

3. Diseño de Software

Ph.D Priscilla Jiménez P.

Diseño a nivel de componentes


¿Qué es un componente?

Visiones de un componente

Principios básicos del diseño de clases

Diseño en el nivel de componentes

Durante el diseño arquitectónico se define un conjunto completo de componentes de software.



El diseño a nivel de componentes define las estructuras de datos, algoritmos, características de la interfaz, mecanismos de comunicación asignados a cada componente del software.

Ilustra el software a un nivel de abstracción cercano al código.

Diseño en el nivel de componentes

¿Quién lo hace?

- Ingeniero de software

¿Por qué es importante?

- Antes de programar es necesario determinar si funcionará.
- Permite revisar detalles del diseño para garantizar consistencia con otras representaciones de diseño (datos, arquitectura, interfaz de usuario)

¿Cuál es el producto final?

- Diseño de cada componente, representado con notación gráfica, tabular o basada en texto.

¿Qué es un Componente?

Un componente es un bloque de construcción de software de cómputo.

Una parte modular, desplegable y sustituible de un sistema. (Especificación de UML)

Los componentes forman la arquitectura del software. Se comunican y colaboran entre ellos y con otras entidades (sistemas, dispositivos, personas, etc.)

Tres visiones de un componente

1. Visión orientada a objetos.
2. Visión orientada a un elemento funcional (módulos) – vista tradicional.
3. Visión relacionada con el proceso.

Visión orientada a objetos

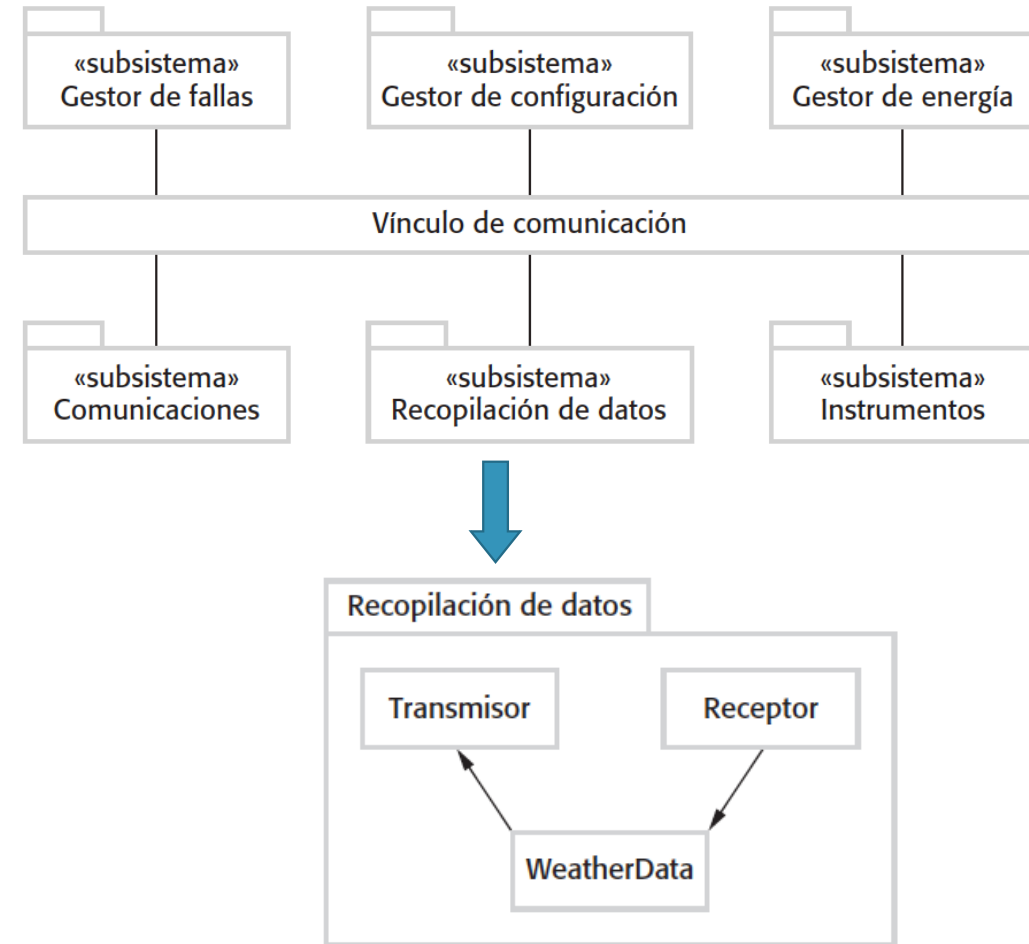
Pasos del diseño del sistema

1. Comprender y definir el contexto y las interacciones externas con el sistema.
2. Diseñar la arquitectura del sistema.
3. Identificar los objetos principales del sistema.
4. Desarrollar modelos de diseño.
5. Especificar interfaces.

Visión orientada a objetos

- Un componente contiene un conjunto de clases que colaboran.
- Cada clase dentro de un componente se elabora por completo para que incluya todos los atributos y operaciones relevantes para su implementación.
- También deben definirse todas las interfaces que permiten que las clases se comuniquen y colaboren.
- Se empieza revisando el modelo de requerimiento y las clases de análisis (componentes q se relacionan y dan apoyo).

