81
241

UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE INGENIERIA

FUNDAMENTOS Y VENTAJAS DE MIGRACION A UNA ARQUITECTURA CLIENTE/SERVIDOR

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

RODOLFO PICHARDO SOBERANES

DIRECTOR DE TESIS: ING. HERIBERTO OLGUIN

CIUDAD UNIVERSITARIA 1997

Agradecimientos

Dedico esta tesis a mis padres, sin ellos no sería quien yo soy, gracias a ellos aprendí que la tenacidad y perseverancia son la base para el éxito; mil gracias por su amor, comprensión, consejos y respaldo.

Agradezco el apoyo incondicional de mis hermanas (Diana, Carmen y Viviana), tíos y en especial a Lourdes que siempre me ha demostrado su amor y comprensión. Sin todos ellos nunca hubiera sido posible esta tesis.

Doy gracias a la UNAM, a innumerables profesores que durante mi trayectoria de estudiante me formaron, y sembraron en mi una semilla que ahora está dando frutos, a ellos les debo lo que soy.

En especial deseo hacer presente mi reconocimiento y agradecimiento al Ing. Heriberto Olguín, no sólo por su profesionalismo, tiempo y valiosos consejos, sino también por ser el fundador del Centro de Cálculo de la Facultad de Ingeniería (CECAFI) del cual me siento muy orgulloso de haber sido becario, al CECAFI le debo mucho del éxito de mi vida profesional, mil gracias Ing. Heriberto.

Mi reconocimiento a la Lic. Amelia Fiel Rivera y Lic. Orfe Castillo Osorio, por su apoyo en la corrección de estilo de la presente tesis, mi más cumplido agradecimiento.

Y el agradecimiento más importante es para Dios, sin él nada sería posible, gracias por darme la oportunidad de vivir.

Abuelos, si me están viendo desde el cielo, esta tesis es también para ustedes, gracias por su amor, los recordaré siempre, y si Dios nos la presta seremos grandes.

Rodolfo Pichardo

INTRODUCCIÓN

La información como un arma competitiva

Las organizaciones operan en un mundo de desaciertos e intervención gubernamental; de políticas impredecibles en los ámbitos monetario, fiscal, impositivo y regulador, de ciclos de negocios y recesiones; de cambios abruptos en las políticas comerciales; de competencia doméstica e internacional, de desacuerdos políticos y sociales, de contracorrientes de cambio en los mercados, y de crecientes costos laborales. A decir verdad, éste es un ambiente implacable y competitivo en el que deben sobrevivir las organizaciones. Para evitar el fracaso, sobrevivir y lograr el éxito, las organizaciones deben explotar las dimensiones de la oportunidad de una gerencia informada, de la diferenciación de productos y servicios, y de una creciente productividad.

Claramente, la información es el arma principal que ayudará a la gerencia, a los productos y servicios, como a la productividad a penetrar en el ambiente competitivo. El encanto de la tecnología informática no hará avanzar estas dimensiones, pero sí lo hará la necesidad de contender y sobrevivir en un medio competitivo y violento, que incluye una competencia internacional más fuerte. Debe quedar claro que las computadoras, la tecnología informática y la información de calidad no son los fines sino simplemente las armas competitivas que apoyan a las organizaciones para alcanzar las metas de los gerentes triunfadores, de productos y servicios excelentes y de una mayor productividad y el éxito a final de cuentas. Cualquiera que sea su industria, las compañías que producen información de la más alta calidad permanecerán como -o se convertirán- en las más fuertes competidoras del ramo. Por otra parte, si una compañía no puede mejorar su información, quedará a la zaga de aquéllas que sí pueden.

En el pasado, la mayoría de los sistemas de información eran diseñados por personas que carecían de una perspectiva táctica y estratégica. Aún más, consideraban a los sistemas de información más como un manipulador de números que como un arma competitiva o un vehículo para extenderse en el mercado y formar un enlace con los clientes. Tendían a ver los sistemas de información como medios para mejorar las tareas de procesamiento de datos y no como herramientas tácticas y estratégicas.

Actualmente, un buen número de organizaciones están empleando los sistemas de información y su tecnología no solamente para aumentar la productividad, sino también para diferenciar sus productos y servicios en el mercado. A continuación se presentan unos cuantos ejemplos de organizaciones que están aprovechando la nueva tecnología informática para diferenciar sus productos y servicios.

1. **Compañías manufactureras.** Se da énfasis a la calidad del producto al inicio del proceso de manufactura, en vez de inspeccionar un producto al final de la línea de producción. Además, el empleo de la automatización flexible con máquinas programadas y controladas numéricamente hace posible utilizar la misma máquina para procesar piezas diferentes. Los datos de diseño y fabricación se almacenan en memorias de computadoras y se comunican de una máquina a otra. El resultado es menor mano de obra, menor desperdicio de material y productos de mayor calidad. Asimismo, en algunas plantas se muestra, mediante pantallas de circuito cerrado de televisión, el precio de las cotizaciones de las acciones de la compañía (la mayoría de los trabajadores poseen acciones de la compañía). Se han adoptado sistemas de control de inventarios, o el enfoque de "utilice una pieza, haga una pieza". El resultado de esto no es sólo una mayor productividad, sino también ahorros sustanciales en los costos de mantener el inventario.

2. **Proveedores.** Cuando el pedido de un cliente entra al sistema de información, pasa por una serie de reglas de decisión programadas, como verificación de crédito, y se coloca en situación de retención, pedido pendiente o aprobación para embarque. Si la orden es un "adelante", el sistema busca en la base de datos el almacén apropiado y la localización exacta del producto (*rack*), a continuación, el pedido aparece inmediatamente en la terminal del gerente de embarque. El gerente decide en qué camión se enviará el pedido e introduce dicho número a la computadora. El sistema calcula automáticamente el tamaño, peso y cargos de envío del embarque, posteriormente se imprimen etiquetas de surtido y empaque y los documentos correspondientes que detallan la información de dicho embarque. En algunos casos, los sistemas automatizados de captura de pedidos y embarque como éste han reducido el tiempo que requieren los clientes para obtener sus productos de una o dos semanas a uno o dos días.

3. **Líneas aéreas.** Las líneas áereas deben tener sistemas de información bastante sofisticados para ser competitivas hoy en día. Estos sistemas proporcionan, entre muchas otras cosas, actualizaciones más rápidas de tarifas, precios internacionales o de agencias de viajes, enlaces directos con los hoteles y agencias de renta de automóviles, como de servicios bancarios. Los programas de estímulos para pasajeros frecuentes requieren bases de datos computarizadas. Algunas de las líneas aéreas más progresistas tienen extensiones de sus sistemas de información incorporadas en las agencias de viaje. Estas agencias, a su vez, ofrecen a sus clientes todo una gama de servicios de viajes, desde asignación de asientos hasta reservaciones en hoteles. Claramente, el enlace con las agencias de viaje tiene la intención de aumentar los ingresos de la línea aérea.

