Procesos de Ingeniería del Software
Tema 3

Tecnologías de componentes y construcción de aplicaciones empresariales

Universidad de Cantabria – Facultad de Ciencias

Patricia López Martínez

Bibliografía

- Bibliografía Básica
  - Martin Fowler (2003): Patterns of Enterprise Application Architecture, Addison Wesley
    - Introducción (Disponible en la preview de Amazon)
  - Ian Sommerville (2009): Software Engineering, 9th Edition, Pearson.
    - Capítulos 17 y 18
  - Alonso et al., “Web Services: Concepts, Architectures and Applications", Springer, 2004
    - Capítulo 1

- Bibliografía complementaria
  - Ivica Crnkovic et al (2002): Specification, Implementation and Deployment of Components, Communications to the ACM, 45 (10)
  - Ivica Crnkovic et al (2006): Component-based Development Process and Component Lifecycle, Intl. Conf. on Software Engineering Advances.

Contenido

- Aplicaciones empresariales
- Evolución de la arquitectura para aplicaciones empresariales
- Tecnologías de componentes

Aplicaciones empresariales

"Enterprise Applications are about the display, manipulation, and storage of large amounts of often complex data and the support or automation of business processes with that data"
(Martin Fowler)

- Satisface necesidades de negocio de una organización
  - También denominadas Sistemas de Información
- Características típicas:
  - Grandes cantidades de datos
  - Persistencia de datos
  - Altamente distribuidas
  - Accesos concurrentes
    - Gestión de transacciones
    - Especialmente importante en aplicaciones web
  - Variedad de interfaces de usuario
  - Interacción con aplicaciones heterogéneas
- Deben proporcionar:
  - Rendimiento
  - Escalabilidad
  - Fiabilidad
  - Disponibilidad
  - Interoperabilidad
  - Seguridad

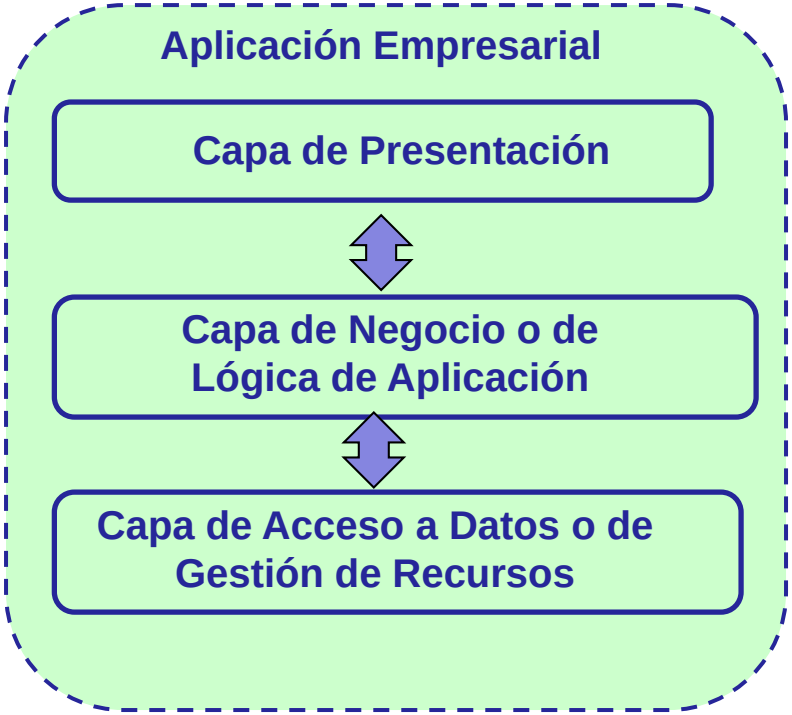
Contenido

- Aplicaciones empresariales
- **Evolución de la arquitectura para aplicaciones empresariales**
- Tecnologías de componentes

