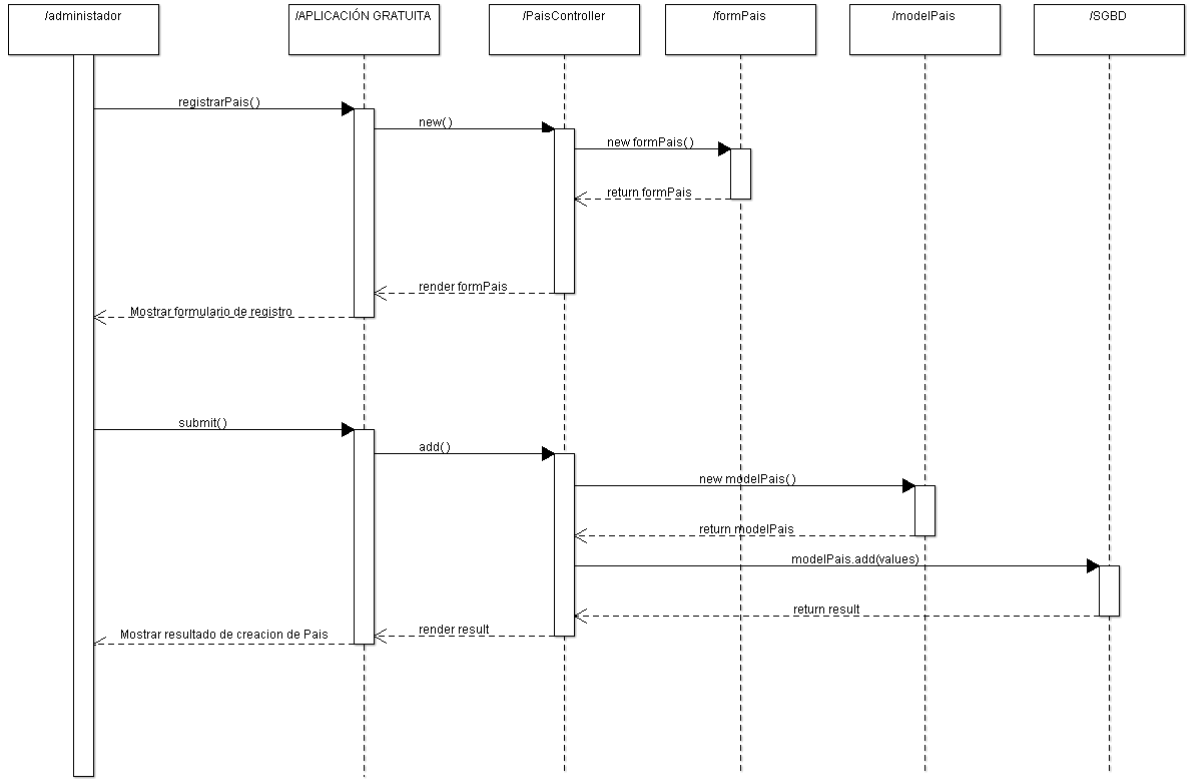


Diagrama de estados

- Diseño
- Diagrama de comportamiento
- Diagrama de máquina de estados o de transición de estados
- Estados por los que pasa un componente del sistema