4. **Transportistas.** Un transportista nacional emplea una red de terminales de computadoras como una herramienta estratégica para atender las necesidades de sus clientes. Los clientes principales de la compañía cuentan en sus oficinas con terminales que están conectadas sobre líneas de transmisión rentadas al sistema de inventarios de transportes. La base de datos permite a la compañía proporcionar a sus clientes más importantes un análisis de las rutas de envío y monitoreo de gastos. El sistema de información es una parte central de la forma en que la compañía opera su negocio.

A decir verdad, el tiempo en que las personas no sabían computación casi ha pasado a la historia, y a medida que crezca la necesidad de una mayor productividad, la tecnología informática se convertirá en forma natural en un soporte para que tanto los trabajadores de la información como los de operaciones mejoren su desempeño.

La arquitectura del sistema de información es a menudo la principal determinante de la posibilidad de evolución y, por consiguiente, de la capacidad de mantener actualizada la información, así como de su más rápida distribución en los niveles donde sea requerida. Una arquitectura rígida representa una gran inversión en tiempo y gastos.

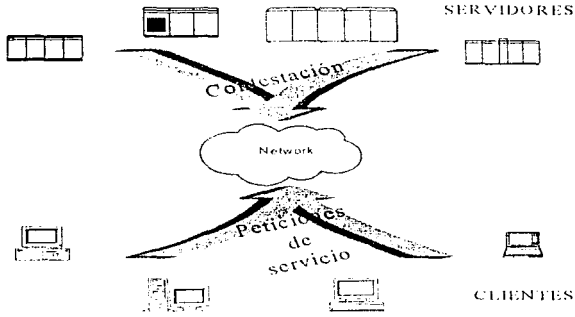
Una arquitectura flexible coloca a la empresa en una posición que le permite un cambio constante. Una empresa puede ser planeada para un fracaso seguro con el uso de una arquitectura rígida, o bien, en una empresa triunfadora que se adecua a las reacciones y modificaciones en la dinámica de sus clientes, proveedores y competidores.

Históricamente, la arquitectura que ha prevalecido en la construcción de aplicaciones ha sido el modelo centralizado *(Host-Centered Computing Architecture)*. En este tipo de arquitectura el usuario realiza sus operaciones desde terminales-tontas[1] y el procesamiento de la información es realizado por el *host*, la información resultante es regresada al usuario, con lo cual termina el ciclo de procesamiento. Resulta importante resaltar que la "revolución" en capacidad de procesamiento que se ha tenido en estos últimos 15 años (la incorporación de minicomputadoras y computadoras personales) ha modificado el tamaño, la relación precio-rendimiento, funcionalidad y ubicación geográfica de los usuarios respecto al equipo de cómputo, pero no se ha modificado en su esencia el modelo de arquitectura. Esto lo analizaremos con más profundidad en el capítulo dos.

[1] Dispositivos que no tienen la capacidad de procesamiento.

Actualmente, las tecnologías se están uniendo por medio de las telecomunicaciones que interconectan las computadoras, las bases de datos, los sistemas de automatización de oficinas, el diseño y la ingeniería asistida por computadora, los sistemas de manufactura asistidos por computadora, los robots, la inteligencia artificial y los sistemas expertos, el correo electrónico, los almacenes automatizados y los sistemas de control de inventarios, los dispositivos de mercadeo electrónico, los equipos de pruebas automáticas, los dispositivos de seguridad y los sistemas de administración de energía. El objetivo de esta unión es "bajar" las aplicaciones existentes en los *mainframes* hacia servidores de PC y LAN, usando para ello la arquitectura cliente/servidor.

Arquitectura cliente/servidor



La figura anterior ilustra el modelo de cómputo cliente/servidor, que puede definirse de la siguiente forma:

La computación cliente/servidor es un modelo de cómputo en el cual una aplicación es particionada entre múltiples procesadores (front-end y back-end) que cooperan (transparente al usuario) entre sí para completar el procesamiento como una sola tarea unificada. Los productos para este ambiente permiten administrar a los procesadores para que juntos realicen una misma tarea, lo cual produce la ilusión de un solo sistema. Los recursos compartidos se encuentran colocados como servidores, que ofrecen uno o más servicios. Las aplicaciones son colocadas como clientes, los cuales accesan los servicios autorizados. La arquitectura es recursiva, en el hecho de que un servidor puede transformarse en cliente y realizar peticiones de servicio hacia otro servidor en la red.

En los años 90 los negocios se enfrentan a una gran competencia en la cual si no se adaptan rápidamente a los cambios (los cuales ocurren aceleradamente) se vuelven obsoletos en un periodo de tiempo muy corto. **El propósito de esta tesis es proporcionar los elementos para conocer las ventajas competitivas y tecnológicas de migración a un ambiente cliente/servidor.**

Tal como se menciona en el objetivo, mediante el análisis de los fundamentos, el procesamiento cooperativo, las necesidades de la existencia , el *middleware*, aplicaciones de dos y tres niveles, el método de implantación, la autoevaluación del personal de la empresa, relativos a la arquitectura cliente/servidor, así como de las estrategias de mercado, en la referente a competencia, problemática de los negocios y la implementación de la arquitectura cliente/servidor. La tesis será un documento de referencia muy útil para cualquier empresa o institución a fin de que puedan contar con elementos sólidos para realizar un estudio de factibilidad técnica y económica con el objeto de migrar o no a la arquitectura cliente/servidor.

La computación cliente/servidor es una nueva, diferente y ventajosa capacidad que permite la implementación de nuevas y mejores soluciones que pueden ser aplicadas a una empresa. El seleccionar el sistema cliente/servidor más óptimo a nuestras necesidades es un proceso particular de cada empresa, existen diferentes caminos por donde ir y una decisión incorrecta puede conducirnos a gastar tiempo y dinero.

Para lograr el objetivo propuesto, decidí dividir esta tesis en cuatro capítulos. El primero tiene como propósito sentar las bases para el entendimiento de la arquitectura cliente/servidor, y explica los elementos necesarios y su evolución para llegar a un modelo de cómputo cliente/servidor, asi mismo, expongo los siguientes temas: conceptos de bases de datos, redes de computadoras, sistemas abiertos e interfaces gráficas. En el capítulo dos, describo en orden evolutivo (de menor a mayor complejidad) las principales arquitecturas en las que se soportan la mayoría de las aplicaciones, éstas van desde arquitecturas centralizadas hasta distribuidas. En el capítulo tres explico lo que es la arquitectura cliente/servidor, detallando los requerimientos de *hardware*, *software*, ventajas, desventajas, presento una metodología de selección de posibles arquitecturas cliente/servidor, y propongo una metodología de implantación. Finalmente, en el capítulo cuatro, que tiene como objetivo dejar en claro las ventajas competitivas que son resultado de la implementación de un sistema cliente/servidor, analizo brevemente los principales problemas de las empresas en cuanto a sistemas de cómputo se refiere, y propongo una solución a estos problemas mediante la implementación de un sistema cliente/servidor.