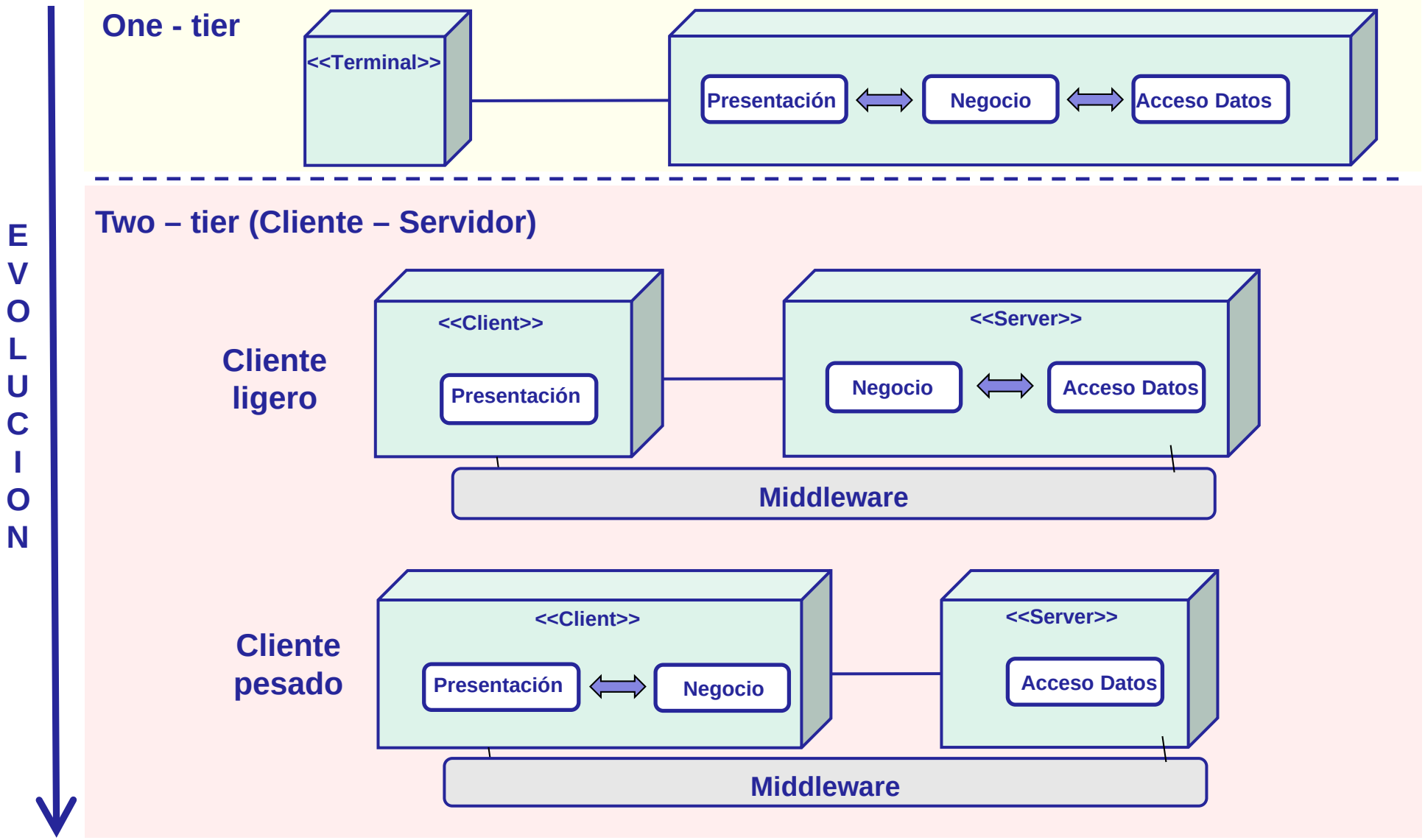
Arquitectura lógica

- Modelo de capas (layers)