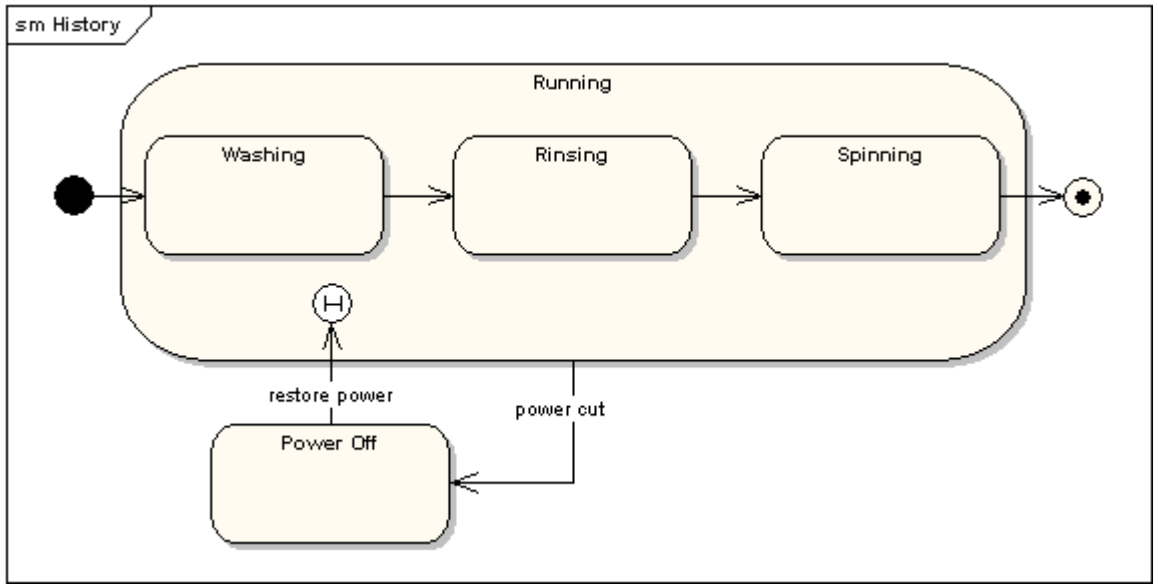
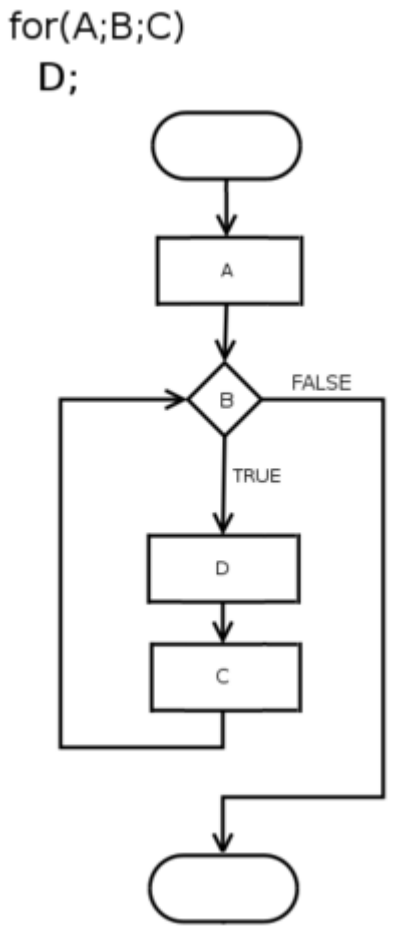


Diagrama de actividad

- Diseño
- Diagrama de comportamiento
- Diagrama de flujo de control
- Define los flujos de trabajo paso a paso



Métricas de diseño

- Cohesión
- Acoplamiento

Foto de krakenimages en Unsplash

Métricas de diseño

Cohesión

Es una medida de la relación (funcional) de los elementos de un módulo.

Es mejor un **alto grado de cohesión**:

- Menor coste de programación
- Mayor calidad del producto

Evitar los *módulos esquizofrénicos.*

Métricas de diseño

Acoplamiento

Es una medida de la interconexión entre los módulos de un programa.

Es mejor un **bajo acoplamiento**:

- Minimizar el efecto onda (propagación de errores).
- Minimizar el riesgo al cambiar un módulo por otro.
- Facilitar el entendimiento.

