

Desarrollo de Aplicaciones Basadas en Componentes

Luis G. Montané Jiménez

Tópicos Selectos de Computación I

Licenciatura en Informática, Facultad de Estadística e
Informática

Febrero 2017



Objetivo de la presentación

- ▶ Abordar los principios teóricos, modelado y diseño de componentes, incluyendo ventajas y desventajas

Agenda

- ▶ **Introducción**
- ▶ **Modelado y Diseño**
- ▶ **Ventajas y Desventajas**
- ▶ **Ejemplos**

Introducción

- ▶ Actualmente sistemas complejos y de alta calidad se deben construir en períodos de tiempo cortos
- ▶ Esto obliga un enfoque de reutilización más organizado
- ▶ La programación tradicional se ha visto incapaz de tratarlos de una forma natural
- ▶ La POO ha sido el sustento de la ingeniería del software para los sistemas cerrados
- ▶ Sin embargo, se ha mostrado insuficiente al tratar de aplicar sus técnicas para el desarrollo de aplicaciones en entornos abiertos

POO

- ▶ Hace prevalecer la visión de **objeto** sobre la de **componente**, estos últimos como unidades de composición independientes de las aplicaciones
- ▶ Asimismo, no toma en cuenta factores necesarios como la distribución, adquisición e incorporación de componentes a los sistemas

Programación Orientada a Componentes (POC)

- ▶ Nace como una extensión natural de la **orientación a objetos** para los entornos abiertos
- ▶ Este paradigma promueve el desarrollo y utilización de componentes reutilizables dentro de lo que sería un mercado global de software
- ▶ La Ingeniería de Software Basada en Componentes (ISBC) se centra en el diseño y construcción de sistemas basados en computadoras que utilizan componentes de software reutilizables

¿Qué es un componente?

- ▶ Un componente es:
 - ▶ Paquete de software (modular, encapsula un conjunto de funciones y es reutilizable)
 - ▶ Puede ser independientemente remplazado (sustituible)
 - ▶ Provee y requiere servicios ambos basados en interfaces específicas

Definición de componentes (1/3)

Szyperski, 2002

- *“Un **componente** es una unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente...”*

Definición de componentes (2/3)

- ▶ Microsoft:

- ▶ *“Pieza de código pre-elaborado que encapsula alguna funcionalidad expuesta a través de interfaces estándar. Los componentes son los “ingredientes de las aplicaciones”, que se juntan y combinan para llevar a cabo una tarea”*

Definición de componentes (3/3)

► Componente

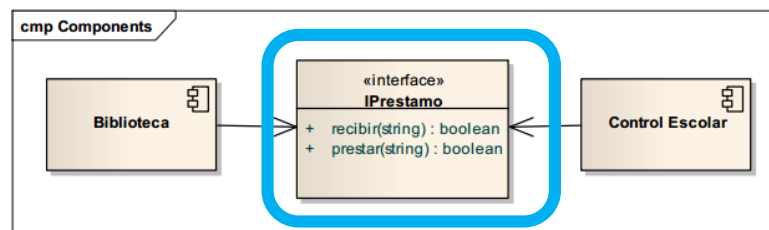
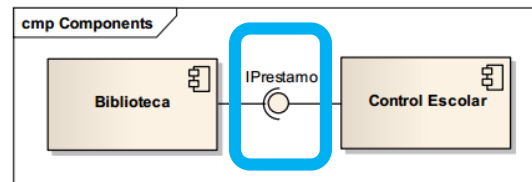
- Una parte reemplazable, casi independiente y no trivial de un sistema que cumple una función clara en el contexto de una arquitectura definida.
- Se requiere que los componentes estén **empaquetados** de forma que permitan su distribución y composición con otros componentes, especialmente con aquellos desarrollados por terceras partes.

Propiedades

- ▶ **Identificable:** Debe tener una identificación que permita acceder fácilmente a sus servicios y que permita su clasificación.
- ▶ **Remplazable:** Se puede remplazar por nuevas versiones u otro componente que lo mejore.
- ▶ **Acceso a través de su interfaz:** Debe asegurar que estas no cambiaran a lo largo de su implementación.
- ▶ **Servicios no varían:** Las funcionalidades ofrecidas en su interfaz no deben variar, pero su implementación sí.