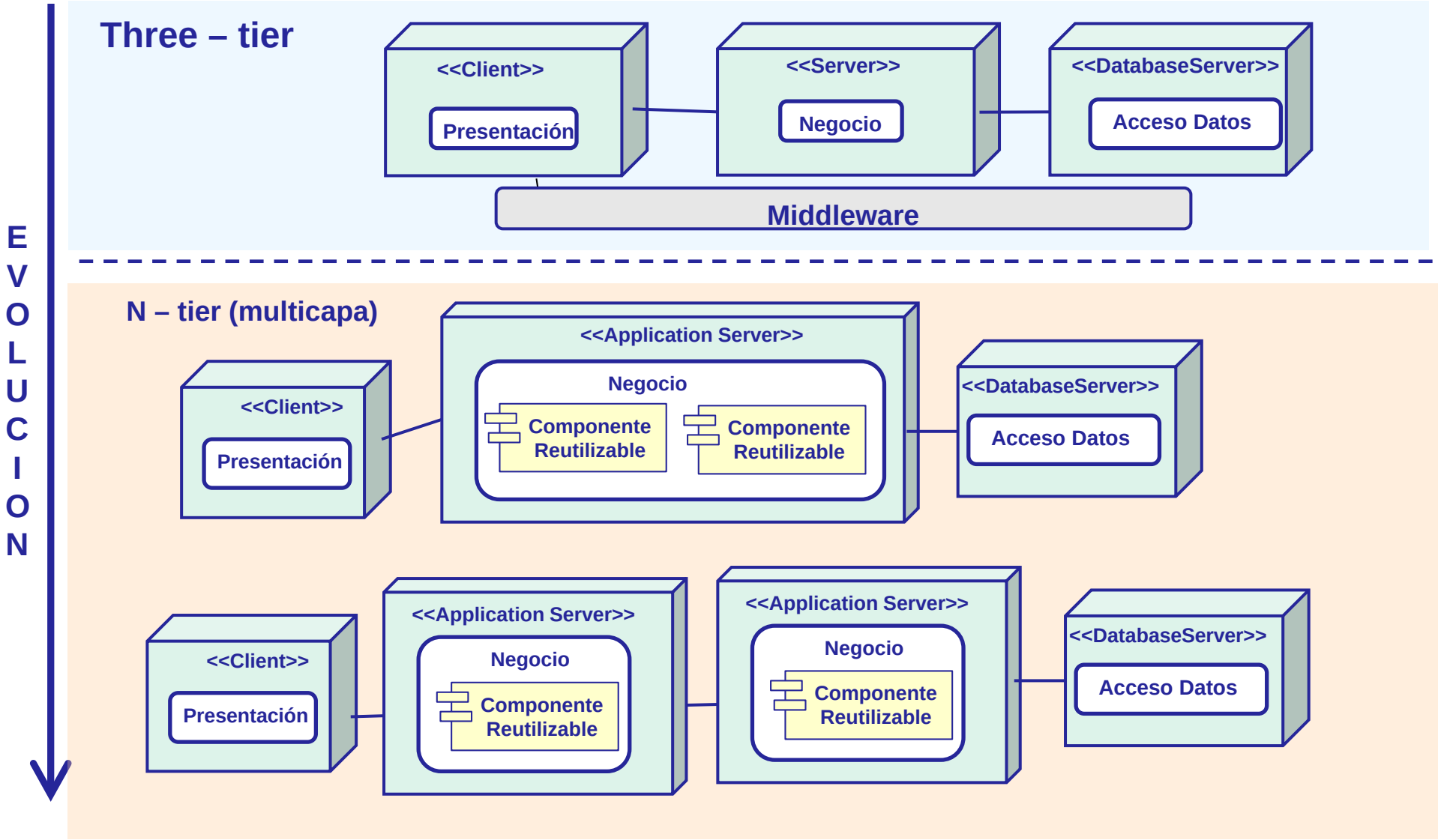
Cliente



Evolución de la arquitectura física



Evolución de la arquitectura física



Evolución del middleware para sistemas distribuidos

EVOLUCION

- Sockets
- Remote procedure call (RPC)
  - Application Programming Interface
- Orientado a objetos distribuidos
  - Java RMI (Remote Method Invocation)
  - CORBA (Common Object Request Broker Architecture) de la OMG => Multilenguaje y multiplataforma
- **Orientado a componentes distribuidos => Servidores de aplicación**
  - OO extendida con composición y reutilización (Componentes reutilizables)
    - **EJB**: Enterprise Java Beans (Extensión de JAVA RMI) (Plataforma Java EE)
    - CCM: CORBA component model (OMG)
    - .NET (Plataforma Microsoft)
- Orientado a servicios
  - Servicios Web

"Redes y Sistemas Distribuidos"

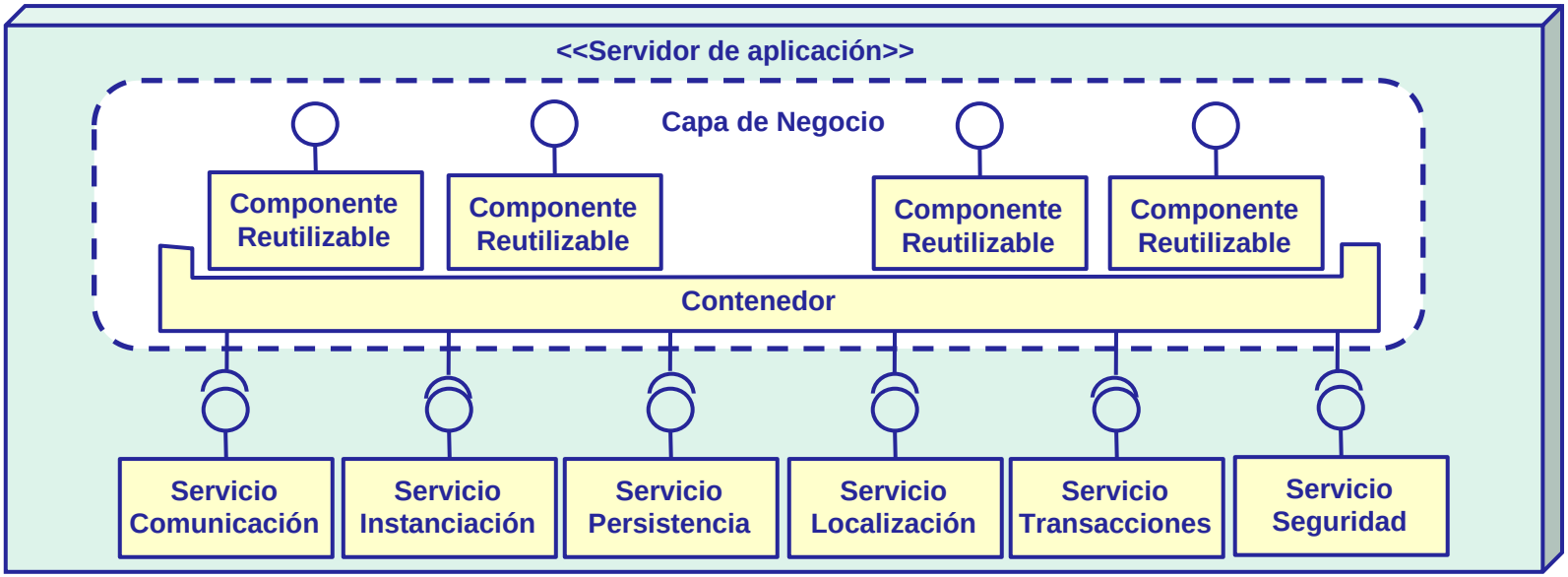
"Procesos de la Ingeniería SW"

