Arquitecturas de Software

Administración de Recursos

Ing. en Sistemas de Información

FRBA – UTN – Argentina - 2014

Sistemas de Gestión de Datos

- Administran un gran volumen de datos: del orden de los millones de registros.
- El dominio de la aplicación se centra en el manejo de sus datos.
- Poseen un gran dinamismo en sus datos: muchas altas, bajas y modificaciones a diario.
- Reciben gran cantidad de consultas variadas sobre sus datos.
- Las operaciones a realizar sobre ellos suelen ser simples, en lo que a su aspecto algorítmico se refiere.
- Operaciones comunes:
  - Altas (INSERT)
  - Bajas (DELETE)
  - Modificación (UPDATE)
  - Consultas (SELECT)

Tipos de Usuarios

- ***Usuarios intensivos***: son usuarios que utilizan el sistema en forma constante, todos los días, varias veces al día. La productividad y efectividad de su trabajo se encuentra estrechamente relacionada con la interacción del sistema (ejemplo: cajera de supermercado)
- ***Usuarios casuales***: son usuarios que acceden al sistema en forma esporádica, pocas veces al día y que no dependen en forma exclusiva del sistema para seguir operando (ejemplo: persona que compra de libros por Amazon).

Arquitecturas de Software

- *Arquitectura es un diseño de alto nivel*. Esto es bastante cierto, en el sentido que un caballo es un mamífero pero la inversa no es verdadera. Existen tareas del diseño que no son arquitectónicas, como la decisión de encapsular estructuras de datos importantes. La interfaz de esas estructuras es, sin duda, una cuestión arquitectónica pero la elección no.

Arquitecturas de Software

- *Arquitectura es la estructura general del sistema*. Esta afirmación implica, erróneamente, que un sistema tiene una sola arquitectura. Múltiples estructuras pueden convivir en una arquitectura dando apoyo ingenieril a un sistema

Arquitecturas de Software

- *Arquitectura es la estructura de componentes de un sistema, sus interrelaciones y los principios y lineamientos que gobiernan su diseño y evolución en el tiempo*. Algunos autores (Bass, Clement & Kazman) consideran que esto último es esencial y parte de una buena práctica profesional. De todos modos no lo consideran parte de la arquitectura.

Arquitecturas de Software

- *Software Architecture in Practice,* Bass-Clements-Kazman, 2003, Addison-Wesley.

Clasificación de Arquitecturas

- Arquitecturas Monocapa
- Arquitecturas Multicapa

Arquitecturas Monocapa

- Se caracterizan por ser sistemas monousuarios, utilizados por una única persona a la vez, sin que exista concurrencia de acceso a la información
- Son comúnmente denominados Desktop, dado que consisten en una aplicación instalada en la PC del usuario, compuesta por un instalador y un ejecutable que permite acceder a la misma.
- No posee conectividad. Es un componente completamente independiente que no interactúa con ninguna otra aplicación, salvo con el sistema operativo en el cual fue instalado.
- La lógica de negocio está centralizada. Todas las reglas propias del negocio que se implementa se encuentran en un solo lugar, en el código de la aplicación instalada en cada máquina.
- En cuanto a la persistencia, los datos de la aplicación se guardan en archivos propios de la aplicación (ejemplo: archivo .doc del Word). Esto permite que la información pueda ser recuperada en un uso posterior de la aplicación. Estos datos se almacenan en cada máquina en la que se encuentre instalado el programa, sin ningún tipo de centralización.