Métricas de diseño: Cohesión y acoplamiento

Cohesión:

- “Un módulo con un alto grado de cohesión hace (idealmente) una sola cosa”

Acoplamiento:

- “Hacer los módulos tan independientes como sea posible”

Guías de un buen diseño

Foto de Jason Leung en Unsplash

Guías de un buen diseño

Dividir los módulos de forma que:

sean tan independientes como sea posible

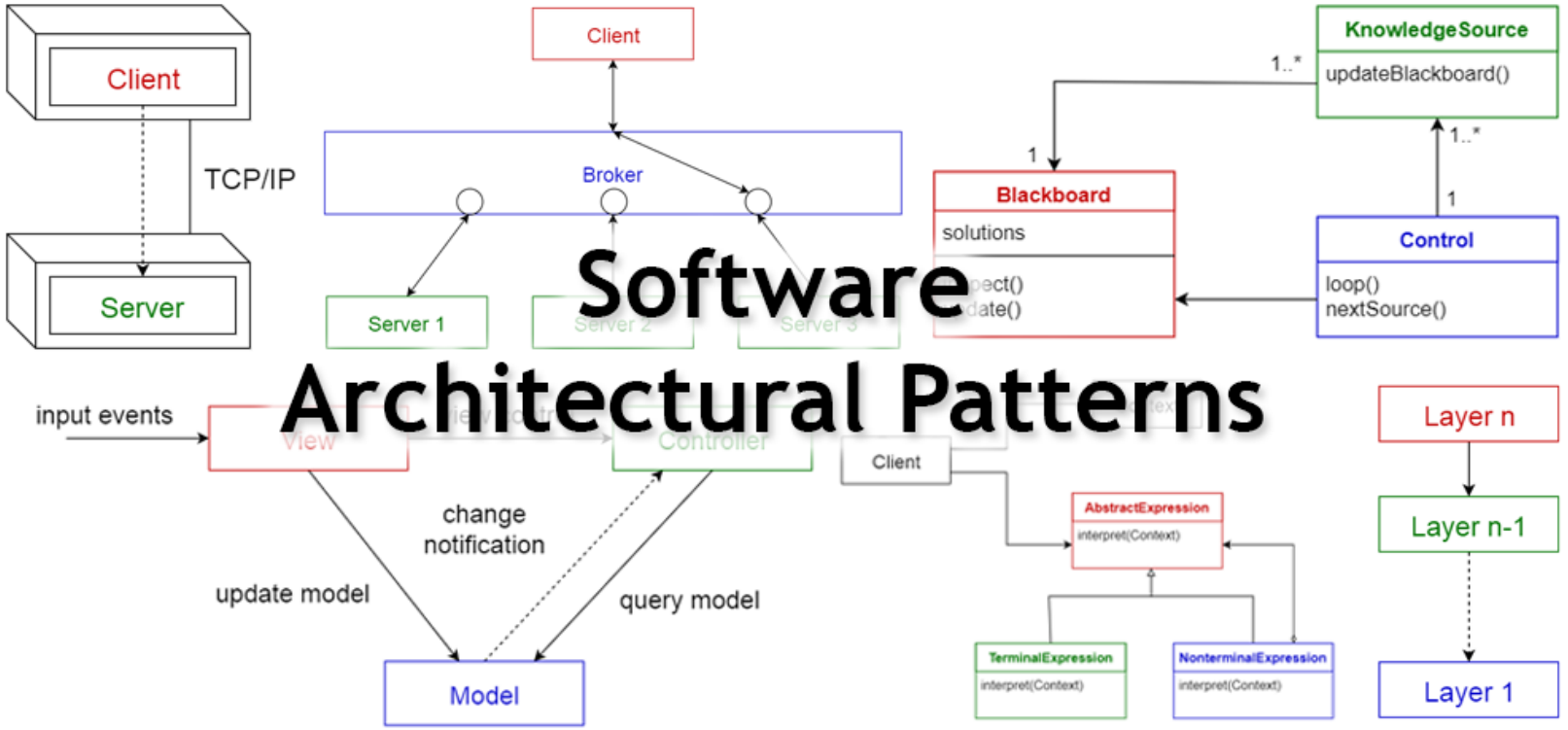
mínimo acoplamiento

cada módulo lleve a cabo una sola función

máxima cohesión

Patrones de diseño de arquitectura

Solución general y reutilizable a un problema común en la arquitectura del software.

