Maestría en Ingeniería
Arquitectura de Software

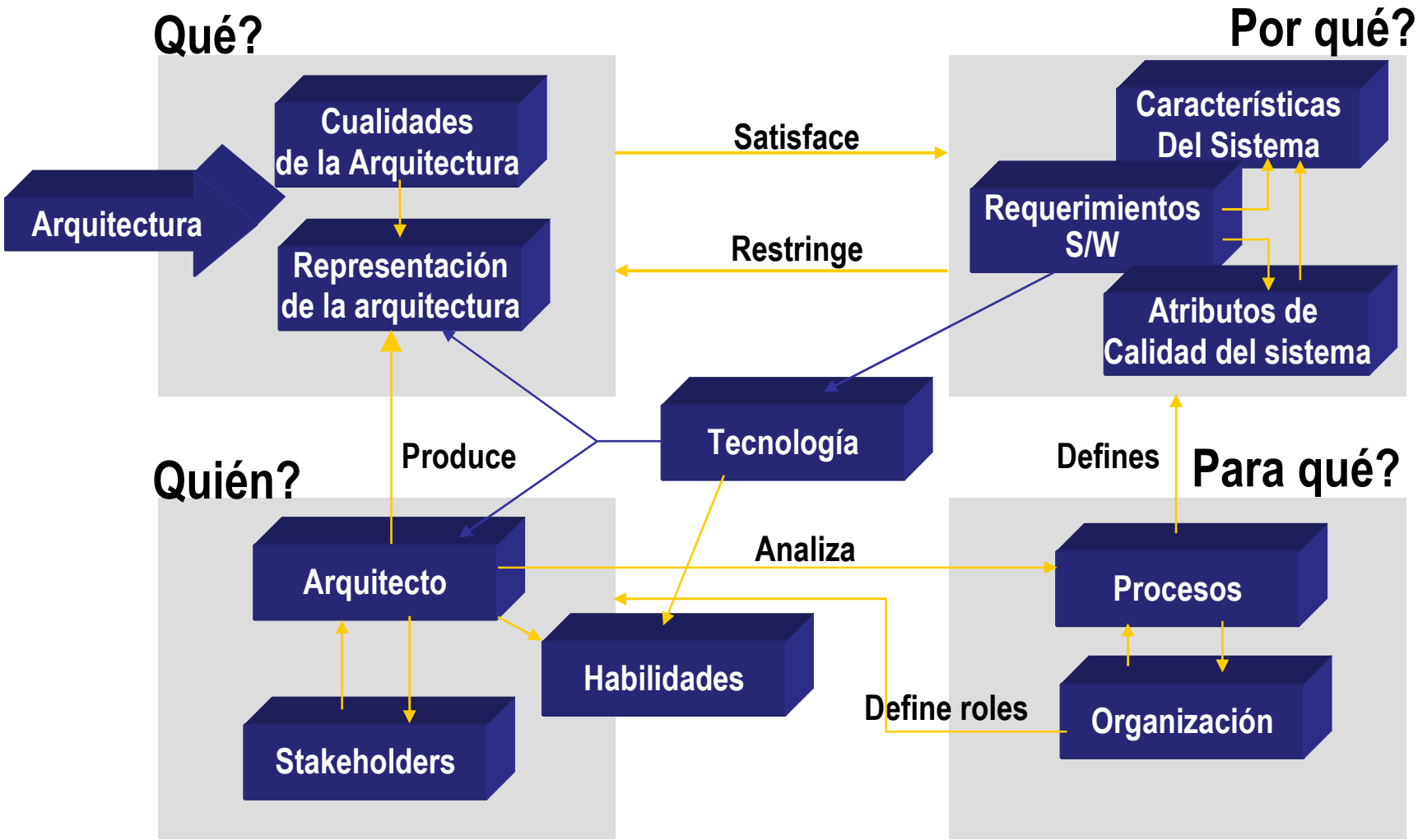
Sesión 3

Fernando Barraza A., Ms.C
fbarraza@javerianacali.edu.co

Sesión 3

- Objetivo: Exponer los conceptos básicos sobre la estructura de una arquitectura de software, así como sus principios y atributos de calidad principales.
- Temas:
  - Conceptos básicos: Componentes, conectores y configuraciones
  - Principios y atributos de calidad: Robustez, reusabilidad, portabilidad, mantenibilidad, etc.