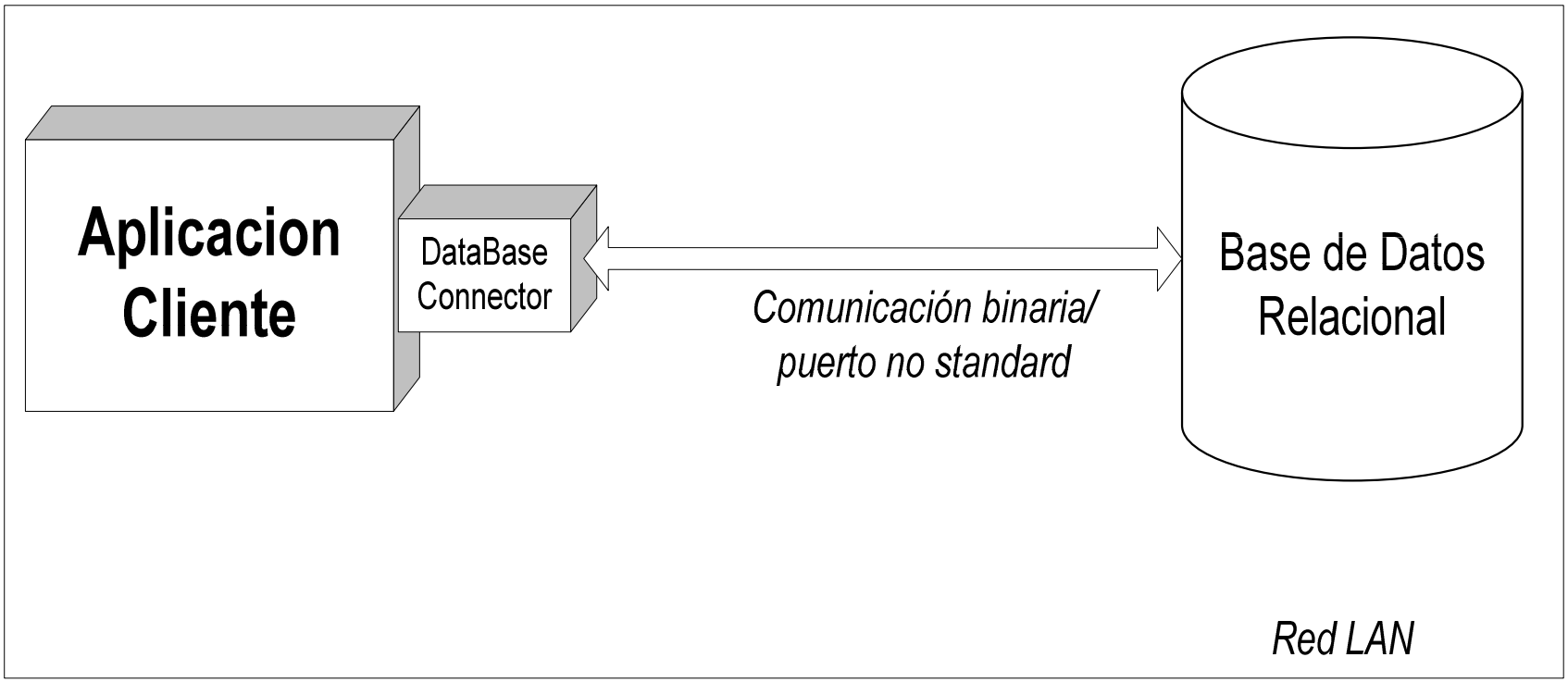
Monocapa Ventajas

- ***Es simple de construir***. La aplicación esta conformada por un único componente.
- ***Facilidad en el rastreo de bugs***: todos los problemas que existan en la aplicación se encuentran en el único componente que conforma la misma.
- ***Acceso directo a datos***: no se requiere una conexión para acceder a los datos de la aplicación.
- ***Fácil puesta en marcha***: solo es necesario instalar la aplicación en la PC.

Monocapa Desventajas

- La integración de datos entre aplicaciones de distintas máquinas es complicada. Cada PC que utilice la aplicación almacena los datos de la operatoria realizada en su propio archivo. Para integrar la información de distintas aplicaciones es necesario realizar una consolidación de los archivos persistidos. (ejemplo: juntar dos archivos de Word de dos alumnos que se encuentran realizando un mismo TP para la facultad)
- Para solucionar los problemas de integración se puede:
  - Hacer una consolidación de datos a mano.
  - Utilizar procesos batch automáticos, corriendo fuera de hora, que integren todos los datos y verifiquen su integridad.
- Normalmente la aplicación requiere una versión distinta, que se adapte a cada Sistema Operativo, por lo tanto se encuentran acoplados al Sistema Operativo para el cual fueron desarrolladas.

Arquitectura Cliente-Servidor



Arquitectura Cliente-Servidor

- ***Aplicación Cliente***: componente Desktop, instalado en la PC de cada usuario, de características similares a las mencionadas en la arquitectura anterior.
- ***Aplicación Servidor***: servidor centralizado, con una Base de Datos que hace de repositorio de la aplicación.

Cliente-Servidor Ventajas

- Al incorporar una base de datos como un módulo de nuestro sistema, ganamos todas las ventajas propias de ella:
  - Concurrencia de consultas
  - Carga de datos simultánea
  - Consulta filtrada y performante sobre gran volumen datos
  - Integridad de los datos almacenados
- Se eliminan los problemas de integración de datos, presentes en la arquitectura anterior. Pueden trabajar varios usuarios a la vez, en forma simultánea, y ambos ver reflejado en forma inmediata, los cambios producidos.
- Se delega gran parte de la complejidad del sistema en la base de datos, la cual es un componente ya programado y probado durante varios años.
- Los usuarios pueden estar en diferentes lugares físicos, y acceder al mismo servidor de base de datos.

Cliente-Servidor Desventajas

- Surgen algunos problemas de acceso o conectividad:
  - La PC del cliente debe tener acceso a la IP del servidor. Existen varias formas de lograr solucionar este problema:
    - Todas las PC se encuentran conectadas a la misma red LAN
    - Se accede a través de Internet (Ejemplo: VPN)
  - La comunicación es binaria por un puerto no Standard (distinto del 80, puerto default del protocolo HTTP). Un Proxy suele bloquear este tipo de comunicación por dos motivos:
    - ***Bloqueo por puertos***: las bases de datos no suelen tener sus servicios de escucha en los puertos que normalmente se encuentran abiertos por default.
    - ***Bloqueo por protocolo***: incluso aunque el servicio de la base de datos sea instalado para utilizar el puerto 80, la información que viaja en esta comunicación no pertenece al protocolo HTTP, por lo tanto también puede ser identificada y bloqueada.

