Club de Investigación Tecnológica

Cliente/servidor

Preparado por: Lic. Edgar Hernández e Ing. Luis Martínez

Diciembre 1997

**Editado y publicado por Rho-Sigma, S.A.
a nombre del Club de Investigación Tecnológica.**

**San José, Costa Rica
Diciembre 1997**

Contenido

Página

I parte

La filosofía cliente/servidor: Conceptos y arquitectura

1. Sistemas Abiertos y Tecnologías Cliente/Servidor

Los sistemas abiertos representan un conjunto de estándares internacionales relacionados con los sistemas de información, que especifican interfaces, servicios y formatos con el propósito de lograr la interoperabilidad y la portabilidad de aplicaciones y datos. Esta clase de sistemas tienen interfaces públicas, lo que significa que cualquier fabricante de *software* o *hardware* las conoce y puede desarrollar productos para ellos. En este caso, los usuarios tienen la posibilidad de adquirir tecnología de diversos proveedores, la cual pueden integrar con un esfuerzo generalmente reducido y beneficiarse de la competencia de mercado.

Los estándares a su vez facilitan el funcionamiento de las organizaciones, sus sistemas e infraestructura. En materia computacional, algunos estándares definen elementos como un conjunto universal de caracteres (por ejemplo, el ASCII), lenguajes de programación (C, Cobol, Pascal, etc.) e interconexión con bases de datos (Microsoft ODBC - *Open Database Connectivity*).

1.1 Estándares [1]

Existen mucho grupos internacionales encargados de establecer estándares, aparte de aquellas empresas cuyos productos se han convertido en estándares "de facto" por su popularidad y antigüedad en el mercado. Un ejemplo muy conocido en esta segunda categoría es el sistema operativo para microcomputadoras, Microsoft DOS.

En el caso de los grupos internacionales, dentro de los más conocidos se pueden mencionar: la Organización Internacional para la Estandarización (ISO), el Comité Consultivo Internacional Telefónico y Telegráfico (CCITT), el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE), el Instituto Americano de Estándares Nacionales (ANSI) y el Instituto Nacional de Estándares y Tecnología (NIST).

Entre los estándares computacionales que son mayoritariamente soportados por los fabricantes de *software* y *hardware* se pueden mencionar:

- POSIX (*Portable Operating System Interfaces for Computing Systems*). El POSIX es una interfaz estándar desarrollada por el Comité Técnico sobre Sistemas Operativos de la IEEE. Está muy estrechamente relacionado al UNIX debido a que surgió de una combinación de llamadas a funciones de su núcleo y de bibliotecas de rutinas que forman parte de su código fuente. Es muy importante ya que tanto los Estados Unidos como los países de Europa, exigen que cualquier gestión por medios computacionales que un usuario desee realizar con agencias gubernamentales, debe cumplir con este estándar.

[1] Este tema es tratado más ampliamente en [Price 1995]

- X/Open. El X/Open es un consorcio de vendedores de *hardware* y *software* que se especializa en los requerimientos de los usuarios. Este grupo no crea nuevos estándares, sino que selecciona aquellos que se han utilizado ampliamente en el mercado. Los requerimientos son obtenidos a nivel internacional, por medio de estudios por correo en 17 países alrededor del mundo. Los resultados se publican en un congreso mundial sobre sistemas abiertos y conforman la base de trabajos técnicos posteriores que terminan por definir especificaciones (interfaces) independientes de los vendedores para la interoperabilidad y la portabilidad de los sistemas.

- *Open Software Foundation* - OSF. El OSF tuvo sus comienzos en el ambiente UNIX y hoy en día está centrado en el área de la computación distribuida. Dos de sus especificaciones, el DCE - *Distributed Computing Environment* y el DME - *Distributed Management Environment* están provocando un impacto muy importante en los sistemas cliente/servidor porque facilitan el flujo de información en ambientes distribuidos heterogéneos tanto de *software* como de *hardware*. Entre los miembros originales de la fundación se encontraban: Digital Equipment Corporation, Hewlett-Packard, IBM, Bull y Siemens AG. Sus dos primeros productos fueron el sistema operativo OSF/1 y la interfaz gráfica Motif GUI.

- *Object Management Group* – OMG. Esta organización tiene como objetivo la definición de estándares para la tecnología de orientación a objetos.

La relación entre los sistemas abiertos y los ambientes cliente/servidor es muy estrecha. Dos de las características primarias de los primeros, la interoperabilidad y la portabilidad, son también básicas en los sistemas cliente/servidor. La interoperabilidad es la que permite la relación entre los clientes y los servidores. En el nivel de *hardware*, es un objetivo esencial de los vendedores de tecnología abierta, que sus productos puedan operar lo más transparentemente posible, con los de los otros vendedores. La portabilidad es igualmente importante porque permite ejecutar los sistemas en una amplia variedad de plataformas de *hardware* y sistemas operativos. La forma más simple de obtener lo anterior, es mediante la definición o el cumplimiento de estándares, como los mencionados con anterioridad y otros que se tratarán más adelante en este documento.

1.2 ¿Qué es la arquitectura de diseño computacional cliente/servidor?

Si se realiza una retrospectiva acerca de la funcionalidad de los sistemas computacionales tradicionales (computador central y terminales "no inteligentes"), en su paradigma de operación, absolutamente todos los requerimientos de servicio del computador (desde la suma de dos valores hasta la presentación de la información) son procesados en el computador centralizado. La información y su posterior presentación "viajan" por el medio de comunicación, y se presentan en un dispositivo (terminal) que tiene como única misión mostrar la salida sin contribuir para nada en el proceso que se solicitó.

Cliente/Servidor es una arquitectura que separa el procesamiento entre clientes y servidores en una red. Los tres componentes esenciales del esquema son: los *clientes* (usualmente PCs o equipos Macintosh), el *servidor* (donde reside por ejemplo la base de datos) y la *red* que transporta requerimientos y posteriormente datos.

Existen varias configuraciones propuestas de modelos cliente/servidor:

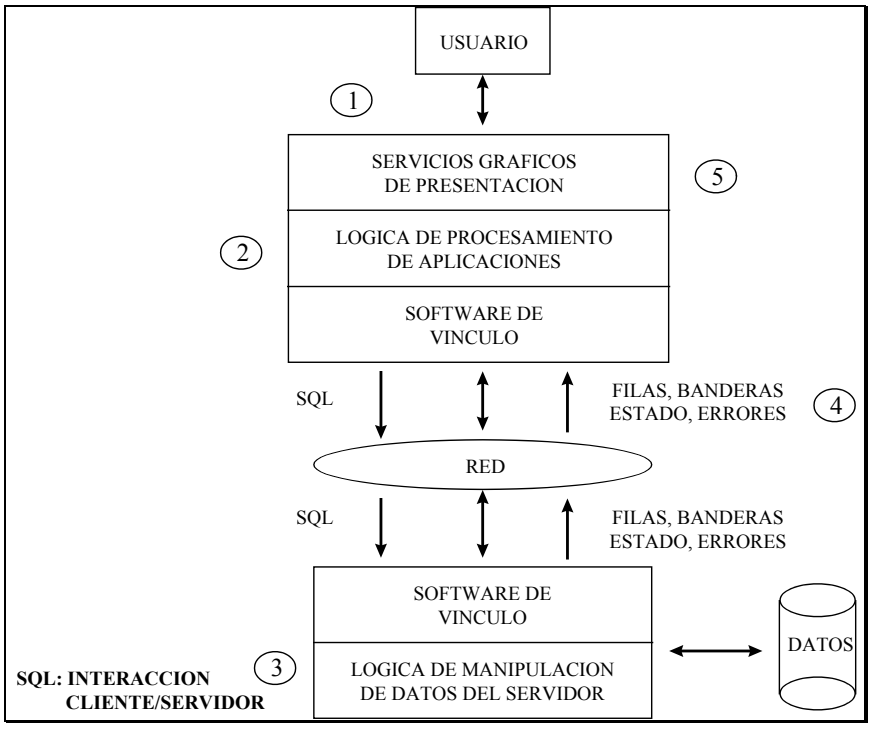
- El cliente procesando solamente el despliegue de información
- El cliente procesando el despliegue y participando en parte del proceso
- El cliente manejando el despliegue, todo el proceso, y accesando los datos del servidor

Las aplicaciones tradicionales se caracterizan por ser muy intensivas en cuanto a la manipulación de los datos, normalmente consisten de cuatro componentes principales: la base de datos, la lógica de la transacción, la lógica de la aplicación y la interfaz de usuario. En las aplicaciones tradicionales –no cliente/servidor- todo este proceso se realiza en un solo procesador central, lo cual impedía al usuario el poder cargar sus propios datos y procesarlos posteriormente a su conveniencia en su propia máquina.

Cliente/servidor divide la aplicación, en esta divisón normalmente se deja la lógica de la transacción y la base de datos en un lugar y la lógica de la aplicación y la interfaz de usuario distribudas en otro lugar, de esta forma el usuario puede tener múltiples fuentes de datos y permite al modelo descentralizar el ambiente.

Más adelante veremos como con la aparición de aplicaciones en Internet, se ha creado una división más (al separar la interfaz de usuario de la aplicación). Esto permitiá a un servidor de aplicación dar servicios al *browser* del Web el cual es independiente de la aplicación. La lógica de la transacción y los componentes de administración de la base de datos permanecen en el mismo lugar.

En forma general, el esquema tradicional de cliente-servidor divide el modelo en dos grandes capas que se subdividen de la siguiente manera:



Capas del Modelo Cliente/Servidor
Figura 1

Visto desde una perspectiva de capas o niveles, se identifican claramente las siguientes actividades:

- La aplicación en el cliente maneja la interfaces de usuario (administración de formas y ventanas).

- La aplicación en el cliente maneja la lógica de la interfaces de usuario (control de la transacción).

- La aplicación en el cliente se encarga de la integridad de manipulación de información acorde con las necesidades que le especifiquemos.

- La aplicación en el cliente realiza todas las validaciones y transformaciones de los datos.

- La aplicación en el cliente procesa consultas *ad hoc*. Se encarga del lenguaje de manipulación de datos (DML). Por ejemplo procesa sentencias INSERT, UPDATE y DELETE.

- El servidor de bases de datos funciona como un "repositorio" de información que administra los accesos concurrentes de los usuarios, maneja los procedimientos almacenados y vela por la integridad de la información.

La mayoría de configuraciones cliente/servidor empleaban hasta hace poco tiempo este modelo de dos niveles, el cual consiste de un cliente (con varias funciones) que invoca servicios de un servidor. Bajo esta forma de trabajar, mientras que la aplicación cliente interactúa con el usuario final, utilizando una interfaces de usuario gráfica, el servidor de base de datos realiza la manipulación de los datos a alta velocidad, protege la integridad de la información, estructura y mantiene los datos acorde con las reglas de diseño de la aplicación comercial. (Nota: cliente/servidor es un concepto lógico. Este no requiere estrictamente que el *front-end* tenga interfaces de usuario gráficas, ni que el *back-end* deba ser un servidor de bases de datos relacional. La tecnología cliente/servidor es meramente un paradigma o modelo para la interacción entre procesos de *software* ejecutando concurrentemente, que pueden o no funcionar en máquinas separadas).

Esta forma dinámica con la cual se puede implementar el concepto permite tener, por ejemplo, las siguientes posibilidades:

- Todos los clientes y servidores pueden residir en la misma máquina. De hecho pueden ser dos procesos trabajando cooperativamente
- Cada uno de los clientes y servidores pueden residir en computadores diferentes conectados por una red de área local
- Algunos clientes y servidores pueden residir en una máquina, y otros clientes y servidores pueden residir en otra máquina

Una de las principales ventajas de este esquema, desde el punto de vista de la administración, es que el administrador del sistema puede distribuir clientes y servidores en distintos módulos de *hardware* acorde con sus necesidades y posibilidades. Esto ofrece al administrador del sistema la flexibilidad para crecer en cuanto al número y el tamaño de los clientes y servidores existentes, o bien reasignando los servidores a diferentes procesadores.

En otras palabras tenemos posibilidades de realizar escalabilidad horizontal (agregar o quitar clientes que accesan datos en el servidor) y escalabilidad vertical (posibilidad de migrar el servicio a una máquina servidora más grande o rápida) sin tener que afectar para nada la funcionalidad de los clientes.

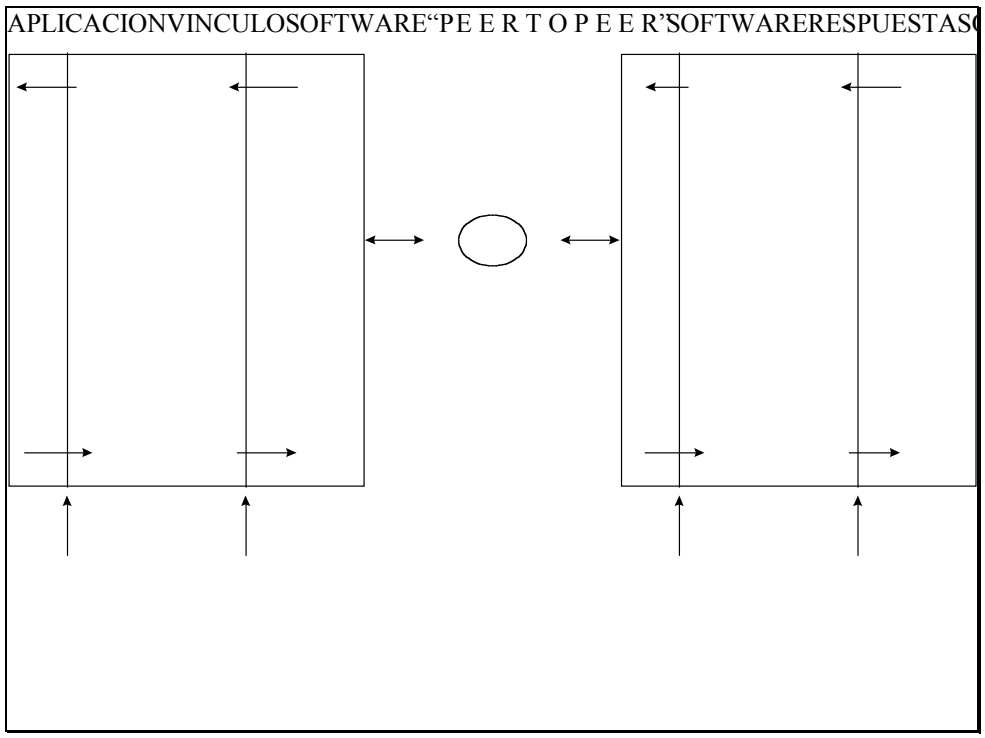
Se puede decir que el modelo computacional cliente/servidor está construido sobre la arquitectura del procesamiento cooperativo, o también sobre lo que se denomina arquitectura *peer-to-peer*[2]. Esta arquitectura usa un protocolo de comunicación (por ejemplo LU 6.2 o TCP/IP) para permitir a los dos procesos tener una conversación interactiva. En este modelo, cada programa ejecuta un conjunto de verbos interactivos para sostener el diálogo que inició alguno de los dos (normalmente el cliente). Cada participante en el diálogo interactivo tiene que poder anticipar y manejar todos los posibles requerimientos y/o réplicas. Esto incluye obviamente la recuperación de errores.

[2] Entiéndase el término *peer-to-peer* como comunicación entre iguales. Algunos autores también denominan a esta clase de comunicación "colegiada".

El *software* que realiza el ligamen coloca una capa de *software* sobre el protocolo *peer-to-peer*, lo cual ampliamente simplifica la programación de la aplicación y oculta los protocolos inferiores tanto del cliente como del servidor. El programa de aplicación establece contacto con el *software* de enlace por medio de un API (*Application Program Interface*). Este API debe existir en cada lado de la conversación (el programador no se preocupa de ello), y es mediante esta transparencia que ofrece el API que se ofrece una ilusión de un sola aplicación a pesar de estar dialogando *software* tanto en el cliente como en el servidor.

Los productos o conceptos de *software* más usados como APIs son los llamados a procedimientos remotos, conocidos por sus siglas en inglés RPC, y el SQL (*Structured Query Language*).

Se muestran a continuación dos gráficas que ilustran la idea:



Llamada a un Procedimiento Remoto
Figura 2

1.3 Características de la computación Cliente/Servidor

De acuerdo con [Bochenski 1994], son diez las características que definen un ambiente cliente/servidor. La existencia de las cinco primeras son de carácter obligatorio, mientras que las otras cinco son de cumplimiento opcional.

a) Una arquitectura cliente/servidor consiste de un proceso cliente y un proceso servidor que pueden ser distinguidos uno de otro y que pueden interactuar bastante independientemente.

b) Las partes cliente y servidor pueden operar, aunque no necesariamente, en plataformas computacionales diferentes.

c) Tanto la parte cliente como la del servidor pueden ser actualizadas individualmente sin que la otra deba serlo también.

d) El servidor es capaz de dar servicio a múltiples clientes en forma concurrente. En algunos sistemas pueden accesar múltiples servidores.

e) Un sistema cliente/servidor incluye algún tipo de capacidad de red.

f) Una porción significativa (a veces la totalidad) de la lógica de la aplicación reside en el cliente.

g) El procesamiento es iniciado usualmente en el lado del cliente, no del servidor. Sin embargo, los servidores de bases de datos pueden iniciar acciones basadas en "disparos automáticos", "reglas del negocio" o procedimientos almacenados.

h) Una interfaz gráfica de usuario amigable generalmente reside en el lado del cliente.

i) La capacidad de un lenguaje estructurado de consultas es una característica de la mayoría de los sistemas cliente/servidor.

j) El servidor de base de datos debería proporcionar seguridad y protección a los datos.

Por su parte [Orfali 1994] resume así las características de los sistemas cliente/servidor:

a) Servicio. El ambiente cliente/servidor conforma una relación entre procesos que se ejecutan en equipos separados. El proceso servidor ofrece servicios, mientras que los clientes los solicitan.

b) Recursos compartidos. Los servidores regulan el acceso a los recursos comunes por parte de los clientes.

c) Protocolos asimétricos. Los clientes pueden pertenecer a una amplia variedad de tecnologías, y son los responsables por iniciar las solicitudes de servicios. Un servidor es un ente pasivo que se encuentra en espera permanente por dichas solicitudes.

d) Transparencia de localización. Aún cuando un proceso servidor puede residir en el mismo equipo que los procesos clientes, es indispensable que los sistemas cliente/servidor oculten a éstos la localización física de los servidores, redireccionando apropiadamente las llamadas a los servicios requeridos.

e) Apertura. El *software* cliente/servidor debe ser lo más independiente posible de las plataformas de *hardware* y sistemas operativos involucrados.

f) Intercambios basados en mensajes. Los servidores y los clientes participan de sistemas "débilmente acoplados", cuya relación se implementa con mecanismos de paso de mensajes (*Message-Passing*).

g) Encapsulación de servicios. Los procesos servidores deben poder ser actualizados sin que esto provoque cambios en los clientes. Para lograr lo anterior, únicamente es necesario mantener inalterada la interfaz de comunicación entre ambos componentes.

h) Escalabilidad. El escalamiento de los sistemas cliente/servidor puede ser horizontal (adición o eliminación de clientes sin afectar significativamente el rendimiento global de un sistema) o vertical (crecimiento hacia configuraciones más grandes y eficientes).

i) Integridad. Los servicios y datos de un servidor se ubican en un lugar centralizado, lo que simplifica su mantenimiento y protección.

1.4 Integración en Cliente/Servidor

La computación cliente/servidor es la unión de lo mejor de los mundos del computador grande o mediano y los computadores personales. No es un tipo particular de *software* o *hardware*, es una arquitectura, una manera de utilizar el *hardware* y el *software* juntos. Es un paradigma en el cual las aplicaciones, los datos y el poder de procesamiento pueden ser distribuidos entre un repositorio centralizado y todos los computadores que lo accesan. El repositorio central es llamado el servidor o *back-end*. Este recibe los requerimientos de todos los otros computadores (clientes o *front-end*s). El servidor procesa los requerimientos que le llegan y regresa los resultados. Los clientes conocen todos los servicios que ofrece el servidor respectivo que están accesando. Para lograr esta comunicación (requerimiento-servicio) utilizan el protocolo o lenguaje común.

1.5 Acerca del *hardware* requerido

Cliente/Servidor es un concepto lógico que se sustenta en la existencia de diversos componentes de *hardware*. Recuérdese que es un asunto recursivo, pues un servidor en un momento dado puede convertirse en un cliente de otro servidor.

Una de las grandes ventajas del modelo cliente/servidor es que permite conectarse tanto con bases de datos locales como remotas, dando gran flexibilidad a la distribución que se quiere dar a la operación.

El servidor no debe ser una gran máquina, puede ser tecnología que no requiere de alta inversión ni costos de mantenimiento. Por otro lado también pueden ser equipos de tecnología abierta (no propietaria), los cuales a pesar de su bajo costo tienen gran poder de procesamiento y capacidad para trabajar con los mejores sistemas operativos y bases de datos existentes en el mercado.

Sin embargo, un servidor tradicional puede ser un equipo de cualquier tecnología: mini computadores (SUN, HP), tecnologías propietarias (DEC VAX, AS/400), equipos de tecnología risc o maiframes tipo IBM 3090. Pueden inclusive ser varios servidores en la misma red, cada uno de ellos ofreciendo un conjunto de servicios diferente (servidor de archivos, servidor de bases de datos, servidor de fax, etc).

1.6 Acerca del *software* requerido

¿Cómo hace el *hardware* para integrar lo mejor de los mundos PC y mini o mainframe? La fortaleza del equipo grande es su alta capacidad de almacenamiento, manipulación de vastas cantidades de datos, confiabilidad, alta disponibilidad y velocidad. La fortaleza primaria del PC es su interacción con el usuario. La aplicación puede dividirse de tal forma que el *back-end* ofrezca los servicios de procesamiento de datos y almacenamiento/recuperación de grandes cantidades de información. Por su parte, el cliente manejaría las interfaces de usuario, consultas *ad hoc*, filtrado y selección de datos. Esto le permite al desarrollador alcanzar un rendimiento óptimo y obtener flexibilidad y seguridad.

Un detalle interesante con respecto del *software*, es el hecho de que el sistema operativo en las plataformas del cliente y los servidores no debe ser idéntico. Muchas instalaciones seleccionan Unix, OS/2 o Windows NT como el sistema operativo del servidor para apoyarse en las facilidades de multitarea que ofrecen estos sistemas.

El *software* puede ser una herramienta de desarrollo (*front-end*) tal como Visual Basic, Delphi, PowerBuilder o SqlWindows, puede ser un lenguaje de consulta ad-hoc (tecnología), o una herramienta de usuario que permita realizar importaciones de datos. Pero, en términos generales, casi todas las aplicaciones tradicionales usarán como lenguaje de manipulación de datos el conocido SQL (*Structured Query Language*).