Arquitectura del Sistema



Sistema	Estación meteorológica
Caso de uso	Reporte del clima
Actores	Sistema de información meteorológica, estación meteorológica
Datos	La estación meteorológica envía un resumen de datos meteorológicos, recopilados de los instrumentos en el periodo de recolección, al sistema de información meteorológica. Los datos enviados incluyen las temperaturas, máxima, mínima y promedio de la tierra y el aire; asimismo, las presiones de aire máxima, mínima y promedio; la rapidez del viento, máxima, mínima y promedio; la totalidad de la lluvia y la dirección del viento que se muestrea a intervalos de cinco minutos.
Estímulos	El sistema de información meteorológica establece un vínculo de comunicación satelital con la estación meteorológica y solicita la transmisión de los datos.
Respuesta	Los datos ya resumidos se envían al sistema de información meteorológica.
Comentarios	Por lo general, se pide a las estaciones meteorológicas reportarse una vez cada hora, pero esta periodicidad puede diferir de una estación a otra y modificarse en el futuro.

Descripción del caso de uso en lenguaje natural estructurado.

Diseño de componente: Transmisor y Receptor administran las comunicaciones, el objeto WeatherData encapsula la información.

Identificación de objetos esenciales del sistema

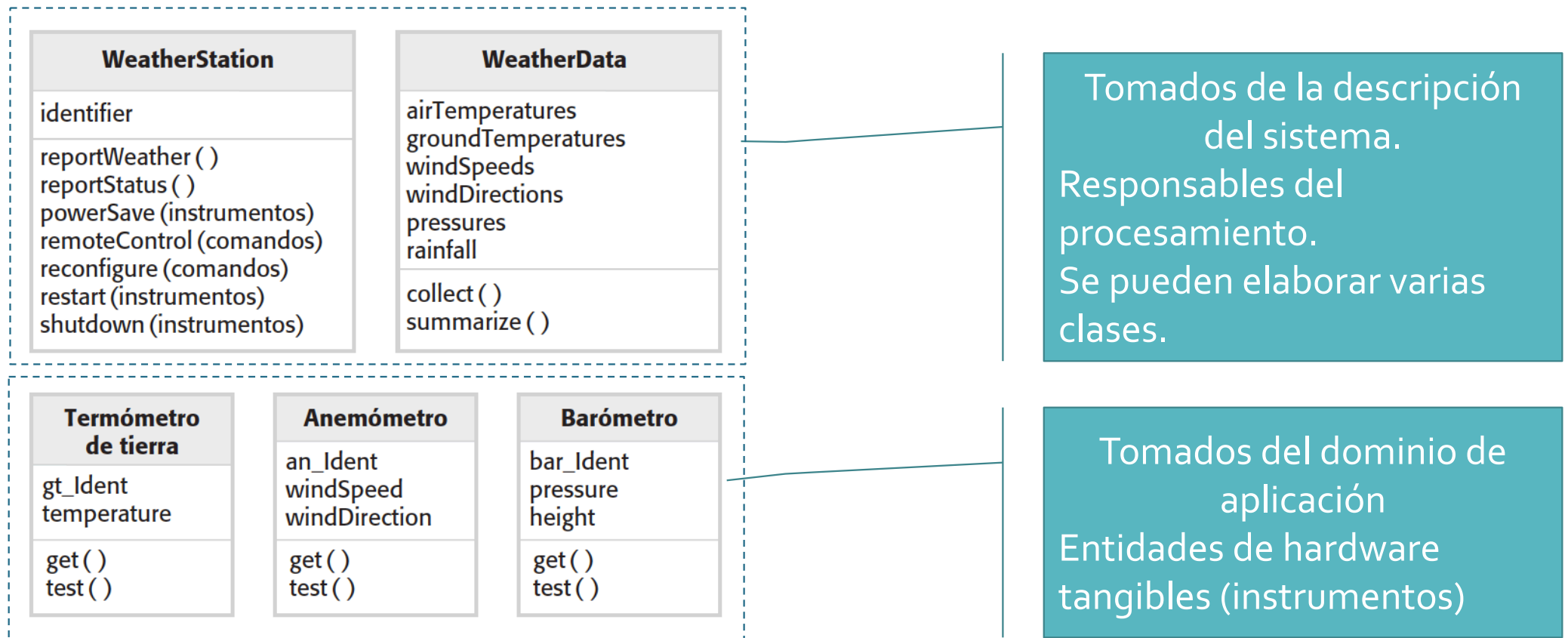
¿Qué ayuda a identificar objetos y operaciones?

- La descripción del caso de uso.