"Servicios Software"

Servidores de aplicación

- Un servidor de aplicaciones proporciona un entorno para la ejecución de componentes de negocio reutilizables en aplicaciones distribuidas multicapa

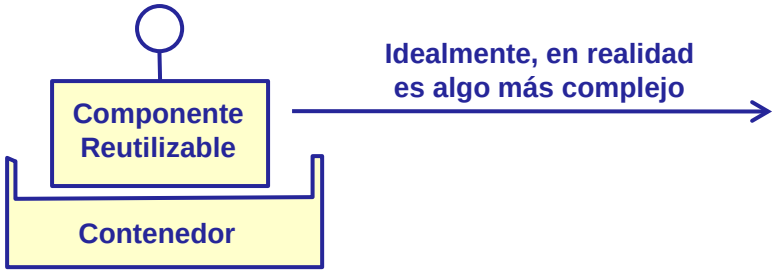
- El servidor de aplicaciones proporciona separación entre:
  - Lógica de negocio => Responsabilidad de los componentes de negocio
  - Gestión de los servicios comunes => Responsabilidad del contenedor
    - El contenedor ofrece capacidad para configurar el modo de uso de los servicios



Contenedor / Componente

- Proceso para añadir un nuevo usuario a un sistema

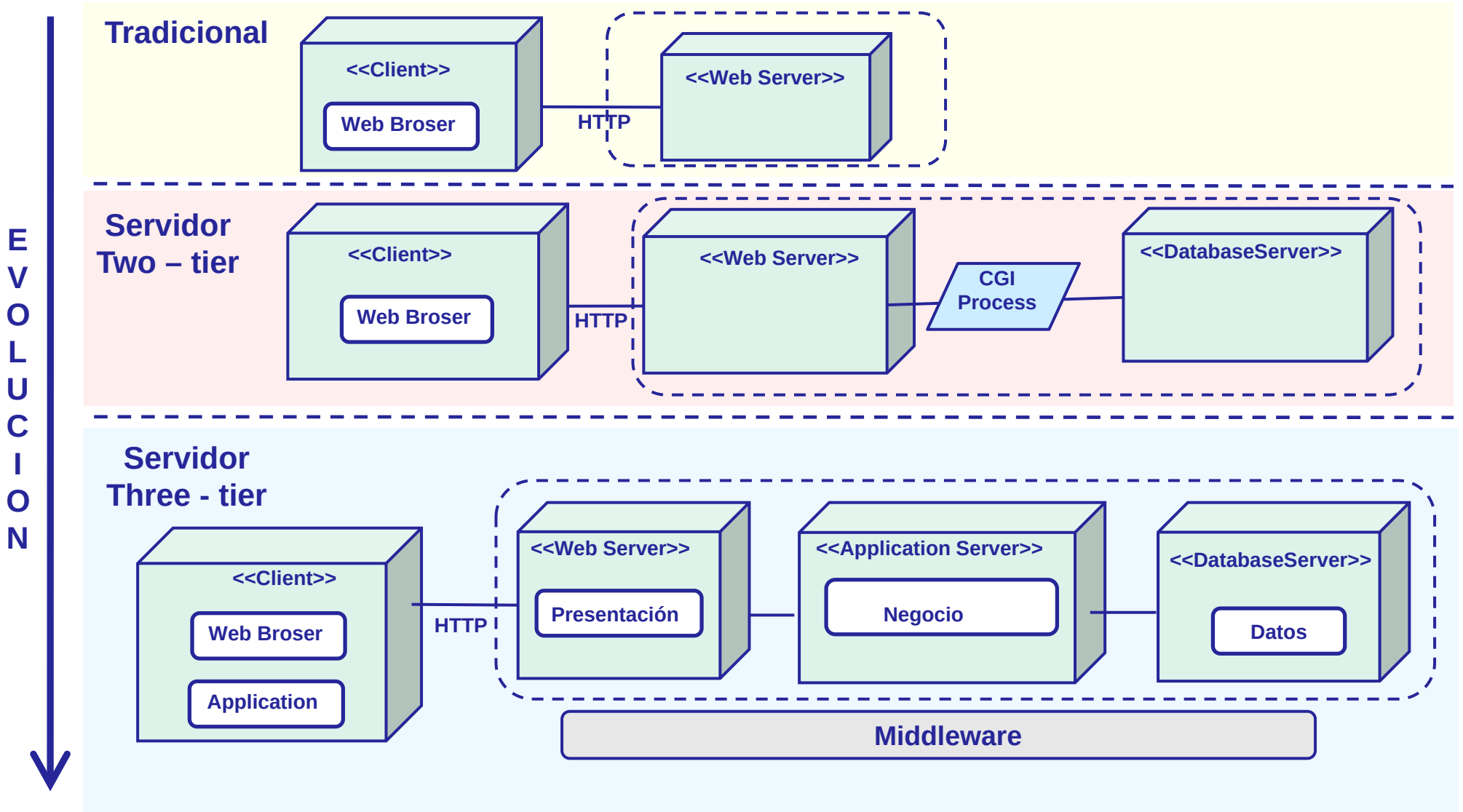
Seguridad	1. Comprobamos que tenemos permiso para añadir usuarios
Transacciones	2. Iniciar una transacción para asegurar la coherencia
Concurrencia, rendimiento	3. Conseguir una tarea ejecutora (unidad de concurrencia) de un posible pool de tareas
Persistencia	4. Acceso a la base de datos
Lógica de negocio	5. Almacenar el usuario
Gestión de recursos	6. Conseguir acceso al servidor de correo
Lógica de negocio	7. Enviar un email de confirmación
Concurrencia, rendimiento	8. Devolver la tarea ejecutora al pool de tareas
Transacciones	9. Finalizar la transacción