Una típica consulta de un cliente puede ser: “Déme los nombres de todos los vendedores quienes han obtenido ventas superiores a los trescientos mil colones el mes pasado”. Esto representa una consulta a la base de datos, uno de los posibles servicios que brinda el servidor. En SQL el anterior requerimiento se puede representar en algo como lo siguiente: SELECT Nombre FROM Vendedores WHERE Mes = “Enero” AND Ventas > 300000. La información resultante de esta consulta, la cual fue originada en el lado del cliente, reside en el servidor y es enviada a este último para su procesamiento o presentación.

1.7 Definición rigurosa de cliente/servidor

La computación cliente/servidor es una arquitectura de procesamiento en la cual una sola aplicación está particionada entre múltiples procesadores los cuales cooperan en una manera unificada para completar la unidad de trabajo como si fuera una sola tarea. Una definición más rigurosa es la siguiente [Boar 1993]:

> "Cliente/servidor es un modelo de procesamiento computacional en el cual una sola aplicación está distribuida entre múltiples procesadores (*front-end* y *back-end*), y los procesadores cooperan (en forma transparente al usuario final) para completar el procesamiento como una sola tarea unificada. Un producto final une a los procesadores para proveer una imagen de un solo sistema (ilusión). Los recursos compartidos son posicionados como servidores ofreciendo uno o más servicios. Las aplicaciones (el que requiere el servicio) son vistas como clientes los cuales accesan servicios autorizados. La arquitectura completa es totalmente recursiva; en un momento dado, los servidores pueden convertirse en clientes y requerir servicios de otros servidores en la red."

Un valor agregado, el cual aparece un tanto oculto en la concepción de esta tecnología, es el hecho de que con este modelo los usuarios podrán explotar al máximo la capacidad de su computador personal. No se debe perder de vista que en el modelo, el cliente es típicamente la aplicación ejecutando en un PC. Ahora bien en este PC no solo ejecutarán aplicaciones desarrolladas por el departamento de sistemas, sino que también existirán diversas herramientas con las cuales el usuario se sienta bien identificado y por las que se siente bien apoyado. Con el apoyo del ambiente de ventanas e interfaces gráficas, el usuario tendrá la capacidad de extraer sus propios datos y manipularlos a su antojo sin depender para nada de sistemas predefinidos o depender del departamento de sistemas, esto es lo que se denomina consultas *ad hoc*. Toda esta manipulación de la información que requiere el usuario, se procesa en forma local, es decir en el PC, con ello estamos distribuyendo el poder de procesamiento en la red, y disminuyendo la eventualidad de saturación en el servidor que almacena la base de datos. Esta interacción es también procesamiento cliente-servidor.

El modelo cliente/servidor combina las fortalezas de un servidor de bases de datos de alto poder, con el diseño visual y facilidad de uso que provee un cliente *front-end* (como PowerBuilder, Visual Basic, SqlWindows y otros). Los servidores de bases de datos relacionales como Informix, Sybase, Oracle, están altamente optimizados para el procesamiento de datos complejos de alto volumen; eso hace que cuenten con características avanzadas de seguridad e integridad, por las cuales el programador no debe preocuparse pues ellas son atendidas por el servidor. Al tener localizados los datos en un servidor de base de datos, los beneficios de velocidad y seguridad se incrementan y la cantidad de tráfico en la red disminuye notablemente.

El modelo cliente/servidor lleva intrínseca la filosofía de "usar la herramienta justa para el trabajo específico" y combina las ventajas del cliente y el servidor para las necesidades de cada quien. El diseñador debe identificar los módulos independientes que pueden ser diseñados e implementados separadamente.

1.8 Principales beneficios

A grandes rasgos los principales beneficios que se obtendrán con la arquitectura cliente/servidor son los siguientes:

- **Mantenibilidad**. La descomposición de sistemas rígidos y monolíticos hacia partes discretas intercomunicadas facilita el mantenimiento y reduce los costos. Como sucede con la mayoría de productos ingenieriles, es más fácil dar servicio, reemplazar y arreglar componentes con interfaces bien definidas, que hacer el equivalente en unidades monolíticas.

- **Modularidad.** La arquitectura cliente/servidor está construida sobre la base de módulos conectables. Tanto el cliente como el servidor son módulos del sistema independientes uno del otro y pueden ser reemplazados sin afectarse mutuamente. Se agregan nuevas funciones al sistema ya sea creando nuevos módulos o mejorando los existentes.

- **Adaptabilidad**. Las facilidades de *software* que existen en los componentes del sistema que actuarán como clientes, permiten a una misma aplicación llegar a diversos elementos de la compañía sin tener que realizar ningún tipo de cambio al sistema.

- **Escalabilidad.** Junto con los atributos de modularidad, adherencia a estándares de la industria, navegación sobre sistemas abiertos y adaptabilidad; las soluciones cliente/servidor pueden ser orientadas a satisfacer las necesidades cambiantes de la empresa.

- **Portabilidad.** Actualmente el poder de procesamiento se puede encontrar en varios tamaños: super servidores, servidores, *desktop, notebooks*, máquinas portátiles. Las soluciones cliente/servidor basadas en estándares permiten a las aplicaciones estar localizadas donde sea más ventajoso u oportuno.

- **Sistemas abiertos.** Los sistemas cliente/servidor ya han alcanzado el nivel de madurez y funcionalidad de los sistemas propietarios, pero bajo la premisa de sistemas basados en estándares de la industria.

- **Autonomía.** Las máquinas cliente pueden ser de diversas configuraciones, tamaños, marcas y arquitecturas. Con una configuración adecuada, cada cliente puede trabajar en forma independiente o como parte de la red distribuida de la empresa.

1.9 Principales desventajas

Aunque las tecnologías cliente/servidor pueden proporcionar numerosos beneficios como los mencionados en el apartado anterior, no conforman la solución perfecta para las necesidades de administración de la información en una empresa, debido a que imponen también ciertas restricciones, algunas de las cuales son:

- Si una parte importante de la lógica de las aplicaciones es trasladada al servidor, éste puede convertirse en un "cuello de botella" del sistema global. En este caso, los recursos limitados

del servidor tienen una alta demanda por un número cada vez más creciente de usuarios [Berson 1996].

- Las aplicaciones distribuidas, en especial aquellas basadas en el modelo cooperativo, son más complejas que las no distribuidas, e imponen cargas adicionales de comunicación y por ende de transferencia de información (datos del usuario y *overhead* del sistema) [Berson 1996].

- Se requiere de un alto grado de compatibilidad y de sujeción a estándares de parte de los dispositivos (*hardware* y *software*) que conforman un sistema cliente/servidor, para que éste pueda funcionar de una manera efectiva y transparente para el usuario.

- Los ambientes y las herramientas cliente/servidor demuestran falta de robustez y confiabilidad cuando se les compara con los ambientes multiusuario tradicionales [Price 1995].

- La mayoría de las herramientas cliente/servidor obligan a los usuarios a aprender esquemas de desarrollo de aplicaciones totalmente nuevos para muchos (bases de datos relacionales distribuidas, programación orientada a objetos, etc.) y sin proporcionar ninguna ayuda de migración [Price 1995].

- La complejidad y el esfuerzo requerido para administrar y soportar un ambiente cliente/servidor grande basado en sistemas abiertos. Hay pocas herramientas bien reconocidas que soportan manejo de configuraciones, monitoreo del rendimiento y distribución de versiones de *software* [Price 1995].

2. Arquitectura y principales componentes de los modelos Cliente/Servidor

2.1 Procesamiento Distribuido, Cooperativo e "Igual-a-Igual" (*Peer-to-Peer*)

Según [Crepau 1989], el Procesamiento de Datos Distribuido "es un conjunto de recursos y actividades de procesamiento de datos geográficamente distribuidos que operan de una manera coordinada para apoyar uno o más elementos de una organización". En vez de colocar información en un repositorio central para su manipulación, el procesamiento distribuido permite asignar poder de procesamiento donde se requiere. Las telecomunicaciones suministran los enlaces entre los sistemas distribuidos, permitiéndoles comunicarse entre sí cuando es necesario. La capacidad para utilizar múltiples sistemas proporciona el respaldo y la disponibilidad requeridos cuando uno de ellos falla. Una ventaja del procesamiento distribuido es que cada elemento o nodo puede operar independientemente de los demás, y posee todos los recursos apropiados para ello. En un ambiente distribuido, las tareas que una vez pudieron ser cumplidas por un único sistema centralizado, son repartidas entre un número variable de sistemas autosuficientes.

El Procesamiento Cooperativo, aunque similar al distribuido, tiene una característica especial que lo diferencia de aquél y es la posibilidad que tienen las aplicaciones de ser divididas en partes que se ejecutan en múltiples plataformas. Es decir, en el Procesamiento Distribuido una tarea puede ejecutarse independientemente en más de un nodo, mientras que en el Procesamiento Cooperativo la misma tarea utiliza diferentes plataformas conectadas para su realización. El objetivo de este último tipo de procesamiento es el de asignar los elementos separados de un proceso computacional a aquellas plataformas o nodos que se encuentren mejor configurados para su ejecución.

En el Procesamiento "Igual-a-Igual" los componentes o nodos utilizan el mismo protocolo de comunicación en la red y mantienen una relación de igualdad en cuanto a las actividades que realizan. Un nodo puede solicitar servicios a otro y a su vez brindarlos a un tercero. Este tipo de procesamiento se considera de bajo nivel debido a que ciertas funciones como la administración de errores y los vencimientos de tiempo (*Timeouts*) en las transmisiones no son controlados por los protocolos, sino que deben hacerlo los programadores de sistemas o aplicaciones.

Los tipos de procesamiento mencionados se ilustran en las figuras que aparecen a continuación.

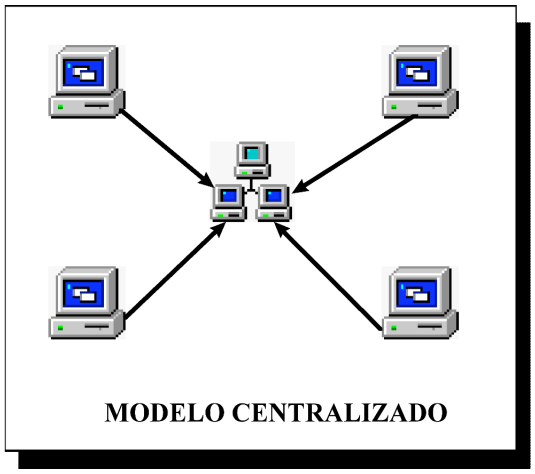


Figura 3a

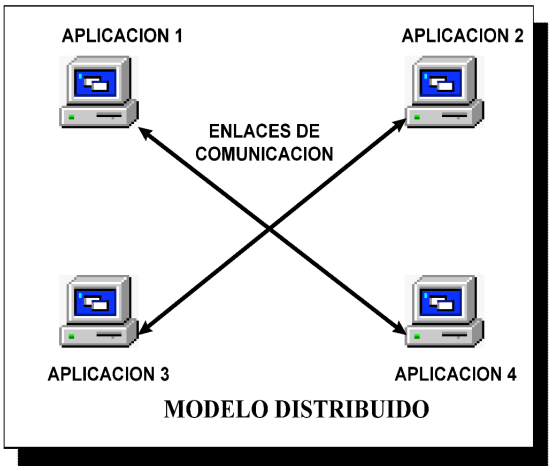


Figura 3b

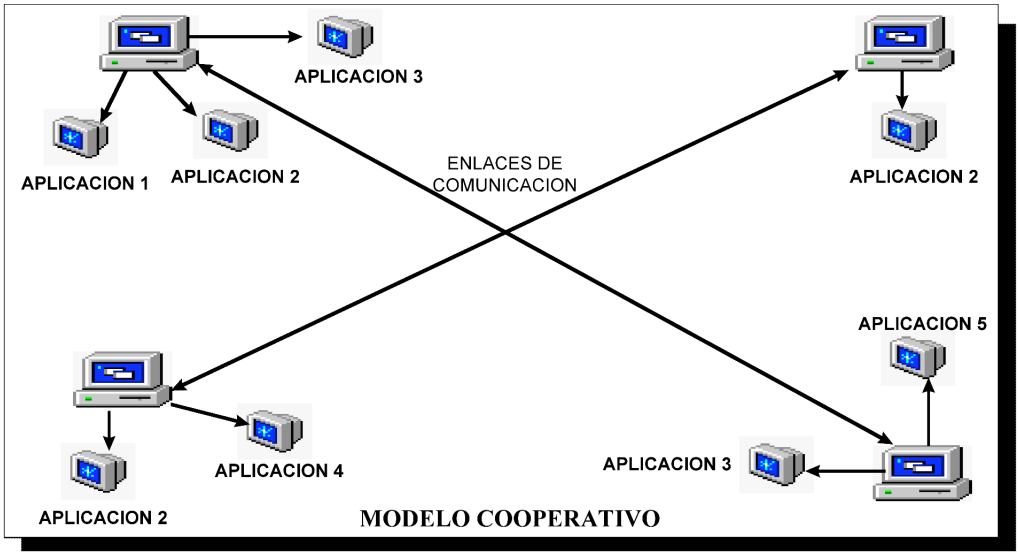


Figura 3c

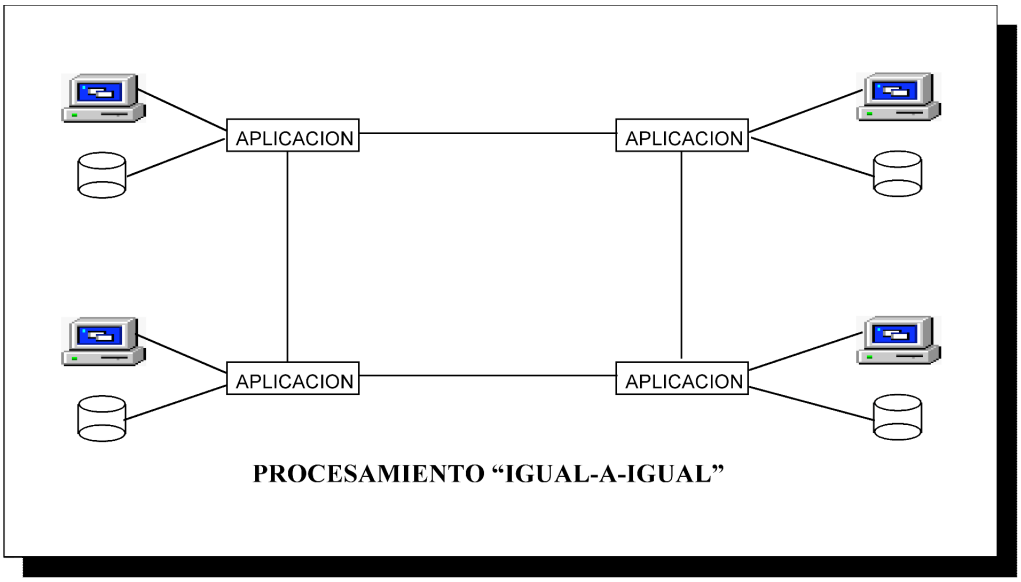


Figura 3d

2.2 Componentes principales de un modelo Cliente/Servidor

Un proceso cliente/servidor toma cualquier tarea y la divide en tres partes, una de las cuales es ejecutada por el **Cliente** que es el nodo (*Front-End*) origen de las operaciones y que interactúa directamente con el usuario final. Otra parte es asumida por el **Servidor** (*Back-End*) que proporciona servicios e información al cliente, y la última es una porción de **Red** (*Middleware*) que une o comunica a las dos anteriores.

A continuación se mencionan algunas características relevantes de estos componentes y de otros considerados también como esenciales.

2.2.1 Front-End

Los productos de *Front-End* o productos de usuario final, se pueden clasificar en aplicaciones y herramientas. Las aplicaciones, desarrollos particulares de *software*, son sistemas del tipo "llave en mano" o paquetes cerrados como por ejemplo, un procesador de palabras. Un alto porcentaje de estos paquetes incorporan características de acceso a servidores de bases de datos, la categoría tradicionalmente más representativa de *Back-End's*.

Las herramientas incluyen generalmente compiladores, lenguajes de programación, generadores de reportes, y otros. Los llamados 4GL's (*Fourth Generation Languages*) han venido evolucionado a partir de lenguajes tradicionales hacia sistemas cliente/servidor que poseen características de consulta e interrogación de bases de datos mediante el estándar SQL (*Structured Query Language*), convirtiéndose así en verdaderos programas de enlace entre los usuarios y aplicaciones locales o remotas.

A pesar de las diferencias fáciles de apreciar entre la naturaleza de ambas categorías, los fabricantes de *software* de hoy en día están constantemente agregando nuevas funcionalidades a los productos de usuario final, haciéndolos cada vez más sofisticados y difíciles de clasificar.

2.2.2 Back-End

Los servidores pueden ser de una naturaleza muy variada: de archivos, de correo electrónico, de impresión y red, entre muchos otros. El *software* del servidor consiste de un proceso lógico que proporciona servicios a procesos solicitantes. Más aún, los servidores permiten compartir las capacidades de dispositivos que ofrecen beneficios importantes.

2.2.2.1 Hardware

El *hardware* utilizado para los servidores debe tener suficiente poder de procesamiento y espacio de almacenamiento para satisfacer las necesidades de las aplicaciones que van a soportar. Este *hardware* puede variar desde poderosos microcomputadores (que emplean hoy en día el término servidor), hasta mini y supercomputadores, e incluyen otros componentes como controladores de discos, buses, unidades de respaldo en cintas, discos compactos, etc.

Los servidores más recientes pertenecen a la tecnología RISC (*Reduced Instruction Set Computer*) que se caracteriza porque disminuye la complejidad de la Unidad Central de Procesamiento (UCP) al usar un conjunto de instrucciones más simple que el implementado en la arquitectura tradicional CISC (*Complex Instruction Set Computer*), la cual utiliza microcódigo para ejecutar sus procesos. Al eliminar la tecnología RISC el modo de direccionamiento indirecto, mantener el tamaño de las instrucciones constante y disponer de conjuntos de instrucciones que se ejecutan en un ciclo o menos de reloj, su rendimiento es superior al *hardware* del tipo CISC.

Sin embargo, tener una UCP rápida no es suficiente si el resto de los componentes del *hardware* no responden de la misma manera. Por ejemplo, hace unos años, los buses estándares eran de tecnología ISA y movían datos a razón de 16 megabits por segundo. Las arquitecturas actuales, que incluyen discos, controladores, buses y memoria *caché* por niveles más rápidos, están implementados bajo la arquitectura EISA (y en menor grado el Micro Canal de IBM) que alcanzan velocidades de transferencia de datos de hasta 33 megabits por segundo.

Los actuales sistemas denominados de “misión crítica” están usualmente protegidos contra la corrupción de la información por medio de esquemas como la **duplicación** (*duplexing*) y el **espejo** (*mirroring*) de discos.

La técnica de *mirroring* involucra el uso de dos disco al mismo tiempo. El segundo disco tiene una copia exacta de los datos contenidos en el primer disco. Si el primer disco falla, el segundo toma su lugar mientras el otro es reparado o sustituido. Si ambos tuvieran porciones dañadas de su superficie, lo más probable es que se encuentren en secciones diferentes en cada uno de ellos, por lo que aún así es factible recuperar la información. Además de estos dos discos, se pueden usar otros más de respaldo.

La técnica de duplicación es similar a la del *mirroring*: uno o más discos adicionales contienen una copia de los datos del disco principal. La diferencia entre ambas es que la duplicación utiliza controladores (canales) independientes para los discos, reduciendo aún más la posibilidad de que el sistema se vea detenido por completo ante la ocurrencia de una falla en las unidades de almacenamiento. Además, la duplicidad de canales puede implicar una mejora significativa en la velocidad de respuesta de la red, al permitir que diferentes usuarios puedan leer simultáneamente el mismo archivo.

Las técnicas anteriores, especialmente la duplicación, forman parte de un concepto denominado "tolerancia a fallas" cuyo objetivo es mantener en operación permanente un sistema servidor, garantizando la integridad de los datos y afectando el rendimiento lo menos posible cuando ocurren fallas. El esquema de tolerancia a fallas más utilizado hoy en día es el conocido como RAID (*Redundant Arrays of Inexpensive Drives*). El arreglo de discos involucra el uso en pararelo de discos de almacenamiento. En los arreglos, los datos son distribuidos a lo largo de varios discos con un mecanismo llamado *stripping* (algo así como desgarrar en). Los archivos se fragmentan en pequeños bloques de datos o tiras que son escritos en diferentes discos. El esquema RAID se divide en cinco niveles que en general cumplen los siguientes objetivos:

- RAID nivel 1: Combina la duplicación con el *mirroring*.
- RAID nivel 2: Se utiliza principalmente en supercomputadores y realiza el *stripping* a nivel de bits.
- RAID nivel 3: Utiliza un disco dedicado exclusivamente al control de paridad. El *stripping* se aplica a nivel de byte o de segmento.
- RAID nivel 4: Similar al nivel 3 pero el *stripping* se efectúa a nivel de bloques de datos.
- RAID nivel 5: El *stripping* se realiza a nivel de bloque pero la información sobre la paridad se distribuye a lo largo de todos los discos en el arreglo.

2.2.2.2 Software

Entre los productos de *software* más utilizados como servidores de procesos se encuentran los administradores de bases de datos. En sus inicios, las técnicas de acceso a la información estaban basadas en manejadores de archivos que permitían a los usuarios compartir datos comunes, aunque no les era posible buscar o tener acceso a registros particulares de información. El nodo cliente tenía la responsabilidad de enviar mensajes e instrucciones al servidor, el cual devolvía archivos completos aún cuando el usuario solamente requería datos

específicos. Los servidores de bases de datos vinieron a resolver muchas de las limitaciones inherentes a los manejadores de archivos. Estos servidores proporcionan servicios como la administración de la concurrencia, el respaldo, la recuperación y la seguridad de la información. Sin embargo, uno de los principales beneficios de los administradores de bases de datos modernos es su capacidad para definir, estructurar y centralizar la forma en que opera una organización. Esto es posible mediante el uso de procedimientos almacenados (*Stored Procedures*), reglas del negocio (*Rules*), disparos automáticos (*Triggers*) y monitores de eventos, que permiten programar parte de la lógica de los sistemas en el servidor de bases de datos, evitando que ésta tenga que repetirse en cada aplicación cliente, con las consiguientes ventajas de la simplificación del mantenimiento de los sistemas de información, la reducción de los requerimientos de *hardware* de los nodos clientes, un mayor control sobre el acceso a los datos, además de que simplifican la consistencia de la aplicación y la actualización de versiones.

Las arquitecturas de bases de datos están fundamentadas en la administración de procesos. Un cliente mediante una aplicación puede crear varios procesos, los cuales estarán asignados a un espacio de direcciones (contexto) y serán ejecutados por un sistema administrador de bases de datos. Cada cliente tendrá entonces su espacio y se encontrará protegido de los demás. Además, si el *hardware* soporta el multiprocesamiento simétrico, los procesos del cliente serán atendidos por una "pila" de procesadores designados para tal efecto.