Interfaces

- Las interfaces son colecciones de uno o más métodos que pueden o no contener atributos



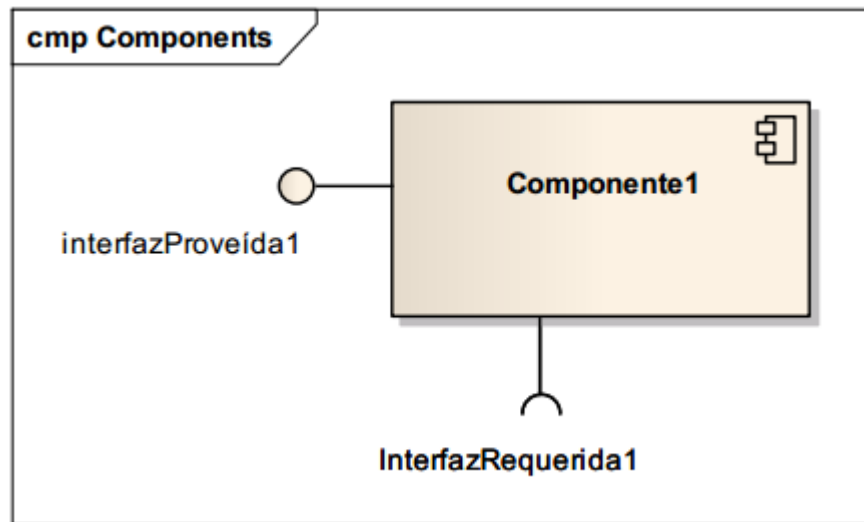
Agenda

- ▶ Introducción
- ▶ **Modelado y Diseño**
- ▶ Ventajas y Desventajas
- ▶ Ejemplos

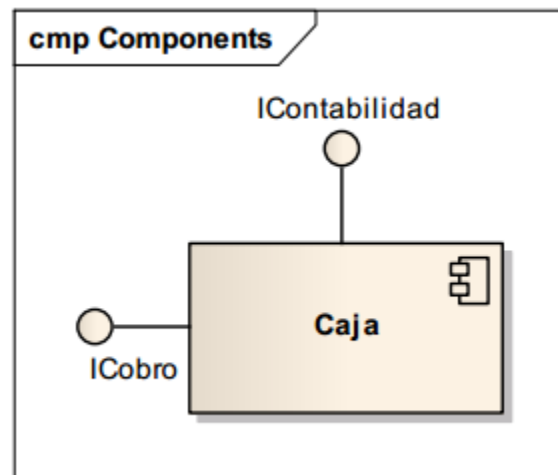
Conceptos clave

- ▶ Existen tres bloques de construcción en la descripción de una arquitectura de software:
 - ▶ Componentes
 - ▶ Conectores
 - ▶ Configuración

Representación

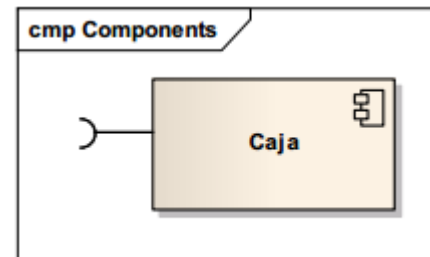
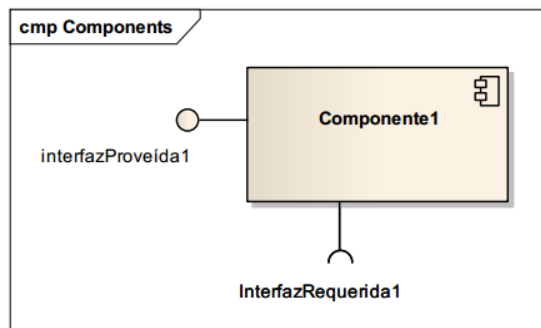