Idealmente, en realidad es algo más complejo

```
public void registerUser (User user) {
    database.storeUser(user);
    mailService.emailUser(user);
}
```

Arquitectura de la WEB: Evolución



Servidor web vs Servidor de aplicación

- **Servidor web**
  - Software o computador que atiende exclusivamente **peticiones HTTP**
  - Las respuestas HTTP correspondientes pueden corresponder a:
    - Páginas HTML estáticas
    - Páginas HTML dinámicas, a través de invocación de Scripts: CGI, Perl, PHP, ASP, Servlets, JSP, etc.
  - El cliente suele ser un web browser (o un servicio web)
  - Ejemplos: Apache, Apache Tomcat y Jetty (Java), IIS (Microsoft)

- **Servidor de aplicación**
  - Software o computador que proporciona acceso a **lógica de negocio** a través de **diferentes protocolos** (incluyendo HTTP)
  - Los clientes pueden ser aplicaciones de escritorio, servidores web u otros servidores de aplicación
  - La lógica de negocio se expone siguiendo algún **modelo de componentes** como EJB, .NET, etc.
  - Gestiona aspectos como seguridad, transacciones, concurrencia, etc.
  - Ejemplos: Tomcat EE, Glassfish, JBoss, etc. (Java), .NET Framework (Microsoft)

- La mayoría de los servidores de aplicación son también servidores web (no al revés)

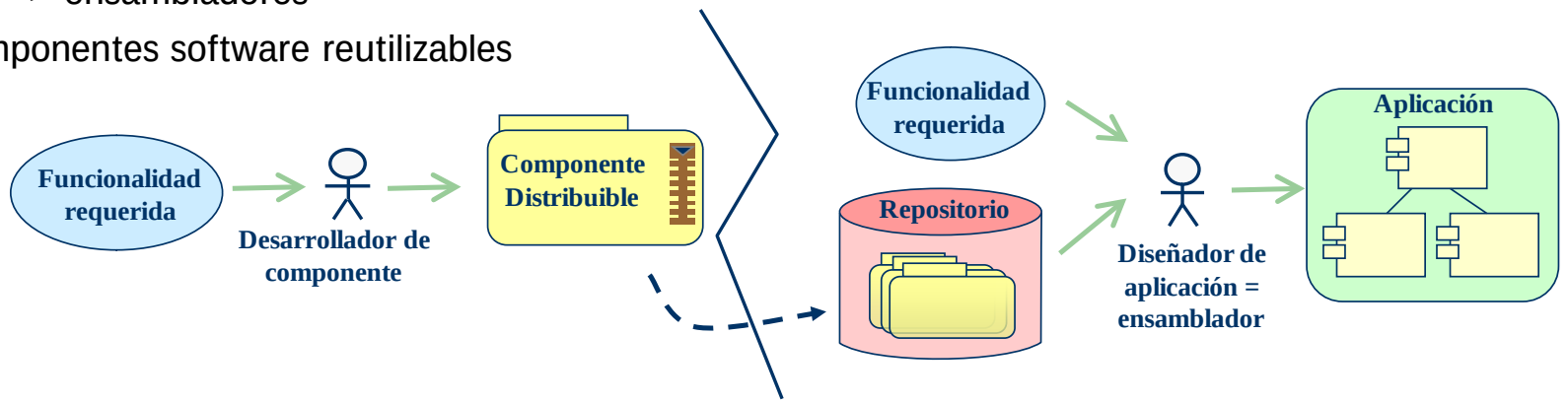
Contenido

- Aplicaciones empresariales
- Evolución de la arquitectura para aplicaciones empresariales
- **Tecnologías de componentes**