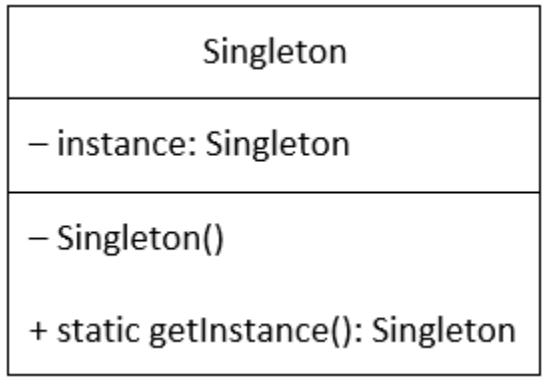


Patrones de diseño

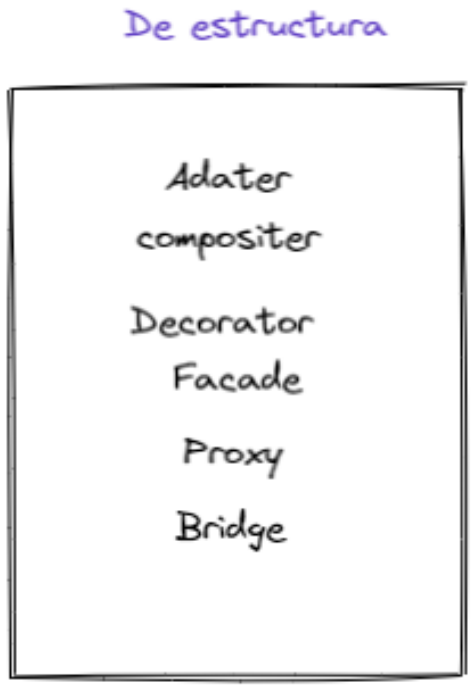
Solución general y reutilizable a un problema común en el diseño del software.

Patrones de diseño: creación

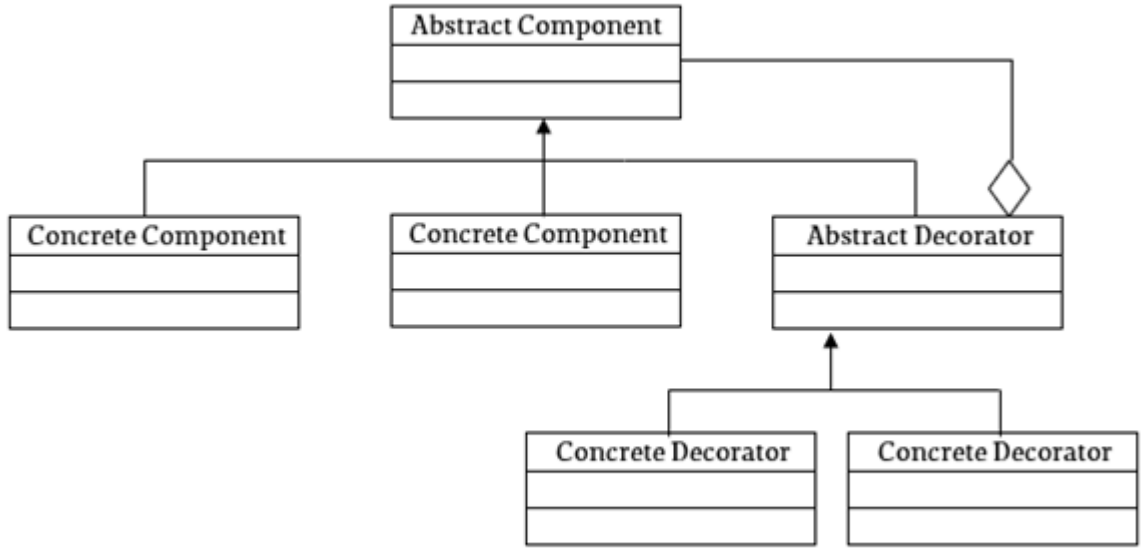
- Ejemplo: **Singleton**
- Crear una sola instancia de una clase con un único punto de acceso