Componente que provee dos interfaces



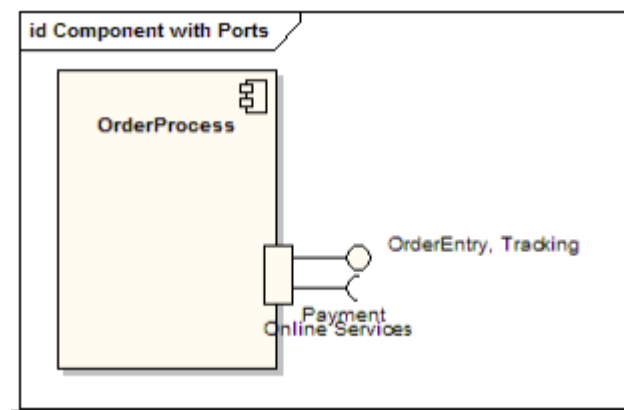
Interfaz requerida/provista

- El conector Ensamble une la interfaz requerida del componente (Componente1) con la interfaz proporcionada de otro componente (Component2); esto permite que un componente provea los servicios que otro componente requiere

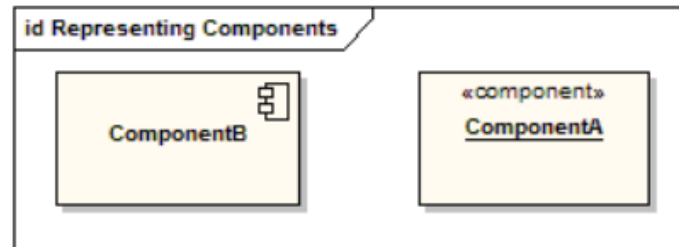
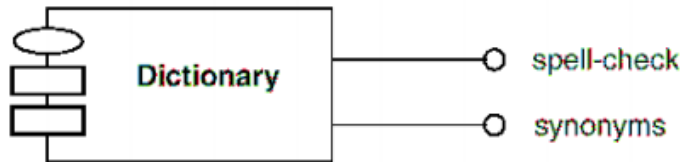


Puertos

- Usar puertos permite que se especifique un servicio o comportamiento a su entorno así como también un servicio o comportamiento que un componente requiere. Los puertos pueden especificar entradas, salidas así como también operar bidireccionalmente.