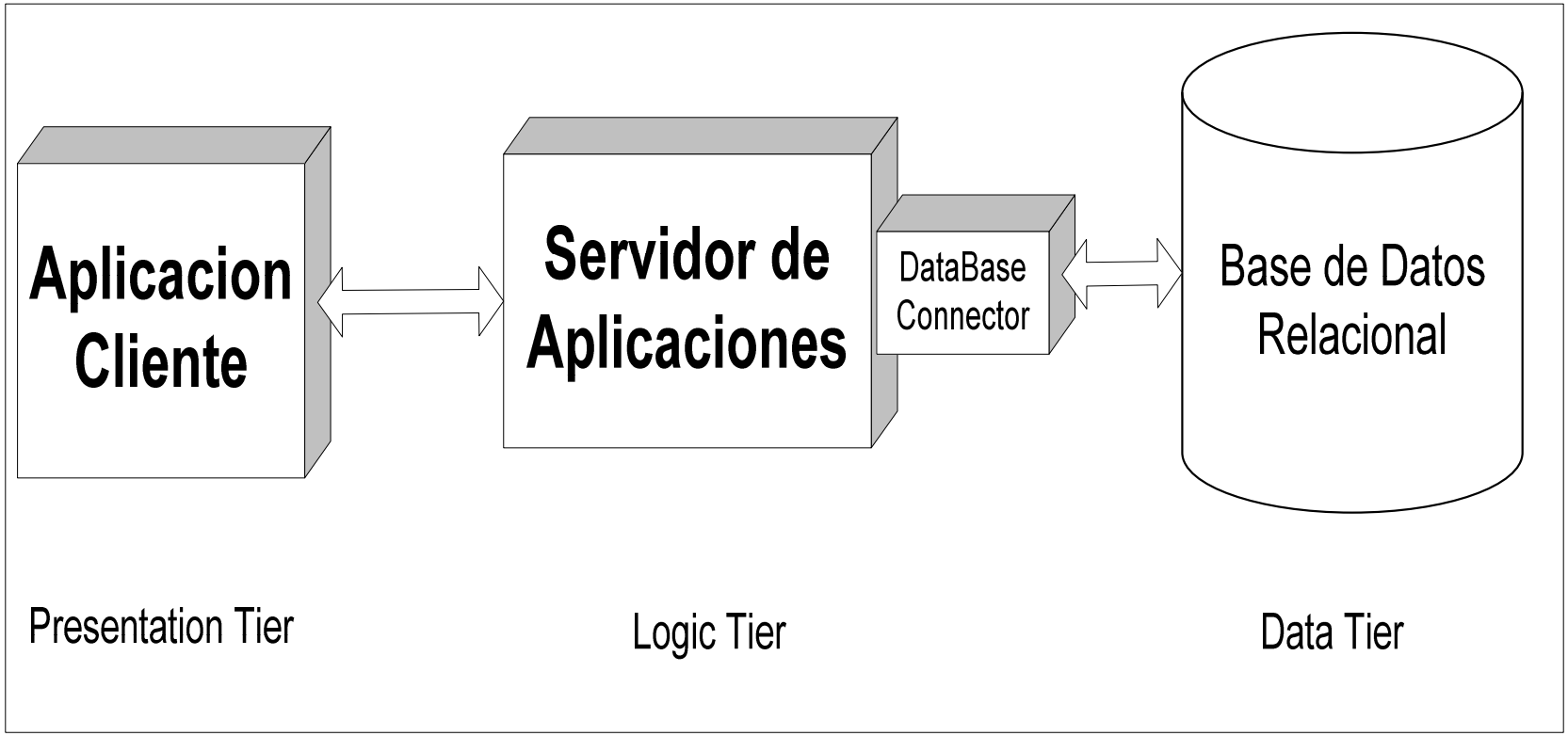
Cliente-Servidor Desventajas

- Requiere instalación: la aplicación Cliente, por poseer características similares a las de la arquitectura Monocapa.
- Al igual que la arquitectura en Monocapa, la aplicación cliente normalmente se encuentra acoplada al sistema operativo.
- Aumenta la complejidad de actualización de la aplicación:
  - Cada vez que surjan cambios o arreglos, será necesario instalar una nueva versión de la aplicación en cada PC. Para superar este problema es se puede optar por alguna de las siguientes alternativas:
    - Crear un módulo de detección de actualizaciones en forma automática (Service Pack, Update Center). Este módulo no debería dejar que el usuario utilice la aplicación por mucho tiempo, con una versión discontinuada. Su implementación puede no ser trivial, dado que hay que tener en cuenta todos los posibles escenarios de actualización que existen.
    - Crear un sitio del cual se pueda descargar la última versión de la aplicación. Corremos el riesgo de que nuestros usuarios prefieran la versión anterior y no instalan la nueva.
    - Enviar personas como instaladores a cada PC para configurar la nueva versión. Esta última opción puede tornarse engorrosa en caso de contar con un gran número de usuarios.

Cliente-Servidor Desventajas

- Aumento de complejidad en rastreo de bugs: cuando surge un error, ahora la causa del mismo puede encontrarse en dos lugares distintos:
  - En la base de datos
  - En la aplicación cliente
- La aplicación se encuentra divida en dos partes, lo cual complica su programación (comparado con la arquitectura anterior)
- La aplicación que el usuario debe instalar en su PC puede terminar ocupando un tamaño significativo, dado que casi todo el código de la aplicación se encuentre en ella. Esto complica aún más la actualización de la aplicación.
- Problemas de seguridad: como el cliente posee en su PC casi todo el código de la aplicación, podría alterarlo para evitar validaciones de dominio (mediante técnicas *cracking* y *decompilación* de código). Además el usuario posee acceso directo a la base de datos. Una mala administración de permisos de usuario podría dejar expuesta toda la información almacenada en la base de datos.