Sin más preámbulos, comencemos a introducirnos a esta arquitectura, que no es una idea nueva, pero que actualmente se ve como la solución mágica a todos nuestros problemas de cómputo, mucho es lo que nos promete, descubramos realmente qué nos ofrece.

ÍNDICE

CAPÍTULO I
FUNDAMENTOS

OBJETIVO

La arquitectura cliente/servidor se fundamenta en diferentes tecnologías, sin la existencia de éstas no sería posible su implantación, por lo que el objetivo de este capítulo es explicar, en forma breve, cuáles son los fundamentos tecnológicos de la arquitectura cliente/servidor.

INTRODUCCIÓN

Los sistemas de información utilizados dentro de una organización frecuentemente se desarrollan independientemente uno del otro; por lo que, en muchas ocasiones distintas plataformas de hardware son adquiridas para diferentes aplicaciones, y es común encontrar diversos tipos de terminales conectadas a distintas plataformas. Esto tiene como consecuencia las siguientes implicaciones :

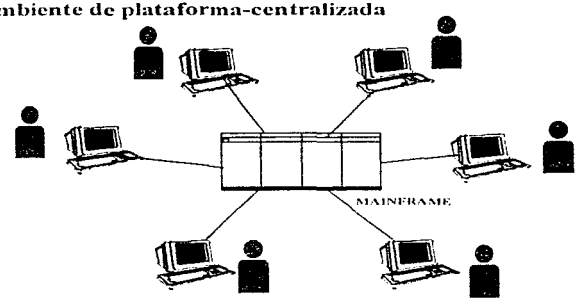
- Un elevado costo en sistemas de comunicación, así como también de plataformas de cómputo.

- La integración de las aplicaciones y datos es realizada a través de papel o medios magnéticos (discos y cintas).

- Los usuarios se enfrentan a la necesidad de aprender a utilizar diferentes aplicaciones y posiblemente diferentes sistemas operativos .

Escenarios como éstos reciben la denominación de sistemas de **plataforma centralizada** (*Platform-Centered*) en donde los usuarios del sistema deben adecuarse a los recursos disponibles en la computadora a la cual se conectan.

Ambiente de plataforma-centralizada



Actualmente, la tendencia es crear sistemas **enfocados hacia los usuarios** (*User-Centered*). En estos ambientes el usuario se conecta a un sistema que cubre sus necesidades. Las tecnologías que han permitido la creación de este tipo de ambientes han sido principalmente:

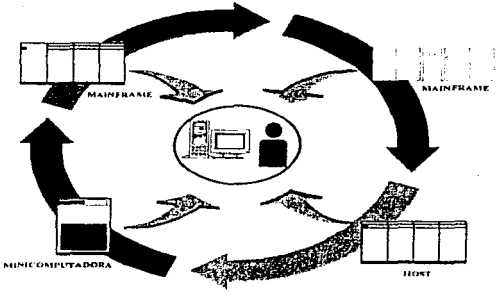
- **GUI** (*Graphical User Interface*). La utilización de una interfaz gráfica[1] permite al usuario contar con una interfaz común para todas las aplicaciones, como también de una cierta facilidad para la integración de datos de diferentes plataformas, lo cual se realiza a través de la capacidad de *Cut and Paste* que poseen los ambientes gráficos.

- **SQL** (*Structured Query Language*). Por medio de SQL el usuario tiene acceso a los datos de la empresa sin importar su ubicación física o plataforma de hardware, esto permite la programación de las aplicaciones independiente del control de los datos.

- **Comunicaciones**. Las redes de área local y la tecnología de comunicaciones proveen del medio necesario para interconectar estaciones de trabajo que se encuentran remotamente, lo cual permite compartir datos y periféricos que se encuentran ubicados en distintas zonas geográficas.

Por medio del uso combinado de estas tecnologías es como el usuario se ha convertido en la parte central del sistema de información, en donde el acceso a los datos es realizado a través de una interfaz común y las aplicaciones utilizan recursos de diferentes plataformas con el objetivo de satisfacer los requerimientos del usuario.

Ambiente de usuario-centralizado



Para lograr un ambiente en donde el usuario es la parte central, las aplicaciones deben ser diseñadas de acuerdo con un modelo básico. Este modelo divide una aplicación en tres capas:

- Interfaz con el usuario
- Procesamiento
- Interfaz con los datos

[1] Son fáciles de aprender y utilizar, por lo que un usuario puede ser entrenado muy rápidamente para el manejo de una nueva aplicación.

Estos tres elementos se pueden encontrar ya sea localmente, o bien, en forma distribuida. La distribución de estas capas a través de diferentes nodos se hace posible por la existencia de un medio de comunicación, al que denominaremos la **red de computadoras**; esta distribución tiene las siguientes ventajas:

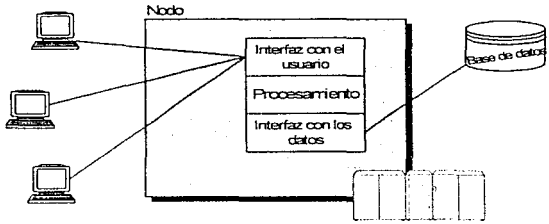
✓ Extiende el sistema hacia nuevos usuarios ubicados en zonas geográficas distintas.

✓ Se comparten funciones entre más de una aplicación .

✓ Se comparte información entre los *sites*.

Además de estas ventajas es posible agregar, de una manera relativamente simple, funciones a las aplicaciones existentes.

Modelo básico de una aplicación



Estas son sólo algunas de las ventajas de esta distribución, que en los siguientes capítulos se explicarán con más detalle.

En este capitulo, explicaré de una manera clara y sencilla algunos de los principales conceptos utilizados en:

Redes de computadoras Debido a que la distribución de una aplicación se sustenta en la existencia de ésta.

Interface con el usuario Se expondrá la tendencia actual hacia los sistema gráficos.

Base de datos Tipos y características.

Al final del capítulo, expondré la necesidad de la integración de los sistemas y analizaré algunas características de los sistema abiertos.

CONCEPTOS DE REDES

Aunque la mayoría de los desarrolladores de aplicaciones cliente/servidor consideran que no se requiere ser experto en redes, es necesario conocer su funcionalidad y razón de ser. La experiencia muestra que existen una gran variedad de problemas (desde los triviales hasta los muy complejos) cuando la red de área local no es planeada y configurada apropiadamente. En un ambiente en donde interactúan bases de datos y aplicaciones de diferentes tipos, en diferentes zonas geográficas, comunicadas a través del uso de una red (en cualquier tipo de configuración) no resultará obvio detectar cuándo un problema es debido a la red y no a la base de datos o aplicaciones, en ocasiones es muy común que el administrador de la base de datos "gaste" mucho de su tiempo investigando por qué la base de datos está lenta, cuando en realidad el problema es generado por la red[2].