Representación

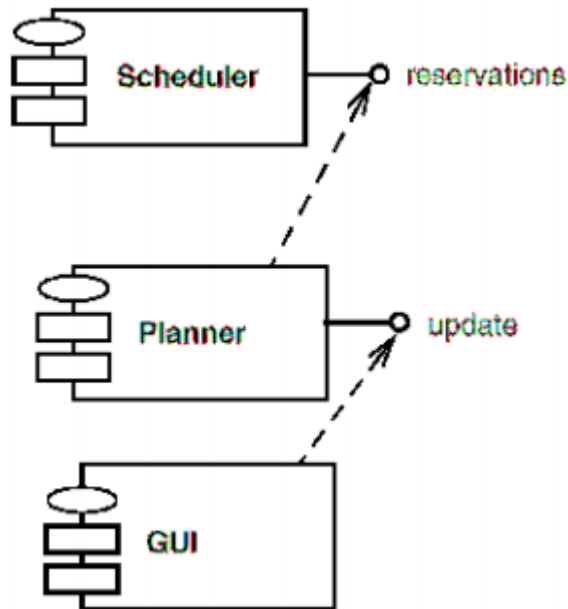


Clasificación de diagramas

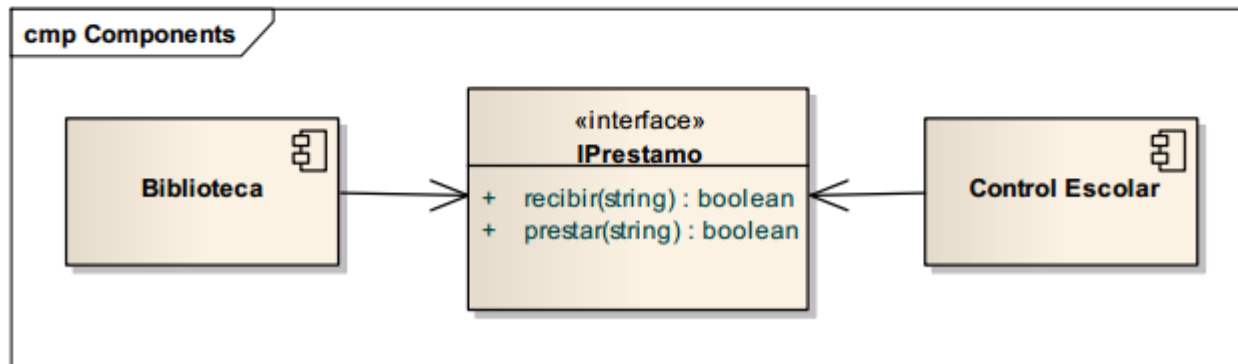
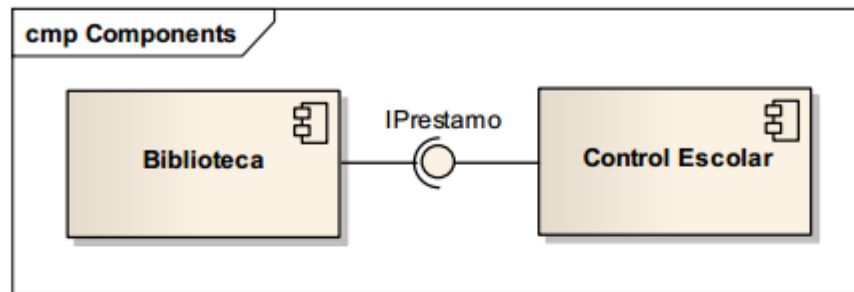
- ▶ Estáticos
 - ▶ Diagramas de casos de uso
 - ▶ Diagramas de clases
- ▶ Dinámicos
 - ▶ Diagramas de estado
 - ▶ Diagramas de actividad
 - ▶ Diagramas de secuencia
 - ▶ Diagramas de colaboración
- ▶ Implementación
 - ▶ *Diagrama de componentes*
 - ▶ *Diagrama de despliegue*

Diagrama de componentes

- Un diagrama de componentes muestra la organización y dependencias entre un conjunto de componentes



Representación y ensambles



Características

- ▶ **Documentado:** Un componente debe estar correctamente documentado para facilitar su búsqueda si se quiere actualizar, integrar con otros, adaptarlo, etc.
- ▶ **Genérico:** Sus servicios deben servir para varias aplicaciones.
- ▶ **Reutilizado dinámicamente:** Puede ser cargado en tiempo de ejecución en una aplicación.
- ▶ **Independiente de la plataforma:** Hardware, Software, S.O.

¿Qué es la arquitectura de un sistema?

- ▶ La vista general sobre la composición del sistema
 - ▶ Componentes y relaciones
 - ▶ Distribución
 - ▶ Estructura
 - ▶ Heterogeneidad: ligas con sistemas externos, diferentes plataformas, etc.

Composición de componentes

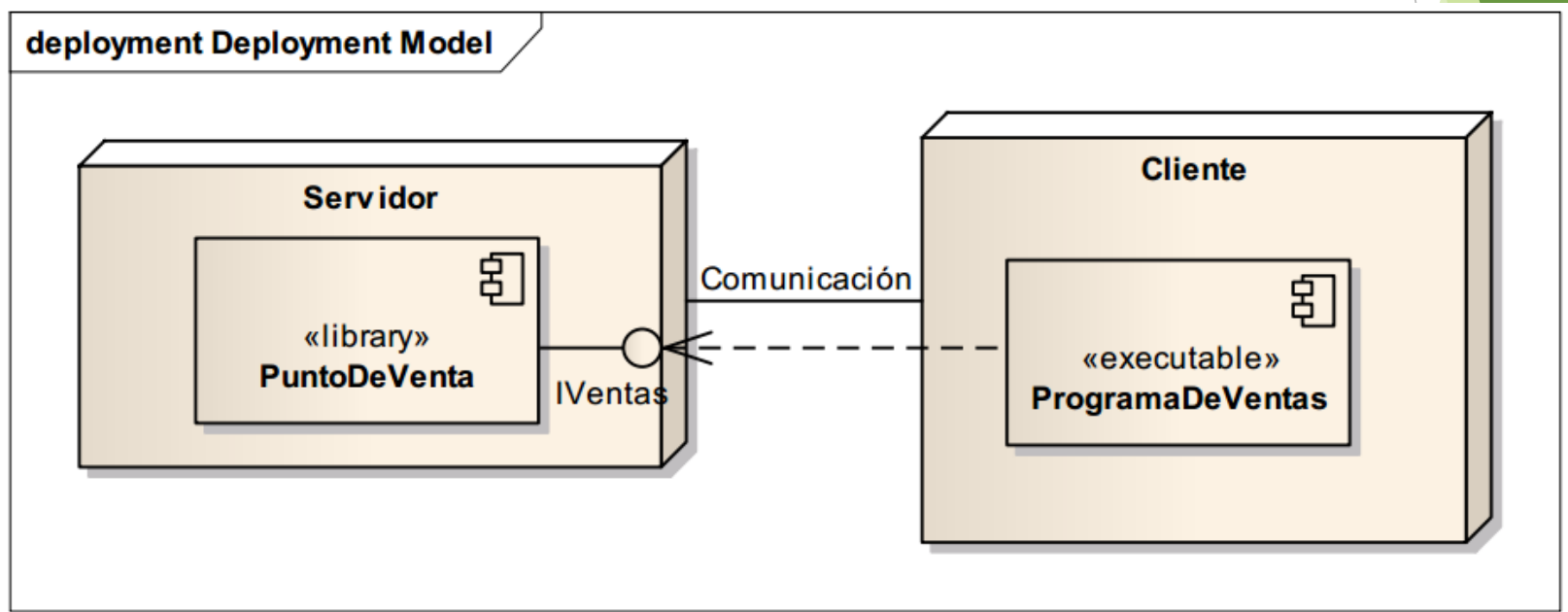
- ▶ Ingredientes arquitectónicos
 - ▶ Modelo de intercambios de datos (transferencia de datos)
 - ▶ Automatización (facilitar la interacción entre componentes)
 - ▶ Almacenamiento estructurado (organización en almacenamiento)
 - ▶ Modelo de objetos subyacente (asegura interoperabilidad entre los componentes)