Elementos relacionados con la arquitectura

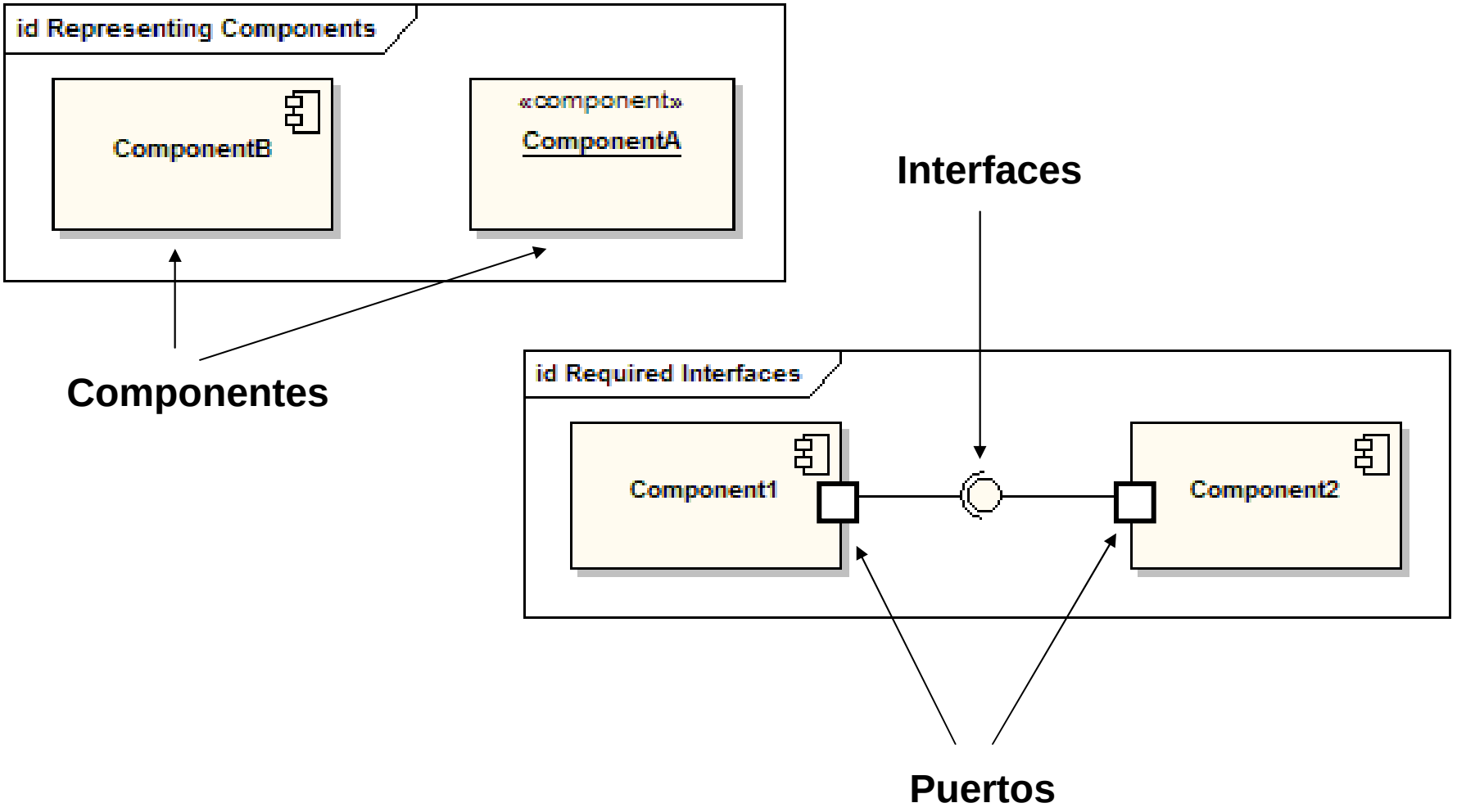


Fuente: Rational Software

Definición de estructura

- Componentes
- Interfaces (puertos)
- Conectores

Diagrama de componentes en UML

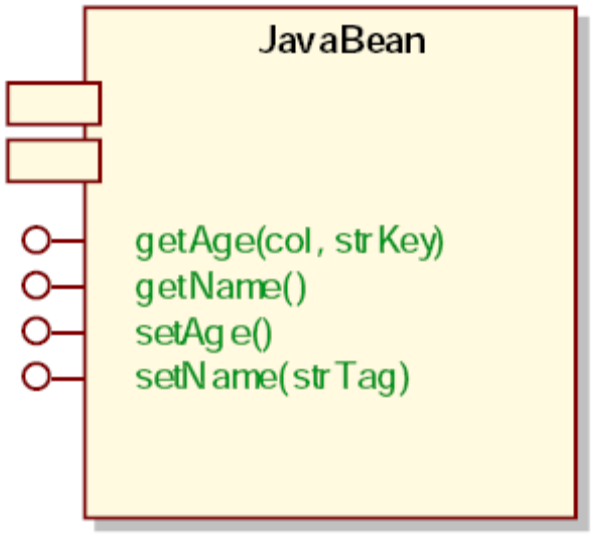


Componentes