En muchas casos el éxito de un sistema cliente/servidor se encontrará sujeto a las capacidades de transferencia de datos de la red que se esté utilizando, por lo que es necesario que los desarrolladores de aplicaciones conozcan más de redes, o bien, trabajen más cercanamente con los expertos de redes desde el inicio de un nuevo proyecto. Esto es de gran importancia debido a que uno de los principales objetivos de un ambiente cliente/servidor es la capacidad de adecuarse a los cambios, por lo que en un futuro cuando se requieran nuevas opciones en el sistema, y se necesite mezclar más software y hardware en la red, deberá mantenerse un rendimiento adecuado para cubrir los nuevos requerimientos.

La finalidad de este tema es explicar algunos de los principales conceptos que se utilizan en el ambiente de las redes. El conocimiento de estos conceptos puede definir el éxito de un proyecto en donde se haga uso de una red; para nuestro caso, el éxito de una implantación de un sistema cliente/servidor dependerá en gran medida del soporte de la red.

Las redes son en la actualidad un requisito indispensable para el desarrollo de un sistema cliente/servidor. Una de las principales razones del porqué la computación cliente/servidor se ha convertido tan popular, es debido al creciente uso de redes, las cuales podemos encontrarlas tanto en las pequeñas como en las grandes compañías.

Componentes de una red

Existen una gran cantidad de componentes que forman una red, pero se cuenta con tres principales categorías, las cuales son esenciales para cualquier LAN (*Local Area Network*):

- Cableado
- Tarjeta de red (NIC, *Network Adapter Card*)
- Sistema operativo de red (NOS, *Network Operating System*)

[2] Los dos principales problemas a los que nos enfrentamos en una red son: un insuficiente ancho de banda y congestionamiento en la red. Estos dos problemas causan que la red sea "lenta".

Cableado

Es el medio físico por el cual son transmitidos los datos (señales eléctricas codificadas) de una computadora a otra, la conexión de las computadoras a este medio es realizada a través del uso de tarjetas de red instaladas en cada una de éstas.

La industria de redes ha estandarizado tres principales medios físicos que pueden ser utilizados en la conexión: cable coaxial, par trenzado (*UTP*) y fibra óptica (*FDDI*). La capacidad de transferencia de estos medios es medida en millones de bits por segundo (*Mbps*).

Cable coaxial

El cable coaxial posee un centro conductor rodeado por una capa de aislamiento y una capa externa no conductiva. Este tipo de cable es resistente a la interferencia y cuenta con un mayor rango de alcance que otros tipos de cable (por ejemplo: par trenzado no blindado).

Con el uso de equipos de comunicación simples, el cable coaxial cubre distancias más largas que el par trenzado y soporta un rango más alto de transferencia de datos. Dependiendo de su uso, existen diferentes tipos de cable coaxial:

- THINNET
- THICKNET

Thinnet

Es un cable flexible y es utilizado básicamente para redes de muy corta distancia entre las estaciones de trabajo, se conecta directamente a la tarjeta de red a través de conectores *BNC* tipo T y requiere de terminadores en sus extremos.

Thicknet

Es un cable rígido y debido a que permite la transferencia de datos a través de distancias largas es utilizado típicamente como *backbone* para interconectar pequeñas redes conectadas bajo *thinnet*.

La tabla 1.1 muestra las ventajas y desventajas del cable coaxial.

Ventajas	Desventajas
✓ Bajo costo de mantenimiento	✗ Limitada distancia y topología
✓ Simple de instalar	✗ Baja seguridad
✓ Buena resistencia al ruido	✗ Dificultad para realizar cambios significativos a la topología
✓ El costo del cable es mayor que el de par trenzado, pero el mantenimiento de los componentes es más económico	

Tabla 1.1 Ventajas y desventajas del cable coaxial

Par trenzado

El par trenzado consiste de dos conductores aislados, trenzados entre sí. Un conjunto de pares trenzados son, en ocasiones, agrupados y encapsulados a través de un aislante.

El par trenzado blindado es menos propenso a la interferencia eléctrica y soporta una mayor velocidad de transferencia que el no-blindado. La tabla 1.2 muestra las ventajas y desventajas del par trenzado.

Ventajas	Desventajas
✓ Es una tecnología muy estudiada	✗ Susceptible al ruido
✓ Facilidad para agregar una computadora a la red	✗ Limitado ancho de banda
✓ Preexistencia de instalaciones realizadas para sistemas telefónicos	✗ Distancias limitadas
	✗ Requiere del soporte de componentes electrónicos de elevado costo

Tabla 1.2 Ventajas y desventajas del par trenzado

Fibra óptica

La fibra óptica es utilizada para el transporte de señales digitales en forma de fotones de luz. Una fibra óptica consiste de un cilindro de vidrio extremadamente delgado llamado núcleo, rodeado por una capa concéntrica de vidrio, conocida como revestimiento.

Existen dos fibras en el cable una de transmisión y otra de recepción. La fibra óptica no se encuentra sujeta a la interferencia, proporciona un alto grado de seguridad y es sumamente rápida (de 100 Mbps hasta 200 000 Mbps). La tabla 1.3 muestra algunas de las ventajas y desventajas de la fibra óptica.

Ventajas	Desventajas
✓ Alto nivel de transferencia	✗ Costo mayor que otros tipos de cables
✓ Es utilizada para largas distancias	✗ Requiere de trayectorias sumamente delicadas, debido a que no soporta doblajes
✓ Inmune a la interferencia	✗ Carece de componentes estándares
✓ Soporte a video, voz y datos	✗ Limitado, prácticamente, a un alto nivel de tráfico
✓ Dificultad para interferir	✗ Requiere de una gran capacidad técnica para su instalación y mantenimiento
✓ Utilizado como *backbone*	✗ Elevado costo de instalación

Tabla 1.3 Ventajas y desventajas de la fibra óptica

Tarjetas de red

Para cada tipo de cableado (thinnet, thicknet, par-trenzado, fibra óptica) existen diferentes conectores para unir la tarjeta de red al cable de comunicación. La tarjeta de red (NIC, *Network Interface Card*) es instalada en cada computadora participante en la red, incluye un puerto de conexión, en la parte trasera de la computadora, mediante el cual se realiza la conexión al medio de comunicación.

La tarjeta de red realiza las siguientes operaciones:

- Transferir los datos desde la computadora local hacia la red y viceversa.
- Crear paquetes de datos para su transferencia.
- Verificar la fuente y el destino de las transmisión.

Para una red tipo ethernet (10base2, 10base5, 10baset), se usará BNC si el cable es tipo thinnet; si se trata de cable tipo thicknet, se utilizará un conector AUI (*Attach Unit Interface*) de 15 pines; pero si el medio es par trenzado, se empleará un conector RJ-45 (comúnmente utilizado en los sistemas telefónicos).

Existen en el mercado tarjetas de red que incluyen más de un tipo de conector, por lo que bastará con configurar la tarjeta para indicar el tipo de medio de comunicación que se va a utilizar.