En otro esquema, el administrador de la base de datos emplea un solo contexto para atender las solicitudes de todos los clientes, descomponiendo los procesos en hilos y ejecutándolos en forma concurrente, para garantizar la mayor equidad en la distribución de los recursos computacionales entre los clientes.

2.2.3 Interfaces Gráficas

Las interfaces gráficas contribuyen de manera significativa a la popularidad de los ambientes cliente/servidor porque facilitan el uso y mejoran la productividad de los sistemas que lo componen. Estas interfaces se caracterizan por ofrecer capacidades como la multi-tarea (*multitasking*), la conmutación entre tareas (*task-switching*) y el intercambio de datos entre aplicaciones. Consisten además de una programación por eventos donde el código responde a acciones tomadas por los usuarios.

Los costos más comunes asociados a las interfaces gráficas se relacionan con la capacitación de los usuarios en el nuevo ambiente y a los recursos dedicados a su desarrollo. En este último aspecto, es posible considerar dos opciones: la utilización de herramientas para construir o "pintar" interfaces o el empleo de bibliotecas de funciones para programarlas, como por ejemplo, el *Software Development Kit* (SDK) de Microsoft. Se ha estimado que al menos el 80% del tiempo efectivo en el desarrollo de sistemas gráficos, se gasta en programación de las interfaces.

En sistemas operativos como Windows 95 o Macintosh, el manejo de la interfaz reside en el lado del cliente, mientras en otros sistemas como UNIX es el servidor el que tiene la responsabilidad de administrar la parte gráfica (excepto en aquellos casos donde se utilicen interfaces como X-Windows o Motif en el cliente).

Las interfaces gráficas se clasifican en dos categorías:

a) Interfaces GUI's (*Graphical User Interfaces*) que implementan el modelo objeto/evento, donde el usuario selecciona primero un objeto gráfico (ventanas de diálogo, menúes, cajas de selección, etc.) y luego las acciones correspondientes que éste deberá ejecutar, las cuales son comúnmente seriales. Estas interfaces se implementan a menudo en aplicaciones para Windows 3.x y OSF Motif.

b) Interfaces de Usuario Orientadas a Objetos (OOUI - *Object-Oriented User Interfaces*) que se caracterizan por interactuar con los usuarios mediante elementos gráficos llamados íconos, que permiten un acceso sencillo a la información representada por múltiples formatos visuales (datos, sonido, imágenes). Las aplicaciones en estas interfaces requieren, por lo general, un elevado nivel de comunicación en tiempo real, mucha interacción entre sí y un procesamiento concurrente de sus instrucciones. Mientras la información es desplegada al usuario, algunas tareas que se ejecutan en un plano secundario se encuentran trasladando información desde y hacia servidores de propósito específico. Algunos ejemplos de estas interfaces son Windows 95, OS/2 Workplace Shell, Macintosh y NextStep.

[Bochenski 1994] resume diez consideraciones importantes relacionadas con las interfaces gráficas:

a) Las GUI's pueden incrementar la productividad haciendo que los sistemas sean más fáciles de usar, y suministrando capacidades como la multi-tarea y el intercambio de datos entre aplicaciones.

b) La facilidad de uso no es automática. Es necesario algún nivel de capacitación para los usuarios.

c) La construcción de interfaces gráficas consume tiempo. El uso de herramientas para "pintar" GUI's puede acelerar el proceso de desarrollo.

d) La pérdida de consistencia en las interfaces puede confundir, irritar y frustrar a los usuarios.

e) Un estilo consistente es lo que le da a las interfaces su particularidad (apariencia *y* percepción).

f) Deben desarrollarse estándares dentro de las empresas para asegurar la programación de interfaces consistentes.

g) El uso de GUI's como *Front-End* puede mejorar el uso y la productividad de ciertos sistemas que, por su naturaleza, son difíciles de manipular.

h) Interfaces gráficas para sistemas como Windows, Macintosh o Presentation Manager, residen del lado del cliente. Por el contrario, en plataformas UNIX, los sistemas basados en X-Windows son administrados por el servidor.

i) Existen actualmente herramientas para construir interfaces gráficas genéricas que sean transportables a diferentes plataformas, como por ejemplo, de Windows a UNIX.

j) Los usuarios finales pueden tener diferentes GUI's ejecutándose al mismo tiempo en sus estaciones de trabajo.

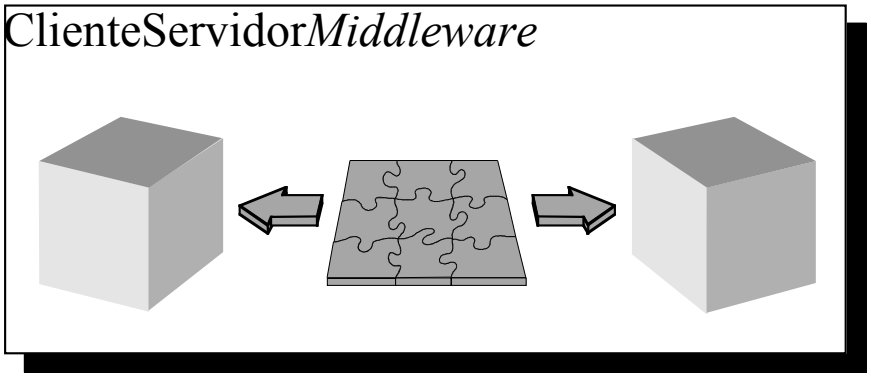
2.2.4 Comunicaciones, Redes y *Middleware*

Los ambientes cliente/servidor de hoy en día, requieren un conocimiento importante del ámbito de las comunicaciones y principalmente de las redes telemáticas. Los desarrolladores de aplicaciones necesitan determinar las capacidades que en estos ámbitos requieren los futuros sistemas, cuál *software* de comunicaciones está disponible y cómo será instalado.

También, se requerirá establecer si el *software* de comunicaciones en el cliente y en el servidor podrá trabajar adecuadamente con el sistema operativo de red disponible. Por último, es imperativo determinar cómo se accederán los datos que se encuentran distribuidos entre los diferentes servidores.

Relacionado estrechamente con el concepto de redes, se encuentra el término ***Middleware*** o *software* de interconexión. En alguna literatura, se interpreta el *middleware* como el conjunto de programas que permiten el acceso a los datos, mientras que en otros escritos se define como el *software* que conecta dos módulos y que les permite comunicarse. En términos generales, es el *software* que facilita la interacción entre los clientes y los servidores, e incluye elementos como las pilas de comunicación, los directorios distribuidos, los servicios de autenticación, las llamadas procedimientos remotos y el manejo de colas. En algunos casos, los desarrolladores de aplicaciones cliente/servidor no tienen que preocuparse de los programas de interconexión como por ejemplo, en el uso de herramientas para interrogación de bases de datos centralizadas. En otros casos por el contrario, es imprescindible conocer las particularidades de dichos programas, porque las aplicaciones así lo requieren (imaginemos el caso de la administración de bases de datos distribuidas, donde la dispersión geográfica de la información debe ser conocida, en mayor o menor grado, por los diferentes procesos que la manipulan). Un ejemplo en este sentido son las Llamadas a Procedimientos Remotos (RPC's).

Algunos productos generales y conocidos de *middleware* son: Novell Netware, TCP/IP, NetBIOS (*Network Basic Input/Output System*) y OSF *DCE (Open Software Foundation Distributed Computing Environment*). Por otro lado, entre los productos específicos tenemos: ODBC (*Open Data Base Connectivity*), EDA/SQL (*Enterprise Data Access/Structured Query Language*), MAPI (*Microsoft Application Programming Interface*), VIM (*Vendor-Independent Messaging*) y SNMP *(Simple Network Management Protocol*).

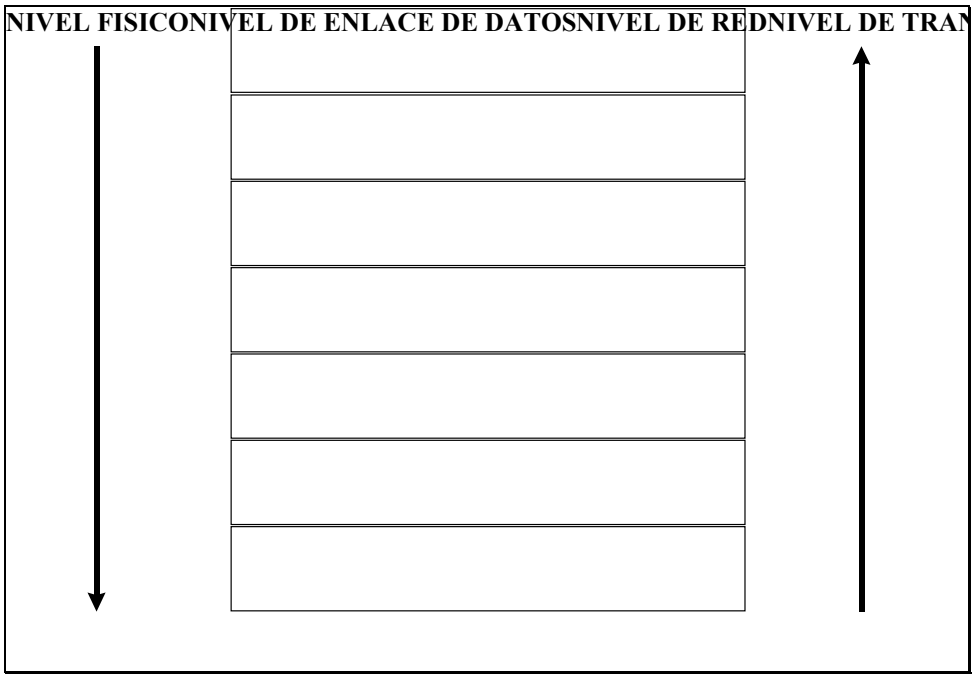


Middleware
Figura 4

2.2.4.1 Modelos de Referencia

Conforme las comunicaciones permiten una integración incremental entre las redes de computadoras, los estándares se vuelven cada vez más importantes, debido a la gran cantidad de dispositivos de diferentes fabricantes que hacen su aparición en el mercado. Sin duda alguna, dentro de los estándares que tienen la mayor difusión, sobresale el **modelo OSI** (*Open Systems Interconnection*) de la organización ISO (*International Standards Organization*).

El modelo OSI comprende siete capas o niveles que abarcan desde la parte física hasta las aplicaciones. Cada nivel en un estrato superior implementa servicios que dependen de las funciones del nivel inferior. Tanto las funciones como los servicios se definen en una especificación de protocolo, el conjunto de los cuales describe las reglas que gobiernan la comunicación entre dos puntos en la red. Esta comunicación es descendente en el nodo que origina el enlace (el flujo de mensajes o paquetes fluye desde los niveles más altos, hasta el medio de transmisión) y ascendente en el nodo receptor.



Modelo de Referencia OSI
Figura 5

El *nivel físico* transporta los bits de un extremo a otro, y es responsable por la sincronización del enlace. El *nivel de enlace de datos* garantiza que la transmisión sea confiable entre los nodos. El *nivel de red* es responsable por enrutar los mensajes, estableciendo, manteniendo y terminando las llamadas conexiones de red, para cumplir este propósito. El *nivel de transporte* es el encargado de transferir los datos entre dos entidades lógicas específicas (procesos). El *nivel de sesión* es el que crea y administra los diálogos de comunicación (sesiones) mediante los cuales los usuarios transmiten o reciben datos. El *nivel de presentación* es el que se relaciona con la sintaxis y la semántica de la información, proporcionando servicios de conversión, compresión y expansión de datos. Finalmente, el *nivel de aplicación* es el que se relaciona directamente con los usuarios, permitiendo a éstos establecer conexiones entre sí y ejecutar sus procesos.

Otro modelo conocido aunque no tan utilizado hoy en día, es el **MAP/TOP** desarrollado por la General Motors en los inicios de la década del 80 para sus aplicaciones de manufactura (y ampliado posteriormente por la Boeing Company), donde la existencia de variados dispositivos de comunicación en las plantas de la fábrica, presentaban un serio problema de interconexión entre los mismos.

Hoy en día el protocolo más comunmente utilizado que implementa el Modelo de Referencia OSI es el TCP/IP.

2.2.4.2 Componentes de una red

Aunque muchos son los componentes que integran una red, tres son los básicos: un sistema de

cableado que interconecta los dispositivos o nodos[3], tarjetas de interfaz y un sistema operativo de red.

El cable puede ser par telefónico, coaxial o fibra óptica. La escogencia del cable depende en gran medida de consideraciones como la topología de la red (bus, anillo, estrella), el tamaño de esta (LAN, WAN, etc.) y la seguridad y el rendimiento esperado. La fibra óptica, por ejemplo, posee un ancho de banda más amplio que el de los otros tipos de cable, permitiendo un alto rendimiento. Además es bastante segura por ser inmune a las interferencias electromagnéticas y requerir la perforación del cable para obtener la información que transita por él, acción ésta fácil de detectar con mínimas medidas de seguridad.

Las tarjetas de interfaz o adaptadores de red, se instalan en cada nodo y sus características dependen del tipo de red al cual se enlazan: Ethernet, Token-Ring, Arcnet o Fibra Optica. Pueden ser internas (en el caso más común) o externas, y su propósito es detectar los mensajes que viajan por la red para determinar cuáles son enviados al nodo al cual están conectados. De la misma forma, cuando el nodo necesita transmitir un mensaje, el adaptador define el momento oportuno para colocarlo en la red.

El sistema operativo por su parte, controla el flujo de mensajes y proporciona otros servicios como el de administración de archivos y de impresión.

Además de la conexión de dispositivos a una red, es muy común encontrar también que se requiere la comunicación entres redes. Esto es especialmente cierto en la actualidad, con el uso tan popularizado de la red de redes Internet. Para lograr esta interconexión, es preciso contar con otro grupo de dispositivos, muy similares en sus funciones, como lo son: Puentes (*Bridges*), Enrutadores (*Routers*), Repetidores (*Repeaters*), Concentradores (*Hubs*) y Compuertas (*Gateways*).

Tomando como referencia el modelo OSI, los puentes operan en el nivel de enlace de datos, conectando redes que tienen diferentes protocolos en el nivel físico. Son fáciles de instalar, ofrecen un buen rendimiento, y no necesitan ser reconfigurados cuando los nodos son trasladados de posición en una red. Sin embargo, presentan problemas a la hora de manejar retrasos producidos por el exceso de paquetes en el medio de transmisión y ofrecen poco apoyo en el aislamiento de fallas.

Un enrutador se desempeña en el nivel de red y puede conectar redes que tienen distintos protocolos en el nivel de enlace de datos. Los enrutadores presentan varias ventajas como por ejemplo, tienen la inteligencia para enviar los mensajes por distintas rutas tomando en cuenta aspectos como la disponibilidad de éstas y las prioridades de los usuarios, pueden soportar cualquier topología que presenten las redes, pueden prevenir que problemas en una red afecten a las otras que se encuentran conectadas, y son capaces de manejar distintos tamaños de mensajes, de acuerdo con el protocolo que se esté utilizando en ese momento. Entre las desventajas que

[3] Las redes inalámbricas (*Wireless Networks*) que utilizan principalmente frecuencias de radio, están tomando cada día más auge, a pesar de que su tecnología está en etapa de investigación y desarrollo.

muestran los enrutadores están su mediano costo, son difíciles de configurar y no son tan eficientes como los puentes.

Los repetidores son dispositivos simples que se encargan de regenerar las señales que viajan por la red que, debido a las distancias involucradas, van progresivamente disminuyendo su intensidad. Los concentradores permiten conectar en un punto central, puentes, enrutadores, repetidores y otros concentradores. Por último, las compuertas enlazan redes de diferentes protocolos mediante la conversión de estas y opera en los niveles más altos (Presentación y Aplicación).

2.2.4.3 Modelos de Comunicación

Los sistemas cliente/servidor modernos se encuentran constituidos por un amplio rango de tecnologías e implementados con diferentes enfoques de comunicación. Sin embargo, de estos últimos, tres son los esquemas más utilizados:

a) Modelo Conversacional. Ocurre entre aplicaciones del tipo *peer-to-peer*, donde se establece una conexión lógica entre dos únicos procesos, aunque no existen límites teóricos a la cantidad de interacciones que puede desarrollar cada proceso.

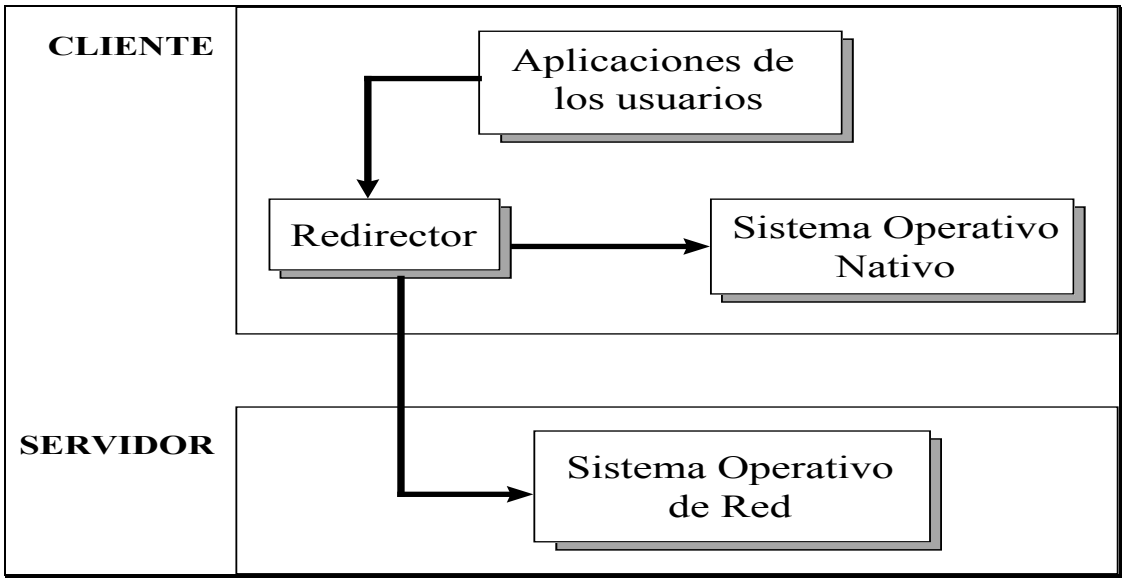
b) Paso de Mensajes. Los mensajes pueden representar solicitudes de servicio de los clientes o respuestas de los servidores a estas, respetando un formato o interfaz específicos. Estos mensajes son recibidos e incorporados en una cola del tipo FIFO (*First In-First Out*) o bien procesados en forma inmediata.

c) Comunicación Interprocesos (IC). Los procesos pueden residir en el mismo equipo o en equipos diferentes. Lo importante aquí es la coordinación en el uso de los recursos compartidos. Existen dos técnicas principales para implantar este tipo de comunicación: los *semáforos* que son variables o banderas que indican la disponibilidad o no de los recursos, y los *conductos* (*pipes*) que conectan diversos elementos, permitiendo el flujo de datos entre ellos.

2.2.4.4 Sistemas Operativos de Red

Un sistema operativo tradicional (conocido como Sistema Operativo Nativo) proporciona múltiples servicios a las aplicaciones que los solicitan, por medio del *hardware* único que controla. Un sistema operativo de red (NOS - *Network Operating System*) extiende estos servicios a muchos computadores diferentes.

Una parte del NOS llamada *Redirector*, reside en cada computador cliente y se encarga de examinar las solicitudes de los usuarios o aplicaciones y de determinar si éstas pueden ser atendidas por el cliente mismo o, en su defecto, deben dirigirse a un dispositivo apropiado en la red. Utilizando al Redirector, un cliente puede acceder los recursos de la red como si los tuviera localmente conectados.



Arquitectura de un Sistema Operativo de Red
Figura 6

Entre las características más sobresalientes de un NOS se pueden distinguir:

a) Independencia del *hardware*. El sistema operativo debe poder utilizarse en el más amplio dominio de *hardware* de red.

b) Soporte de múltiples servidores. La comunicación entre servidores debe ser tan transparente como sea posible.

c) Administración de la red. El NOS debe tener integradas funciones de administración y mantenimiento de la red, como programas utilitarios para respaldos, monitoreo del rendimiento, tolerancia a fallas, y otros.

d) Seguridad, protección y control de acceso a los recursos.

Los sistemas operativos de red suministran dos tipos de servicios: básicos y extendidos. Las tablas siguientes resumen algunas características relevantes de cada tipo:

Servicios Básicos
1. Coordinación de multitareas. El sistema operativo asigna a las diferentes tareas (porciones de un programa) intervalos fijos de ejecución y coordina su ejecución concurrente para garantizar que las tareas avancen hasta su conclusión.
2. Priorización de tareas. Las tareas deben ejecutarse de acuerdo con prioridades que permitan diferenciar el nivel de servicio ofrecido por los servidores a los clientes.
3. Intercomunicación de procesos. Procesos independientes deben poder intercambiar y compartir datos.
4. Manejo de hilos (*threads*). Los hilos son grupos de instrucciones que son estructuradas para maximizar la concurrencia o avance de las tareas. Estos grupos se asignan a eventos para organizar su ejecución en el tiempo.
5. Protección de tareas. El sistema operativo debe proteger a las tareas de interferir unas con otras en el manejo de sus propios recursos.
6. Sistema de administración de archivos multiusuario. Debe permitirse el acceso a los archivos por parte de múltiples tareas, garantizando la integridad de la información.

Servicios Extendidos
1. Diversidad de protocolos de comunicación. Los NOS deben facilitar la comunicación de los servidores entre sí y con clientes que pertenezcan a la mayor cantidad de plataformas de *hardware* posibles.
2. Soporte de objetos (BLOB's - *Binary Large Objects*). La naturaleza cada vez más gráfica de las aplicaciones computacionales, obliga en la actualidad, a construir sistemas operativos que manejen representaciones poco tradicionales de datos en formatos como sonido, vídeo, imágenes y otros.
3. Directorios Globales. Los clientes en una red deben conocer la localización de los servidores y los servicios que éstos proveen, por medio de un directorio general que se actualice dinámicamente conforme nuevos servidores se incorporen a la red o bien otros desliguen de ésta.
4. Servicios de autenticación y autorización. Un NOS debe garantizar a los servidores que los clientes que solicitan los servicios son quienes realmente dicen ser y que además tienen permiso para hacerlo.
5. Sincronización de clientes y servidores. La sincronización de los relojes entre los clientes y los servidores es esencial en aplicaciones como bases de datos y en todas aquellas dependientes de eventos.
6. Control del Multiprocesamiento. El término multiprocesamiento se refiere a la capacidad de un sistema operativo de administrar varios procesadores en un mismo dispositivo. El multiprocesamiento puede ser Asimétrico o Simétrico. En el primer caso, un único procesador ("Maestro") es el que posee la capacidad de ejecutar el sistema operativo y es el responsable de coordinar a los demás procesadores ("Esclavos"), los cuales están dedicados a funciones específicas como Entrada/Salida, Gráficos, y otros. En el Multiprocesamiento Simétrico, todos los procesadores tienen

iguales responsabilidades y pueden sustituirse unos a otros en sus labores. Las aplicaciones son divididas en hilos que son asignados para su ejecución concurrente en los procesadores disponibles por una rutina del sistema operativo llamada calendarizador (*Scheduler*).
7. Administración de Transparencia. La transparencia es la capacidad que tiene un NOS de aislar de consideraciones físicas a los clientes de una red, con el objetivo de simplificar y hacer más eficiente su funcionamiento. Puede establecerse en varios niveles: localización, nombramiento y replicación de los recursos, fallos y administración de la red, sincronización de tiempo, etc.