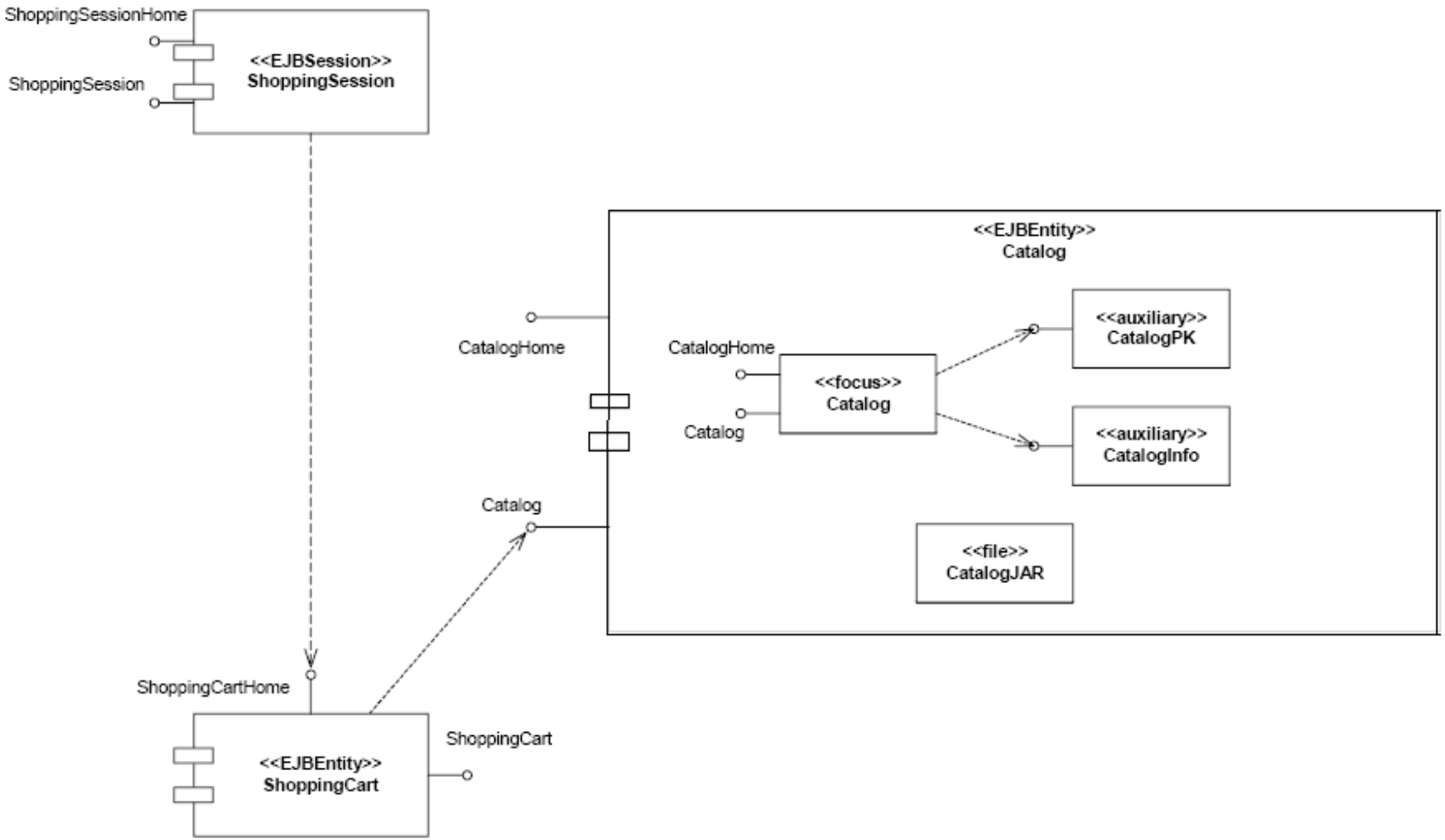
- Representan elementos computacionales y almacenamientos de un sistema. Por ejemplos:
  - Servidor de aplicaciones
  - Unidad física de software
  - Base de datos relacional
- Un componente se define siempre dentro de una familia de componentes