En las tarjetas de red, por lo general, deben de configurarse los siguientes parámetros:

- **IRQ** (Interrupción). Se deberá indicar el nivel de interrupción que se le asignará a la tarjeta, cuidando no "chocar" con algún otro dispositivo conectado a la computadora.

- **Puerto I/O** (entrada/salida). Especifica el canal a través del cual la información es transferida entre el hardware de la red de la computadora y el CPU.

- **Dirección de memoria base**. Define la dirección de memoria que será utilizada por la tarjeta de red para intercambiar información entre la computadora local y otra computadora conectada en la red.

- Para algunas tarjetas se requiere especificar el espacio de memoria RAM reservado para el buffer.

Estándares relativos a las redes

Los estándares han incrementado su importancia en la medida en que los requerimientos de interconexión de redes han aumentado. Cuando una red requiere ser conectada a otra, los estándares permiten este acoplamiento de forma relativamente simple. Las empresas que tienen planeado la implementación a largo plazo de sistemas cliente/servidor deben conocer los diferentes estándares de redes, con objeto de desarrollar un plan de implementación.

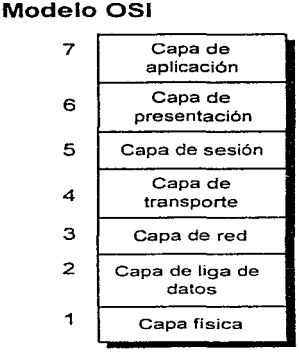
En el pasado, algunas grandes compañías, como IBM y DEC establecian sus propios estándares en el diseño de redes. Estos estándares describían los mecanismos requeridos para el intercambio de datos de una computadora hacia otra. Los desarrollos no fueron del todo compatibles. Las computadoras conectadas a un sistema IBM a través de su arquitectura SNA (*System Network Architecture*) no podían comunicarse en forma directa con redes que utilizaban la arquitectura DNA (*Digital Network Architecture*).

Actualmente, las organizaciones encargadas del establecimiento de estándares[3] desarrollaron modelos que son globalmente reconocidos y aceptados como estándares para el diseño de cualquier tipo de red de computadoras.

Existen dos modelos, el modelo OSI y IEEE, que establecen los estándares por seguir, y describen el funcionamiento de la red a través de capas funcionales.

Modelo OSI

En 1984, después de siete años de trabajo, la ISO publicó un modelo llamado *Open System Interconnection* (OSI). Este modelo es utilizado para describir el flujo de datos desde la conexión física de la red hasta la aplicación del usuario final, y en la actualidad es el más utilizado en el ambiente de redes.

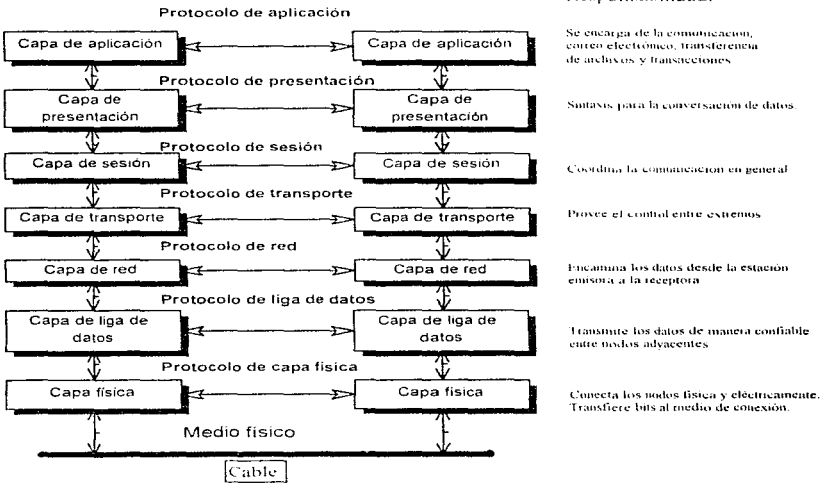


[3] La Organización Internacional de Estándares (ISO, International Standard Organization) y el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE).

Relación de las capas en el modelo OSI

El propósito de cada una de las capas es proporcionar servicio a la siguiente capa más elevada. Lógicamente las capas se comunican directamente con su capa asociada en otro nodo, físicamente, cada capa sólo se comunica únicamente con su capa adyacente

Relaciones entre las capas

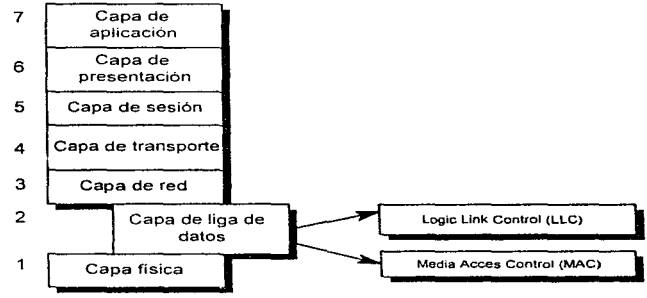


Excepto por la capa más baja en el modelo, ninguna otra capa transmite información en forma directa a su capa asociada en otro nodo. La información enviada debe pasar a través de las capas más bajas y escalar las capas en la computadora que recibe, hasta llegar a la misma capa que envió los datos.

Modelo IEEE 802

Otro modelo de red fue desarrollado por el Instituto de Ingenieros Eléctricos y Electrónicos, el cual ha proliferando en el ambiente de las redes de área local. Este proyecto fue llamado 802, debido al año y el mes en el cual inició (febrero de 1980).

Modelo IEEE



El modelo 802 está de acuerdo con el modelo OSI, pero se decidió definir con más detalle la capa de liga de datos. El proyecto 802 divide la capa de liga de datos en dos subcapas:

☞ MAC (*Media Access Control*) Control de acceso al medio, y

☞ LLC (*Logic Link Control*) Control de la liga lógica.

La subcapa de control de acceso al medio *MAC* tiene una comunicación directa con la tarjeta de red y es responsable de mantener la comunicación libre de errores.

La subcapa de control de liga lógica *LLC* se encarga de la administración de la comunicación (transferencia de datos).

Este proyecto da como resultado diferentes documentos, los cuales definen los estándares requeridos para tres topologías:

☞ 802.3 define los estándares para las redes tipo BUS, como Ethernet, el cual utiliza un mecanismo llamado *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD).

☞ 802.4 define los estándares para las redes tipo *Token-Passing Bus* (por ejemplo. ArcNet).

☞ 802.5 define los estándares para las redes de topología *anillo*.