Desarrollo de software basado en componentes (CBD)

> "Creo que la industria del software carece de unas bases robustas en parte debido a la falta de una sub-industria de componentes. Tal industria sería increíblemente exitosa"
> (Douglas McIlroy, OTAN Software Engineering Conference, 1968)

- Construcción de aplicaciones mediante ensamblado de módulos software reutilizables, que han sido diseñados previamente con independencia de las aplicaciones en las que van a ser utilizados
  - Industrialización del desarrollo de software
  - Programadores => ensambladores
  - Mercado de componentes software reutilizables

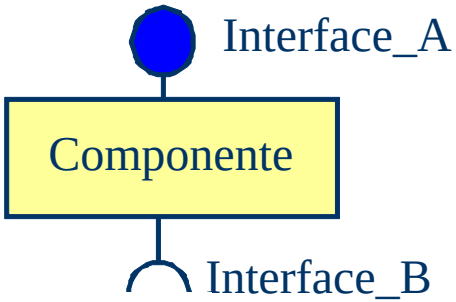


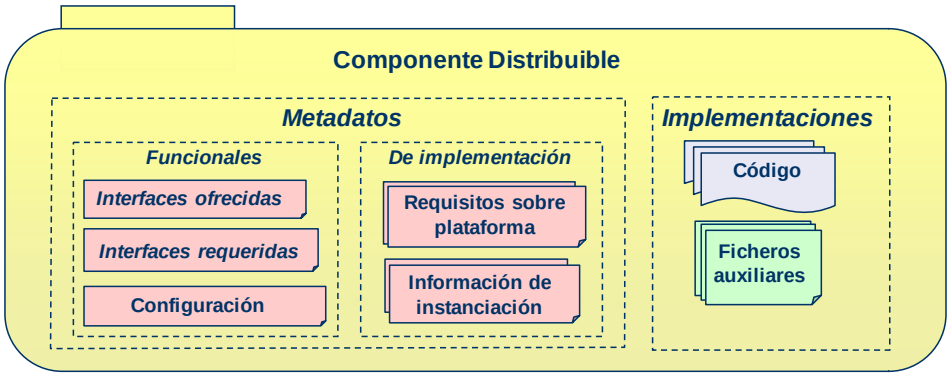
- Objetivos
  - Construir software más rápidamente adaptable a cambios
  - Gestionar la creciente complejidad del software

Definición de componente software

"Un componente software es una unidad de composición con un conjunto de interfaces definidas por contrato y dependencias explícitas del contexto. Un componente puede ser desplegado de manera independiente y está sujeto a composición por terceros" (Szyperski, 1998)

- Unidad de **composición**
  - Composición basada en interfaces => Contrato de uso
  - Independiencia de la implementación
- Unidad de **despliegue independiente**
  - Contrato de instanciación
- Composición por terceros => Manejo **opaco**
  - Metadatos (descriptores, anotaciones, etc.)





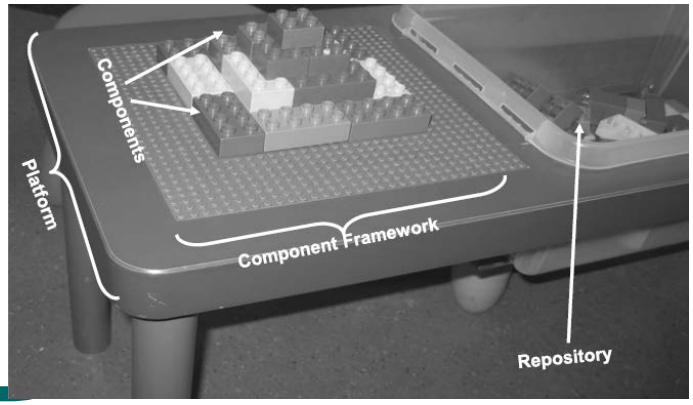
Modelo de componentes vs Framework de componentes

"Un componente es un elemento software conforme a un **modelo de componentes** estándar y que puede ser independientemente desplegado y compuesto sin modificación de acuerdo a un **estándar de composición**" (Councill and Heineman, 2001)