Definiciones de arquitectura de software

- ▶ Perry & Wolf
 - ▶ Arquitectura de software = {Elementos (Qué), Forma (Cómo), Razonamiento (Por qué)}
- ▶ Kruchten
 - ▶ Trata del diseño e implementación del más alto nivel de estructura del software
 - ▶ Arquitectura = abstracción, descomposición, estilo, estética
- ▶ Shaw & Garlan
 - ▶ Arquitectura de software [es un nivel de diseño que] comprende la descripción
 - ▶ Elementos de cuales el sistema está construido
 - ▶ Interacciones entre estos elementos
 - ▶ Patrones que guían esta composición y restricciones de estos patrones

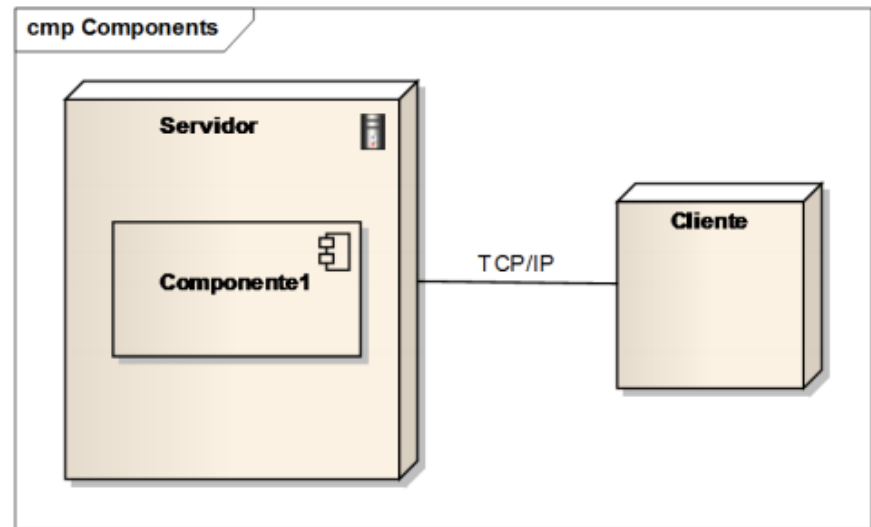
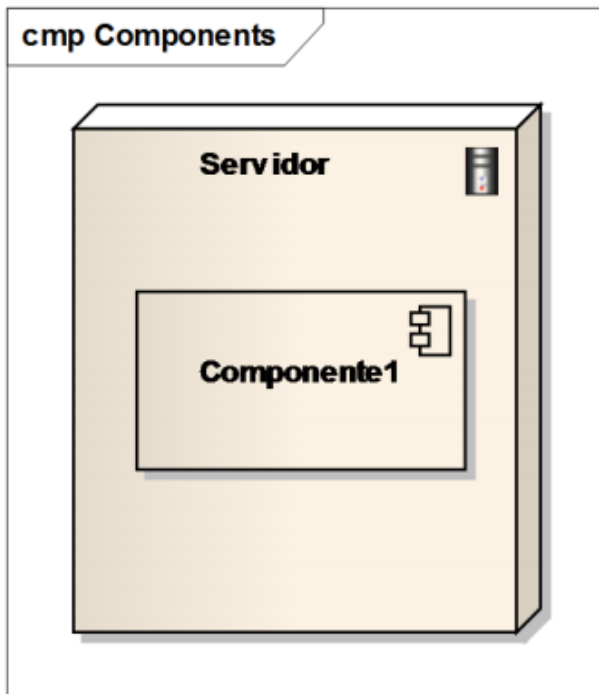
Diagrama de despliegue

- Muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos



Nodos

- Los nodos son los elementos donde se ejecutan los componentes.



Agenda

- ▶ Introducción
- ▶ Modelado y Diseño
- ▶ **Ventajas y Desventajas**
- ▶ Ejemplos

¿Por qué una arquitectura es importante?

Muchos sistemas terminan por ser un desastre

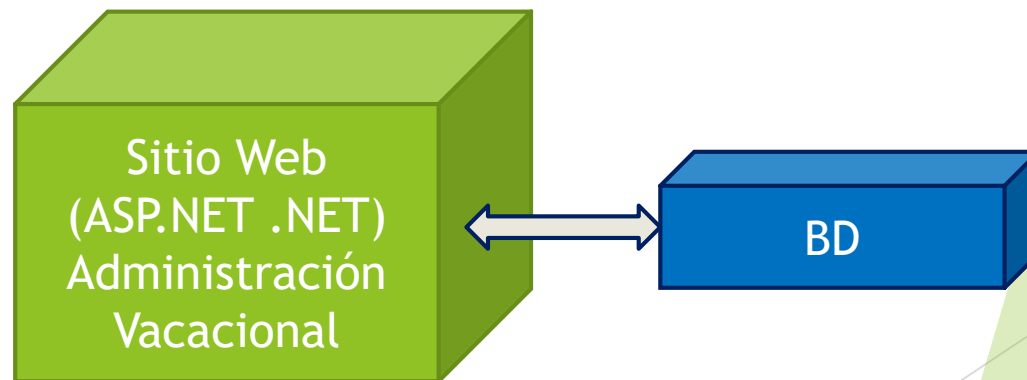
- ▶ Limitado rol dado a la arquitectura
- ▶ Arquitectura se convierte en un factor que se considera tardíamente en el desarrollo de software

¿Cómo el diseño de una arquitectura beneficia el desarrollo de software?