Patrones de diseño: estructura

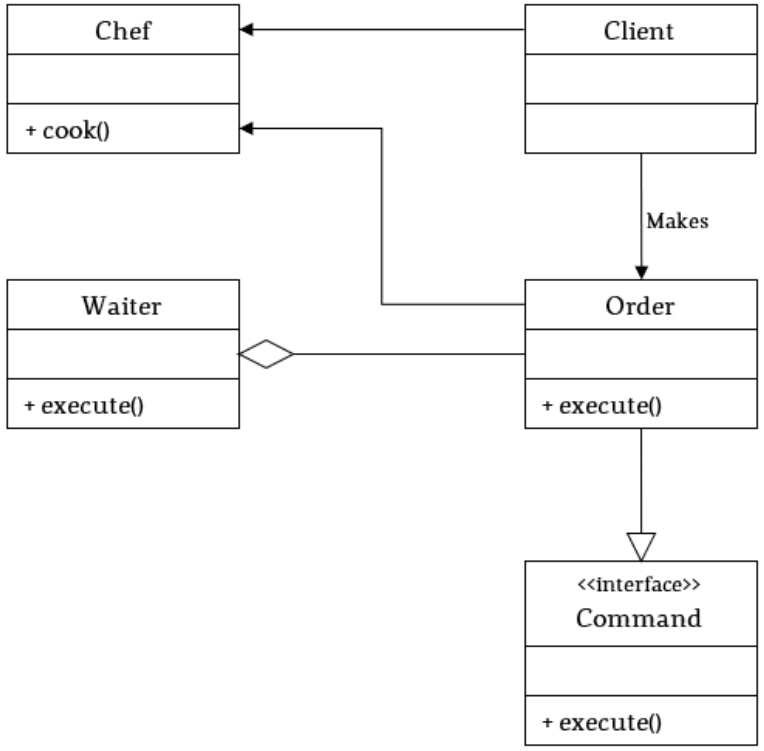


- Ejemplo: **Decorator**
- Añadir funcionalidad en tiempo de ejecución



Patrones de diseño: comportamiento

- Ejemplo: **Command Interpreter**
- Reducir acoplamiento entre clases



Principales errores del diseño

Foto de Kenny Eliason en Unsplash

Principales errores en el diseño

- Prisa por comenzar a programar:

  “Cuanto antes se comience a escribir código, más tarde se acabará el programa”.

- Falta de detalle en las especificaciones de diseño
- No documentar el diseño.
- No tener en cuenta el entorno físico.

Documentación final de diseño

Foto de ron dyar en Unsplash

Documento final: Diseño del Software (I)

1. **Introducción**
    - 1.1. Propósito del documento
    - 1.2. Entornos hardware y software
    - 1.3. Principales funciones del software
    - 1.4. Bases de datos externas
    - 1.5. Restricciones y limitaciones
    - 1.6. Referencias

2. **Descripción del diseño**
    - 2.1. Diagramas
    - 2.2. Descripción de datos
    - 2.3. Descripción de interfaces
    - 2.4. Descripción de comunicaciones

Documento final: Diseño del Software (II)

3. **Descripción de los módulos (para cada módulo)**
    3.1. Descripción
    3.2. Descripción de la interfaz
    3.3. Módulos relacionados
    3.4. Organización de los datos

4. **Descripción de archivos externos y datos globales**
    4.1. Descripción
    4.2. Métodos de acceso

Documento final: Diseño del Software (III)

5. **Especificaciones de programas**

6. **Referencias cruzadas con los requisitos**

7. **Plan de pruebas**
    7.1. Estrategia de integración
    7.2. Estrategia de pruebas
    7.3. Consideraciones especiales

Matriz de trazabilidad de requisitos

- Al finalizar el diseño (detallado) de la aplicación, es necesario comprobar que todos los requisitos educidos en la fase de análisis del problema tienen su correspondiente representación en el dominio de la solución a implementar.

	Elementos Funcionales			
Requisitos	Clase1.método1	Clase1.método2	…	ClaseN.métodoM
Requisito 1	X	X		
Requisito 1.1		X		
…				
Requisito N	X			X
Requisito N.M				X