- Interoperabilidad => Componentes conformes a un **modelo de componentes**
  - Estandariza el modo en que se especifica, implementa y despliega un componente software
  - Vista del componente cómo **abstracción arquitectural**

- Composición y Ejecución => Infraestructura de soporte (**Middleware/Framework de componentes**)
  - Proporciona los servicios necesarios para instalar, ejecutar, y gestionar los componentes a lo largo de todo su ciclo de vida
  - Vista del componente como **implementación**

Modelo de componentes + framework = **Tecnología de componentes**

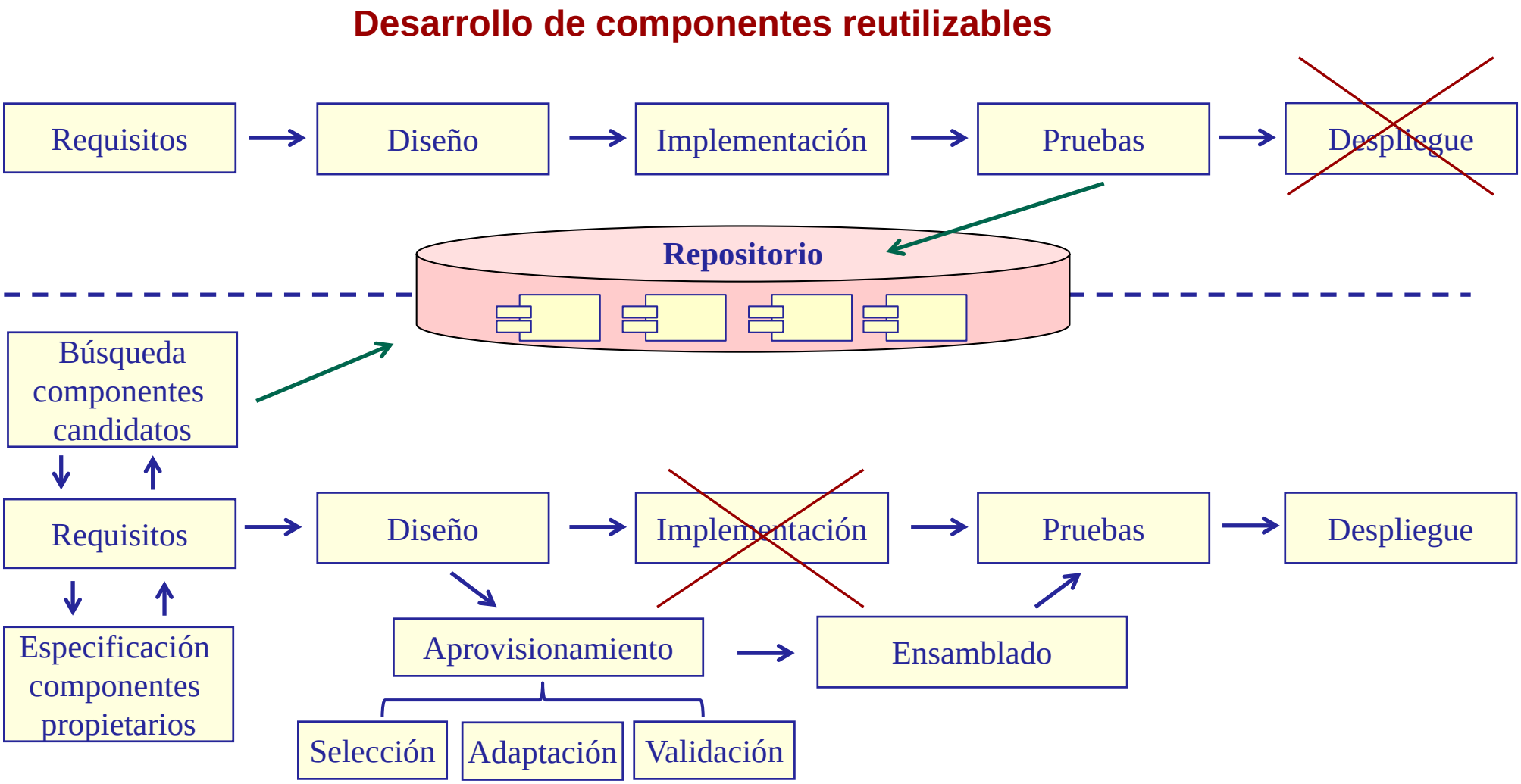


Tecnologías de componentes

- Una **tecnología de componentes**, por tanto, incluye:
  - **Modelo de componentes:** Marco conceptual para el diseño de componentes y de aplicaciones generadas por ensamblado de componentes
    - Define de forma exacta qué se entiende por componente, cómo se especifican componentes e interfaces, modos de interacción entre componentes, modo de empaquetamiento y despliegue, etc.
  - **Framework (middleware) de componentes**: Infraestructura de soporte necesaria para la ejecución de aplicaciones basadas en componentes
    - Proporciona servicios de soporte comunes a todos los componentes: gestión del ciclo de vida (instalación, activación, ejecución, actualización, etc.) , persistencia, comunicación, transacciones ...
    - Comúnmente denominados servidores de aplicación (sobre todo en dominio Java)
    - Generalmente basados en el modelo de contenedor/componente