Sugerencias para identificar las clases de objetos:

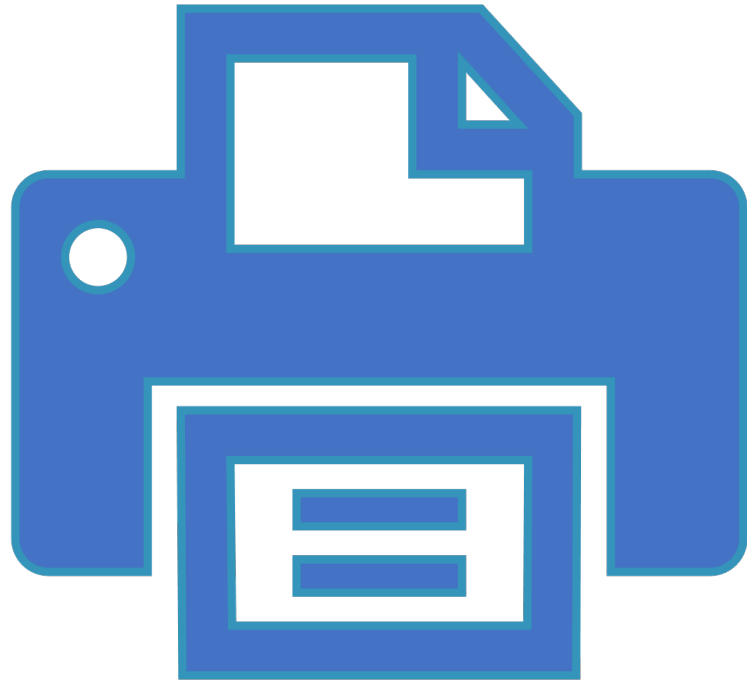
- Análisis gramatical de una descripción en lenguaje natural del sistema a construir. Objetos y atributos son sustantivos; operaciones o servicios son verbos.
- Utilice entidades tangibles (cosas) en el dominio de aplicación (estación), roles (administrador, médico), eventos, interacciones (reuniones), ubicaciones (como oficinas), unidades organizacionales (compañías).
- Análisis basado en escenario.

En la práctica se deben usar varias fuentes de conocimiento para describir clases de objetos.