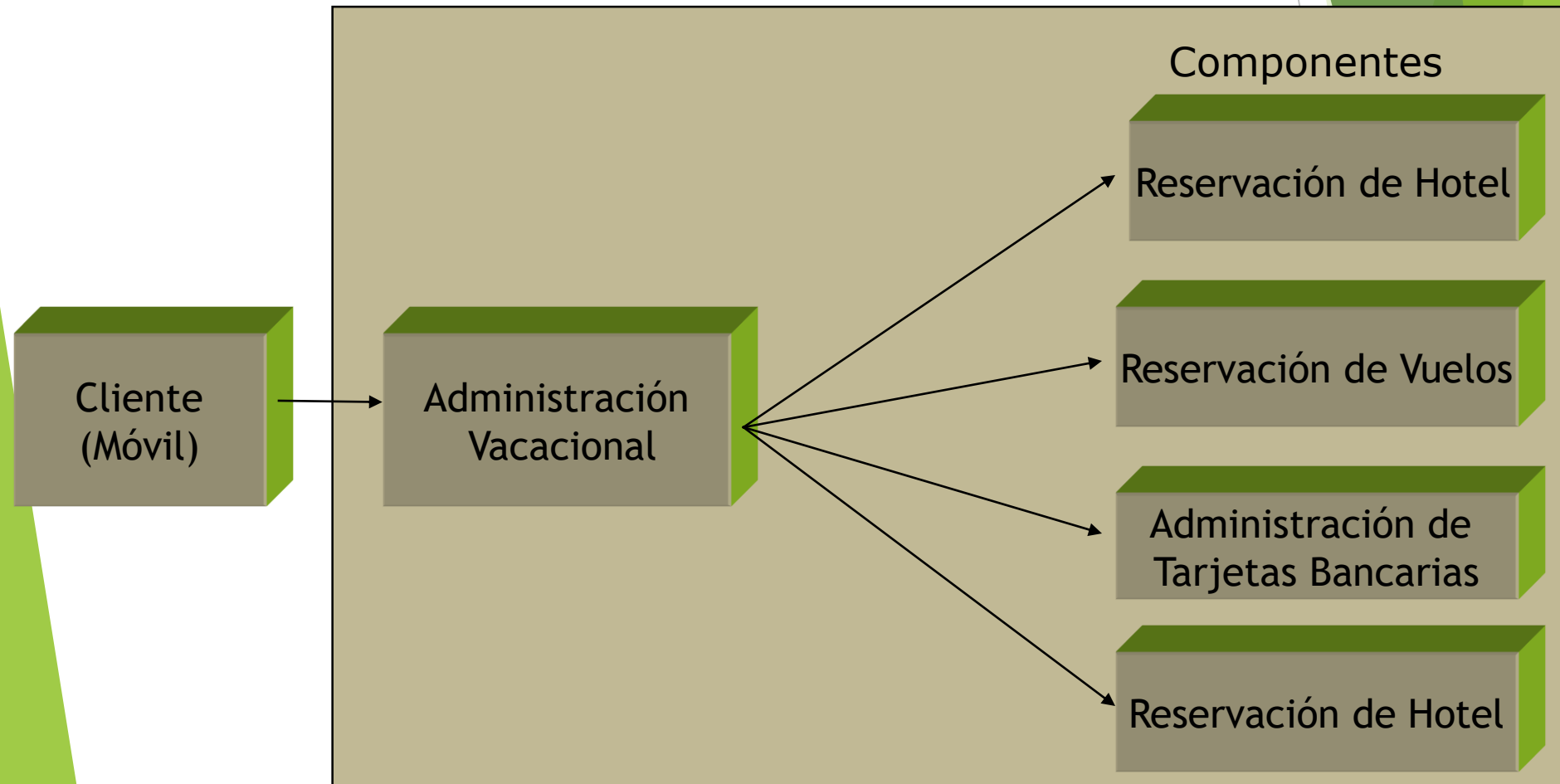
- ▶ El manejo de proyectos: en el desarrollo top-down, componentes pueden ser asignados a diferentes grupos de trabajo
- ▶ La coordinación centralizada de proyectos: se conoce como todas las piezas deben ser colocadas
- ▶ Las decisiones acerca del middleware, plataforma, estructura deben ser centralizadas

Mala arquitectura = escalabilidad pobre y difícil mantenimiento

- ▶ Toda la lógica del negocio esta codificada en el sitio Web
- ▶ Al estar todo mezclado es difícil de mantener



Buena arquitectura = buen desempeño



Ventajas/Desventajas

▶ Ventajas

- ▶ Reutilización del software
- ▶ Simplificación de pruebas
- ▶ Simplificación del mantenimiento
- ▶ Rendimiento
- ▶ Mayor calidad

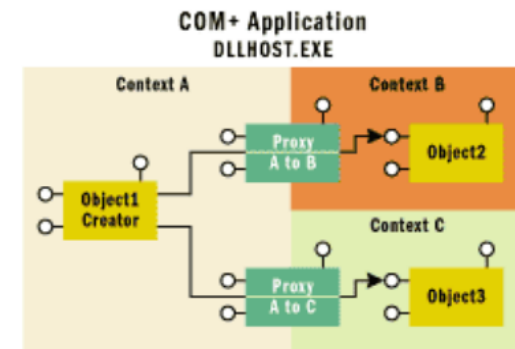
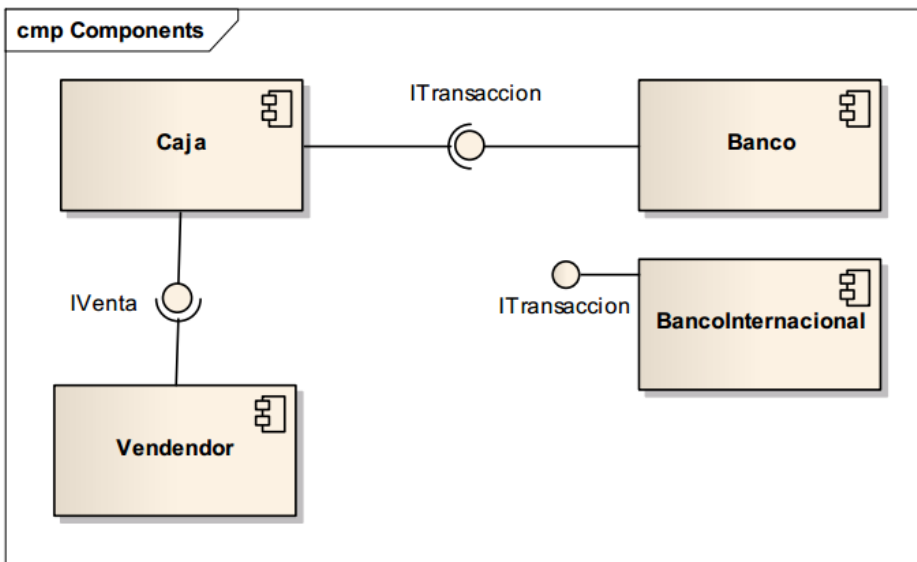
▶ Desventajas