Los sistemas operativos de red pueden abarcar desde ambientes pequeños con un reducido número de computadores (LANtastic y Novell Netware Lite), hasta redes medianas y super redes (Microsoft Windows NT, Banyan Vines, Microsoft LAN Manager y Novell Netware). Las redes pequeñas casi siempre operan con un esquema *peer-to-peer* donde no existen dispositivos que asumen la función de servidores. Por el contrario, todos los nodos involucrados comparten las mismas responsabilidades sobre una base de igualdad operativa. Mientras tanto, en las redes de mayor escala, los NOS brindan acceso a servidores especializados como los de impresión y de bases de datos, además de funciones para interconexión con redes de área metropolitana (MAN - *Metropolitan Area Network*) y de área extensa (WAN - *Wide Area Network*).

2.2.4.5 Mensajería y Llamadas a Procedimientos Remotos (RPC's)

En el mecanismo de Llamadas a Procedimientos Remotos, un cliente invoca una función o servicio de un servidor remoto y detiene su ejecución hasta recibir una respuesta. El *software* que maneja los RPC's obtiene los parámetros del cliente que serán enviados, forma un mensaje con ellos y lo dirige hacia el servidor. Este recibe el mensaje, interpreta los parámetros, ejecuta el servicio y devuelve el resultado al cliente.

La mensajería es un mecanismo de comunicación entre aplicaciones que utiliza colas de entrada y salida de mensajes, por lo que no se necesitan conexiones dedicadas y privadas para que los nodos intercambien información. Estos se comunican entre sí poniendo y retirando mensajes de las colas respectivas y, a diferencia de los RPC's, no requieren que sus envíos reciban respuesta, y pueden atender cada mensaje en el momento que lo deseen, siendo entonces una técnica de procesamiento completamente asincrónica. Este esquema es muy flexible debido a que los mensajes pueden ser guardados en disco (colas persistentes) o en memoria (colas no persistentes), y las colas pueden residir local o remotamente.

2.2.4.6 Pilas

Las pilas son grupos de niveles de funciones en un modelo de referencia de red (como por ejemplo el OSI), los cuales proporcionan algunos servicios de comunicación. Cada pila tiene un conjunto de protocolos e interfaces de programación de aplicaciones (API's) bien definidos, siendo las de más bajo nivel aquellas que controlan el medio físico de transmisión y los adaptadores de comunicación con la red. En cada nivel, los clientes y los servidores cooperan conjuntamente para ejecutar los servicios que se implementan a partir de los servicios del nivel inmediatamente anterior. Entre más alto se encuentre un nivel en la pila, más abstractos se vuelven los servicios.

Las pilas más utilizadas en redes de comunicación son: TCP/IP (UNIX e Internet), Novell Netware IPX/SPX, IBM NetBEUI, IBM APPC/SNA y Named Pipes.

2.2.4.7 Ambiente de Computación Distribuida (DCE)

El estándar OSF DCE o *Distributed Computing Environment* es muy importante porque proporciona una solución para integrar servidores de diferentes fabricantes en un único ambiente cliente/servidor heterogéneo, además de definir un enfoque integral para la seguridad y la comunicación entre procesos. Un cliente DCE puede, por ejemplo, interactuar con uno o más servidores que pertenezcan a plataformas de *hardware* y sistemas operativos distintos.

En realidad, el DCE permite especificar un entorno que abarque diferentes arquitecturas, protocolos y sistemas operativos, para crear, ejecutar y dar mantenimiento a aplicaciones distribuidas, por medio de componentes o servicios, algunos de los cuales son:

a) Servicio de Hilos

b) Seguridad

c) Tiempo Distribuido (sincronización de relojes)

d) Llamadas a Procedimientos Remotos

e) Servicios de Directorio

f) Archivos Distribuidos

La importancia actual del DCE se debe al respaldo que ha recibido de los principales fabricantes de *hardware* del mercado, entre los que se encuentran: IBM, DEC, Hewlett-Packard, Gradient, Cray, Siemens y Tandem.

2.2.4.8 Protocolos de Red

Existen varios protocolos estándares que definen el camino de las redes de computadoras de hoy en día. Entre éstos se pueden mencionar el IEEE 802, el TCP/IP, la Conmutación de Paquetes (*Packet Switching*), el *Frame Relay* y el Modo de Transferencia Asincrónica (ATM - *Asynchronous Transfer Mode*).

El proyecto **IEEE 802**, es en realidad un conjunto de definiciones de protocolos, que se aplica especialmente a las redes de área local y actúa en los niveles físico y de enlace de datos. El 802.3 conocido como CSMA/CD (*Carrier Sense, Multiple Access with Collision Detection*) es para redes Ethernet con topología de bus, mientras que el 802.4 describe una arquitectura de Token-Bus y el 802.5 una de Token-Ring.

Por otro lado, el **TCP/IP** (*Transmission Control Protocol/Internet Protocol*) diseñado por el Departamento de Defensa de los Estados Unidos, es un protocolo público, no propietario, utilizado ampliamente en redes de área extensa (WAN). Este protocolo abarca los niveles tres y cuatro del modelo OSI (red y transporte), y forma la base de la gran red mundial Internet.

La **Conmutación de Paquetes** es una tecnología asociada a las redes de área extensa, donde todos los mensajes son ensamblados en unidades llamadas paquetes. Cada paquete se compone de un encabezado con información de control y un detalle con los datos por transmitir. Cuando un paquete llega a un nodo, éste lee el encabezado para averiguar si es para él. Si no lo es, redirige el paquete hacia su destino. La conmutación de paquetes está fundamentada en las técnicas de conexión. Una red de paquetes proporciona un circuito virtual que da la impresión de que dos nodos se comunican por medio de un enlace punto a punto. La velocidad máxima de transmisión que puede alcanzarse con la conmutación de paquetes es de 56 kilobits por segundo. El X.25 es la implementación estándar de este protocolo.

El ***Frame Relay***[4] está diseñado para interconectar redes de área local entre sí y con redes de área extensa. Entre sus características sobresalen su alta capacidad de transmisión, alta eficiencia, baja sobrecarga[5] (*overhead*), y una confiable transmisión de la información sobre enlaces públicos de comunicación. Este esquema no incorpora funciones de identificación y corrección de errores, y trabaja a velocidades de 1.5 (DS1) y 2 megabits por segundo. La única desventaja del *Frame Relay* es que no es muy conveniente para transportar voz y datos al mismo tiempo, debido a que utiliza paquetes (*frames*) de longitud variable que pueden llegar a producir retrasos considerables.

[4] *Frame Relay* y ATM fueron cubiertos en el informe del Club titulado Redes Empresariales de Banda Ancha por Aníbal Mayorga.

[5] La sobrecarga es información adicional a la del usuario que debe adjuntarse a los mensajes para que sean administrados por un protocolo de comunicación. El ideal es mantener la razón datos del usuario/datos del protocolo lo más alta posible.

El **Modo de Transferencia Asincrónica**, también conocido como *Cell Relay*, está tomando mucho auge en la actualidad. Su principal ventaja radica en que permite integrar redes de área local y área extensa con voz, datos, imágenes y video en una misma red, a velocidades que oscilan desde los 4 megabits por segundo hasta los 45 megabits por segundo. En esta tecnología, la información es dividida en celdas (*cells*) de 53 bytes de longitud, que puede ser transmitida por medio de esquemas basados en paquetes o basados en circuitos.

2.2.5 Procesamiento Distribuido y Transacciones

Un sistema de procesamiento distribuido es un esquema cimentado en un conjunto de elementos independientes que están conectados por una red de computadoras, y cuyo propósito es cooperar entre sí para poder ejecutar sus propias tareas (procesos). Los datos involucrados en esta tareas pueden estar distribuidos o localizados en un lugar central. El concepto más importante en un sistema distribuido es el de **transacción**, que se define como una unidad de trabajo. A partir de su origen, usualmente un cliente, es administrada por uno o más servidores y retorna finalmente a su punto de partida. Una vez concluida, todas las partes involucradas deben estar de acuerdo en aceptarla o rechazarla.

De acuerdo a [Gray 1993], toda transacción debe reunir cinco características:

a) **Atomicidad**. La transacción es indivisible. Todas las acciones que comprende se ejecutan como un todo, o se cancelan completamente.

b) **Consistencia**. Una vez concluida la transacción, el sistema debe quedar en un estado consistente.

c) **Aislamiento**. El comportamiento de una transacción no debe verse afectado por otras transacciones que se ejecuten concurrentemente.

d) **Persistencia**. Los efectos de una transacción después de que se ha aplicado (mediante una operación *commit*), deben ser permanentes, aún cuando el sistema falle.

e) **Serialización**. Si una transacción que está siendo ejecutada depende de cierta información, ésta debe ser bloqueada para prevenir que otra transacción la modifique.

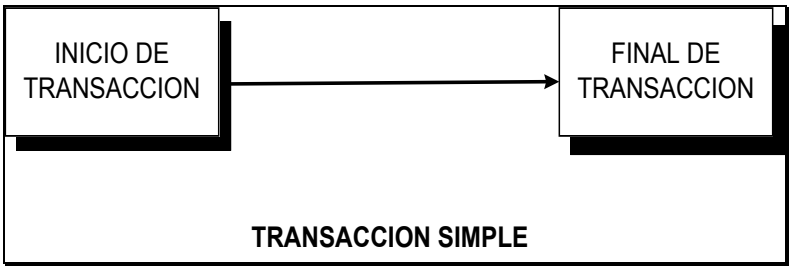
Todo lo anterior significa que una transacción es la unidad básica de recuperación, consistencia y concurrencia en un modelo cliente/servidor. Por lo tanto, todas las aplicaciones que participen de un sistema distribuido, tienen que sujetarse al esquema transaccional.

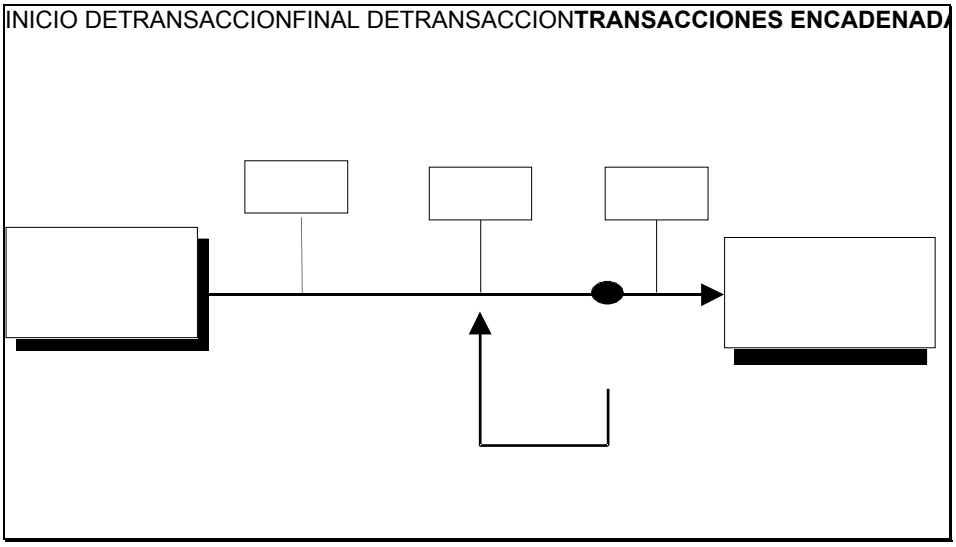
Las transacciones pueden dividirse en tres categorías:

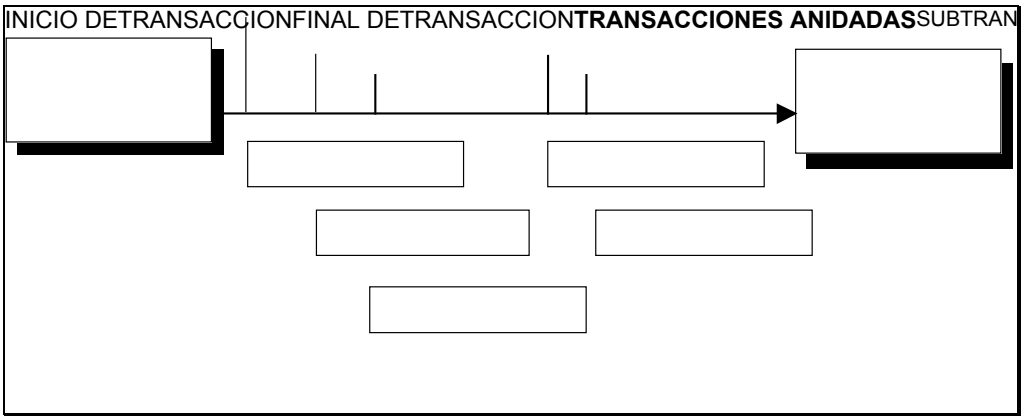
- Transacciones simples. Son únicas y de corta duración. Representan la forma más sencilla de transacción. Por su naturaleza, no son adecuadas para procesos fuera de línea o que se extiendan en el tiempo, y su función es liberar lo antes posible los recursos críticos del sistema que estén siendo utilizados por las aplicaciones. La principal desventaja de esta categoría es que si una transacción incluye un gran número de operaciones y falla por cualquier motivo, todo el proceso debe repetirse desde el inicio.

- Transacciones encadenadas. Este tipo de transacción utiliza el concepto de "Puntos de Sincronización" (*Syncpoints* o *Savepoints*) los cuales indican que cierto trabajo acumulado hasta ese momento, ha sido guardado en forma temporal. Si una parte de la transacción falla, el sistema retorna al último punto de sincronización ejecutado, e intenta a partir de ahí, recuperarse del error. Una variante a este esquema es el uso de operaciones "commit" en lugar de puntos de sincronización.

- Transacciones anidadas. En esta categoría, cada transacción tiene la capacidad de dividirse en subtransacciones y éstas a su vez en otras más, creando una estructura jerárquica. Cuando una subtransacción realiza un *commit*, sus resultados solo son accesibles por la transacción-madre. Además, una operación *commit* se vuelve permanente cuando todas las transacciones en orden ascendente han hecho también su *commit* respectivo. Por el contrario, una operación *rollback* de una transacción implica que todas las subtransacciones asociadas en orden descendente, ejecutan a su vez un *rollback*.







Categorías de Transacciones
Figura 7

El protocolo utilizado por el procesamiento distribuido para aplicar las transacciones terminadas exitosamente, es llamado *Commit* de Dos Fases (*Two-Phase Commit*). Este protocolo implementa un mecanismo rígido que sincroniza las actualizaciones en diferentes nodos, centralizando la decisión de aplicar el *commit* en un nodo coordinador, pero permitiendo que cada participante la acepte o no. Su utilidad radica en que garantiza la integridad de las actividades en un ambiente distribuido, a pesar de las condiciones adversas que éste enfrente. En la primera fase, el coordinador de transacciones envía a los demás nodos el aviso de que deben prepararse para ejecutar el *commit*. Si todos los nodos responden que pueden efectuar tal operación, entonces el coordinador, en la segunda fase, transmite la instrucción de aplicar el *commit*. Con solo que un nodo falle en ejecutar dicha instrucción, será suficiente para que el protocolo anule la transacción.

2.2.5.1 Monitores del Procesamiento de Transacciones

Los sistemas computacionales que supervisan el procesamiento transaccional "en línea" se denominan Monitores del Procesamiento de Transacciones o MPT (*Transaction Processing Monitors*). Estos monitores garantizan la consistencia de las operaciones sobre los datos entre aplicaciones que participan en ambientes de procesamiento cooperativo. La administración y el soporte de las transacciones son dos de los servicios más importantes que deben proporcionar los programas de *middleware*. Las recientes generaciones de monitores suministran otros servicios, útiles inclusive para ambientes "fuera de línea" como el correo electrónico, la automatización de flujos de procesos y los sistemas orientados al soporte de decisiones. Algunos de estos servicios incluyen:

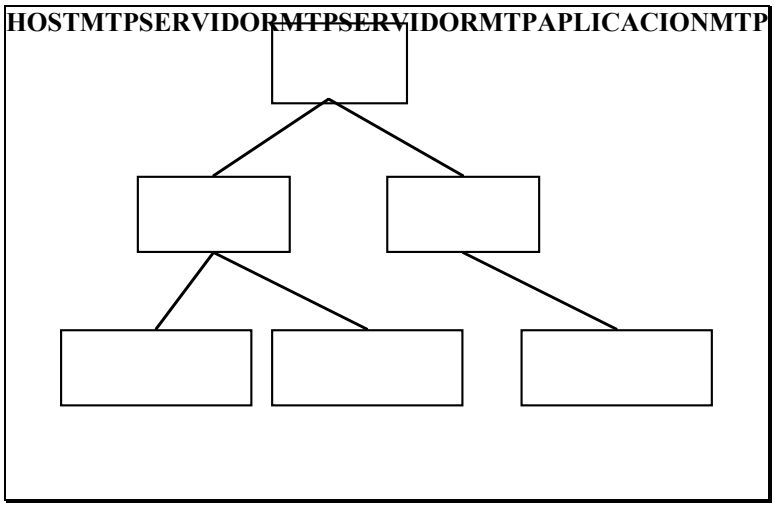
- La entrega garantizada de mensajes entre aplicaciones ejecutadas en sistemas operativos similares o diferentes.

- Escalabilidad automática mediante la replicación de aplicaciones.

- Cargas de trabajo reducidas de los servidores, al administrar las transacciones de los clientes.

- Administración simplificada de los sistemas, debido a que los monitores tienen la capacidad de balancear las cargas de trabajo, recolectar estadísticas del sistema y supervisar el rendimiento de éste.

- Mejoramiento en la disponibilidad de los sistemas al permitir la recuperación y el restablecimiento de las operaciones, utilizando los recursos de los clientes.



Monitores de Procesamiento de Transacciones
en ambientes multinivel
Figura 8

2.2.5.2 Objetos Distribuidos

Los cambios en la tecnología informática han sido permanentes desde la aparición del primer computador comercial en los años de 1940. El *hardware,* especialmente, se muestra como un área de innovaciones revolucionarias más que evolutivas, que hacen prácticamente impredecible su comportamiento futuro. El *software* por su parte, ha avanzado más lentamente, en muchos casos sin una dirección específica, pero siempre determinado de alguna manera por el *hardware*.

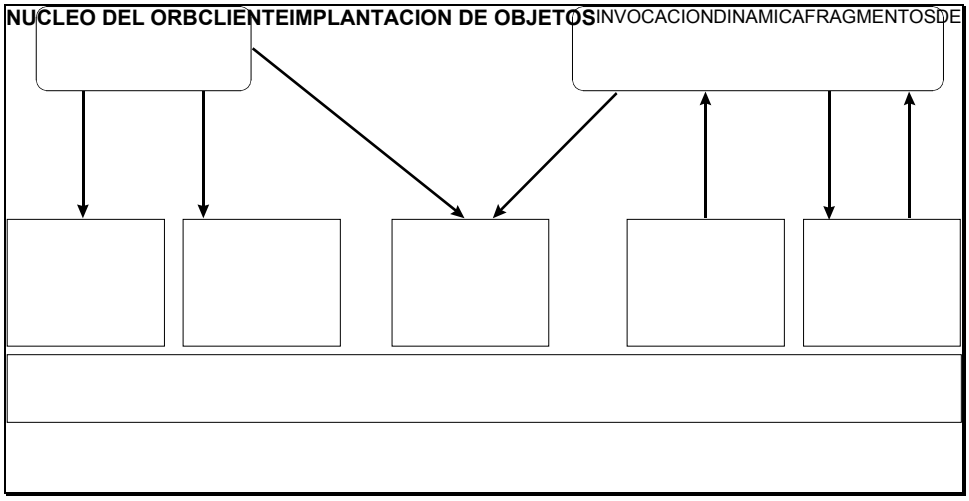
En un amplísimo universo tecnológico colmado de ideas, descubrimientos y posibilidades casi infinitas, solo aquellos conceptos verdaderamente relevantes permanecen, aunque se requiera de un tiempo considerable para su aceptación. La teoría de objetos es un ejemplo clásico de estas palabras. Por muchos años sus conceptos fueron casi ignorados ante el resplandor fulgurante que emitían otras propuestas, especialmente en los campos de la programación estructurada y los sistemas de información. Hasta que la experiencia no demostró lo limitado de las pretensiones en dichos campos, es que se dirigió la atención hacia las "viejas soluciones". Y es que no solo la tecnología de objetos ha demostrado hasta el momento una pequeña parte de sus potencialidades, sino que por primera vez un desarrollo de *software* ha motivado el creciente interés de los fabricantes de equipo y programas, por construir dispositivos electrónicos, sistemas operativos, lenguajes de programación y muchas otras aplicaciones que lo soporten.

Los ambientes cliente/servidor han llevado el concepto de orientación objetos al nivel del procesamiento distribuido, confiriéndole un significado ya de por sí evidente en su naturaleza y constitución. Y es que si quisiéramos buscar el origen mismo de la filosofía cliente/servidor, probablemente la hallaríamos en el campo de los objetos, debido a que su comportamiento es, en su forma más pura, una relación cooperativa de clientes y servidores.

La utilidad real de los objetos distribuidos radica en la forma en que los sistemas cliente/servidor deben ser desarrollados. La nueva visión concibe un sistema como un conjunto de componentes reutilizables (estructuras, plataformas o *frameworks*) que son ensamblados y extendidos para incorporar los requerimientos particulares de los usuarios. Un componente puede ser modificado sin alterar a los demás componentes ni la forma en que éstos se relacionan. Esto permitirá obtener sistemas menos propensos a errores, más fáciles de mantener y con tiempos de desarrollo significativamente bajos.

Anticipándose a la inevitable explosión de objetos en todos los campos de la tecnología computacional, a principios de 1990 fue creado en Estados Unidos el Grupo de Administración de Objetos (OMG - *Object Management Group*) como una asociación internacional sin fines de lucro, con la responsabilidad por reducir la complejidad, bajar los costos y acelerar la introducción de nuevas aplicaciones orientadas a objetos. Para cumplir sus propósitos, el OMG ha definido una arquitectura estándar denominada CORBA[6] (*Common Object Request Broker Architecture*) que especifica los servicios, facilidades y aplicaciones que deben cumplir los desarrollos orientados a objetos para poder operar entre sí.

[6] Un resumen de esta arquitectura puede encontrarse en [Orfali 1995]. Para un mayor detalle, léanse [OMG 1992] y [OMG 1993].