Ejemplo: objetos de la estación meteorológica

El conocimiento del dominio de aplicación se usa para identificar otros objetos, atributos y servicios. El enfoque es en los objetos sin pensar en cómo podrían implementarse.



Ejemplo: Taller de impresión avanzada

Objetivo:

Obtener los requerimientos que plantea el cliente en el mostrador.

Presupuestar trabajo de impresión.

Pasar el trabajo a una instalación automatizada de producción.

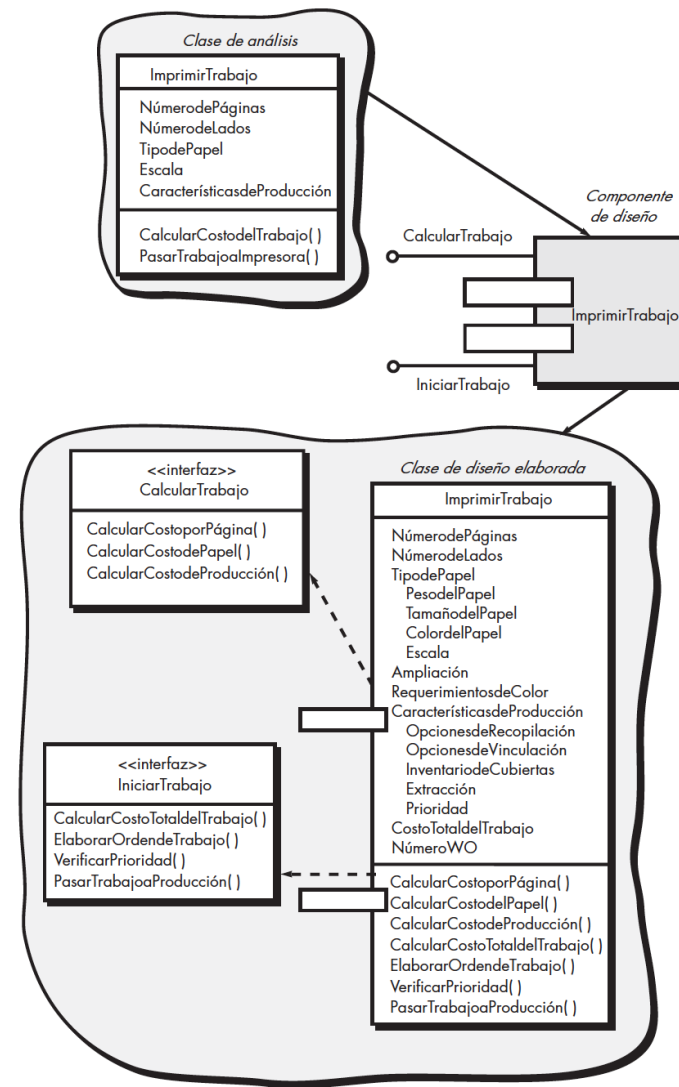
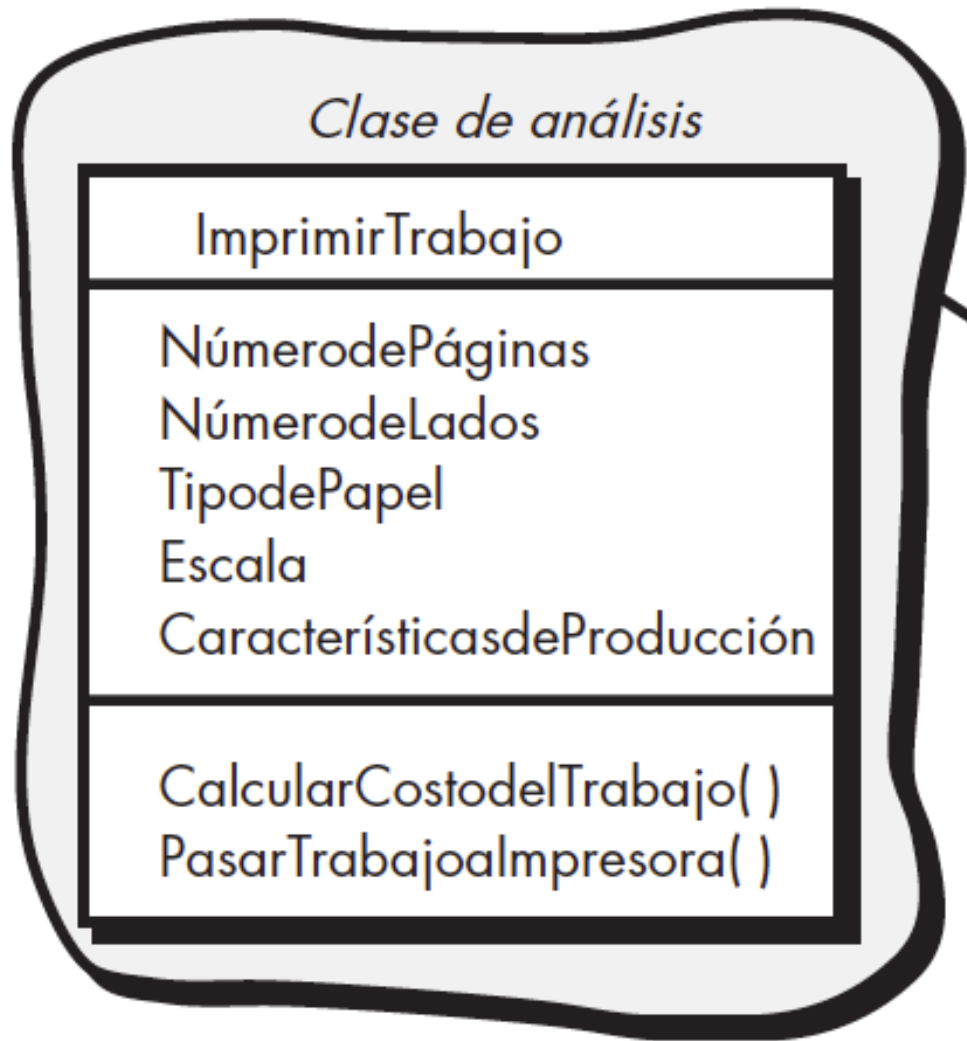