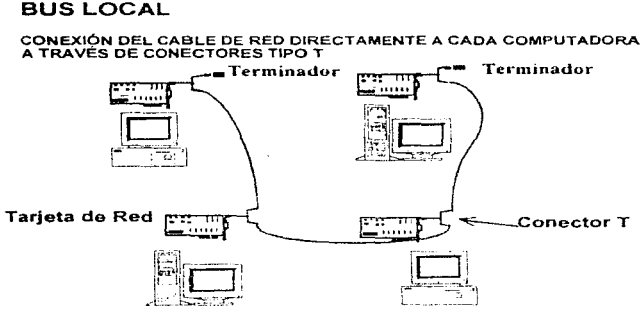
Características físicas de las redes de área local

Topología

La forma en que las computadoras en una *LAN* son conectadas es llamada *topología*. Las tres topologías más utilizadas son BUS, STAR BUS y RING.

Topología BUS

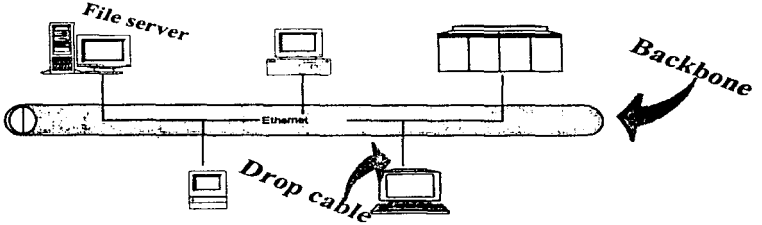
En la topología tipo bus se conecta cada una de las computadoras a un cable y en cada terminación del cable se coloca un *TERMINADOR*. La señal enviada por el cable viaja de ida y regreso de un terminador a otro.



Los mensajes en el bus viajan desde un extremo de la red al otro. La información es "pasada" por todas las computadoras que forman la red, cada computadora verifica la dirección destino en el mensaje, si la dirección es igual a la de la computadora el mensaje es recibido, si la dirección no es la misma, el mensaje viaja por el bus a la siguiente computadora en la red.

La red más simple de este tipo utiliza conectores tipo T para conectar el cable con la tarjeta de red. Un terminador es conectado al último conector T, en cada extremo de la red. Si una de las computadoras en la red falla, no se afectara el funcionamiento de la LAN pero si la conexión hacia una de las computadoras se pierde, o si el cable se abre, se perderá la conectividad causando que todas las computadoras conectadas, al segmento que se abrió, no puedan comunicarse. Una estructura más complicada de este tipo de topología se obtiene por medio del uso de *drop cables* y de un *backbone* como medio de distribución de los datos.

BUS CON DROP CABLES Y BACKBONE



La tabla 1.4 muestra algunas ventajas y desventajas de la red tipo BUS.

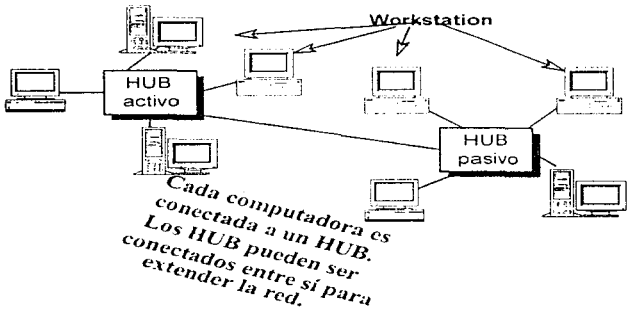
Ventajas	Desventajas
✓ La falla de una computadora no afecta el funcionamiento de la red	✗ Si el cable por medio del cual se mandan los datos se "abre" puede afectar la funcionalidad de una gran cantidad de computadoras en la red
✓ Flexibilidad y facilidad en la conexión de una computadora en la red	✗ Longitud limitada del cable, al igual que el número de computadoras
✓ Conectores y cables de bajo costo	✗ Degradación en el *performance* de la comunicación

Tabla 1.4 Ventajas y desventajas de la red tipo bus

Topología en estrella tipo BUS

En una red en la que se utiliza esta configuración, cada computadora es conectada a un dispositivo especial llamado *HUB*, el cual provee de una conexión común, para que todas las computadoras pueden comunicarse entre sí. Un *HUB* puede ser conectado a otro con el objeto de extender la red.

ESTRELLA



La tabla 1.5 muestra las ventajas y desventajas de esta topología.

Ventajas	Desventajas
✓ Facilidad para conectar una nueva computadora	✗ La falla del *HUB* deja fuera de la red a todas las computadoras conectadas a éste
✓ Monitoreo y administración central	

Tabla 1.5 Ventajas y desventajas de la topología en estrella tipo bus

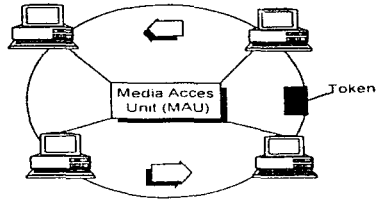
Topología TOKEN-RING

En este tipo de configuración, las computadoras son conectadas en un "loop" continuo en el cual un "TOKEN[4]" es pasado de una computadora a otra. Aunque el nombre de Token Ring hace referencia a un anillo, físicamente la red se implementa como una estrella y de manera eléctrica como un anillo (lógicamente). Las computadoras se encuentran conectadas a un *HUB* central llamado *MAU* (*Media Access Unit*), el cual está alambrado en una configuración de estrella. Las computadoras usan el token para intercambiar datos y deben esperar a un token "libre" para poder transmitir mensajes.

El token contiene la dirección del nodo que envía como también del nodo que recibe. Cuando la computadora destino recibe el mensaje, regresa el token a la computadora origen con objeto de verificar la transferencia y confirmar que la información fue recibida. El nodo que envía pasa el token al siguiente nodo en el anillo para que esa computadora pueda ahora transferir información en la red.

[4] El TOKEN contiene un *header*, en donde se almacena la dirección destino y fuente de los datos, que es el medio por el cual se transmiten los datos sobre la red.

TOKEN RING



El siguiente cuadro muestra las ventajas y desventajas de esta topología:

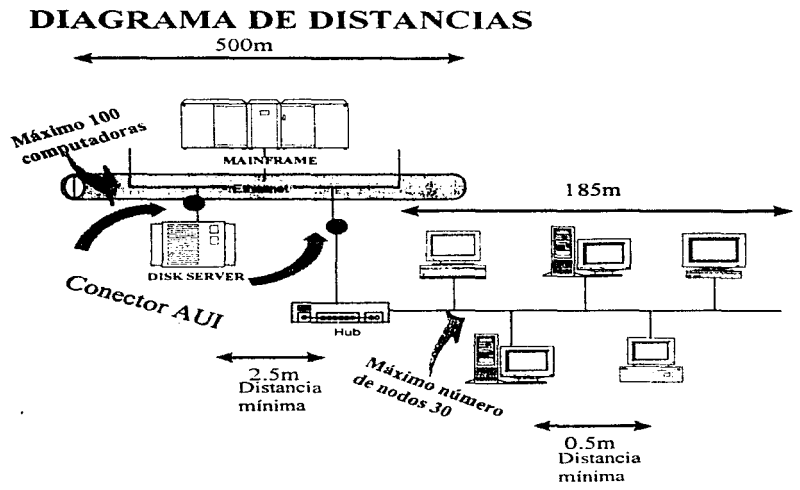
Ventajas	Desventajas
✓ La falla del cableado afecta a un número pequeño de computadoras ✓ La misma forma de acceso para todas las computadoras ✓ Limitada degradación del *performance* al crecer la red	✗ El cableado y las conexiones tienen un costo elevado

Tabla 1.6 Ventajas y desventajas de la topología token-ring

El siguiente cuadro muestra impedancias, longitudes, distancias y número de segmentos para los diferentes tipos de cables.