Detalle de los componentes

- Puede estar hecho de un conjunto interno de clases, paquetes de clases e interfaces
- Las interfaces exponen los puntos visibles de entradas a los servicios desde otros componentes



Ejemplo


ShoppingSessionHome
ShoppingSession
<<EJBSession>>
ShoppingSession
<<EJBEntity>>
Catalog
CatalogHome
CatalogHome
Catalog
<<focus>>
Catalog
<<auxiliary>>
CatalogPK
<<auxiliary>>
CatalogInfo
Catalog
<<file>>
CatalogJAR
ShoppingCartHome
<<EJBEntity>>
ShoppingCart
ShoppingCart

Interfaces

- Conectan los componentes.
- Los puntos de interfaz se llaman generalmente puertos (ports)
- Los puertos pueden definir interfaces tanto simples como complejas, desde una signatura de procedimiento hasta una colección de rutinas a ser invocadas en cierto orden, o un evento de multicast.

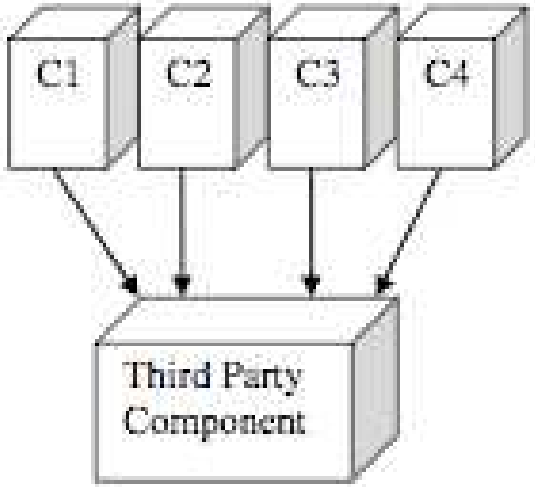
Conectores

- Son entidades que representan interacciones entre componentes.
- Los conectores también tienen interfaces que están definidas por un conjunto de roles.
- En la implemenación se utilizan diversas tecnologías, por ejemplo:
  - Invocador RPC.
  - Sockets
  - Memoria compartida.

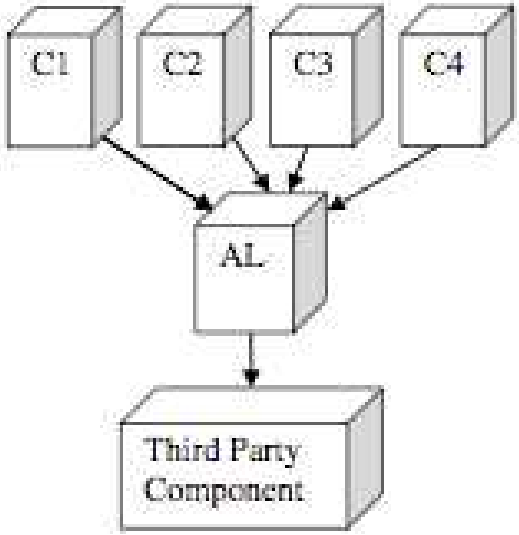
Propiedades de una AS

- Son inherentes a la estructura y configuración de una Arquitectura de Software definidas por la dependencia, comunicación, colaboración entre sus componentes.
- Algunas medidas usuales:
  - Dependencia: Acoplamiento
  - Colaboración: Cohesión
  - Comunicación: Latencia

Dependencia



Los componentes C1, .. C4 son dependientes del componente de tercera parte, si este ultimo se reemplaza por otro componente on una interfaz diferente, muy seguramente se requeriran cambios en C1, …, C4.