Fig 10.1 (Pressman, 2010) - Elaboración de un componente de diseño



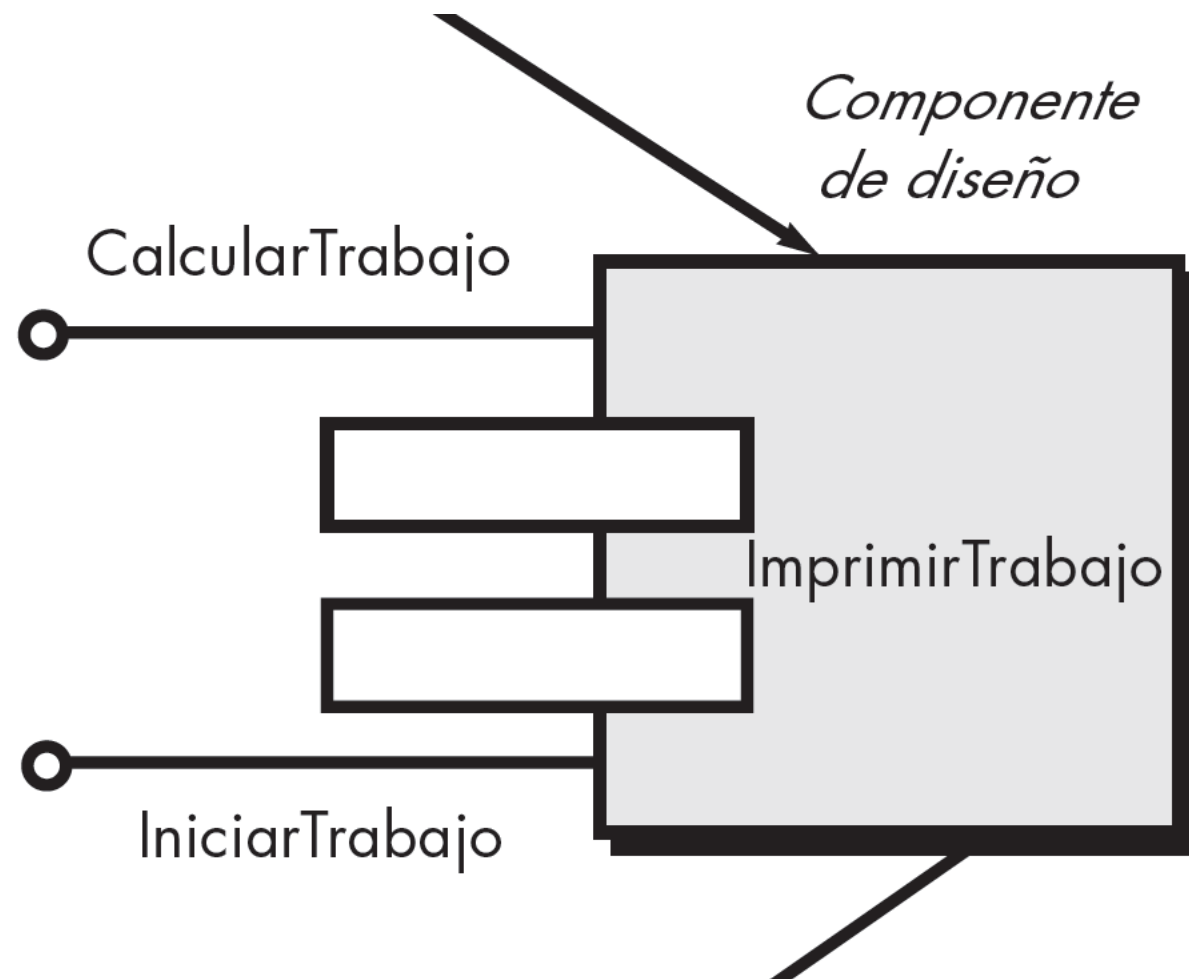
Ejemplo: Taller de impresión avanzada

Ingeniería de requerimientos

- Clase llamada **ImprimirTrabajo**

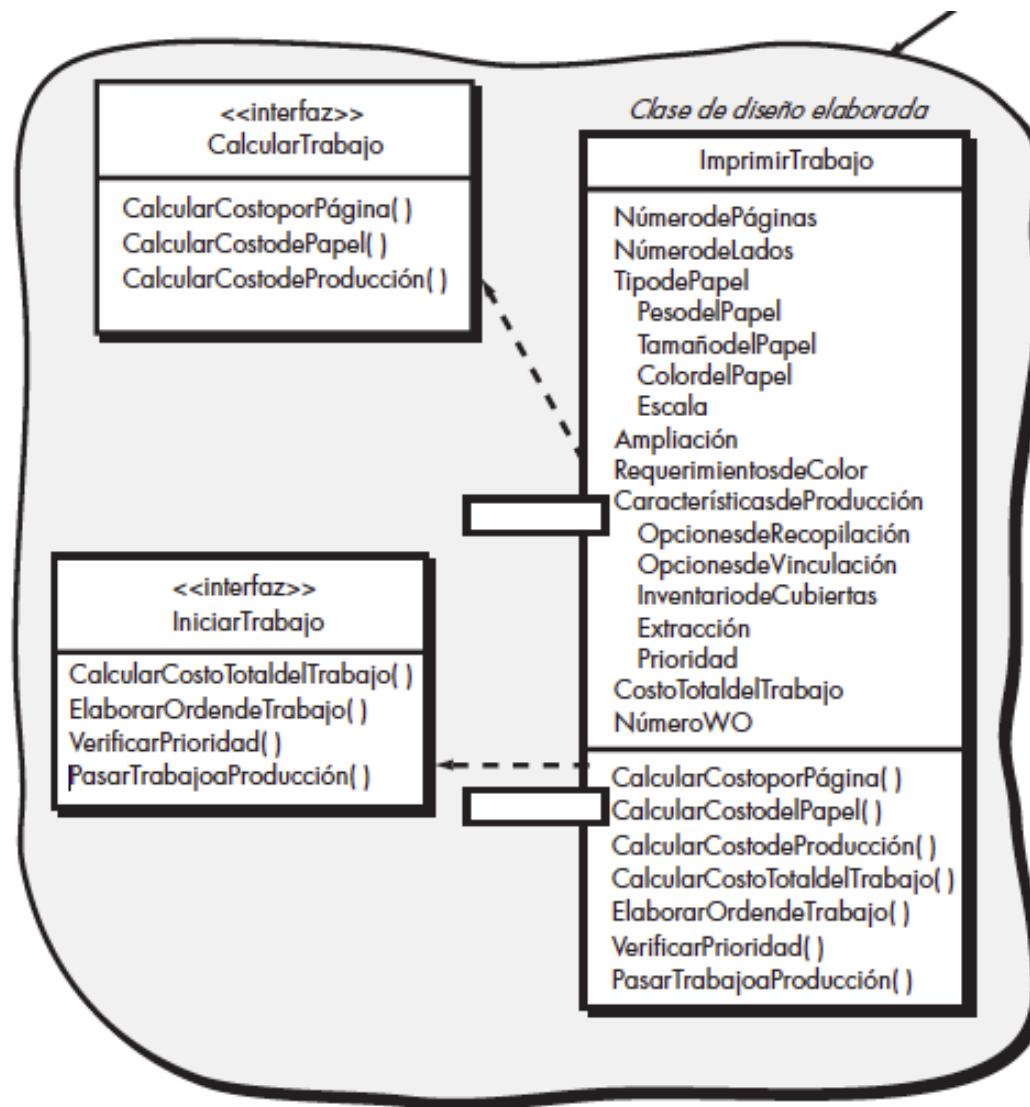
Ejemplo: Taller de impresión avanzada

Se crea un componente de diseño a partir de la clase de análisis



Ejemplo: Taller de impresión avanzada

Clase de diseño elaborada del componente IniciarTrabajo



CalcularTrabajo quizá colabore con otro componente TabladeValuación que contiene info sobre los precios.

IniciarTrabajo quizá colabore con otro componente FilaDeTrabajos para determinar tipo y prioridades.