	Thinnet (10Base2)	Thicknet (10Base5)	Twisted-Pair (10BaseT)	Token-Ring
Topología	Local bus	Bus	Star	
Tipo de Cable	RG-58	Thicknet, 3/8 " Par-Trenzado Drop Cable	Par-Trenzado sin revestimiento	Par-Trenzado sin/o con revestimiento
Impedancia	50 Ohms	50 Ohms		
Resistencia del terminador	50 Ohms	50 Ohms	85-115 Ohms UTP 135-165 Ohms TP	100-120 Ohms UTP 150 TP
Longitud máxima de segmento	185 m	500 m	100 m	Desde 45 hasta 200 m
Mínima distancia entre computadoras	0. 5 m	2.5 m	2.5 m Entre el repetidor (*HUB*) y la computadora	2.5 m
Máximo número de segmentos conectados	5	5	Conexión en estrella no soporta el manejo de segmentos	33 Media Access Units (MAU)

El diagrama siguiente muestra gráficamente la información anterior.



NOS (Network Operating System)

Los sistemas operativos tienen a su cargo la coordinación de las funciones de la computadora, así como la administración de los recursos. Existe una diversidad de sistemas operativos, desde los existentes para sistemas mono-usuario hasta sistemas multiusuario. Como consecuencia de la necesidad del uso de redes de computadoras, surge una nueva generación de sistemas operativos, que se les denomina NOS (*Network Operating System*, en español "sistemas operativos de red"). Esta nueva generación incluye todas las funciones ya conocidas para un sistema operativo nativo (ejemplo MSDOS), pero además cuentan con nuevas funciones que le permiten el control y administración de los recursos que se comparten a través de la red. Un NOS debe distinguir entre las funciones "locales" y las "remotas", para hacer esto se cuenta con un controlador llamado **redirector**[5], el cual tiene a su cargo la determinación del tipo de función por realizar (local o remota). En caso de que la función sea local, el sistema operativo nativo ejecuta la función; en caso contrario, el NOS se encargará de la ejecución de la función y pasará la petición al dispositivo apropiado en la red.

Los sistemas operativos de red deben ocultar la ubicación física de los recursos utilizados por las aplicaciones, creando la imagen (ilusión) de un solo sistema, en donde en realidad pueden existir una gran cantidad de máquinas interconectadas.

[5] Utilizando el **redirector** los clientes accesan a los recursos como si éstos existieran conectados localmente a la computadora.

Para un ambiente cliente/servidor, los NOS deben ofrecer las siguientes características:

- **Localización transparente**. Usuarios, servicios y recursos son agregados y borrados de la red constantemente, pero nunca están sujetos a una localización fija. El sistema operativo es el encargado de localizar la ubicación de los recursos requeridos, para esto existirá un directorio global, distribuido a través de replicación[6] en los diferentes nodos que conforman la red.

- **Administración transparente**. Los servicios y funciones del NOS deben estar integrados a la administración de las funciones del sistema operativo local, debiendo existir una replicación transparente al usuario, por ejemplo: si un archivo es replicado en varias máquinas, el sistema operativo de red debe encargarse de sincronizar las actualizaciones, protegiendo a los usuarios contra fallas en la red (recuperación transparente y reconexión de la sesión).

- **Control de la seguridad transparente al usuario**. El usuario debe poder accesar los recursos, no importando donde se encuentre, ya sea de un cuarto de hotel, oficina, hogar, etc.; el NOS se encargará de controlar los accesos a los recursos .

- **Enlaces transparentes**. Los modernos NOS ocultan la complejidad de controlar múltiples protocolos y ofrecen en diferentes formas la capacidad de RPC (*Remote Procedure Call*), lo cual permite generar funciones que se ejecuten en el servidor.

- **Independencia de plataformas de hardware**. Los NOS deben ofrecer la capacidad de operar en más de una plataforma de hardware.

Los sistemas operativos de red más populares en un ambiente ethernet son: Novell Netware, LAN Server, LAN Manager y Windows NT. Los cuales fueron diseñados para un ámbito de un número pequeño de servidores. Existe una nueva generación de NOS que incrementa el número de servidores en la red.

Los principales contendientes en esta nueva era son OSF (*Open System Foundation*), DCE (*Distributed Computing Environment*), Novell Netware ver 4. x y Sun ONC (*Open Network Computing*) y las principales compañías que están presentes son: DIGITAL EQUIPMENT, HEWLETT-PACKARD, IBM, Microsoft y TANDEM.

GUI (Graphical User Interfaces)

Una vez que ya hemos conocido los elementos principales que conforman una red, nuestro siguiente paso es conocer la primera capa del modelo básico de una aplicación, que es la interfaz con el usuario, el objetivo de este tema es analizar el impacto en el uso de las interfaces gráficas y sus características más importantes.

[6] La replicación permite la rápida disponibilidad de los recursos y mantiene un "performance" adecuado a los requerimientos.

Tradicionalmente, entre más avanzadas eran las herramientas de programación, el desarrollo y el uso de aplicaciones se volvía más complicado, y la curva de aprendizaje para estos productos (herramientas de desarrollo y aplicaciones finales) era muy larga, lo cual ocasionaba altos costos de entrenamiento. La mayoría de estas aplicaciones se diseñaban pensando en un uso intensivo del teclado (ambientes de texto). La introducción de las interfaces gráficas permitió:

- ✓ **Programar aplicaciones más amigables**. En un ambiente gráfico el usuario simplemente selecciona la aplicación que desea ejecutar de un conjunto de imágenes gráficas o íconos, y dentro de una aplicación el usuario puede seleccionar opciones, o bien, introducir datos, a través del uso de menús tipo "pop-up", cajas de scroll, o por medio de "Radio Buttons" o "Check Buttons".

- ✓ **Establecer estándares en la programación de las aplicaciones**. Las operaciones básicas, como lo son el abrir y cerrar un archivo, se realizan de la misma forma en cualquier aplicación. Los usuarios que aprenden a utilizar una determina aplicación, en una ambiente gráfico, pueden rápidamente aprender a utilizar otra.

Dos interfaces gráficas que han sido muy populares para ambientes de computadoras personales son Windows de Microsoft y OS/2 Presentation Manager de IBM; y para las plataformas RISC y UNIX, las interfaces gráficas más comúnmente usadas son X-Windows, Open Look y Motif. Todos estos productos permiten a un usuario utilizar el *mouse* para seleccionar elementos gráficos en el monitor .