Solo el componente AL depende del Componente del tercero. Si este ultimo Es reemplazado solo AL debe revisar su Interfaz con dicho componente.

Colaboración

- Indica como dos o más componentes trabajan en conjunto para implementar una funcionalidad requerida por la aplicación.
- En metodologías “orientadas por responsabilidades”, cada componente asume por “contrato” una responsabilidad en el sistema.
- No es buena idea cargar de muchas responsabilidades a un mismo componente (baja cohesión)

Comunicación

- Resuelve como la información de datos y control fluye entre los componentes
- Se implementa mediante patrones o estilos
- Ejemplo de patrón de AS: Cliente-Servidor

La AS resuelve requerimientos no-funcionales

- **Restricciones técnicas:**
  - Requerimientos sobre las tecnologías que ciertas aplicaciones deben usar
  - “La aplicación debe desarrollarse en lenguaje Java y correr bajo sistema operativo Windows XP SP2”.
  - Usualmente son requerimientos no negociables.

La AS resuelve requerimientos no-funcionales (2)

- **Restricciones del negocio:**
  - Relacionados con el contexto del negocio sin tener razones técnicas
  - “La aplicación debe tener interfaz con las herramientas X y Y”
  - Al igual que las restricciones técnicas suelen ser no negociables.

La AS resuelve requerimientos no-funcionales (3)

- **Atributos de calidad:**
  - Definen aspectos relacionados con los usuarios de la aplicación y otros stake-holders
  - Usualmente son negociables y en ocasiones se contradicen.
  - Son medibles.

Algunos atributos de calidad en una AS

1. Desempeño
2. Capacidad
3. Disponibilidad
4. Administración
5. Seguridad
6. Facilidad de uso
7. Portabilidad
8. Confiabilidad
9. Mantenibilidad
10. Escalabilidad

Desempeño

- Define el rendimiento de una aplicación en función de la cantidad de trabajo que debe tomar una solicitud en un momento dado, y / o los límites de tiempo que se deben cumplir para su correcto funcionamiento.
- Usualmente tiene dos metricas asociadas:
  - Troughput
  - Tiempo de respuesta

Troughput

- Es la medida que indica la cantidad de trabajo que puede hacer una aplicación en una unidad de tiempo.
- Ejemplos de medidas:
  - Transacciones por segundo (tps)
  - Mensajes procesados por segundo (mps)

Tiempo de respuesta

- Se refiere a la latencia que una aplicación tiene al procesar una transacción.
- Frecuentemente es medido en milisegundos, desde el momento que una aplicación recibe una entrada hasta que la procesa, sin incluir su despliege.
- El tiempo puede estar delimitado por “deadlines”, los cuales indican los tiempos máximos permitidos para una transacción.