Especificación de interfaz

- Se definen las interfaces entre los componentes del sistema.
- La especificación de una interfaz debe ser precisa, para que sea factible usar un componente sin que otros tengan que saber como se implementó.
- Ayuda a que los objetos y subsistemas se puedan diseñar en paralelo. (Es muy útil para desarrolladores)

Especificación de interfaz

- El diseño de interfaz se preocupa por la especificación del detalle de la interfaz hacia un objeto o un grupo de objetos.
- Definir las firmas y la semántica de los servicios que ofrecerá el objeto o un grupo de objetos.
- Se especifican usando UML similar a las clases, pero no se incluyen atributos, y debe especificarse como << interface >>

Se usa UML con la misma notación de un diagrama de clases.

Especificación de interfaz


No hay una relación 1:1 entre objetos e interfaces, un objeto puede tener muchas interfaces o un grupo de objetos pueden accederse a través de una sola interfaz.

«interfaz» Reporte
weatherReport (WS-Ident): Wreport statusReport (WS-Ident): Sreport

«interfaz» Control remoto
startInstrument (instrument): iStatus stopInstrument (instrument): iStatus collectData (instrument): iStatus provideData (instrument): string

Visión orientada a objetos

Esta actividad se aplica a cada componente.



Se debe especificar: atributos, operaciones e interfaz.



Además, se diseñan algoritmos asociados a cada operación.

Visión orientada a objetos

Pasos del diseño del sistema

- Comprender y definir el contexto y las interacciones externas con el sistema.
- Diseñar la arquitectura del sistema.
- Identificar los objetos principales del sistema.
- Desarrollar modelos de diseño.
- Especificar interfaces.

Modelos y diagramas más
específicos

Modelos de diseño

- Los modelos de diseño son el puente entre los requerimientos y la implementación de un sistema.
- Cuando hay vínculos cercanos entre especificadores, diseñadores y desarrolladores, los modelos pueden ser abstractos. (involucra varias discusiones)
- Cuando los vínculos son indirectos, se necesitan modelos detallados.

El nivel de abstracción y detalle también depende del tipo de sistema.

Modelos de diseño útiles

Modelos del subsistema

- Exponen los agrupamientos lógicos de objetos en subsistemas coherentes. Se representa mediante una forma de diagrama de clases en el que cada subsistema se muestra como un paquete con objetos encerrados.

Modelos de secuencia

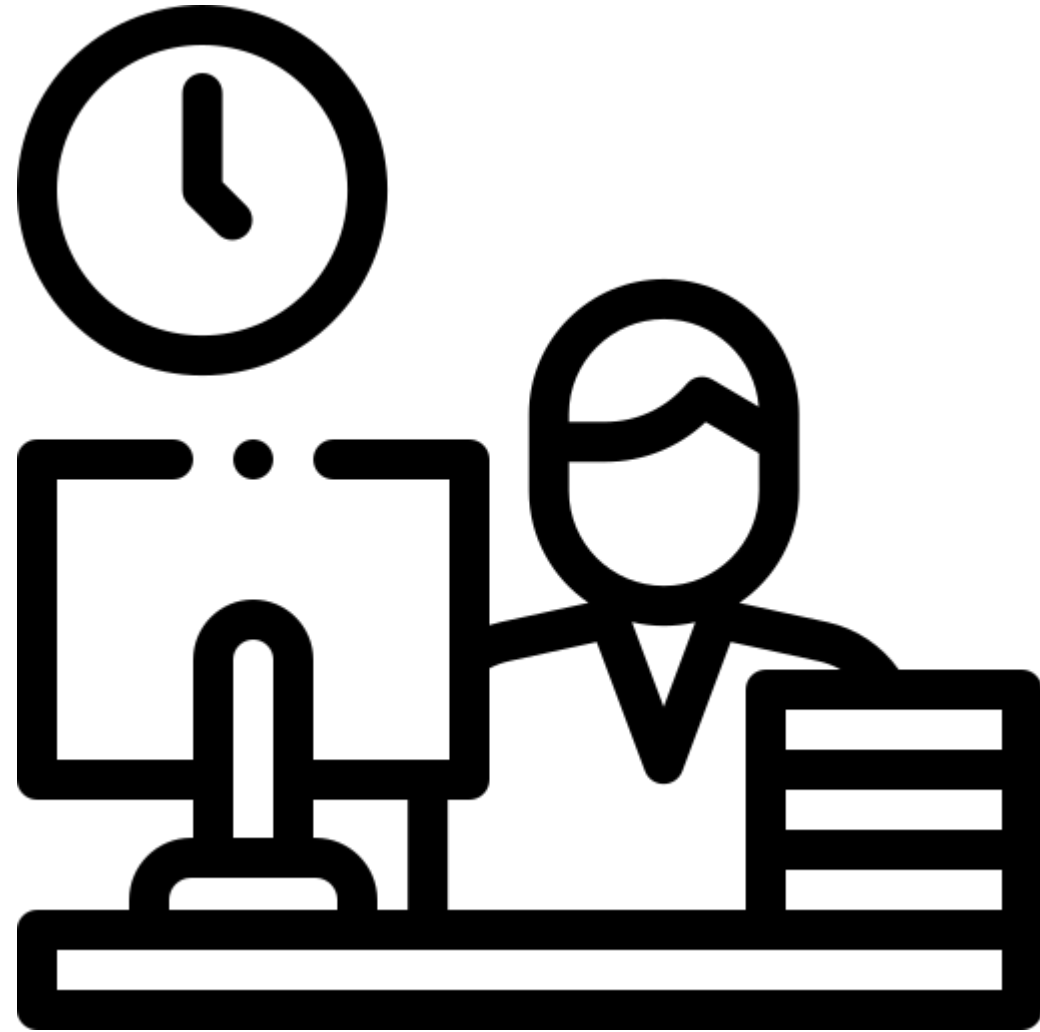
- Ilustran la secuencia de interacciones de objetos. Debe haber un modelo de secuencia por cada caso de uso que identifique. Modela el comportamiento combinado de un grupo de objetos.