Arquitectura Multicapa



Arquitectura Multicapa con Cliente Desktop

- Este tipo de arquitectura intenta atacar los principales problemas presentados en la arquitectura Cliente-Servidor.
- El Cliente sigue siendo una aplicación instalable, pero ahora no accede directamente a los datos, sino que lo hace a través de la aplicación servidor.
- El cliente se considera liviano (thin-client) porque el usuario no posee la aplicación entera en su poder, sino solo una cáscara del sistema. Muchas reglas del negocio pueden ser ahora implementadas en la aplicación servidor.
- Uno de los temas centrales que surgen en esta arquitectura, es el método de comunicación que se va a utilizar entre la aplicación cliente y la aplicación servidor.
- La aplicación cliente ahora no puede aprovechar los beneficios de un driver de base de datos, sino que debe utilizar algún método propio de conexión.

Arquitectura Multicapa con Cliente Desktop

Existen diversas tecnologías para solucionar este problema, dependiendo del lenguaje utilizado:

- Comunicación tradicional a bajo nivel, mediante la utilización de Sockets
- Comunicación mediante RPC (Remote Procedure Call) o RMI (Remote Method Invocation): ambas alternativas presentes en la mayoría de los lenguajes de programación actuales, proveen mecanismos automáticos para la ejecución de código remoto, junto con la serialización de los parámetros de entrada y salida.
- Mediante el uso de WebServices. La aplicación cliente puede utilizar el protocolo HTTP para intercambiar mensajes con la aplicación servidor, mediante archivos en formato XML.

Arquitectura Multicapa con Cliente Desktop Ventajas

- La aplicación cliente puede poseer una interfaz gráfica más robusta, interactiva y performante que las que se pueden lograr actualmente con tecnologías Web.
- Como la aplicación servidor es la encargada del acceso a la Base de Datos, la lógica de la pantalla no se encuentra tan acoplada a la fuente de origen de datos.
- La aplicación cliente no posee acceso directo a la base de datos y las validaciones de negocio mas importantes pueden ser programadas en la aplicación servidor, eliminando de esta forma algunos problemas de seguridad presentes en la arquitectura anterior.
- El programa instalado en la PC del usuario no necesita contar con toda la lógica del negocio, por lo tanto suele ser bastante más liviano que en una arquitectura Cliente-Servidor. Gracias a esto, la actualización de versiones de la aplicación es más fácil.

Arquitectura Multicapa con Cliente Desktop Desventajas

- La aplicación cliente es más liviana y fácil de actualizar que en la arquitectura Cliente-Servidor, pero sigue siendo necesario desarrollar un módulo de actualización para la instalación de futuras versiones.
- La lógica de la aplicación ahora se encuentra dividida en tres partes. Esto complica la detección y arreglo de fallas.
- Es necesario determinar qué bibliotecas de código serán utilizadas desde el cliente y cuáles desde el servidor. Su desarrollo se vuelve más complicado que en una arquitectura en dos capas.
- La interacción de la capa Servidor en los pedidos a la base de datos puede representar un cuello de botella en la performance de la aplicación.
- Si no se utiliza WebServices para la comunicación entre la aplicación cliente y la aplicación servidor, esta arquitectura presenta los mismos problemas de conectividad que la arquitectura Cliente-Servidor.

Arquitectura Multicapa con Cliente Web

- Las aplicaciones Web intentan eliminar muchos de los problemas de conectividad presentes en las arquitecturas anteriores. Para ello centran sus energías en el desarrollo de aplicaciones que se ejecuten dentro de un Browser de Internet.
- Por aplicaciones Web no nos referimos a simples páginas HTML o portales de contenido, sino a aplicaciones enteras y complejas, con mucha lógica de negocio involucrada.
- Esta arquitectura es una de las más recientes y se encuentra en constante evolución.
- El desarrollo de aplicaciones Web es un tema amplio y completo que por si solo podría requerir varios libros para ser explicado correctamente. Solo nos limitaremos a nombrar sus características más relevantes para nuestro estudio.

Arquitectura Multicapa con Cliente Web Composición

- Aplicación cliente ejecutando en un Browser de Internet. El código de la aplicación podrá estar compuesto por diversos elementos, dependiendo de la plataforma de desarrollo elegida. Constantemente surgen nuevos elementos que intentan mejorar la experiencia de las aplicaciones Web. Algunos de los elementos más comunes son:
  - HTML
  - JavaScript
  - Hojas de estilo (CSS)
  - Otros elementos no Standard (plugins):
    - Java (Applets, JavaWebStart)
    - Flash (ActionScript, Flex)
    - ActiveX (COM)
- Application Server. Al tratarse de aplicaciones Web, el servidor debe poder lidiar con el protocolo HTTP. Al igual que en la aplicación cliente, existen muchos servidores dependiendo de la tecnología elegida. Algunos de los más comunes son:
    - Apache (PHP)
    - Tomcat (Java)
    - Internet Information Server (ASP .NET, ASP)
    - ColdFusion (Flash)
- Base de datos

Arquitectura Multicapa con Cliente Web Ventajas

- Eliminan el problema de acceso presente en todas las demás arquitecturas. El puerto 80 y el protocolo HTTP suelen estar permitidos en casi cualquier ambiente..
- Eliminan los problemas de instalación de la aplicación..
- Al no requerir instalación, el usuario siempre accede a la última versión disponible de la aplicación. Esto evita problemas de actualización.
- Al accederse a través de Internet, se eliminan los problemas de conectividad.
- Como la aplicación cliente no se instala ni se ejecuta directamente, sino a través del Browser, se eliminan los problemas de acoplamiento al sistema operativo.
- Al no requerir instalación y poder ser accedidas desde cualquier PC con acceso a Internet, suelen ser ágiles para un usuario casual.
- Al igual que en la arquitectura Multicapa con cliente Desktop, el usuario solo posee una parte de la aplicación. Por lo tanto se eliminan muchos problemas de seguridad.