Estructura del CORBA
Figura 9

En esta arquitectura, los clientes solicitan servicios a otros objetos o al ORB por medio de llamadas a través del mismo ORB. Este localiza el código que implementa la función requerida, ensambla los parámetros y transfiere el control a la interfaz de implantación de objetos, que contiene la definición de los datos de los objetos y los métodos o servicios que estos proporcionan. Para satisfacer una solicitud, la interfaz de implantación de los objetos puede requerir de los servicios del ORB, con el que se comunica utilizando el Adaptador de Objetos.

La arquitectura CORBA está compuesta por cuatro elementos principales:

- Un Agente coordinador de las solicitudes de los objetos (*Object Request Broker* – ORB o *Corredor de Objetos*). Representa el bus de interconexión por medio del cual los objetos clientes solicitan servicios a los objetos servidores. Estas solicitudes pueden ser estáticas (compiladas previamente en las aplicaciones y establecidas con un lenguaje de definición de interfaces- IDL) o dinámicas a través del uso de API's (*Application User Interfaces*) de invocación. Los clientes también pueden interactuar directamente con el ORB para demandar funciones específicas.

- Servicios de Objetos. Proporcionan las operaciones básicas para el modelamiento lógico y el almacenamiento físico de los objetos, además de que definen las operaciones que las clases deberían implantar o heredar. Entre las operaciones propuestas se encuentran: nombramiento (directorios de nombres para la localización de componentes[7]), administración del ciclo de vida (para crear, copiar, mover y eliminar componentes), persistencia (interfaz simple para almacenar componentes en repositorios de información como bases de datos y archivos planos), notificación de eventos (solicitud y distribución de eventos entre componentes), control de la concurrencia (administra los bloqueos sobre transacciones o hilos), relaciones (para crear asociaciones o enlaces entre componentes, utilizados entre otras cosas para forzar

[7] Un componente es un conjunto de objetos distribuidos, empacados de forma binaria y asequibles a los clientes remotos por medio de procedimientos de invocación.

la integridad referencial) y externalización (para controlar el flujo de datos hacia y desde los objetos). Además de estos servicios básicos, existen cinco más extendidos, incorporados a mediados de 1995: consulta, autorización de uso, propiedades, seguridad y tiempo.

- Facilidades comunes. Conforman una colección de componentes que definen las reglas de asociación entre los objetos de aplicación. Algunas facilidades posibles podrían ser: catalogación y edición de clases y objetos, rutinas de impresión, reporte de errores y ayuda, interfaces con sistemas externos, funciones de correo electrónico, y otras más.

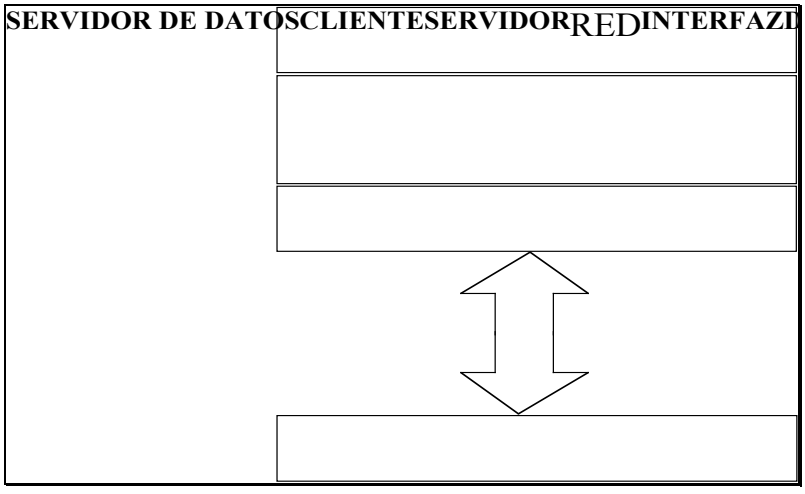
- Objetos de aplicación. Son componentes específicos de las aplicaciones de los usuarios que deben ser definidos con el Lenguaje de Definición de Interfaces (IDL).

Aparte del CORBA, otras arquitecturas han sido propuestas para estandarizar el área de los objetos distribuidos. Entre éstas se encuentran el Modelo de Objetos Comunes (COM - *Common Object Model*) de Microsoft con su Clases de Fundamentos Estándares (SFC - *Standard Foundation Classes*) y el Novell AppWare Distributed Bus.

2.3 Arquitecturas de Tres Niveles (*Three-Tiered Architectures*)

La primera generación de aplicaciones cliente/servidor estuvo fundamentada en la idea de que los clientes llevaban la principal carga de trabajo, incluyendo el manejo de la interfaz con el usuario y el código ejecutable de los programas. Unicamente la administración de los datos era potestad de los servidores.

Uno de los más grandes problemas presentes en este enfoque es el control de las versiones de los programas, así como la administración de la configuración de estos. Es muy costoso y complejo estar constantemente actualizando módulos de aplicaciones, más aún si éstos involucran actividades de red o se ejecutan en diferentes plataformas de equipos y sistemas operativos.

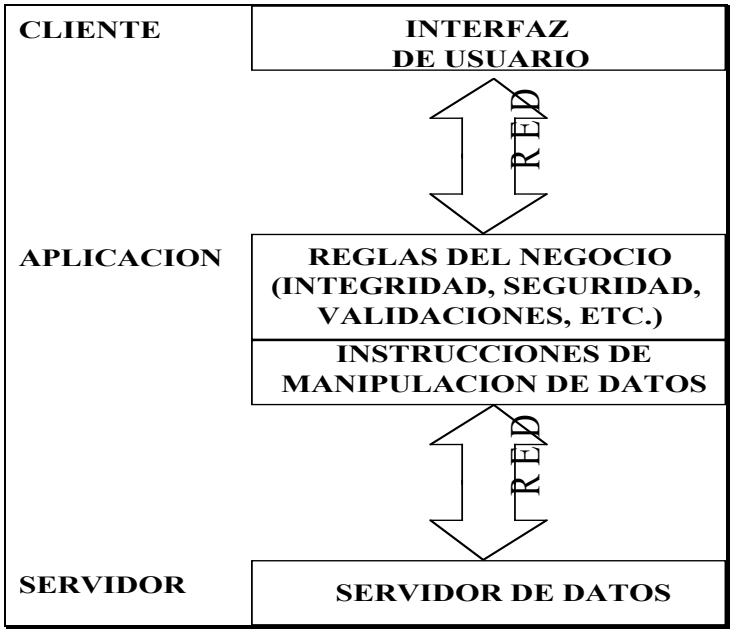


Modelo Tradicional Cliente/Servidor (2 Niveles)

Figura 10

Una forma de resolver estos inconvenientes es identificar las áreas más susceptibles a los cambios en las aplicaciones automatizadas. Es un hecho que tales cambios ocurren con poca frecuencia en la interfaz con el usuario y en el modelo de datos subyacente, no así en las reglas del negocio y en la forma en que se implementan los procesos de la organización. Esta apreciación nos conduce a un nuevo modelo cliente/servidor donde se identifican tres niveles básicos:

- El nivel de presentación o interfaz con el usuario
- El nivel de aplicación donde se codifican los procesos de la empresa y
- El nivel de datos



Modelo de Tres Niveles
Figura 11

Al esquema anterior se le conoce como Modelo de Tres Niveles (*Three-Tiered Model*) o configuración cliente/servidor de segunda generación. Esencialmente, el modelo separa, en módulos diferentes, las actividades de interacción con el usuario y la forma de trabajar de la organización. La ventaja principal de introducir un nivel de aplicación entre la interfaz del cliente y los servicios de datos del servidor es minimizar el costo y la complejidad del mantenimiento del *software*, especialmente si consideramos que la modificación del código de la aplicación es centralizada y por lo tanto, la cantidad de elementos modificados es muy reducida.

Otro beneficio derivado del modelo es el rendimiento incremental que puede obtenerse al reducir el tráfico de mensajes del cliente al servidor y viceversa. Una transacción, que como la de retirar dinero de un cajero automático produce un intercambio importante de mensajes entre el cliente y el servidor, puede limitarse a una única solicitud al servidor, a través de la interfaz con el usuario. Y como es usual que se utilice el mismo *hardware* para implementar los niveles de aplicación y datos, el ahorro en la transmisión de mensajes es verdaderamente significativo.

Una tercera ventaja del esquema de tres niveles tiene que ver con la cantidad de conexiones que se producen en una red, entre los clientes y los servidores, cuando se intenta hacerla crecer. Con el modelo tradicional de dos niveles, la cantidad de conexiones posibles entre "N" clientes y "M" servidores puede llegar a ser de "N x M", y su crecimiento es geométrico. Con la existencia de tres niveles, esta cantidad de conexiones se reduce a "N + M", con un crecimiento aritmético.

En este caso es obvio el ahorro potencial en cuanto a costos, complejidad y esfuerzo para llevar a cabo actualizaciones de *software*, el control de versiones de programas o simplemente incrementar el tamaño de una red.
Finalmente, el modelo de tres niveles ayuda a mejorar la disponibilidad, confiabilidad y administración de una red, al permitir que los servidores de aplicaciones sean replicados con el objetivo de aumentar el rendimiento global del sistema o evitar que éste quede inutilizado ante la ocurrencia de una falla.

Más adelante veremos cómo con la llegada de Internet muchos de los conceptos han evolucionado y obligado a pensar en un nuevo paradigma de desarrollo de aplicaciones cliente/servidor. De hecho en este capítulo de dos niveles versus tres niveles, Internet introduce el concepto de múltiples niveles. Es posible dividir funciones tanto en el lado del cliente como en el lado del servidor, agregando servidores de aplicación que manipulen la lógica de la aplicación y la conectividad en la interfaz del usuario, incorporando servidores especializados que realicen alguna función especial, o que ofrezcan lógica transaccional adicional. De alguna manera se agregará *inteligencia* al *browser* del lado del usuario y al servidor del Web.

Conclusiones - parte 1

En esta primera parte hemos visto las bases tecnológicas del modelo; preparamos el camino con definiciones, características, nomenclaturas, requisitos técnico/humanos, ventajas y desventajas de la arquitectura. Se ha hecho énfasis en la faceta transaccional del modelo y las capas que lo componen con sus diferentes variantes. Notamos como uno de los grandes éxitos del modelo es precisamente el hecho de ser modular, esto permite el desarrollo de esquemas flexibles y no tan monolíticos y de difícil dinamismo como sucede con los sistemas desarrollados en tecnologías de *mainframes*.

De una forma técnica pero sin ahondar en mucho detalle introdujimos al lector en un paradigma, que si bien no es complejo, sí requiere de una adecuada preparación y conocimiento antes de dar el paso para adoptarlo. A pesar de que se insiste lo suficiente en lo que representa el concepto, no se debe nunca de perder de vista que siempre intervendrán en gran medida componentes de software y hardware para los cuales existen proveedores que buscan realizar su trabajo. Existen numerosos productos para el cliente y para el servidor -tanto en hardware como en software- y de redes propiamente, que nos facilitan el trabajo enormemente. Sin embargo a la hora de seleccionar se debe de tomar cuidadosamente la decisión, o en su defecto asesorarse muy bien, porque no es solo la parte tecnológica la que intervendrá, influirá bastante la parte organizacional y por consiguiente humana que deberá adoptar la tecnología; aquí participan tanto los desarrolladores como los usuarios finales del sistema. De hecho uno de los grandes beneficiados con el modelo es precisamente el usuario, dado que al tener de su lado (el cliente) tecnología más afín a su uso diario, le permite integrarse más facilmente a los sistemas y a su vez estos, estar más cerca de lo que el usuario necesita.

Los productos y sistemas deben obligatoriamente ser flexibles y abiertos para adoptar tecnologías que irrumpen el mercado y que están relacionadas con el modelo cliente/servidor. En la segunda parte profundizaremos en estos conceptos como por ejemplo: *groupware, workflow*, EDI y correo electrónico, entre otros, así como algunos más novedosos: los almacenes de datos y el impacto de las intranets.

II parte

Aplicaciones y herramientas cliente/servidor

Resumen Ejecutivo

En la primera parte analizamos el área técnica y conceptual de la tecnología cliente/servidor. En esta segunda parte profundizaremos en algunas tecnologías que incorporan el modelo o que lo utilizan como forma de operación. Algunas de estas tecnologías incluso aparecieron antes del éxito del modelo cliente/servidor y posteriormente lo adoptaron para potenciar su éxito.

Las aplicaciones pueden ser tanto conceptos específicos y modalidades de trabajo como pueden ser también herramientas que se utilizan en el día a día de todas las organizaciones. Con esto pretendemos aclarar que cliente/servidor no son solo transacciones que se pasan de equipo a equipo sobre un medio, ni tampoco son solo capas complejas abstractas que requieren de alta tecnología. El modelo que encierra cliente/servidor ha evolucionado de tal forma que los conceptos como el *groupware*, el *workflow* o el intercambio electrónico de datos lo aplican exitosamente. De hecho hemos tomado algunos de estos conceptos para explicar su funcionalidad dentro del paradigma del cliente/servidor.

Por último, cerramos esta segunda parte con una breve reseña de algunas herramientas, específicamente los sistemas operativos, los sistemas administradores de bases de datos y las aplicaciones basadas en objetos, todo desde la óptica y funcionalidad de la tecnología cliente/servidor.

3. Groupware

Groupware o "Computación de Trabajo en Grupo" es el *software* que facilita el trabajo realizado por un grupo de personas. Este término ha sido usado para referirse a programas que van desde el correo electrónico (*E-Mail*) que permite enviar mensajes a los usuarios a través de las facilidades de una red de comunicación, hasta la automatización de flujos de trabajo o procedimientos repetitivos de una oficina (*Workflow Automation*), pasando por el Intercambio Electrónico de Datos (EDI - *Electronic Data Interchange*) que combina la mensajería electrónica con el procesamiento de transacciones. El *Groupware* es un caso específico de procesamiento cooperativo, donde la información y su manejo son compartidos entre los usuarios participantes, conectados a través de una red de comunicación. En este entorno, el *Groupware* actúa como el coordinador de las actividades del grupo de usuarios interconectados.

Algunos productos de *Groupware* son diseñados para que, una vez adquiridos, las empresas puedan adaptarlos de acuerdo con sus necesidades específicas de información. La mayoría de ellos incorporan capacidades de enrutamiento de mensajes, la ejecución de acciones basadas en la ocurrencia de eventos y un seudo-lenguaje de programación (*Scripting Language*) mediante el cual se indica a las aplicaciones, por parte del usuario o de un programador, que deben efectuar ciertas instrucciones relacionadas con procesos de flujos de trabajos.

La combinación de varias de las tecnologías anteriormente mencionadas, ha dado paso a la creación de términos que algunas veces son tomados como sinónimos pero que en la mayoría de los casos guardan importantes diferencias. Los términos más empleados son: "Computación de Flujos de Trabajo" (*Workflow Computing*), *Groupware*, "Computación Colaborativa" (*Collaborative Computing*), "Flujos de Trabajo" (*Workflow*) y "Correo Electrónico Inteligente" (*Intelligent E-Mail*).

Por su parte, las universidades y otros centros de investigación han efectuado estudios sobre la computación colaborativa bajo el nombre de "Trabajo Cooperativo Soportado por Computador" (CSCW - *Computer-Supported Cooperative Work*), que incluyó en sus inicios el Correo Electrónico, las Teleconferencias, y más recientemente, herramientas de procesamiento distribuido y automatización de tareas separadas en el tiempo. Tales herramientas son, en estos momentos, parte importante de procesos de re-ingeniería en las empresas, cuyo objetivo es volver a diseñar actividades que ya no son eficientes, o que surgieron sin diseño. Sin embargo, es esencial considerar algunas limitaciones que presentan los productos de *Groupware*:

a) Ninguna herramienta comprende todas las actividades de una empresa susceptibles de ser apoyadas informáticamente.

b) Los productos son de aparición reciente, por lo que aún no se encuentran verdaderamente "maduros", es decir, puede ser común encontrar que faltan, en dichos productos, características consideradas obvias.

c) Las personas tienden por lo general a trabajar individualmente, por lo que una transición hacia el trabajo en grupo puede ser difícil de llevar a la práctica.

d) Un mal uso del *Groupware* puede generar una explosión inconveniente de información internamente en una empresa, incluso exponer datos y procedimientos a personas que no debieran tener acceso a ellos.

En la actualidad, existe un buen número de productos de *Groupware* disponibles en el mercado, pero el primero de ellos en adquirir relevancia fue **Notes** de Lotus Corporation, el cual incorpora diferentes tipos de aplicaciones, permite compartir información y está basado en el concepto de bases de datos. Otros productos similares incluyen: Word Perfect Office de WordPerfect Corporation, Cooperation de NCR y TeamLinks de Digital Equipment Corporation.

3.1 Workflow

Los productos de *Workflow* ayudan a simplificar y automatizar los procesos de trabajo que involucran el manejo de documentos, por lo general, entre diversos usuarios. Estos procesos se caracterizan, entre otras cosas, por estar dispersos físicamente, requerir más de una persona para su conclusión y representar información crítica que debe ser manipulada oportunamente.

[Dyson 1990] clasifica los productos de *Workflow* en tres arquitecturas primarias:

a) Arquitectura basada en clientes o usuarios: Automatiza las interacciones de un usuario con otros clientes o con aplicaciones.

b) Arquitectura basada en objetos o agentes: La información contenida en documentos se incorpora a "formas" electrónicas y es visualizada como objetos activos que realizan acciones específicas, como por ejemplo, la consulta a una base de datos. Estas acciones se definen e implementan en el diseño de las formas mismas.

c) Arquitectura basada en servidores: Las herramientas de esta categoría le permiten a un administrador de flujos de trabajo hacer un seguimiento de las tareas que ocurren en el sistema, facilitando sus labores de control y análisis histórico y estadístico.

Por su parte, [Marshak 1993] editor del *Workgroup Computing Report*, definió en 1993 un modelo del *Workflow* constituido por tres R's:

a) Rutas: Son los caminos que deben recorrer los objetos (documentos, mensajes, eventos, etc.)

b) Reglas: Define cómo va a ser trasladada la información y a quién se enviará.

c) Roles: Definen la posición de los usuarios involucrados y las funciones de las tareas que realizan.

3.2 Intercambio Electrónico de Datos (EDI)[8]

El EDI puede definirse como un conjunto de procesos automatizados que facilitan el intercambio y la manipulación de datos entre computadoras pertenecientes a compañías que mantienen una relación de negocios: uno de cuyos principales propósitos es reducir el retraso producido por el manejo de papelería y servicios de entrega como el correo. Entre las aplicaciones de EDI más comunes podemos encontrar, el intercambio de datos de negocios (como por ejemplo, órdenes de compra y facturas) y la transferencia electrónica de fondos.

Entre las ventajas más relevantes del EDI se encuentran los ahorros de costos al reducir la papelería involucrada, procedimientos más ágiles (sustitución de actividades manuales por automatizadas), reducción de errores en la digitación de información (se lleva a cabo una sola vez en el EDI) y la creación de nuevas oportunidades de negocios al eliminar barreras geográficas y permitir una integración global a gran escala.

Las desventajas más notorias del EDI se encuentran en la inversión inicial que debe realizarse para lograr su implementación, y el alto nivel de compatibilidad que debe existir (tanto en *hardware* como en *software*) entre las empresas que deseen interconectarse.

3.3 Administración de Documentos Multimedios (Imágenes Electrónicas)

La historia de la tecnología de documentos multimedios se remonta a la década de 1960 donde la gran cantidad de papeles en una empresa empezó paulatinamente a ser sustituida por microfilms y sistemas informáticos de recuperación de datos (*Text Retrieval*). Veinte años después y hasta la fecha, las alternativas dominantes están representadas por microcomputadoras, *scanners*, dispositivos de almacenamiento óptico, y otros similares que conforman opciones eficientes y económicas.

Los sistemas electrónicos de documentos son aplicaciones cliente/servidor orientadas a bases de datos. Los clientes actúan como *Front-End's* para obtener información digitalizada (texto e imágenes) que será almacenada por los servidores. Las bases de datos que utilizan éstos pueden ser del tipo relacional (las más frecuentes), hipermedios, orientadas a objetos o inteligentes, y tienen la responsabilidad de organizar y clasificar la información, transferirla a las aplicaciones de acuerdo con reglas definidas por los usuarios, y protegerla de accesos no autorizados.

[8] Para obtener información más detallada de este tema, puede leerse el informe del Club, "Intercambio Electrónico de Datos" por Carlos Gallegos y Luis Calderón, 1995.

3.4 Programación y Calendarización

Las aplicaciones de programación y calendarización son un claro ejemplo de la categoría de *Groupware*, debido a que sus principales componentes como la programación de reuniones, los calendarios compartidos y las listas de "Qué Hacer", corresponden a actividades cimentadas en el procesamiento colaborativo. Los clientes, mediante interfaces apropiadas, se ponen de acuerdo en ejecutar labores de grupo (por ejemplo, reuniones) y compartir recursos (como salas de exposición). Por su parte, los servidores ejecutan tareas de fondo en forma permanente y verifican la ocurrencia de eventos que controlan y ponen en funcionamiento las actividades de los clientes.

3.5 Conferencias

Las conferencias o reuniones electrónicas son aplicaciones de *Groupware* relativamente recientes, que manipulan datos e imágenes. En las conferencias "en línea" es posible que varias personas, en forma simultánea, compartan y modifiquen un mismo archivo examinando cada usuario lo que los demás están realizando, o bien conversen entre sí por medio de un micrófono mientras se observan unos a otros a través de cámaras de video instaladas en sus respectivas computadoras. Asimismo, la interacción entre los clientes puede surgir por conferencias "en diferido" como es el caso de los boletines electrónicos (CompuServe e Internet entre otros), donde los usuarios pueden participar de discusiones en grupo sin estar sujetos a lugares y tiempos específicos.

3.6 Correo Electrónico

Un sistema de correo electrónico (*E-Mail System*) es una de las aplicaciones más comunes y antiguas en la filosofía computacional Cliente/Servidor, junto a otras también conocidas como los servidores de archivos e impresión.

Su propósito es enviar mensajes a otras computadoras mediante el uso de capacidades de red, y asume la misma función que tienen los memorándums, faxes, teléfonos y el servicio postal tradicional en las oficinas, pero brindando mayores ventajas que éstos. La arquitectura del correo electrónico consiste de dos componentes básicos:

- La interfaz con el usuario (*Front-End*)
- El servicio de entrega de los mensajes (*Back-End*)

Entre los dos componentes anteriores se encuentra el mecanismo de transporte que debe ser proveído por un sistema de red de comunicación.

Componentes de un Sistema de Correo Electrónico
Figura 12

La interfaz con el usuario es conocida como el Agente Usuario, mientras que el servicio de entrega se denomina Agente de Transporte. El correo electrónico se está modularizando cada vez más, por lo que en un futuro cercano es muy probable que ambos componentes sean adquiridos por separado por las empresas.