Modelos de máquina de estado

- Muestran cómo los objetos individuales cambian su estado en respuesta a eventos. Resume el comportamiento de un objeto o un subsistema, en respuesta a mensajes y eventos.

Práctica 11

Diseño en el nivel de componentes
– Visión orientada a objetos.



Tres visiones de un componente

1. Visión orientada a objetos
2. Visión orientada a un elemento funcional (módulos) – vista tradicional.
3. Visión relacionada con el proceso

Vista tradicional

Visión orientada a un elemento funcional

Un componente es un elemento funcional de un programa que incorpora:

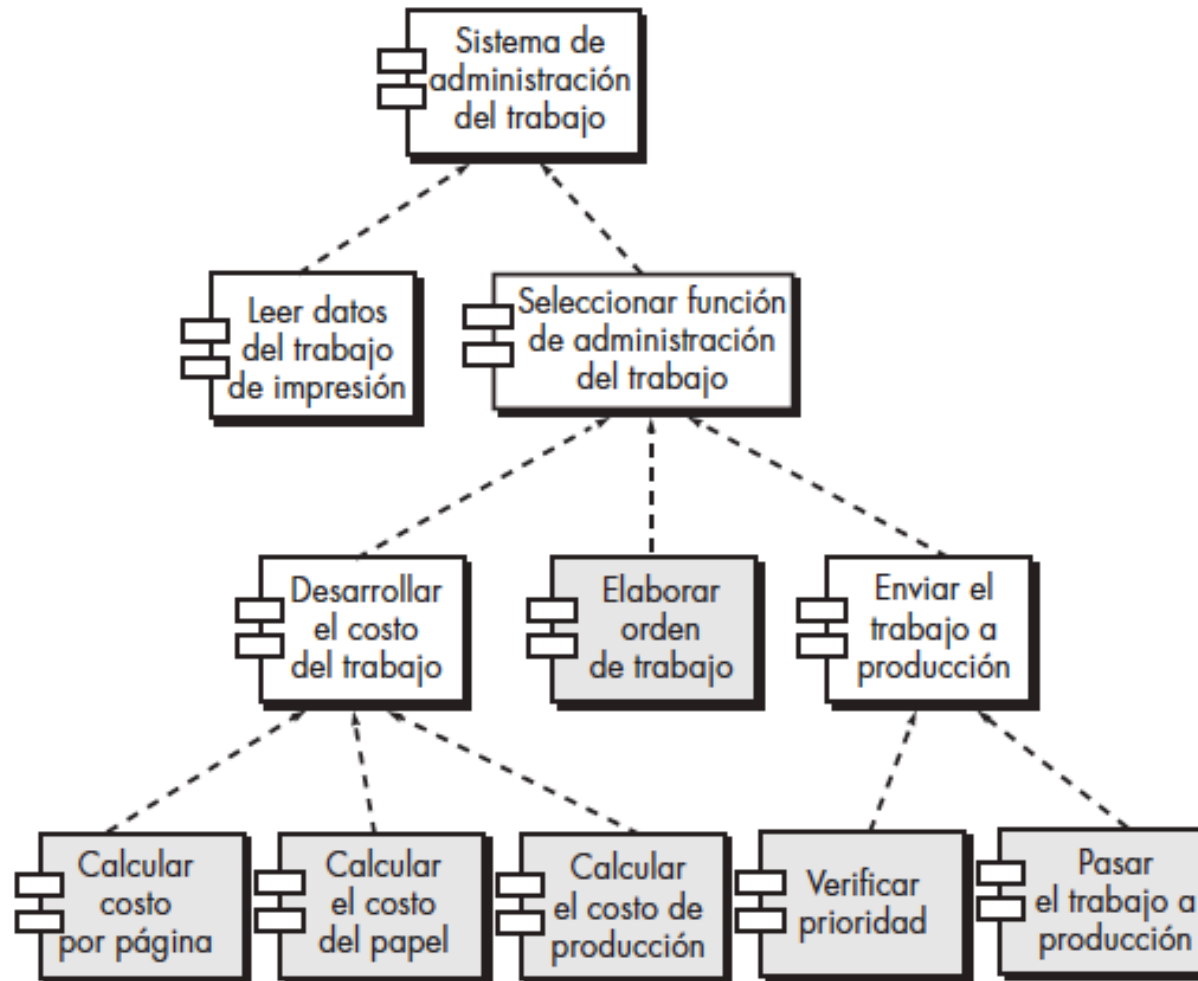
- la lógica del procesamiento,
- las estructuras de datos requeridas para procesar la lógica,
- y una interfaz que permita la invocación del componente y el paso de datos.