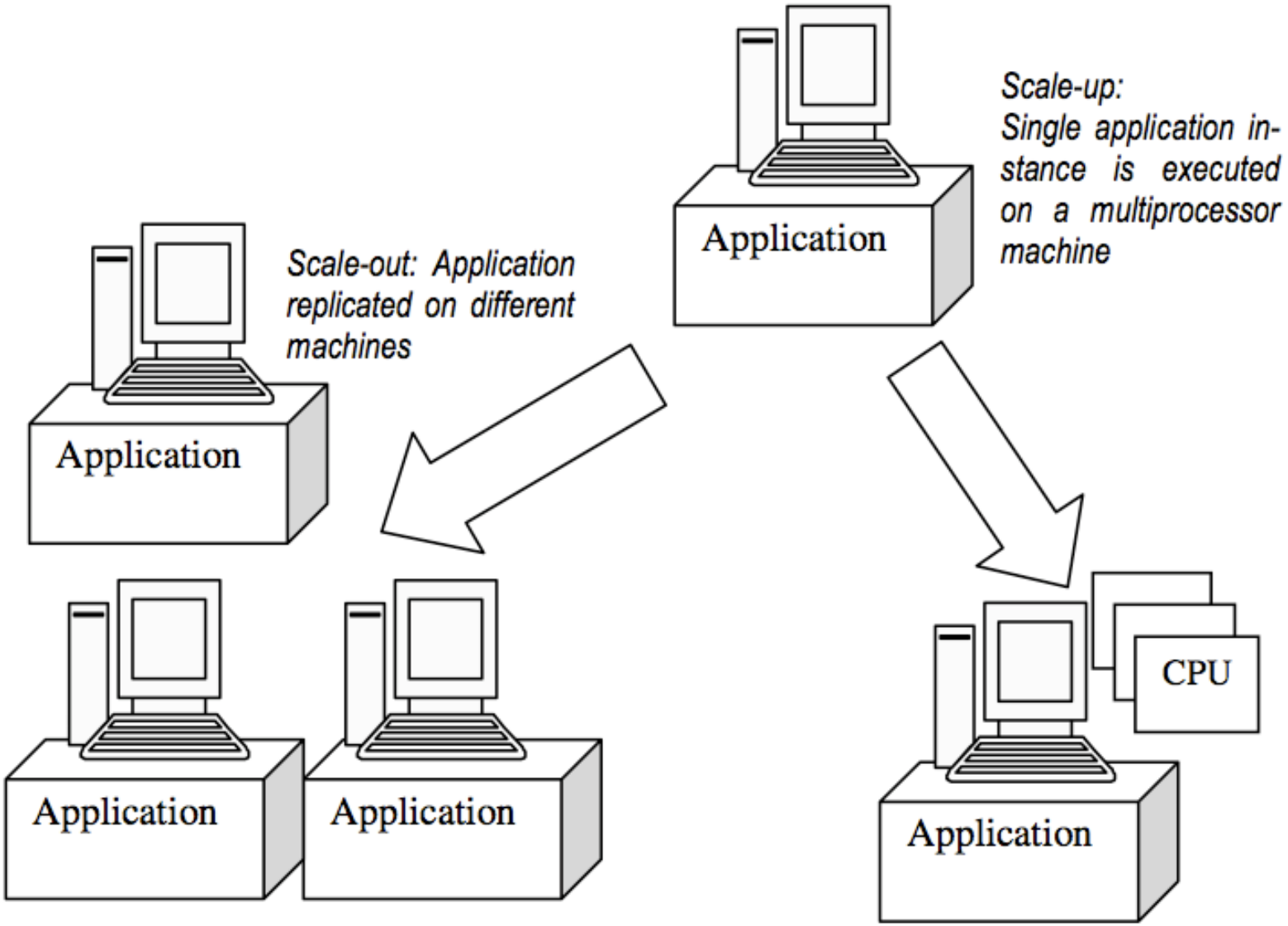
Escalabilidad

- Es la capacidad de la aplicación para funcionar correctamente si el tamaño de procesamiento se incrementa.
- El tamaño de procesamiento se refiere a:
  - Requerimientos de carga
  - Conecciones simultaneas
  - Tamaño de los datos
  - Despliegue

Requerimientos de carga

- Una arquitectura escalable debe permitir adicionar capacidad de procesamiento para incrementar el throughput y decrementar el tiempo de respuesta.
- Dos estrategias para escalar una aplicación:
  - Scale-Up
  - Scale-Out

Scale Up / Scale Out



Mantenibilidad

- Es una atributo de calidad que mide que tan fácil es cambiar una aplicación para soportar nuevos requerimientos funcionales y no funcionales.
- Las medidas de mantenibilidad son relevantes solo en el contexto de una solución y se indican mediante la descripción de los componentes de de la arquitectura y como interactuan entre ellos y con el entorno

Seguridad

- En el plano arquitectónico, la seguridad se reduce a la comprensión de los precisos requerimientos de seguridad para una aplicación y la elaboración de mecanismos de apoyo a ellos.
- Existen diferentes medidas aplicables dependiendo del requerimiento de seguridad

Requerimientos de Seguridad Comunes

- Autenticación: La aplicacion puede verificar la identidad de los usuarios u otras aplicaciones con las cuales se comunica
- Autorización: Usuarios o aplicaciones autenticadas tienen definidos los derechos de acceso a los recursos del sistema
- Encripción: Los mensajes enviados y recibidos desde y hacia la aplicación están encriptados
- Integridad: Asegura que el contenido del mensaje no puede ser alterado durante su tránsito
- Non-repudiation: Quien envia el mensaje (sender) tiene prueba de entrega y el receptor (receiver) está seguro de la identidad de quien lo envía. Es decir, no puede refutar su participación en el intercambio del mensaje.