La interfaz da la capacidad al usuario de interactuar con el correo electrónico, y al igual que otras interfaces, debe ser consistente para los usuarios finales si pretende facilitar su uso. La red de comunicación soporta el mecanismo de transmisión de mensajes, que impone la restricción a los usuarios emisores y receptores de utilizar un mismo protocolo de transporte. El módulo servidor es el que brinda las funciones básicas de correo que comprenden:

- La base de datos o repositorio de mensajes
- El "*software*" de enrutamiento de mensajes
- Los servicios de direccionamiento

Los correos electrónicos modernos están siendo cada vez más incorporados dentro de otras aplicaciones. Tal es el caso de los sistemas operativos como Microsoft Windows 95 y Microsoft Windows NT, IBM OS/2, Novell Netware, UNIX y algunos otros.

3.6.1 Características relevantes

Muchas son las características que podríamos enumerar en un producto de correo electrónico, pero son algunas de ellas las que provocan el verdadero impacto cuando se utilizan en una organización o en un conjunto de ellas. En ocasiones, ciertos rasgos que muestran los paquetes son visualizados de manera diferente dependiendo del lugar, empresa o compañía que los utilice. Por ejemplo, una característica considerada como interesante en un lugar, puede ser imprescindible para otro.

Las opciones de un correo electrónico abarcan capacidades como las listas de distribución y correspondencia, la impresión de textos y copias, la organización y el almacenamiento de mensajes, el enrutamiento de formas, la seguridad de las transmisiones y la autenticación de los mensajes.

Otras características incluyen:

- La notificación audible de los mensajes recibidos
- La priorización de mensajes
- El manejo de los mensajes (recepción, lectura, retransmisión, etc.) por medio de reglas lógicas establecidas por el usuario

3.6.2 Ventajas y Desventajas

Como todas las aplicaciones informáticas, el correo electrónico tiene beneficios y debilidades.

Entre los primeros podemos encontrar:

a) La velocidad de entrega de los mensajes. A diferencia del correo postal tradicional que puede durar días, semanas o meses en ser entregado, el correo electrónico es casi inmediato.

b) Los mensajes pueden ser entregados y leídos a conveniencia del emisor o el receptor, sin importar el lugar en que se encuentren o la hora escogida.

c) Todo correo puede entregarse a una o múltiples personas simultáneamente. Para esto, se confeccionan **listas de distribución** que contienen el nombre de las personas a las cuales se quiere hacer llegar los mensajes. Es fácil incluir o eliminar nombres de estas listas.

d) Los mensajes pueden ser categorizados, permitiendo llevar un seguimiento de lo que se ha transmitido, por quién y cuándo. Si un usuario no recuerda las particularidades de un correo en especial, puede recuperarlo, volverlo a leer y retransmitirlo a otros usuarios agregándole más información si así lo desea.

e) Los usuarios no están restringidos a mantener una copia electrónica de los mensajes, ya que pueden imprimir éstos en cualquier momento.

Así como podemos determinar beneficios importantes para el correo electrónico, también es posible especificar algunas desventajas que su empleo puede acarrear. Entre ellas encontramos:

a) La sobreutilización puede conducir al abuso. Algunos usuarios pueden acostumbrarse a transmitir mensajes de manera excesiva, reduciendo su tiempo efectivo de trabajo y el de las personas con quienes se comunican. En este caso, es común encontrar que un usuario que retorna al trabajo después de varios días de ausencia, encuentre decenas de mensajes en su directorio personal. Para minimizar este problema, algunos productos ofrecen características de filtración de la información que permiten seleccionar ésta basada en parámetros como la prioridad, el emisor, la fecha de recepción, el tamaño del mensaje y otros más. Por ejemplo, el correo electrónico para Internet Z-Mail de la empresa Z-Code Software, utiliza filtros y un sistema basado en reglas (como los sistemas expertos) para manejar grandes volúmenes de mensajes.

b) En muchas situaciones, los mensajes hablados son más claros y se transmiten más rápidamente que los escritos. En este sentido, la sustitución del correo electrónico por el correo de voz puede conducir a mejores resultados.

c) El empleo extensivo del correo escrito dentro de una organización, puede conducir a la impersonalización de las relaciones internas, creando un clima frío y de poca motivación.

d) La adquisición de un correo electrónico involucra, por lo general, la compra de dos productos: la parte servidora y la función cliente, esta última con un valor que dependerá de la cantidad licencias (usuarios) que se deseen comunicar. Por lo tanto, el precio total de un paquete de esta naturaleza puede ser muy alto.

4. OLTP, OLCP y OLAP

El Procesamiento Transaccional En Línea (OLTP - *On-Line Transaction Processing*), también conocido como Procesamiento Operacional, es el modelo más común de *software* de aplicaciones que existe e incluye sistemas como los de manufactura, toma de pedidos, reservaciones aéreas, y otros. Se caracteriza porque las transacciones ejecutadas son generalmente simples, homogéneas y de corta duración, los datos son representaciones de hileras de números, caracteres y fechas, y son accesados por gran cantidad de usuarios concurrentes. Además, involucra operaciones normales de mantenimiento como inserciones, modificaciones y eliminación de información.

El Procesamiento Complejo En Línea (OLCP - *On-Line Complex Processing*) es también de tipo operacional pero difiere del OLTP en que la información manipulada es de carácter complejo como, por ejemplo, texto, gráficos, imágenes, audio y vídeo. También, las transacciones incluyen muchas líneas de código, son de media y larga duración y son pocos los usuarios que concurrentemente accesan los datos. La tecnología de objetos, mediante aplicaciones de control de flujo de tareas (*Workflow*), es la que ha dado sustento y reconocimiento a este modelo de procesamiento.

El Procesamiento Analítico En Línea (OLAP - *On-Line Analytical Processing*) está ganando rápidamente popularidad gracias a la tecnología emergente de los almacenes de datos. Este modelo trata en forma exclusiva con datos de solo lectura, y está orientado por la necesidad de las organizaciones de transformar sus datos en información. Por lo general, se accesan grandes cantidades de datos a través de procesos de consulta *ad-hoc*, sobre bases de datos multidimensionales con el propósito de servir a sistemas de apoyo a las decisiones. Si las bases de datos consultadas son del tipo relacional, este procesamiento se denomina ROLAP.

Los almacenes de datos (incluyendo los supermercados de datos) hacen un uso extensivo de los modelos de procesamiento descritos anteriormente. El OLTP y el OLCP definen las operaciones (estados) de una empresa en puntos específicos del tiempo. El OLAP por su parte, tiene la función de ayudar a interpretar integral y estratégicamente el significado de los datos operativos, suministrando herramientas de muy alto nivel que pueden categorizarse en tres niveles:

- Consulta y reporte de la información. Facilitan a los usuarios generar sus propios reportes y gráficos, mientras mantiene un control estricto sobre el acceso a la información.

- Análisis multidimensional. Facilitan el recorrer la información por niveles, del resumen al detalle y de éste a la consolidación, para identificar tendencias desde diferentes perspectivas.

- Exploración de datos. En las dos categorías anteriores, los usuarios conducen el proceso de búsqueda de información. Sin embargo, también es posible ejecutar procesos automatizados que descubran patrones ocultos en los datos. A esta técnica se le denomina exploración.

5. Almacenes de Datos (*Data Warehousing*)

De acuerdo con [Tash 1996] de Database Decisions, un almacén de datos es "un conjunto de representaciones temporales de la información de una empresa, obtenidas tanto de bases de datos de producción como de fuentes externas de información". Los datos están orientados a temas específicos (manufactura, contabilidad, etc.), no son volátiles (las fuentes de información son de lectura únicamente) y varían con el tiempo.

Dependiendo de la perspectiva que se utilice para analizar la información los almacenes de datos pueden dividirse en:

- Almacenes (*Warehouses*). Reflejan una estrategia de Arriba-Hacia-Abajo (*Top-Down*) de la empresa.

- Supermercados o Centros Comerciales (*Marts*). Reflejan un enfoque de Abajo-Hacia-Arriba (*Bottom-Up*), enfocado a los niveles departamentales o de grupos de trabajo.

Uno de los beneficios directos de los almacenes y supermercados es su capacidad para proporcionar acceso a información precisa y consistente a partir de fuentes heterogéneas. Aquí, típicamente se tendrá un servidor o más dedicados a acceder y organizar los datos del almacén. El problema principal que enfrenta es la diversidad de formatos y representaciones en que se puede encontrar esta información.

6. Herramientas Cliente/Servidor

Aunque el ámbito de las tecnologías Cliente/Servidor es muy amplio, algunas áreas se han visto particularmente favorecidas con el desarrollo de programas informáticos, cuyo uso y conocimiento se ha extendido rápidamente entre los usuarios de aplicaciones, sean éstos técnicos o no. Tres de estas áreas son:

- **Sistemas Operativos**.

En la década de 1970, un proveedor de equipo, IBM, dominaba tanto el mercado del hardware como el de los sistemas operativos. Estos último se encontraban divididos en dos grandes grupos: UNIX y sus diferentes versiones, y los sistemas operativos propietarios de empresas competidoras de IBM como Burroughs, Univac, Honeywell y Control Data. Con la introducción al mercado computacional de las tecnologías de minicomputadoras y microcomputadoras, el panorama empezó a cambiar radicalmente. A mediados de los años 80, existían cuatro contendientes principales en el campo de los sistemas operativos: DOS (Microsoft), OS/2 (IBM), UNIX y Apple. Hoy en día, la popularidad de que gozan las interfaces gráficas, ha orientado el nuevo rumbo de los sistemas operativos tanto actuales como futuros. Por el lado de empresas como Microsoft, resaltan el Windows 95 y el Windows NT; el Apple System 7 y 8 de Apple Computer Inc., el Novell Netware de Novell, contendiente del Windows NT como sistema operativo de red; y las versiones de UNIX de compañías como NeXT, Sun y Santa Cruz Operation.

- **Sistemas Administradores de Bases de Datos**

Un componente esencial de un elemento servidor, es indudablemente el que corresponde a los Sistemas Administradores de Bases de Datos (SABD's). El modelo relacional es el predominante en este campo y muchos son los productos que se disputan el mercado internacional: Oracle, Sybase, Informix On Line, CA/Ingres, GUPTA SQLBase, Microsoft SQLServer e IBM DB2/6000. La mayoría de estos productos son funcionalmente más robustos que sus similares de ambientes multiusuario, con una ventaja adicional: sus capacidades han sido incrementadas con el concepto de "Servidor Inteligente de Bases de Datos", mediante la incorporación de reglas del negocio, disparos automáticos *triggers* y procedimientos almacenados *stored procedures* que permiten que la programación de la integridad de la información y el mantenimiento de la base de datos se realice en el servidor de la base de datos y no en los programas cliente, ajustándose de esta forma a un modelo cliente/servidor más natural. Estas capacidades se ven reforzadas con otras como la *replicación* (actualización local de los datos, dejando al SABD la tarea de distribuir dicha actualización a otras bases de datos en distintas localizaciones separadas geográficamente) y el *multiprocesamiento simétrico* (utilización de arquitecturas de procesamiento paralelo).

Oracle es una de las bases de datos más vendidas mundialmente, y se encuentra disponible en prácticamente cualquier plataforma de *hardware* y sistema operativo del mercado. Entre sus características principales sobresalen su alta portabilidad para *hardware* y *software*, un amplio soporte de protocolos de comunicación, capacidades de procesamiento distribuido y paralelo y

un diccionario de datos activo que contiene un conjunto de tablas de solo lectura con información acerca de la base de datos y sus componentes (tablas, vistas, índices, sinónimos, etc.).

Sybase es una base de datos relacional que permite integrar aplicaciones y datos heterogéneos. Algunas fortalezas importantes son su arquitectura de multihilos de 32 bits, utilitarios de respaldo y recuperación de datos dinámicos (operación continua), soporte de tipos de datos definidos por el usuario, operaciones de entrada/salida asincrónicas y bloqueos multinivel.

El Informix On-Line Dynamic Server está basado en la Arquitectura Dinámica Escalable (DSA – *Dynamic Scalable Architecture*) de Informix y se caracteriza por sus capacidades distribuidas, el soporte del multiprocesamiento simétrico y mutiprotocolos de red, el procesamiento paralelo de consultas y la administración de grandes bases de datos.

El CA/Ingres de Computer Associates proporciona una tecnología avanzada de bases de datos relacionales para el soporte de sistemas OLTP (*On-Line Transaction Processing*) y de apoyo a la toma de decisiones. Consiste de tres componentes: el servidor de bases de datos, un utilitario para la administración del conocimiento, y programas para la administración de objetos. Entre sus propiedades distintivas destacan el bloqueo multinivel, el soporte de bases de datos muy grandes, el respaldo y la recuperación de información en línea y la capacidad de procesamiento paralelo de consultas.

El Gupta SQLBase (hoy Centura SQLBase), es considerada una base de datos líder en el campo cliente/servidor. Implementa la integridad referencial declarativa y el concepto de cursores de información "hacia delante" y "hacia atrás". También, incluye características como la compresión de datos entre el cliente y el servidor, la optimización de consultas basadas en costos, el respaldo y la recuperación de datos en línea, y el manejo de datos multimedia de forma similar a los demás tipos de datos (enteros, caracteres, fechas, etc.).

El Microsoft SQL Server, posee el mismo "motor" de bases de datos que Sybase, debido a un acuerdo firmado por ambas compañías para mercadearlo en diferentes plataformas de *hardware* y sistemas operativos. Mientras que Sybase tiene en UNIX y Novell Netware sus principales versiones (Sybase System 10 y 11), Microsoft SQL Server se sobresale por su desempeño en los sistemas operativos Windows NT y OS/2.

DB2 de IBM, tuvo sus orígenes en el ambiente de mini y supercomputadoras como una base de datos propietaria. La versión más reciente, DB2/6000 para la plataforma RISC 6000 y el sistema operativo AIX, incorpora tecnología avanzada que cubre los campos de las arquitecturas cliente/servidor, computación distribuida, sistemas abiertos y redes telemáticas. Esta base de datos soporta tipos de datos y funciones definidas por los usuarios, el particionamiento de bases de datos, la programación visual y orientada a objetos, el almacenamiento y la administración de información multimedia y posee un conjunto de programas utilitarios basados en el concepto de *Data Warehousing*.

Aunque la tendencia de los últimos años ha estado orientada hacia los SABD's relacionales, algunas nuevas arquitecturas de bases de datos están surgiendo con fuerza. Entre éstas

encontramos SABD's: Orientados a Objetos, para la Recuperación y Manipulación de Texto, basados en la Inteligencia Artificial y Multimedios. Los caminos que están siguiendo algunas áreas de la computación (interfaces gráficas, sistemas operativos, etc.) hacia el futuro cercano, hacen suponer que estas categorías de bases de datos, van a ir paulatinamente adquiriendo mayor relevancia.

- **Aplicaciones basadas en objetos**

En el lado del cliente, la década de los años 90 ha estado orientando el desarrollo de herramientas informáticas basadas primero en el concepto de *objetos* y posteriormente en el de *componentes* (colecciones de objetos de muy alto nivel que realizan tareas generales o específicas).

En el terreno de los lenguajes de programación se encuentran productos como el Visual Basic, Visual Fox y Visual C++ de Microsoft, PowerBuilder de Sybase, Centura (anteriormente GUPTA) de Centura Corporation, Paradox SQL Link y Delphi de Borland, DataEase SQL de DataEase International, EDA/SQL de Information Builder, y Forté, de Forté Software Inc. La mayoría de estos productos incluyen además de un lenguaje de programación orientado a eventos y objetos, herramientas utilitarias como generadores de reportes, consultas, y monitoreo de transacciones en bases de datos[9].

Las herramientas CASE cubren un amplio espectro de áreas, entre las que destacan: el Modelamiento de Bases de Datos (como por ejemplo: S-Designor de Powersoft, y ERWin de Logic Works), el Modelamiento de Aplicaciones Orientadas a Objetos (Rational Rose de Rational Software Corporation y Andersen Consulting's Foundation), el Desarrollo del Ciclo de Vida de los Sistemas de Información (Teamwork de Cadre), la Administración de Configuraciones y Actualización de Versiones (Polytron Version Control System de Intersolv), y otras más.

El correo electrónico es hoy en día una de las aplicaciones cliente/servidor más utilizadas por los usuarios, debido esencialmente a la expansión tan acelerada en que se ha visto envuelta la red más grande del mundo: Internet. Los paquetes de correo electrónico pueden agruparse en tres grandes niveles:

- Los basados en redes de área local, como por ejemplo: cc:Mail de Lotus, Microsoft Mail, DaVinci Mail, BeyondMail y QuickMail (para microcomputadores Macintosh).

[9] En el último año hay una tendencia a proveer buenos ambientes de desarrollo para Java (Symantec, Borland, PowerSoft, Microsoft y Oracle Generator).

- Para mini y supercomputadoras: el *Professional Office System* (PROFS) y el *Distributed Office Support System* (DISOSS) de IBM, VMSmail y All-in-One de Digital Equipment Corporation, HPDeskmanager de Hewlett-Packard y *Comprehensive Electronic Office* (CEO) de Data General.

- Sistemas Públicos de Correo Electrónico. Esta categoría puede ser accesada desde diversas plataformas que incluyen micro, mini y supercomputadoras. Los usuarios se pueden conectar a estos sistemas a través de un modem o de una red de conexión por paquetes, y por lo general aunque no necesariamente, utilizan la red Internet como infraestructura de transmisión. Algunos ejemplos en esta clasificación son el EasyLink, el MCI Mail de MCI International, CompuServe de Compuserve Inc., y Eudora.

Los sistemas operativos conforman el fundamento para el desarrollo de las demás herramientas basadas en objetos y componentes. A partir de la infraestructura que proporcionan (bibliotecas de clases, métodos, eventos, interacciones, etc.), pueden construirse aplicaciones de la más variada naturaleza utilizando el concepto de línea de ensamble, es decir, escoger aquellos componentes que en forma integrada cumplan determinados servicios específicos y rodearlos de una cubierta (interfaz) fácilmente adaptable a las necesidades particulares de los usuarios. Es tan significativo este campo, que la mayoría de las empresas más importantes del mercado computacional han creado o se han unido para crear productos en esta categoría que se conviertan en los estándares de la industria. Ejemplo de lo anterior son: Microsoft OLE 2.0 Object Linking and Embedding, Component Object Model (COM) de Microsoft y Digital Equipment, Common Object Request Broker Architecture (CORBA) del Object Management Group, Opendoc de Apple Computer Inc., System Object Model (SOM) y Distributed System Object Model (DSOM) de IBM, OpenStep de Sun, NeXT y Hewlett-Packard y Taligent de Apple Computer Inc., IBM y Hewlett-Packard.

Conclusiones - parte 2

Es interesante notar cómo conceptos eminentemente comerciales hacen uso de la tecnología para conseguir sus fines cuales son mejorar la agilidad de las empresas. Cuando un usuario utiliza su correo, realiza computación de trabajo en grupo o automatiza sus procesos de trabajo, no tiene por qué preocuparse de la base tecnológica que apoya al producto que utiliza. Menos tiene que saber que su herramienta utiliza tecnología cliente/servidor. La tecnología simplemente le permite alcanzar sus fines u objetivos de trabajo. Eso es lo que demuestra la madurez de la tecnología: es confiable, es segura y ya nadie cuestiona su funcionalidad. Se da por un hecho que el producto tiene una base comprobada y sobre la cual existe una innumerable cantidad de productos respetables y que a su vez son bien conocidos por los profesionales del campo. Es decir no se navega sobre aguas desconocidas o que lleven a lugares difusos.

Cliente/servidor es una tecnología que permite la evolución de las aplicaciones y de los sistemas. Citamos en esta entrega el caso de los almacenes de datos. Esta es una tecnología que ha evolucionado y es, en este momento, en la madurez plena del cliente/servidor donde ha encontrado el nicho tecnológico adecuado para acentuarse en cualquiere de sus ramificaciones de almacenes o minería de datos, lo que a su vez le permitirá continuar su evolución sin tener que sacrificar el nivel alcanzado. Nos referimos a la nueva química entre los almacenes de datos y el Web.

III parte

La tecnología cliente/servidor en las empresas

Resumen ejecutivo - parte 3

Cerramos esta serie de tres artículos con una visión de la tecnología desde una perspectiva estrictamente empresarial, complementado con un pequeño matiz de su operación en el mundo de la tecnología de Internet.

En la informática moderna, ya no es solamente el Ingeniero de Sistemas quien decide la forma y estilo que tendrán los sistemas, el aporte del usuario final y del usuario gerencial es fundamental y tiene un sitio primario en la elaboración de las aplicaciones. La tecnología cliente/servidor aporta muchísimo a este concepto. Aquí veremos cómo los conceptos de downsizing, upsizing y rightsizing son una consecuencia inevitable de la necesidad que se tenía de acercar más los sistemas al usuario; a la vez permitieron en alguna medida a las empresas reducir los costos de mantenimiento de sistemas. Esta última frase es totalmente contundente, por ejemplo, no es lo mismo los contratos de mantenimiento preventivo/correctivo que se pagan por las grandes computadoras que la garantía que ofrecen los grandes servidores usados hoy día. También las herramientas de desarrollo permiten un desarrollo mucho más rápido con el consiguiente beneficio para la empresa.

La empresa debe planificar y debe adaptarse al cambio. En esta parte hablaremos sobre aspectos de reingeniería y planificación, considerando cómo estos conceptos son complementados por la tecnología cliente/servidor.

Por útimo, cerraremos este informe con algo que recientemente se ha amalgamado perfectamente con la tecnología cliente/servidor, nos referimos a la entrada en escena de la tecnología de Internet.

7. Downsizing, Upsizing y Rightsizing[10]

Downsizing es el proceso de migrar sistemas de información que se ejecutan en equipos grandes, hacia plataformas más pequeñas y menos costosas. Por lo general, aunque no necesariamente, este traslado ocurre desde supercomputadoras hacia minicomputadoras o redes de área local que se cimentan en microcomputadores. El llevar a cabo procesos de *downsizing* no implica que los sistemas se vayan a reducir o simplificar. Este término no tiene que ver con el tamaño de las aplicaciones, la cantidad de registros en una base de datos, el número de módulos de una aplicación o la cantidad de transacciones por unidad de tiempo que puedan producir los programas informáticos. Más bien a menudo estos aspectos permanecen invariables, con la excepción de que se utiliza *hardware* y *software* menos costosos para su implantación. El menor costo es casi sinónimo de menor tamaño, de ahí que el concepto acuñado fuera el de *downsizing* (por su significado en el idioma inglés).

Al ser uno de los objetivos principales del *downsizing* el incrementar la productividad en la utilización de los recursos de cómputo, permitiendo a los usuarios finales un acceso directo a los datos y los sistemas, se hace clara la relación entre este concepto y el de cliente/servidor. La introducción de las microcomputadoras a principios de los años 80, inició una revolución en el desarrollo de nuevos productos de *hardware* y *software* que hizo posible la aparición del *downsizing*. Estos mismos desarrollos, entre ellos, sistemas operativos de red, procesamiento distribuido, bases de datos relacionales, lenguajes de programación visual y otros más, conforman la base de los ambientes cliente/servidor. Con el paso de los años, se ha incrementado aceleradamente la producción de herramientas cada vez más poderosas y sofisticadas, que casi igualan e incluso superan en ocasiones a las disponibles en ambientes de mini y super computadoras.