Vista tradicional

Visión orientada a un elemento funcional

Dentro de la arquitectura de software, el módulo cumple 3 funciones:

- Como componente de control que coordina la invocación de todos los demás componentes del dominio del problema.
- Como componente del dominio del problema que implanta una función completa o parcial que requiere el cliente.
- Como componente de infraestructura que es responsable de las funciones que dan apoyo al procesamiento requerido en el dominio del problema.



Vista tradicional

Los componentes se obtienen del conjunto de diagramas de flujo de datos.

La interfaz del módulo se define explícitamente.

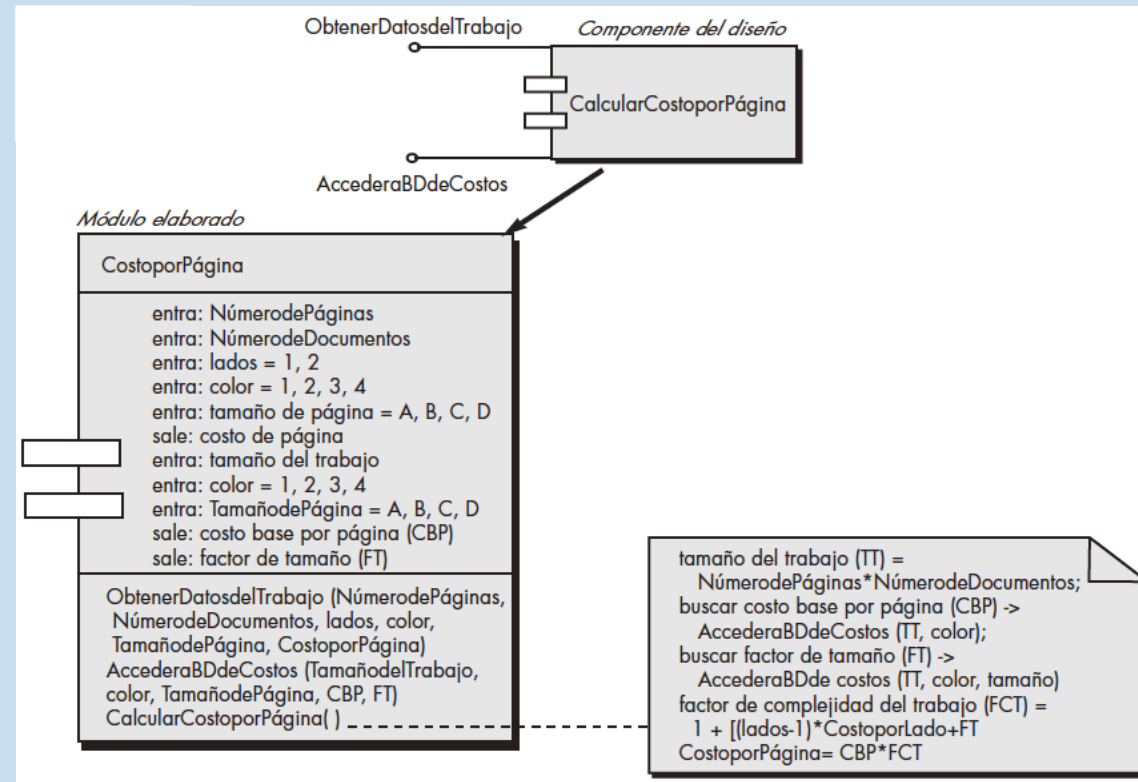
Se define las estructuras de datos.

El algoritmo que permite que el módulo cumpla su función prevista.

Ejemplo:

Módulo: CalcularCostoPorPágina

- Objetivo: Calcular el costo de impresión por página con base en las especificaciones dadas por el cliente.
- Datos requeridos: # pags, # docs, impresión por uno o dos lados, req. de color, y req. de tamaño.
- Los datos son pasados a través de la interfaz del módulo



Diseño en el nivel de componentes para CalcularCostoporPágina.

Figura 10.3 (Pressman, 2010)

Tres visiones de un componente

1. Visión orientada a objetos
2. Visión orientada a un elemento funcional (módulos) – vista tradicional.
3. Visión relacionada con el proceso

Visión relacionada con el proceso

La visión OO y la tradicional suponen que el componente se crea con base a las especificaciones obtenidas del modelo de requerimientos.



Otro enfoque se basa en componentes reutilizables de software y patrones ya existentes.

Conforme se desarrolla la arquitectura de software, se escogen los componentes a reutilizar.

Se dispone de
descripción de su
interfaz,

funciones que
realizan,

y la comunicación
y colaboración
que requieren.

Puntos importantes

- El proceso de diseño en el nivel de componentes incluye una secuencia de actividades que reduce poco a poco la abstracción con el que se representa el software.
- Es posible adoptar tres puntos de vista en el nivel de diseño:
 - Enfoque OO.- Clases de diseño que provienen del dominio del problema y de la infraestructura.
 - Enfoque OM (tradicional).- Requiere representación de estructuras de datos, interfaces y algoritmos para un módulo del programa.
 - Enfoque relacionadas en el proceso.- Reutilización de componentes y patrones de software.



Bibliografía y Recursos útiles

- Sommerville, I. (2011). Ingeniería de Software. 9^{na} Edición. (Cap. 7.1).
- Pressman, R. (2010). Ingeniería del Software. Un Enfoque Práctico. 7^{ma} Edición. (Cap. 10.1, 10.3).