Arquitectura Multicapa con Cliente Web Desventajas

- Si bien las aplicaciones WEB no requieren instalación, es necesario tener el Browser perfectamente configurado. Los siguientes elementos suelen traer conflictos:
  - Permitir la apertura ventanas emergentes (Popups)
  - Permitir la ejecución de JavasScript
  - Permitir la ejecución de Plugins
  - Contar con los Plugins en la versión correcta
  - Tener habilitado un nivel de seguridad que permite acceder a la aplicación
- La performance del protocolo HTTP deja mucho que desear a la hora transferir grandes volúmenes de datos. El protocolo HTTP es un protocolo orientado a caracteres y no binario, con lo cual se añade un gran overhead al tamaño de los mensajes transmitidos.
- Actualmente la performance de ejecución de código JavaScript en los Browser es inferior a la de otros lenguajes existentes: Java, .NET, PHP.
- La calidad de la interfaz gráfica de usuario que se puede lograr, en especial en lo que se refiere a la interacción con componentes gráficos, suele ser inferior a la que es posible lograr mediante una aplicación Desktop.

Arquitectura Multicapa con Cliente Web Desventajas

- Las aplicaciones Web logran desacoplarse del Sistema Operativo pero son dependientes del Browser de Internet sobre el cual ejecutan. Actualmente existen muchas diferencias en la forma en que los Browsers tratan el contenido de las aplicaciones Web. Algunos de los problemas más frecuentes son:
  - No todos los Browsers renderizan HTML de la misma forma, con lo cual una misma aplicación puede visualizarse de formas distintas en Browsers diferentes.
  - JavaScript es interpretado en forma distinta, y con distintos niveles de performance, en cada Browser.
  - Las hojas de estilo (CSS) poseen propiedades que funcionan en algunos Browsers y otros no, y muchas propiedades que funcionan en todos los Browsers son renderizadas de forma distinta.
- Si bien se evitan los problemas de actualización de la aplicación, el cliente descarga el 100% de la aplicación cada vez que accede a la misma.
- Si el usuario no tiene acceso Internet (problemas con el ISP) no se puede acceder a la aplicación..
- Actualmente el tiempo de desarrollo requerido para una aplicación con arquitectura Web suele ser bastante mayor que el requerido para desarrollar la misma aplicación en arquitectura Desktop.
- La cantidad de lenguajes diferentes que conviven en una misma aplicación suele complicar el mantenimiento de la misma.

Arquitectura Orientada a Servicios (SOA)

Extendamos el concepto de arquitectura de software:

- **Arquitectura empresarial:** enfoque para optimizar a lo largo de una organización los procesos comúnmente fragmentados (tanto manuales como automatizados) en un contexto integrado responsable de cambiar y apoyar la aplicación de la estrategia de negocio.

- **Arquitectura tecnológica:** descripción de la estructura e interacción de servicios de plataforma y componentes lógicos y físicos de tecnología. Esto es, hardware, software e infraestructura de red necesaria para soportar el despliegue de aplicaciones esenciales y críticas.

Arquitectura Orientada a Servicios (SOA)

Entremos en la orientación a servicios:

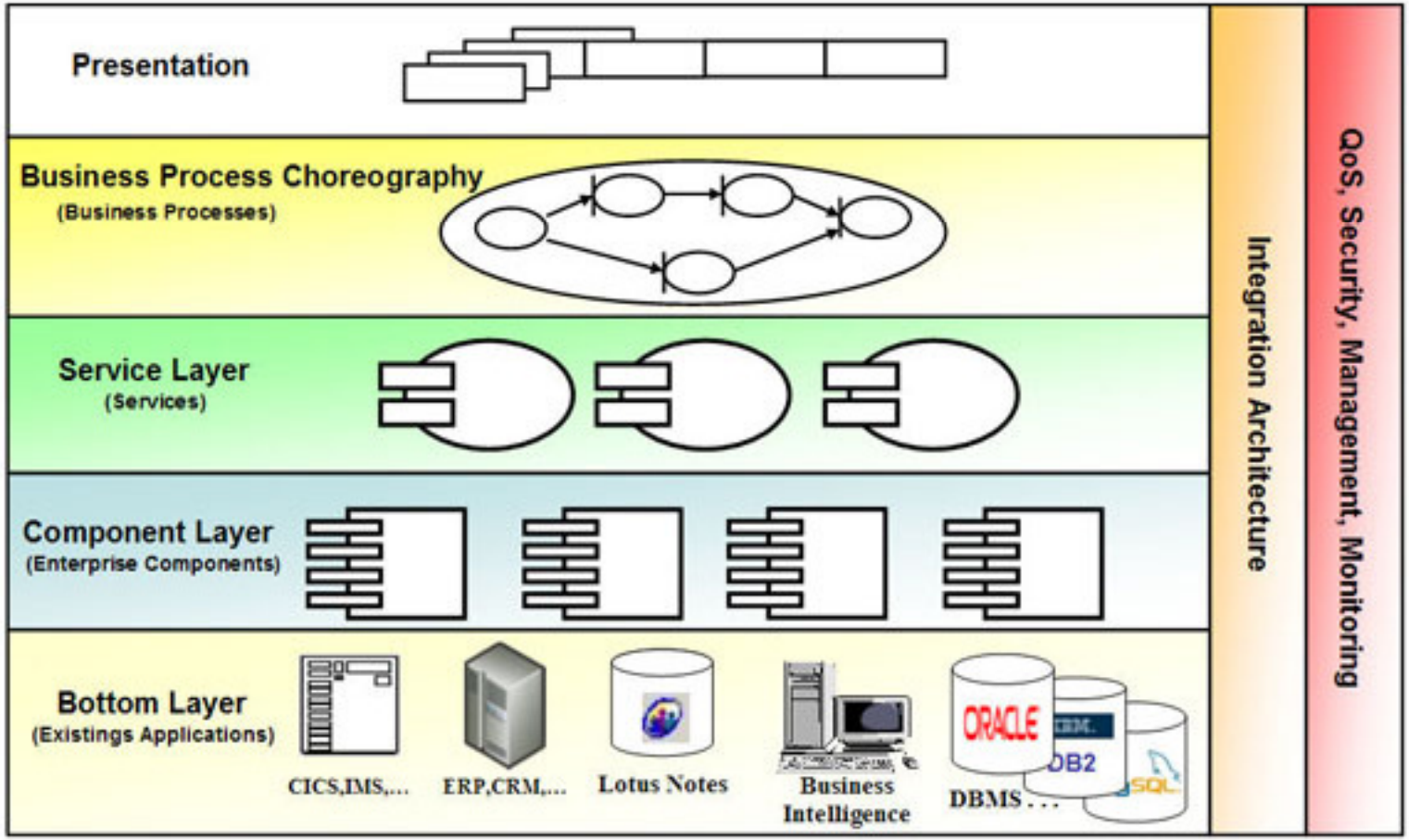
- **Arquitectura Orientada a Servicios:** aquella que soporta un enfoque basado en un diseño de servicios que reflejan las actividades de negocio del mundo real de acuerdo con los procesos de negocio de la organización.

- **Orientación a servicios:** enfoque basado en un diseño de servicios que reflejan las actividades de negocio del mundo real de acuerdo con los procesos de negocio de la organización.

- **Servicio:**
  - Es una representación lógica de una actividad repetible del negocio que tiene un resultado definido (por ejemplo, chequear crédito de cliente, proveer datos de pronóstico de tiempo, consolidar reportes repetitivos)
  - Es auto-contenido
  - *Puede* resultar de la composición de otros servicios
  - Es una “caja negra” para quienes lo consumen

Arquitectura Orientada a Servicios (SOA)

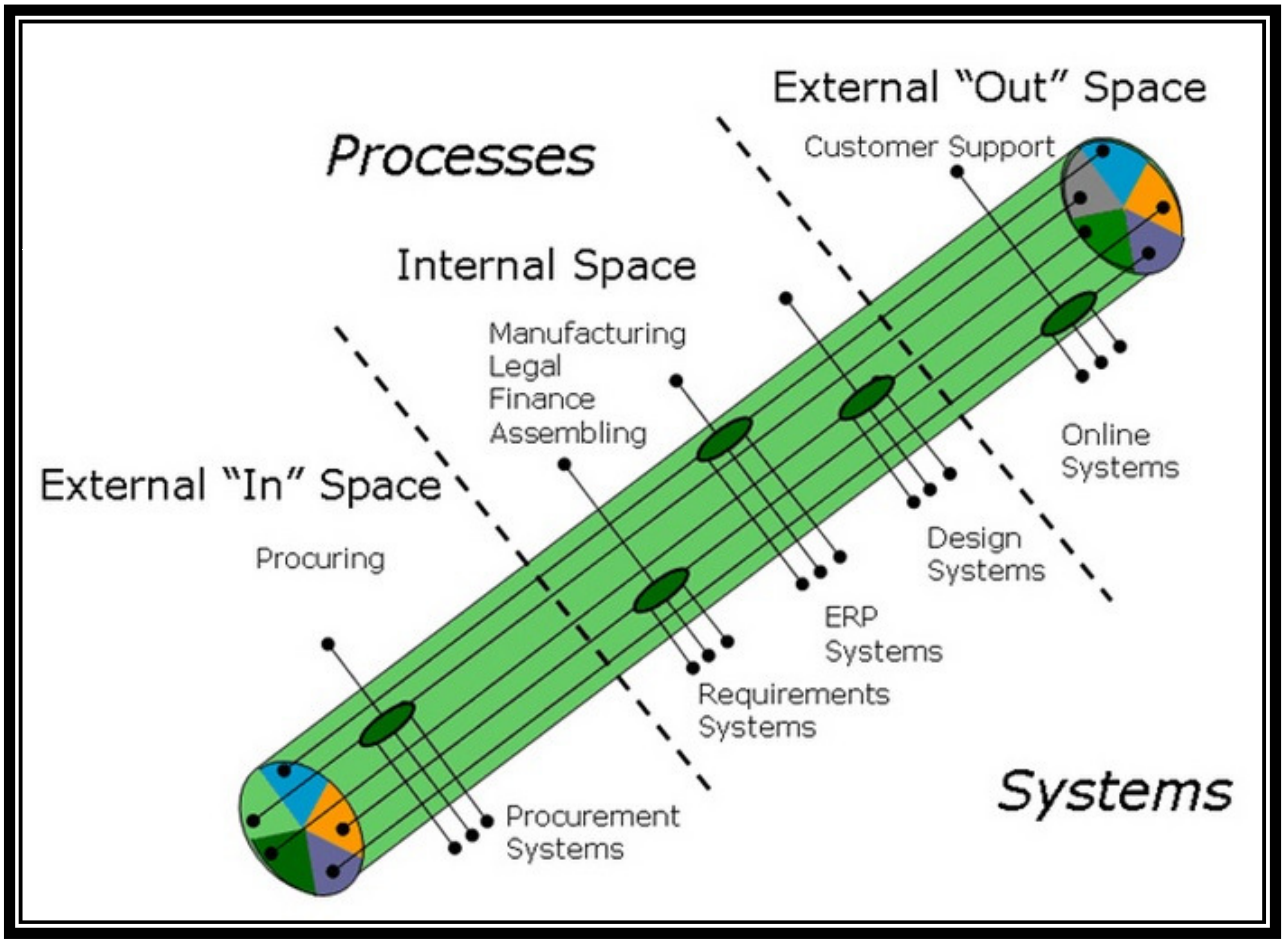
SOA propone una modelo de siete capas entre proveedores y consumidores de servicios:



Arquitectura Orientada a Servicios (SOA)

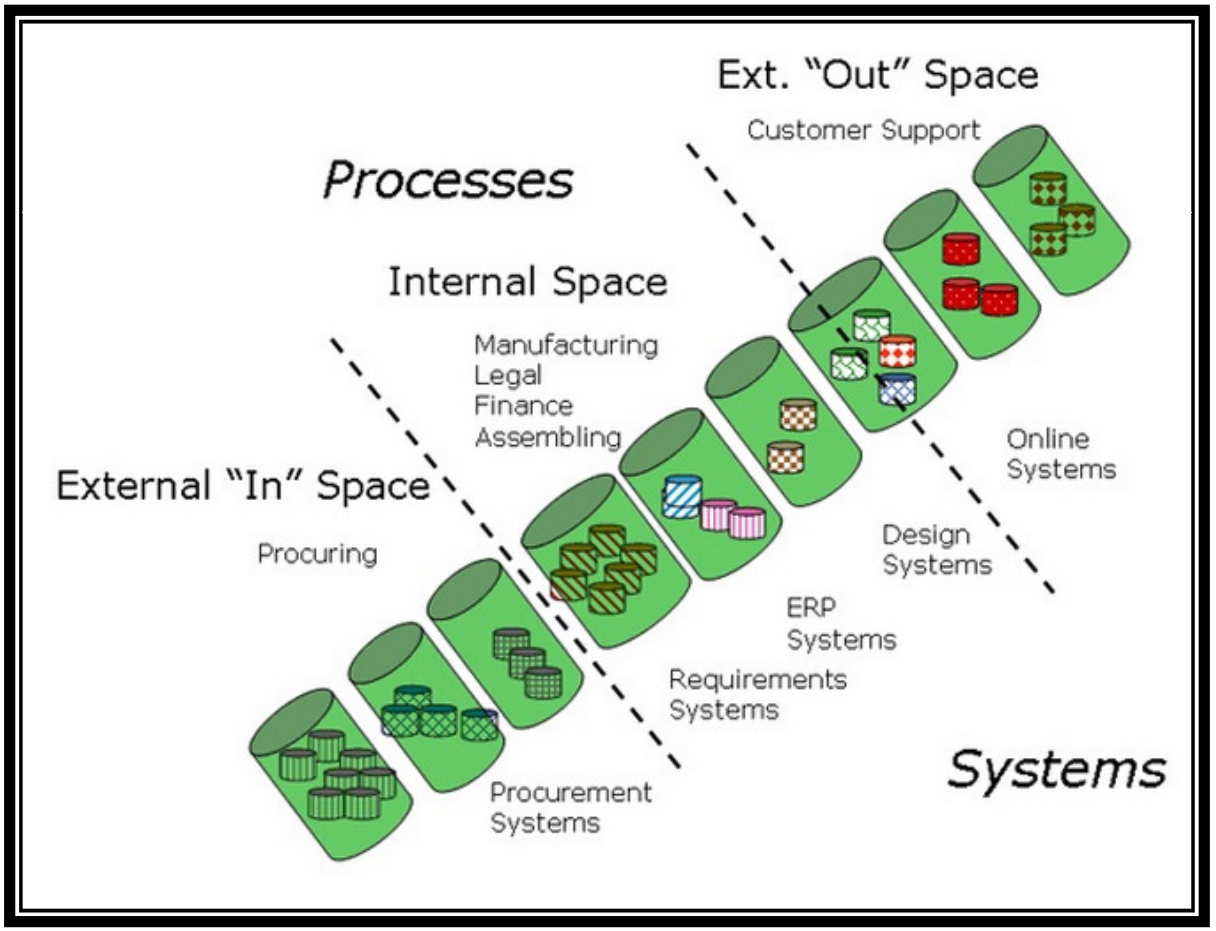
El problema: lograr una arquitectura en la cual los procesos de negocio son soportados por sistemas que permiten el libre intercambio de información