Varias son las razones fundamentales para adoptar el *downsizing*:

- El ahorro de costos, al utilizarse equipos y herramientas menos sofisticados y más ampliamente difundidos para los cuales existe un mayor número de proveedores en el mercado. Sin embargo, este ahorro es relativo como se podrá apreciar en el siguiente apartado.

- Mejorar la productividad de los usuarios finales y del personal de desarrollo de sistemas informáticos con productos fáciles de usar y adaptar, y que permitan representar e implementar los procesos de la organización de formas más naturales.

[10] Este capítulo está basado en el libro de [Guengerich 1994]. Su lectura es altamente recomendada para Gerentes y Administradores de Sistemas de Información por su enfoque empresarial.

- Las empresas reducen su dependencia en cuanto a la programación de los sistemas de información, porque la cantidad de profesionales que pueden encontrarse en el mercado que dominen herramientas cliente/servidor es mayor, y porque el tiempo de aprendizaje de estos sistemas es por lo general menor que el requerido para aplicaciones similares en otros ambientes (por ejemplo, mini y super computadoras).

A pesar de que la filosofía de *downsizing* ha logrado una amplia aceptación, en particular por su estrecha vinculación con los ambientes cliente/servidor, existen costos asociados que las empresas rara vez alcanzan a identificar, antes de asumir un proceso de cambio tecnológico crítico y radical. Estos costos pueden resumirse a continuación en:

- Costos de *hardware*. Aunque los costos usuales de una infraestructura de mini o super computadoras son elevados, la configuración de las redes de área local que son el fundamento del *downsizing*, puede rápidamente llegar a igualar o incluso superar a aquella. Dispositivos de comunicación como tarjetas de red, concentradores, servidores de archivo, bases de datos e impresión, arreglos de discos para respaldos, cableado y algunos otros, además del equipo micro computacional de los usuarios finales con características como multimedia y capacidades de transmisión *fax-modem,* elevan significativamente el valor de la inversión inicial. También, es indispensable considerar los costos incrementales y recurrentes para mantener un apropiado rendimiento del entorno en general, la integridad de la información, la seguridad de los datos y la disponibilidad continua de la red, sin contar con factores adicionales como la integración con otros sistemas de cómputo fuera de la organización.

- Costos de *software*. El *software* es hoy en día, la porción más cara y compleja de administrar de un entorno computacional. Los conceptos de licencias, cantidad de usuarios concurrentes, cambio de versiones y otros por el estilo, son solo la "punta del *iceberg*" en ambientes cliente/servidor, que pocos se encuentran en configuraciones centralizadas grandes.

- Costos de capacitación. El trasladar mucho del procesamiento de la información a manos de los usuarios finales, incrementa su grado de responsabilidad e independencia, por lo que necesitan una adecuada capacitación que les ayude a asumir en forma conveniente sus nuevas actividades. También el personal de cómputo deberá estar involucrado en un plan de capacitación institucional, quizá el más complejo, debido al cambio de visión y forma de trabajo que implican la nueva tecnología de redes, los lenguajes de programación visual orientados a eventos, las herramientas CASE, las interfaces gráficas y muchos elementos adicionales que nacen con los ambientes cliente/servidor.

- Costos de personal técnico especializado. A menudo, el *downsizing* obliga a las empresas que lo adoptan a contratar nuevo personal con conocimientos especializados y experiencia acumulada en áreas como la administración de bases de datos, comunicaciones, sistemas operativos de redes, como único camino para implantar, integrar y dar soporte a la reciente tecnología.

Con el aumento en la cantidad de computadoras personales en las empresas, algunos

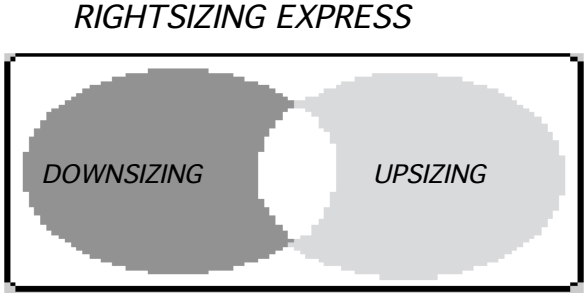
departamentos empezaron a gastar sus presupuestos en programas como hojas electrónicas y manejadores de archivos. Pronto los usuarios se dieron cuenta de la necesidad de compartir los datos y posteriormente, los sistemas y otros dispositivos como impresoras y unidades de almacenamiento. De esta forma, hicieron su aparición las redes de computadoras de área local. A este proceso de movilización de equipo independiente hacia plataformas de redes se le conoce como *upsizing*. A partir de aquí, los beneficios, desventajas y costos del *upsizing* se fusionan con los del *downsizing*.

El concepto de *rightsizing* es el de colocar las partes apropiadas de los sistemas de información en las plataformas de *hardware* y *software* correctas [Bochenski 1994], por lo que involucra un balance de los otros dos conceptos: *downsizing* y *rightsizing*. El término *rightsizing* es preferido al de *downsizing,* debido a que este último, ha tomado en los últimos años una connotación negativa dentro de las organizaciones, dado que se considera que la reducción de costos en materia tecnológica, aunado a un mayor nivel de automatización de los procesos internos implica una reducción en personal. En muchos casos esto ha sido así efectivamente, por la influencia de doctrinas como la globalización y las economías de escala. Sin embargo, no se ha podido hasta el momento demostrar que la siguiente fórmula es realmente válida:

Automatización = Aumento de la productividad = Disminución del recurso humano

Por el contrario, la automatización de tareas repetitivas presenta un terreno fértil para utilizar la capacidad humana en actividades más creativas y analíticas, creando nuevos procesos y permitiendo que otros que han funcionado en forma independiente, puedan ser integrados.

Un estudio publicado por la revista PC Week del 10 de Setiembre de 1990, volumen 7, número 36, muestra que de 200 compañías que adoptaron el *downsizing*, el 16% de las mismas redujo su personal de *staff* de sistemas, 14% lo aumentó y un 70% se mantuvo igual. La razón esencial de los resultados anteriores radica en que el *downsizing* de la empresa no es equivalente al *downsizing* de la tecnología.



***Downsizing*, *Upsizing* y *Rightsizing*: Tres caras de la misma moneda**
Figura 13

Existen casos en que el *rightsizing* no es una opción adecuada para una empresa, aunque éstos representan situaciones muy específicas. Tres de estos casos son:

- Aplicaciones con elevados niveles de seguridad. Es bien conocido que los ambientes centralizados disponen de mejores esquemas y herramientas para proteger la información que guardan. Un ambiente cliente/servidor, por naturaleza distribuido, es más propenso a presentar fallas de seguridad importantes. En la cadena de dispositivos de *hardware* y *software* involucrados, muchos son los puntos débiles que son atacados con frecuencia por "intrusos": cables y dispositivos de comunicación (escucha indiscreta, intervenciones radiales y telefónicas), servidores de archivos (virus), servidores de bases de datos (enmascaramiento[11]), y otros más.

- Aplicaciones actuales para las que no se justifica migrar hacia plataformas menores, por causas como rendimiento, estabilidad (poca ocurrencia de fallas) y satisfacción de las necesidades de información de los usuarios.

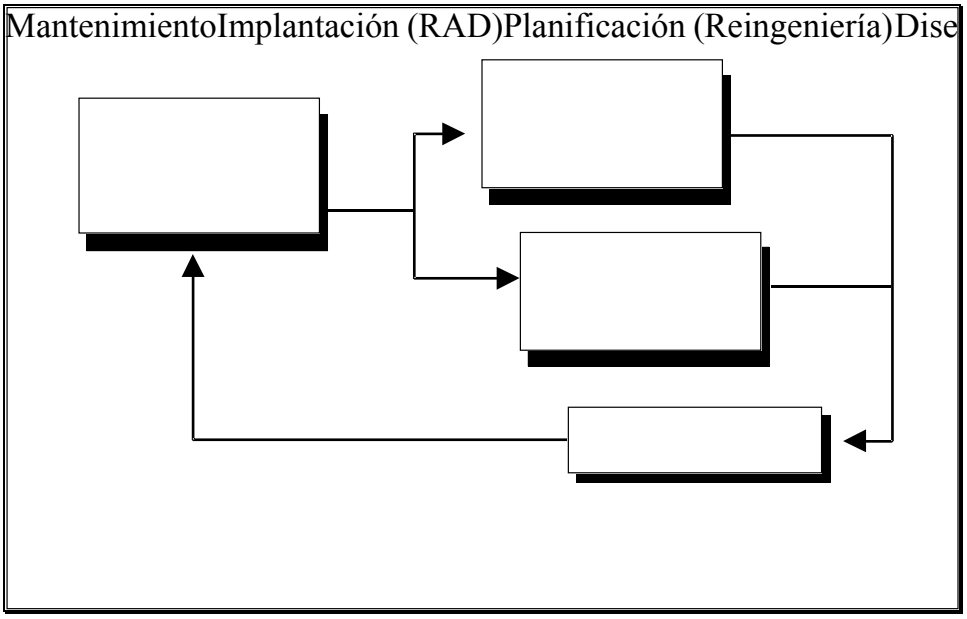
- Aplicaciones que poseen características que se adaptan mejor a ambientes de mini o super computadoras. Algunas de estas características incluyen respuestas en tiempo real, simulaciones complejas, etc.

Por el contrario, hay aplicaciones que por sus propiedades son firmes candidatas al proceso de *rightsizing*, en especial aquellas que involucran la captura y edición de datos, gráficos y soporte a las decisiones. Inclusive, el *rightsizing* deja abierta la posibilidad de que una aplicación se divida de tal forma que las diferentes partes sean implementadas en distintas plataformas (micro, mini o super computadoras).

7.1 Planificación

El proceso de *rightsizing* de los sistemas de información tiene como fundamento el ciclo de vida tradicional de un sistema, pero se ha extendido para adaptarse a los nuevos requerimientos técnicos y organizacionales de las empresas de hoy en día. En su aplicación, las diferentes fases identificadas pueden traslaparse y parte de los sistemas surgen en el transcurso mismo del proceso, es decir, no es razonable ni conveniente tratar de identificar desde un inicio todos los elementos que estarán involucrados en el ciclo de vida debido, principalmente al balance que se debe buscar (y probar) en las distintas plataformas.

[11] El enmascaramiento es la capacidad de un programa "intruso" de identificarse ante un sistema como un usuario o proceso válido para obtener información confidencial.



Metodología de desarrollo del *Rightsizing*
Figura 14

Una de las fases mostradas en el cuadro anterior, la planificación, incluye dos etapas básicas:

- Realización de un análisis costo/beneficio. Este análisis debe ser lo suficientemente detallado para que la Alta Administración tenga todos los elementos de juicio que le permitan determinar la factibilidad del proyecto. Entre las actividades conducentes al análisis se encuentran: la estimación de costos de operar el sistema actual y de operar el sistema propuesto, el costo de las fases subsiguientes, una descripción de los beneficios intangibles, y la identificación de los riesgos asociados con llevar o no a la práctica el proyecto (sistema de información).

  La estimación de costos debe contemplar el *hardware*, el *software* y el personal inicial, el desarrollo, la operación y el mantenimiento del sistema. Para los sistemas existentes, únicamente los costos de operación y mantenimiento pueden estar disponibles. Después de ejecutada la estimación, es preciso comparar estos costos en el tiempo, utilizando fórmulas como por ejemplo el valor presente neto o los flujos de caja futuros descontados.

  Los beneficios intangibles pueden incluir el mejoramiento de la imagen de la empresa al disponer de procesos más simples y eficientes que se reflejen en sus productos o servicios, una mayor motivación de los usuarios por estar más cerca de la tecnología y por ende ser más autosuficientes en sus procesos, y una flexibilidad creciente para responder en forma oportuna a los cambios futuros en los requerimientos de información de la empresa.

La identificación de los riesgos debe cubrir áreas como la financiera, la técnica, la sistémica (políticas y procedimientos), la de recursos humanos y cualquier otra que se considere crítica en la organización. Es indispensable, en estas áreas, determinar el impacto tanto interno como externo (competitividad) que tendrá el asumir el cambio o, en su defecto, retrasarlo o ignorarlo.

- Búsqueda de alternativas de financiamiento. La consecución de los fondos para la realización de un proyecto de *rightsizing* puede tener dos presentaciones: el ingreso directo de dinero proveniente del capital (presupuesto) de la empresa dueña del proyecto o de un préstamo de una entidad bancaria, o la concesión financiera de parte de las compañías que proveen la tecnología en esquemas como el *leasing* de equipo, la retribución de regalías a la empresa cuando se venden copias del sistema final a otras organizaciones, o la inversión de recursos en el proyecto mismo (alianza estratégica).

7.1.1 Reingeniería

La reingeniería es un conjunto de conceptos, métodos y técnicas para llevar acabo una nueva nueva forma de actuar y un rediseño radical de los procesos de una empresa, orientados al mejoramiento crítico de áreas como los costos, el control de calidad, el nivel de servicio, etc. En la reingeniería se examinan los procesos y estructuras de una organización y se buscan formas de mejorar el cumplimiento de los objetivos de esta, implantando cambios que faciliten el llegar paulatinamente a dicho mejoramiento. Es un examen objetivo de la verdadera esencia del trabajo realizado por las personas.

Tres son los elementos que caracterizan a la reingeniería:

- Cambios dramáticos. Se pretende al menos, alcanzar un 80% de mejoramiento en las áreas expuestas con anterioridad.

- Quebrantamiento de reglas. Representa acciones radicales cuyo propósito es replantear y reformar suposiciones y creencias institucionales.

- Orientación a procesos. El énfasis está centrado en los procesos exclusivamente.

Al igual que con el término *downsizing*, la reingeniería ha sido mal interpretada en muchas ocasiones (aún por compañías y profesionales dedicados a aplicarla), al considerársele como un mecanismo para la reducción automática de personal, cuando la experiencia indica que no necesariamente tiene que ser de esta forma.

A menudo las empresas no desean enfrentar una evaluación profunda de la manera en que se ejecutan sus procesos internos y, en lugar de revisar continuamente las actividades cotidianas, ponen en funcionamiento un conjunto de acciones que posponen pero que no eliminan los problemas presentes. Entre las más comunes se pueden encontrar el mejoramiento incremental en la calidad de los procesos, recortes indiscriminados de presupuestos y personal, y adquisición de nueva tecnología. Llegará el momento en que estos paliativos no serán suficientes y una organización deberá enfrentar el problema de cambiar radicalmente o sencillamente desaparecer.

Algunos signos dentro de las empresas son alarmas que indican con un alto grado de probabilidad, que se hace necesario un proceso de reingeniería. Los signos más significativos pueden ser:

- Amplia insatisfacción de los usuarios con los sistemas de información.

- Disminución de la posición competitiva de la empresa o decrecimiento de su participación en el mercado.

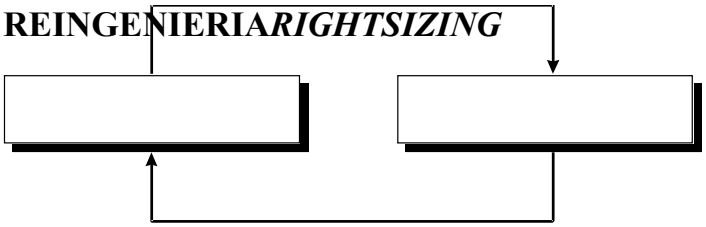
- Mala comunicación entre las dependencias internas que muestra un claro desconocimiento, en cada una de ellas, de lo que las demás hacen.

- Crecimiento rápido no esperado, que impide que los procesos actuales cumplan a cabalidad sus propósitos.

- Competencia. Otras compañías en el mismo segmento de mercado pueden estar tomando acciones que mejoren su presencia general.

Aunque el *rightsizing* de los sistemas de información puede ser el resultado esperado de un proceso de reingeniería, no siempre representa la respuesta en este sentido. Lo que sí es seguro es que la perspectiva de la reingeniería afecta significativamente la decisión de cómo ejecutar el *rightsizing*. La relación estrecha entre ambos términos sucede porque el *rightsizing* posee un enfoque de procesamiento de información radicalmente diferente de los que existen en los sistemas centralizados o poco integrados (computadoras personales o minicomputadoras con bajos niveles de integración, por la ausencia de verdaderas redes telemáticas), porque los usuarios adquieren una mayor responsabilidad por los sistemas que utilizan (lo que los obliga a conocer y evaluar de manera detallada las particularidades de sus procesos), y porque los principales beneficios del *rightsizing* solo pueden ser obtenidos cuando una organización efectúa cambios profundos en la forma de administrar y transformar la información, ya que no tiene sentido sustituir o emular ambientes de computación grandes con equipo y aplicaciones menos costosas con el único fin de reducir costos.

La reingeniería no concluye con el *rightsizing*. Más bien, ambos conceptos se alimentan mutuamente para garantizar un proceso evolutivo continuo dentro de las empresas. Esto quiere decir que ninguno de los dos es estático y beneficioso en un punto específico del tiempo. Sin un análisis, una evaluación y cambios frecuentes, su utilidad será nula, los costos asociados prohibitivos y los recursos obtenidos, desperdiciados.



Simbiosis de la Reingeniería y el *Rightsizing*
Figura 15

7.2 Diseño de sistemas: Diseño Conjunto de Aplicaciones (JAD - *Joint Application Design*)

Un proyecto exitoso de *rightsizing* debe incluir métodos y técnicas capaces de identificar las áreas de cambio en materia de sistemas de información, y producir los resultados esperados bajo las condiciones definidas. Para lograrlo, dos son los métodos más reconocidos de que dispone una organización en la actualidad: el **Diseño Conjunto de Aplicaciones (JAD)** y **el Desarrollo Rápido de Aplicaciones (RAD)**.

El JAD es un método de diseño que utiliza un alto nivel de acoplamiento organizacional y de trabajo en grupo para producir sistemas de información que sean útiles y aceptados por los usuarios finales. Los usuarios son los miembros más importantes del equipo de trabajo, que incluye también a personal de sistemas. Este equipo se reúne inicialmente para discutir, conceptualizar y analizar los procesos factibles y necesarios de automatizar, las consideraciones estratégicas, la disponibilidad de tiempo y recursos y la limitación de equipo existente, basados en su conocimiento de las actividades y propósitos de la organización. Esta forma de trabajo reduce los posibles problemas de comunicación entre las partes y garantiza que el producto final sea lo que realmente se requiere.

Debido a que el JAD asegura el trabajo en grupo y la cooperación, los usuarios y el personal de sistemas se comunican más efectivamente y enfrentan el objetivo común, de tener éxito en el proyecto. Los fundamentos del JAD son las técnicas conocidas de entrevistas, prototipos y análisis de documentos. La participación activa de los usuarios obliga a utilizar representaciones menos técnicas (como Diagramas de Flujo de Datos, Cartas estructuradas, etc.) y más generales mediante el uso de proyectores de video, pantallas de despliegue gráfico, y otras más. Además, las guías altamente estructuradas que implementa el JAD ayudan a producir resultados precisos. Todo lo discutido y aprobado debe quedar documentado en forma apropiada y ser distribuido a las partes involucradas para su análisis y recomendaciones pertinentes.

7.3 Desarrollo Rápido de Aplicaciones (RAD - *Rapid Application Development*)

A diferencia del JAD que se ubica principalmente en el nivel de diseño de procesos en el ciclo de vida de un sistema, el RAD es una técnica de implantación que tiene como propósito eliminar la realización de un diseño riguroso de un sistema y su revisión posterior, antes de proceder a programarlo. Un término más conocido para esta técnica es el de Prototipos, que consiste en desarrollar partes de un sistema de información (las mínimas necesarias) para ir evaluando su factibilidad y cumplimiento con los requerimientos de información establecidos. Es un desarrollo incremental en que se entregan prototipos totalmente funcionales. En el RAD el usuario también desempeña un papel muy importante que se ve facilitado por el uso de herramientas gráficas por medio de las cuales se mejora la comunicación y se reduce el tiempo total de desarrollo. El RAD brinda apoyo en cuatro áreas específicas:

- Evaluación de productos y tecnologías con las cuales la empresa no había tenido experiencia previa.

- El diseño de interfaces de usuario que sean fáciles de aprender y de usar.

- Definición iterativa de las especificaciones de los requerimientos de información, independientemente de la interfaz del usuario. En este caso se conduce al usuario a la definición de sus necesidades de información que justifique el costo del esfuerzo involucrado para desarrollar las aplicaciones.

- Evaluación del rendimiento. Este punto es crítico en organizaciones donde no se tiene la experiencia ni la información suficiente (volumen de datos, cantidad de usuarios, etc.) para predecir a priori, el comportamiento futuro de un sistema. Los prototipos ayudan a establecer rápidamente, situaciones y condiciones especiales que pueden afectar el rendimiento de las aplicaciones.

Las herramientas para el desarrollo de prototipos caen en varias categorías, algunas de las cuales son: herramientas CASE, Programación Visual y acceso a bases de datos.

8. Tecnologías cliente/servidor desde una perspectiva empresarial

Una idea muy arraigada es que cliente/servidor implica Sistemas de Información Ejecutivos, sistemas de recuperación de información o algún otro tipo de sistema en un computador personal que maneje solamente información de consulta. Estas son solo algunas de sus aplicaciones, pero en general, el concepto ha demostrado ser muy robusto en el área de sistemas altamente transaccionales.

Según [Martin 1994], en un estudio realizado en una variedad de compañías que han migrado sus aplicaciones al área del cliente/servidor, a la pregunta de "por qué la migración", se han encontrado una variedad de respuestas diferentes:

- Un desarrollo de sistemas más rápido
- Una mejor producción en el rendimiento de los sistemas
- Un desarrollo de sistemas más barato
- Menor entrenamiento para el desarrollador
- Menor costo de entrenamiento para el usuario final
- Herramientas más amigables para el programador
- Interfaces gráficas más amigables para el usuario
- Oportunidad de reutilización de código
- Oportunidad de reutilizar el *hardware*
- Incremento en la eficiencia
- Incremento en la flexibilidad
- Ventaja estratégica
- Moda

Esta diversidad de respuestas pueden ser afines (algunas de ellas) a ciertas empresas, sin embargo los objetivos obviamente pueden variar sustancialmente dependiendo del nivel en que se encuentre la madurez informática de la empresa.

Pueden existir muchas razones por las cuales una empresa tenga una estrategia para querer desarrollar esta tecnología. Nosotros pensamos que no importa cuál sea esa estrategia, cualquier objetivo debe estar enfocado en querer mejorar la calidad en el manejo de la información. Esta debe ser oportuna y obtenida de una manera que al usuario le resulte fácil lograrlo. La herramienta justa en el momento en que se requiera. La tecnología cliente/servidor facilita o propicia esta filosofía, se sabe que la mayoría de deficiencias que se le han encontrado a la arquitectura cliente/servidor, están localizadas en la administración, no en la tecnología en sí.