- Ejemplos de tecnologías de componentes:
  - COM , COM+, y .NET (Multilenguaje y monoplataforma)
  - Sun Microsystems: JavaBeans (GUIs), Enterprise JavaBeans (Multiplataforma y monolenguaje)
  - OMG: CORBA Component Model (Multilenguaje y multiplataforma) – En desuso
  - OSGI Alliance: OSGI Bundles (Java)
    - Base de los plug-ins de Eclipse

Ciclo de vida de aplicaciones basadas en componentes



CBSE: Ingeniería del Software basada en componentes

- CBD implica un cambio profundo en los procesos de desarrollo de software
  - Provoca la aparición de una nueva disciplina dentro de la IS

- **Ingeniería Software basada en Componentes** (CBSE)
  - Define los procesos, métodos y herramientas que dan soporte al desarrollo de sistemas software construidos por ensamblado de componentes:
    - Soporte para la construcción de componentes software reutilizables
    - Soporte para la construcción de sistemas por composición de dichos componentes
    - Soporte para el mantenimiento de este tipo de sistemas, bien por sustitución o por introducción de nuevos componentes

Ventajas del uso de CBD

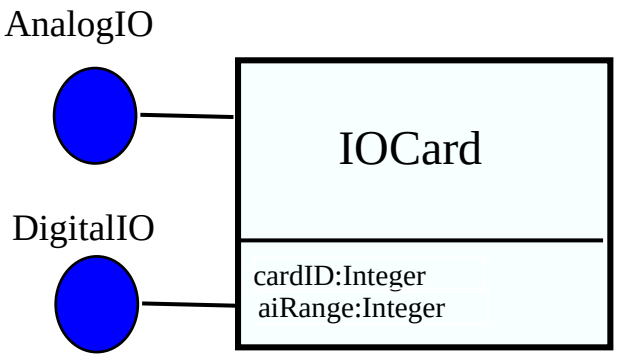
- Mejora de la productividad
  - Gracias a la reutilización de software
  - En una primera fase se produce el efecto contrario, al invertir tiempo y esfuerzo en conseguir reusabilidad
- Disminución de la complejidad del software
- Mejora de los tiempos de acceso al mercado
  - Consecuencia de los dos aspectos anteriores
- Incremento de la calidad del software
  - Siempre que se emplee un mercado de componentes certificados
- Mejora del mantenimiento
  - Errores mas fáciles de detectar y subsanar
  - Posibilidad de actualización dinámica

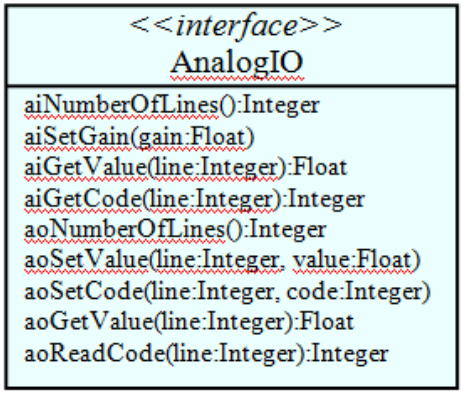
Limitaciones actuales del CBD

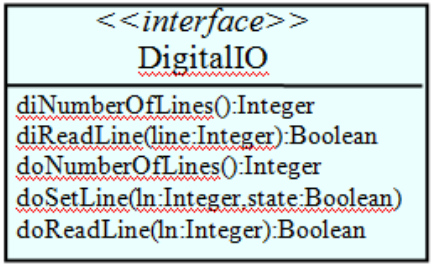
- Disponibilidad de componentes
  - Sólo existe en algunos campos: GUIs, ofimática, etc
  - Sí existen a nivel intraempresarial
- Falta de estandarización
  - Intereses empresariales
  - Interoperabilidad a través de pasarelas “bridges”
- Falta de procesos de certificación que garanticen la calidad de los componentes
- Falta de un proceso de ingeniería software basado en componentes bien definido
  - Riesgo para las empresas de cambiar sus procesos de desarrollo

Ejemplo de componente reutilizable

- Componente que permite la lectura y escritura de valores analógico/digitales a través de tarjetas de adquisición de datos
  - Requisitos
    - Independiente del fabricante
    - Independiente del número de líneas de la tarjeta (1..*)







- Se han desarrollado tres implementaciones (en Ada):
  - PCI9111 de la empresa AdLink Tech CO.
  - PCM3718 de Advantech
  - Tarjeta Virtual

```
interface Hello {
        string sayHello ();
    };

component HelloComp {
    supports Hello;
}
```