Las dificultades centrales son información integrada y acceso a esa información



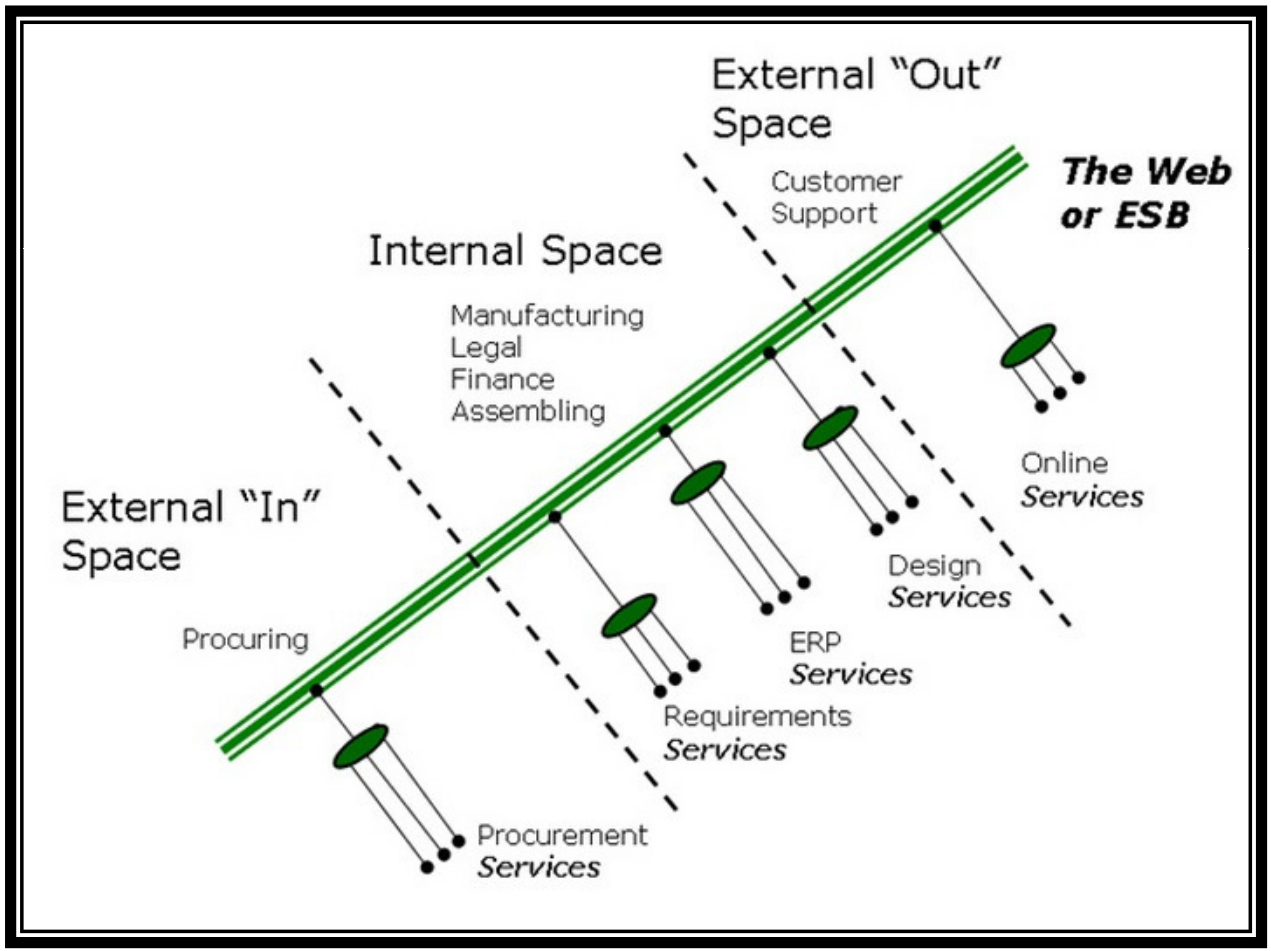
Arquitectura Orientada a Servicios (SOA)

Pero existen silos de información: lo común es encontrar que cada proceso de negocio tiene su propio sistema el cual a su vez tiene sus interfaces y formatos de información particulares



Arquitectura Orientada a Servicios (SOA)

Flujo de Información Sin Fronteras: con SOA, las aplicaciones se reemplazan por servicios que interactúan entre sí



Arquitectura Orientada a Servicios (SOA)

Beneficios:

- Mejora de performance de procesos del negocio
- Reducción de costos de IT
- Mejora de la efectividad de las operaciones del negocio
- Mejora de la efectividad de IT
- Mejora de la eficacia de gestión
- Reducción de riesgos

Arquitectura Orientada a Servicios (SOA)

Referencias:

- The Open Group SOA Source Book: colección de material para arquitectos empresariales que trabajan con arquitectura SOA (http://www.opengroup.org/soa/source-book/intro/index.htm)
- The Open Group Interoperable Enterprise Business Scenario: propuesta para Enterprise Services Bus (ESB) (http://www.opengroup.org/soa/source-book/soa/soa_bif.htm)