- ▶ Cambia el enfoque de los diseñadores
- ▶ Posiblemente requiera de aprender nueva tecnología y teoría
- ▶ Seguridad

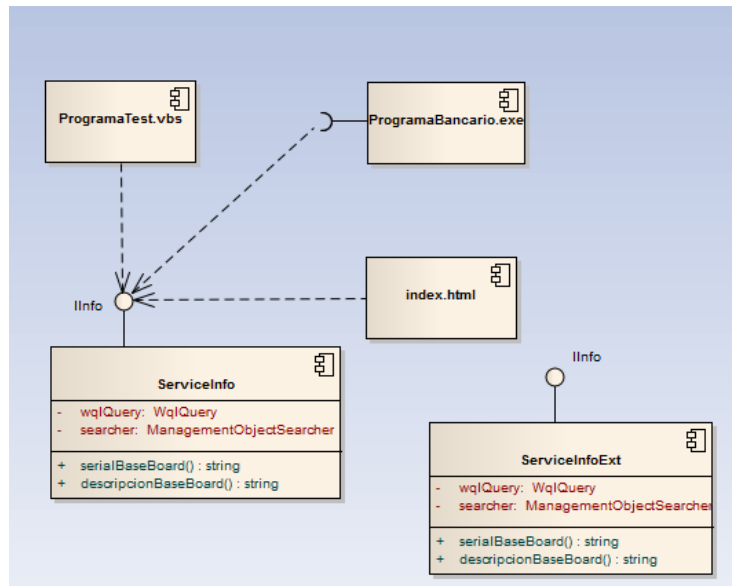
Agenda

- ▶ Introducción
- ▶ Modelado y Diseño
- ▶ Ventajas y Desventajas
- ▶ **Ejemplos**

Ejemplos (1/2)



Ejemplos (2/2)



Modelo de componentes

- ▶ Define la forma de sus interfaces y los mecanismos para interconectarlos entre ellos.
- ▶ Determinan los mecanismos de composición
 - ▶ Ejemplos:
 - ▶ DCOM, COM, JavaBeans, CORBA

Plataforma de componentes

- ▶ Es un entorno de desarrollo y de ejecución de componentes que **permite aislar la mayor parte de las dificultades conceptuales y técnicas** que conlleva la construcción de aplicaciones basadas en los componentes de ese modelo.
- ▶ En este sentido, podemos definir una plataforma como una implementación de los mecanismos del modelo, junto con una serie de herramientas asociadas.

Referencias

- ▶ Perdita Stevens; R. J. Pooley (2007). Utilizacion de UML en Ingenieria del Software con Objetos y Componentes. 2/ed. Publisher: Pearson Education.
- ▶ Roger S. Pressman (2002). Ingeniería de Software Un Enfoque Práctico. McGrawhill. Steve McConnell.
- ▶ Clemens Szyperski (2002). Component Software: Beyond Object-Oriented Programming. 2nd. Addison-Wesley Longman Publishing.

Desarrollo de Aplicaciones Basadas en Componentes

Luis G. Montané Jiménez

Tópicos Selectos de Computación I

Licenciatura en Informática, Facultad de Estadística e
Informática

Febrero 2017