El proceso de transición hacia esta tecnología debe ser muy cuidadoso y realizado en forma paulatina. Si se está migrando desde plataformas monolíticas, como lo son la mayoría de las basadas en *supercomputadores*, se debe tener presente el impacto que el cambio puede tener en los desarrolladores, no se deben seleccionar aplicaciones críticas para la empresa ni intentar cambiar todos los sistemas de un solo golpe, insistimos que el cambio debe ser paulatino. De hecho se puede utilizar parte de la infraestructura existente, muchas de las herramientas de desarrollo que corren en el cliente pueden accesar las bases de datos residentes en los *supercomputadores*, esto permitirá tanto a desarrolladores como a usuarios finales "romper el hielo" con la nueva tecnología. El departamento de sistemas de información debe entender claramente que el cambio hacia esta nueva tecnología debe ir de la mano con un cambio en la calidad del servicio que se ofrece al usuario final. El concepto de computación orientada al usuario nunca debe ser perdida de vista. Los sistemas deben estar totalmente orientados hacia las exigencias del usuario y no el usuario adaptado al sistema.

Por otro lado cada vez con mayor razón se deben seleccionar herramientas que permitan al usuario participar en el proceso. Esto es, el usuario debe contar con las facilidades para desarrollar sus "propias" aplicaciones acorde con sus necesidades. Esta facilidad permitirá además reducir el *backlog* que presente el departamento de sistemas. Esta facilidad de poner la computación en manos del usuario, le permitirá a la empresa obtener una ventaja competitiva, dado que la información llegará cada vez con más prontitud a los sitios que requiera la empresa.

8.1 El Síndrome de las Grandes Promesas por Aplicaciones (SGPA)

El Síndrome de las Grandes Promesas por Aplicaciones representa la separación cada vez más extensa de lo que prometen los ambientes cliente/servidor y los resultados que verdaderamente se obtienen. Tanto los vendedores de tecnología como los profesionales del área de sistemas de información, se han encargado durante años de promocionar el término Cliente/Servidor como la solución a todos los problemas de manejo de datos.

En la década del 70 el énfasis estuvo en el acceso a la información en equipos centralizados. En los 80 fue el procesamiento distribuido para la toma de decisiones y en los 90, los sistemas cliente/servidor para ayudar a mejorar el servicio al cliente y elevar la productividad. Sin embargo, hasta el día de hoy las soluciones presentadas para soportar los niveles operativos y de toma de decisiones en este último campo, no han sido del todo satisfactorias. De esta forma, el contraste entre los beneficios esperados por los ambientes cliente/servidor y la realidad experimentada, se ha hecho cada vez más evidente, agravado por la diferencia en perspectiva de lo que los vendedores creen que los usuarios conocen del término y lo que éstos en realidad

dominan.

El desarrollo efectivo de aplicaciones cliente/servidor puede ser crítico para una empresa, por lo que es imprescindible reducir el síndrome GPA. De acuerdo a [Zeitz 1995] esto es posible lograrlo si se lleva a cabo lo siguiente:

- Lograr un acuerdo a nivel corporativo de cambiar la forma de hacer las cosas en la empresa
- Generar expectativas realistas de lo que las tecnologías cliente/servidor pueden hacer
- Involucrar a los altos ejecutivos en el proceso de cambio
- Seleccionar los recursos de administración y consultoría adecuados
- Ofrecer capacitación en todos los niveles
- Asignar y dedicar suficientes recursos
- Contratar expertos de reconocida trayectoria
- Los proveedores deberán contar con centros de soporte donde los clientes puedan asistir a demostraciones, realizar prototipos y capacitarse

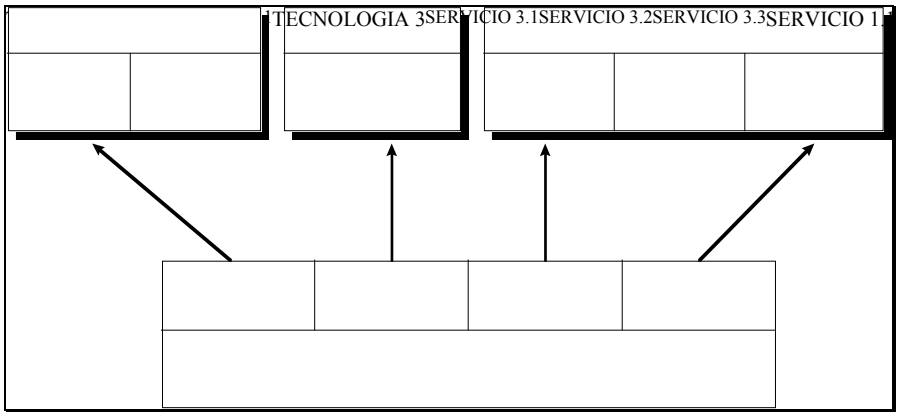
8.2 Integración de tecnologías

Modernos procesos como la globalización y la apertura de mercados han hecho que las compañías traten de utilizar al máximo sus recursos, sin importar dónde se encuentren localizados éstos. Lo anterior es especialmente cierto para mejorar la productividad en empresas que manejan estructuras planas y con un alto nivel de trabajo en grupo.

Muchos ejecutivos han diseñado planes para lograr que sus organizaciones se comuniquen y cooperen internamente así como, con sus clientes y proveedores con esquemas como la administración de documentos, el correo electrónico, las videoconferencias y el control de flujos de procesos. La tecnología subyacente que ha proporcionado el soporte para lograr esto ha sido la computación colaborativa. En palabras de [Adhikari 1995]: “Si las infraestructuras organizacionales, técnicas y de *software* son conformadas correctamente, pueden resultar en un intercambio efectivo y dinámico.”

En la actualidad, la computación colaborativa ha sido desarrollada alrededor de los ambientes cliente/servidor. Sin embargo, la presencia de algunos obstáculos ha impedido que la misma pueda propagarse en todos los estratos de una organización. Uno de estos obstáculos es que la tecnología es vertical, es decir, está estructurada para procesar información a partir de fuentes directas de datos (nivel operativo), integrar esta información por áreas funcionales (nivel táctico) y finalmente resumirla para apoyar la toma de decisiones (nivel estratégico). Esta verticalidad permite la comunicación entre diferentes grupos pero no una colaboración global entre ellos. Es

más, distintos grupos pueden implantar distintas tecnologías, complicando aún más la integración en la empresa. Una solución adoptada por algunas compañías ha sido la creación de núcleos de integración, consistentes de varias plataformas de *hardware* y *software* que brindan servicios comunes a los disponibles en las tecnologías existentes dentro de las empresas.



Núcleos de Integración
Figura 16

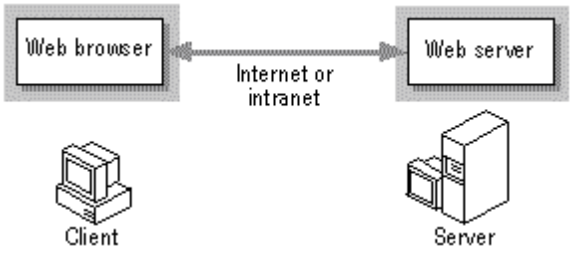
8.3 Internet/Intranet y arquitecturas Cliente/Servidor

La mayoría de las empresas u organizaciones, se enfrentan hoy en día al cuestionamiento de mejorar la calidad y alcance de sus sistemas para lograr un mejor posicionamiento en el mundo de los negocios. La tecnología cliente/servidor no escapa a estos cuestionamientos. El incremento de aplicaciones del tipo OLTP obliga a pensar en novedosas aplicaciones cliente/servidor, si bien no en nuevos tipos, si en aplicaciones que exploten al máximo los conceptos de la tecnología y que a la vez desarrollen el procesamiento de transacciones y que aporten a otros conceptos como lo son los almacenes de datos. A estas novedades se le unen conceptos de distribución de aplicaciones, LANs remotas o computadores móbiles que utilizan los conceptos de replicación, siempre bajo el amparo de aplicaciones desarrolladas en el paradigma de cliente/servidor.

En esta búsqueda por el mejoramiento, nos encontramos con autores que cuestionan el modelo y preven el ocaso de la tecnología cliente/servidor. Por ejemplo en el número de abril de la revista Information Week [Caldwell 1996] aparece un artículo titulado *Client-Server, Can It be Saved?*, el autor expone la tesis que el cliente/servidor es un modelo caro y complicado y explica que las Intranets son la respuesta a estos problemas. Desde nuestro punto de vista, el Internet no es otra cosa más que un servidor mucho más grande con clientes accesándolo desde todos los frentes, uno de los puntos a su favor (con relación a la tesis de este autor) es que la empresa no necesita estandarizar en equipos (desktops) ni en sistemas operativos para que la tecnología Internet funcione adecuadamente, una infraestructura cliente/servidor existente puede servir como el *backbone* para una intranet. Además con la existencia de herramientas de desarrollo de bajo costo disponibles en la misma gran red, la creación de sistemas puede ser más barata que en el

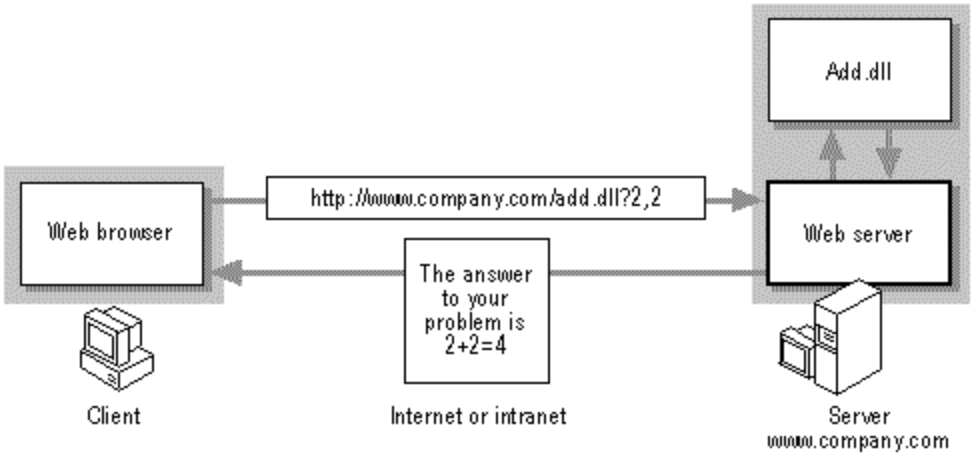
mundo tradicional cliente/servidor. Otro punto a favor del desarrollo de aplicaciones bajo el paradigma de las intranets, es que dada la popularidad del mundo Internet, prácticamente todo el mundo está familiarizado con los conceptos del Web, esto redunda en una gran reducción de costos de entrenamiento para usuarios finales, así como también se reducen los costos de mantenimiento/distribución de aplicaciones y documentación.

Sin embargo hoy por hoy la mayoría de aplicaciones en intranets funcionan como sistemas de distribución de información, desafortunadamente el Web fue diseñado para un ambiente de visualización de documentos, esta óptica hace que el concepto pierda muchas potencialidades ligadas al mundo de los negocios interactivos; las aplicacions para el procesamiento de transaciones son todavía pocas, pero debido a su interrelación con aplicaciones cliente/servidor esta cantidad deberá aumentar; definitivamente la visualización de documentos en el WEB no es la aplicación final del Internet. La visualización de documentos al estilo de Internet mezclado con el rendimiento de procesamiento transaccional que enriquece al cliente/servidor, ofrece un potencial muy fuerte para el desarrollo de un nuevo estilo de aplicaciones. La unión de las tecnologías hará que el Internet evolucione, a la par del cliente/servidor, hacia un nuevo concepto no solo de páginas estáticas sino también de páginas dinámicas y de una funcionalidad de solo lectura, a una funcionalidad transaccional. No cabe ninguna duda que la aparición en masa de la tecnología de Internet y el aporte posterior que ha representado el concepto de Intranets, representó un cambio sustancial en la forma de buscar soluciones a los problemas informáticos. Sin embargo nosotros creemos que los modelos basados en Intranets son un complemento o una extensión de la arquitectura cliente/servidor, no su reemplazo. De hecho la forma misma en que se plantea el concepto técnico de Intranet, es una forma de aplicar tecnología cliente/servidor. Veamos el siguiente gráfico:



Aquí se puede notar como con el simple hecho de utilizar un *browser*, el cual a su vez está accesando un servidor de páginas, implementa a su manera el concepto básico de cliente/servidor. Internet aporta, entre otras cosas, su facilidad de acceso, su interfaz gráfica estándar, su acesabilidad, etc y el cliente/servidor aporta robustez en la integridad de los datos, recuperación ante errores, actualización de datos en línea e interactividad. El Internet se hará más poderoso y el cliente/servidor se hará menos caro para desarrollar e implementar aplicaciones.

Otro ejemplo más claro es el siguiente:



Aquí se representa el concepto de cliente/servidor en toda su dimensión. Un cliente localizado en cualquier lugar del mundo o en el ámbito de definición de la Intranet corporativa, envía un requerimiento vía una dirección URL a una rutina localizada en el servidor (también se envían sus parámetros). El servidor replica diciendo: *la respuesta a su problema es tal*. Ese es el concepto bajo el cual operan las intranets el mismo es una forma de trabajar de cliente/servidor. Acceso un servicio remoto (un servidor proveedor de servicios) este me sirve como un medio para posteriormente continuar procesando localmente en mi máquina. El servidor me ofrece información, el cliente me permite procesarla.

Este particionamiento de funciones reduce significativamente las labores que podrían ser muy pesadas para un solo servidor. Recuérdese que para que el web pueda operar, se requiere de la existencia no solo de servidores, también requiero de un *browser* el cual a su vez necesita de una máquina con su propio sistema operativo (Microsoft Windows o Macintosh OS), controlando el despliegue (se tiene una interfaz de despliegue gráfica), además de tener que controlar los discos locales, la impresoras y la tarjeta de red. La aplicación está construida bajo el supuesto de que el usuario final dispone de un *browser* (de esta forma llegará a un número mayor de personas). En este ambiente se desarolla un servidor de aplicaciones que se comunica con la base de datos y producce documentos HTML para su posterior visualización. En fin por todos lados se respira el concepto de separación de labores y de procesamiento de funciones específicas, algo en lo cual el concepto cliente/servidor es definitivamente robusto.

Conclusiones - parte 3

Una de las preguntas que deben hacerse los encargados de dirigir las empresas, es si su organización está haciendo un uso adecuado de la tecnología. Se deben preguntar si sus sistemas de información o de procesos en verdad contribuyen a una adecuada gestión de sus empleados. Distintos miembros del Club de Investigación Tecnológica han proclamado que muchas veces la tecnología "llega" obsoleta a nuestro país, y eso es en ocassiones cierto. Hoy por hoy la tecnología cliente/servidor no es precisamente la tecnología de punta, no es lo último en innovación. Sin embargo las tecnologías que hoy marcan la pauta hacen uso de ella. Es difícil encontrar una empresa de éxito que no implemente de una u otra manera los conceptos de cliente/servidor.

Los profesionales en sistemas, por el rol que desempeñan en su organización, deben convertirse en gestores de cambio para bien de su empresa. Deben evitar que su empresa caiga en huecos que la lleven a retrasarse tecnológicamente. La tecnología cliente/servidor permite ese desarrollo. Como ya se mencionó, no es tecnología muy novedosa, pero sigue siendo totalmente segura y vigente. La tecnología es una herramienta que puede permitir a las empresas encarar la competencia con mejores armas. En este punto el profesional en sistemas juega un rol primordial, dado que debe ser el impulsor de su uso efectivo, nada se gana con disponer de la tecnología si no se utiliza de la mejor manera. La inversión en tecnología podría convertirse en arma de doble filo si no permite una recuperación de la inversión. Es por ello que confiamos en los principios que encierra esta tecnología: transcienden las modas y permiten hacer realidad aplicaciones confiables y evolucionables.

El futuro en este campo es impredecible. El avance tecnológico es vertiginoso e impresionante, de tal forma que es imposible saber lo que nos depara el futuro. Ante esta incertidumbre, es bueno estar en el presente con las herramientas que mejor han funcionado con hechos en las empresas; cliente/servidor es una de ellas.

El enfoque de este informe tiene matices técnicos y estratégicos. Hemos querido resaltar los beneficios y posibles inconvenientes que pueda tener la tecnología cliente/servidor con miras a lograr un uso más adecuado del ambiente computacional.

Bibliografía

[Adhikari 1995] Adhikari, Richard, *All Together Now*, Revista Computerworld Client/Server Journal, Estados Unidos, Junio 1995.

[Ayer 1995] Ayer, Steve, *Object-Oriented Client/Server Application Development Using Object PAL and C++*, Editorial McGraw-Hill, Inc., Estados Unidos, 1995.

[Berson 1996] Berson, Alex, *Client/Server Architecture*, Second Edition, Editorial McGraw-Hill, Inc., Estados Unidos, 1996.

[Boar 1993] Boar, Bernard, *Implementing Client/Server Computing, A Strategic Perspective*, Editorial McGraw Hill, Estados Unidos, 1993.

[Bochenski 1994] Bochenski, Barbara, *Implementing Production-Quality Client/Server Systems*, Editorial John Wiley & Sons, Estados Unidos, 1994.

[Byte 1994] Byte Magazine, Advanced Operating Systems, Byte Special Report, Revista Byte, Estados Unidos, Enero 1994.

[Caldwell96] Caldwell, Client-Server: Can It be Saved ?, Informationweek, Abril 8, 1996 .....

[Coulouris 1988] Coulouris, George F. y Dollimore, Jean, *Distributed Systems: Concepts and Design*, Editorial Addison-Wesley publishing Company, Estados Unidos, 1988.

[Gray 1993] Gray, J., Reuter, A., *Transaction Processing: Concepts And Techniques*, Editorial Morgan Kaufman Publishers Inc., Estados Unidos, 1993.

[Griffith 1995] Griffith, Vicki, *An Overview of Client/Server Remote Access Technology*, Gupta Developers Conference White Paper, Estados Unidos, 1995.

[Guengerich 1994] Guengerich, Steven, *Rightsizing Information Systems*, Second Edition, Editorial Sams Publishing, Estados Unidos, 1994.

[Hatfield 1996] Hatfield, Bill, *Developing Powerbuilder Applications*, Fourth Edition, Editorial Sams Publishing, Indiana, Estados Unidos, 1996.

[Ingraham 1995] Ingraham, Bud, *Client/Server: Snakepit or Salvation?*, Gupta Developers Conference White Paper, Estados Unidos, 1995.

[Jenkins 1995] Jenkins, Avery, *Centralization Strikes Again*, Revista Computerworld Client/Server Journal, Editorial Advisory Board, Estados Unidos, Agosto 1995.

[Larson 1995] Larson, James, *Database Directions*, Editorial Prentice Hall, New Jersey, Estados Unidos, 1995.

[Marion 1994] Marion, William, *Client/Server Strategies: Implementation in the IBM Environment*, Editorial McGraw-Hill, Inc., Estados Unidos, 1994.

[Martin 1994a] Martin, Richard, *The Premise and the Promise*, Client/Server Strategies, 1994.

[Martin 1994b] Martin, Richard, *The Moving Wave of Techonology*, Client/Server Strategies, 1994.

[Martin 1994c] Martin, Richard, *Clients (vs. Servers)*, Client/Server Strategies, 1994.

[Martin 1994d] Martin, Richard, *Server (vs. Clientes)*, Client/Server Strategies, 1994.

[Martin 1994e] Martin, Richard, *No Job Too*, Client/Server Strategies, 1994.

[Martin 1994f] Martin, Richard, *Explicit Acts of Architecture*, Client/Server Strategies, 1994.

[Martin 1994g] Martin, Richard, *Two Tiers or Three?,* Client/Server Strategies, 1994.

[Martin 1994h] Martin, Richard, *Selecting an Appropiate Client/Server,* Client/Server Strategies, 1994.

[Martin 1994i] Martin, Richard, *The Seven Habits of Succesful Client/Server Projects*, Client/Server Strategies, 1994.

[Martin 1994j] Martin, Richard, *Integrating Legacy & Client/Sever Systems,* Client/Server Strategies, 1994.

[Mullender 1989] Mullender, Sape, *Distributed Systems*, Editorial ACM Press, Estados Unidos, 1989.

[Nutt 1992] Nutt, Gary J., *Open Systems*, Editorial, Prentice Hall, Estados Unidos, 1992.

[OMG 1992] Object Management Group, *The Common Object Request Broker: Architecture And Specification*, Editorial John Wiley & Sons, Inc., Estados Unidos, 1992.

[OMG 1993] Object Management Group, *Object Management Architecture Guide*, Editorial John Wiley & Sons, Inc., Estados Unidos, 1993.

[Orfali 1994] Orfali, Robert y otros, *Essential Client/Server Survival Guide*, Editorial John Wiley & Sons, Estados Unidos, 1994.

[Orfali 1995] Orfali, Robert y Harkey, Dan, *Client/Server Computing*, Byte Special Report, Editorial McGraw-Hill Inc., Estados Unidos, Abril 1995.

[Price 1995] Price Waterhouse Interamerica Consulting Group, *Estándares de Sistemas Abiertos de Software*, Informe No. 18 del Club de Investigación Tecnológica, Costa Rica, 1995.

[Purba 1994] Purba, Sanjiv, *Developing Client/Server Systems using Sybase Sql Server*, Editorial John Wiley & Sons, Estados Unidos, 1994.

[Rajan 1995] Rajan, Sundar, *Upsizing from Xbase to Client/Server: Tips and Techniques*, Gupta Developers White Paper, Estados Unidos, 1995.

[Symons 1995] Symons, Van, *AS/400 Client/Server Computing: Putting the World of Information in the Palm of Your Hand*, Gupta Developers Conference White Paper, Estados Unidos, 1995.

[Tash 1995] Tash, Jeffrey B., *Client/Server Infrastructure Road Map*, Revista Computerworld Client/Server Journal, Editorial Advisory Board, Estados Unidos, Agosto 1995.

[Waterson 1995] Waterson, Christopher y Sendowski, Menachem, *Bechtel Electronic Time Record: A Three-Tiered Application Success Story*, Gupta Developers Conference White Paper, Estados Unidos, 1995.

[Zeitz 1995] Zeitz, William, *GAP Syndrome*, Revista Computerworld Client/Server Journal, Estados Unidos, Junio 1995.

Otras referencias útiles

Revista Client/Server Journal, Framingham, Massachusetts, Estados Unidos, Junio 1995.

Revista Client/Server Journal, Framingham, Massachusetts, Estados Unidos, Agosto 1995.

Revista Client/Server Journal, Framingham, Massachusetts, Estados Unidos, Noviembre 1995.

Revista Information Week, Abril 1996.

Revista Database Programming & Design, Miller Freeman Publications, San Francisco, Estados Unidos, Enero 1996.