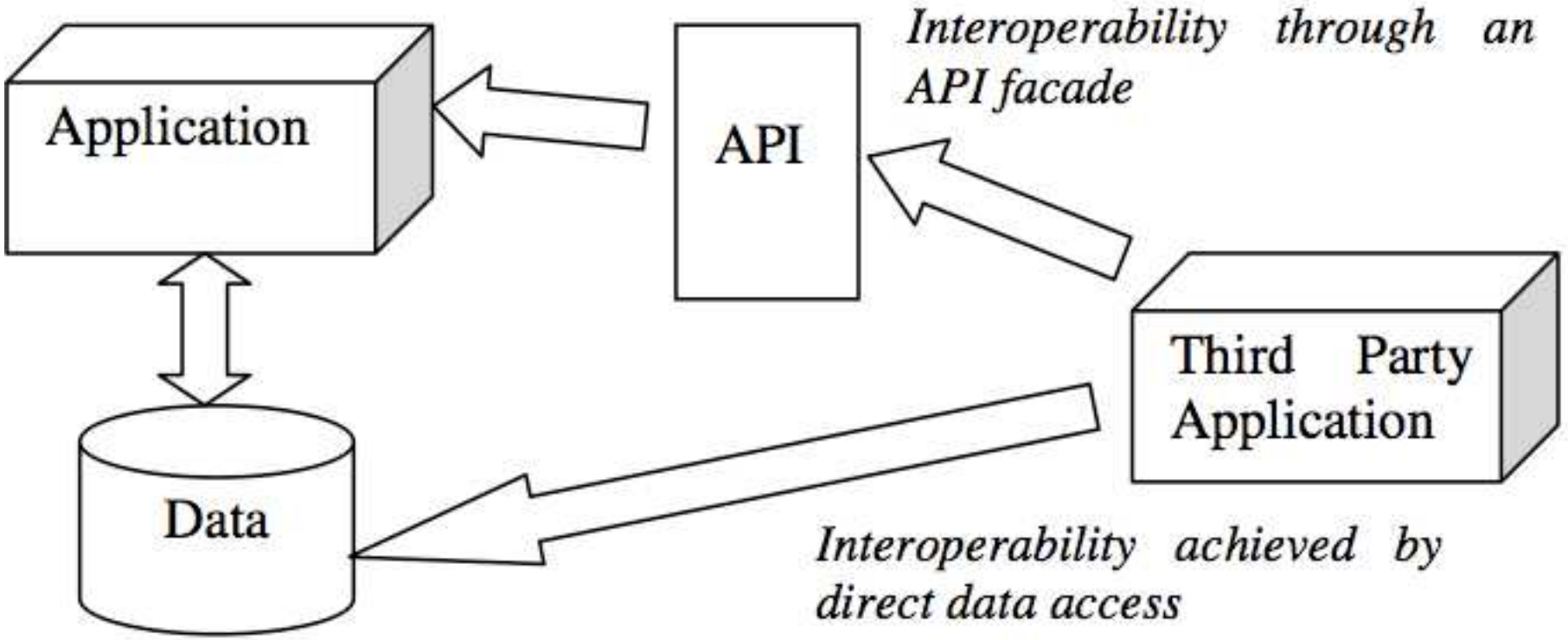
Disponibilidad

- Puede ser medida por la proporción de tiempo requerido que la aplicación es usable. Ej: 99.99%
- Las arquitecturas redundantes o con alta recuperabilidad son utilizadas para obtener alta disponibilidad
- Arquitecturas con registro dinámico de componentes facilitan el mantenimiento y favorecen la disponibilidad del sistema

Integración

- Facilidad con la que una aplicación puede incorporarse en un contexto de sistema más amplio.
- Las estrategias más usada para alcanzar alta integrabilidad de las aplicaciones son a través de:
  - Integración de los datos
  - Integración por API

Estrategias de integración



Otros atributos de calidad

- Portabilidad: Capacidad de la aplicación para moverse de la plataforma original de desarrollo a otras plataformas
- *Testability*: Facilidad para ser probada. En general entre mas complicado el diseño más dificil de probar
- Supportability: Facilidad para detección y diagnosticos de fallas (ej: Logs)

Créditos

- Essential Software Architecture. Ian Gorton.
- Describing Software Architectures. Gert Florijn.
- Workshop de Arquitectura I/T, Bogotá 2005. Carlos Bittrich. IBM.