El concepto de las interfaces gráficas se basa en la premisa que los usuarios respondan e interactúen a través del uso imágenes gráficas, de una forma mucho más fácil y rápida que en un ambiente de texto (en donde sólo existen carácteres).

En general, las interfaces gráficas permiten al usuario explotar en forma más simple y rápida la información, incrementando su productividad. Las interfaces gráficas no sólo permiten la facilidad de aprendizaje y uso, sino también contar con una interfaz consistente en diferentes ambientes, reduciendo con esto el tiempo que es requerido para aprender a usar una aplicación.

Actualmente, se encuentran disponibles una gran cantidad de herramientas para el desarrollo de aplicaciones gráficas. Estas herramientas permiten a los programadores crear aplicaciones completas dentro de un ambiente gráfico sin tener que escribir una gran cantidad de líneas de código fuente, por lo que el tiempo de desarrollo es reducido.

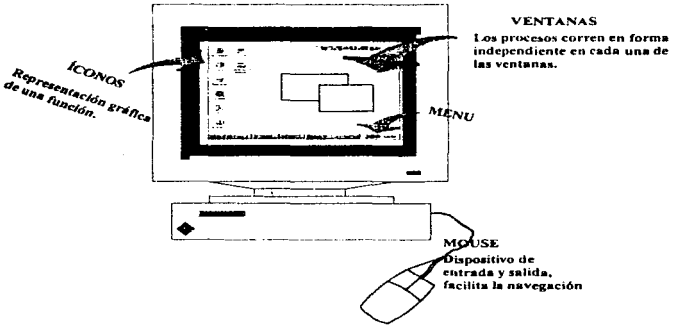
Típicamente una GUI despliega múltiples programas en ventanas independientes en la pantalla, los usuarios pueden mover estas ventanas, modificarles el tamaño y cambiarse fácilmente entre programas. Los programas tienen una interfaz con el usuario consistente.

Los programas que corren bajo una GUI usan un ambiente gráfico para toda la salida a la pantalla, esto permite desplegar texto en diferentes tipos y tamaños, así como integrar y manipular fácilmente texto y gráficas. El uso de gráficas resulta en un contenido de información más rico y denso de lo que es posible en modo carácter.

Los usuarios encuentran generalmente una interfaz gráfica fácil de aprender, debido a que las GUI establecen una consistencia en la interfaz con el usuario que permite llevar el conocimiento de una aplicación a otra. La facilidad de operación es cada vez más importante debido a que el uso de las computadoras se extiende a usuarios menos sofisticados. Por supuesto, los usuarios poderosos también se benefician, si las funciones básicas de un programa son fáciles de aprender, se puede usar más tiempo en aprender funciones más complicadas.

Con las GUI las aplicaciones son fáciles de aprender y al mismo tiempo muy poderosas, visualmente atractivas e interactivas, divertidas de usar, pero diseñadas para trabajo en serio.

INTERFAZ GRÁFICA



Las interfaces gráficas han tenido un papel relevante en la popularidad de los sistemas cliente/servidor, algunas consideraciones importantes al respecto son:

- Disminuyen el tiempo de entrenamiento de los usuarios, ya que son fáciles de usar y entender.
- Al encontrar una interfaz gráfica, los usuarios realizan su trabajo de una forma menos confusa y con mayor rapidez, por lo que se vuelven más productivos.
- Las interfaces gráficas incrementan la productividad, haciendo que el sistema sea más fácil de operar y ofreciendo capacidades como *multitasking* (multitarea) e intercambio de datos a través de las aplicaciones.
- El aprendizaje no es automático; debe existir un cierto grado de entrenamiento hacia los usuarios.
- La programación de interfaces gráficas puede consumir demasiado tiempo, sino se utilizan las herramientas existentes para su rápida programación.
- La carencia de consistencia en las interfaces gráficas pueden confundir, irritar y frustrar a los usuarios.
- Existe una guía de estilo definida para diferentes plataformas, la cual le da a la interfaz gráfica lo que se conoce como *Look and Feel*.
- En nuestra empresa debemos definir un estándar de interfaz, de tal forma que se garantice que la adquisición de una interfaz gráfica cumpla con estos estándares o bien los desarrollos propios de la empresa cumplan con ellos.
- Actualmente existen diferentes interfaces gráficas, todas ellas programadas con diferentes técnicas.

- Existen herramientas que nos permiten construir interfaces genéricas, las cuales son portables a una gran variedad de patrones de interfaces gráficas.
- Los usuarios pueden tener diferentes interfaces gráficas, como X-Windows y Microsoft Windows, las cuales operen en su computadora al mismo tiempo.
- Orientada hacia el uso extensivo de iconos.
- Tiene buena estética en pantalla: se ve bien y resulta agradable trabajar con ella.
- Ofrece elementos estándares, tales como menús, elementos de ventanas, controles de diálogos, etc.

Actualmente resulta difícil determinar dónde termina la interfaz con el usuario y dónde comienza a trabajar el sistema operativo, muchas personas confunden el sistema operativo con la interfaz, tal es el caso de Microsoft Windows, que incorpora las capacidades de multitasking y manejo de memoria virtual, lo cual confunde al usuario ya que no distingue si Windows corre bajo o por encima de DOS, la verdad es que DOS continua siendo el SO y Windows la interfaz gráfica[7] (para el caso de windows 3.1 y 3.11).

La nueva generación de sistemas operativos que está surgiendo incorpora la interfaz gráfica a su ambiente, ya no existe la división entre el sistema operativo y la interfaz gráfica, ejemplo de esto es Windows 95 y Windows NT. Estos nuevos sistemas operativos incorporan internamente el manejo de redes, multprocesamiento, multitarea, *multithread* (multi-hilos) y seguridad.

Existe la posibilidad de programar las interfaces gráficas a través del uso de las API[8] (*Application Program Interface*), o bien, con herramientas que nos facilitan esta labor, por ejemplo, una aplicación podría contener diferentes iconos y, dependiendo de lo que desee realizar, el usuario presionaría el ícono correspondiente. El programador deberá crear los íconos e incorporar el código asociado con dichos iconos, y también cuidar las ligas entre su aplicación y el resto del ambiente gráfico. Esto no es una tarea fácil por lo que la programación de una interfaz gráfica podría consumir demasiado tiempo.

Afortunadamente, el programador cuenta con herramientas que le permiten facilitar su programación, y no partir desde cero, herramientas tales como Power Builder, SQL*Windows, VisualBasic, y otras más. Estas herramientas permiten realizar desarrollos de una forma muy rápida y simple, si se deseará obtener un desempeño elevado sería necesario programar la interfaz gráfica a través del uso de las API y apoyándonos en lenguajes de tercera generación como lo es lenguaje C. En la metodología de programación de estas herramientas se utilizan técnicas como la **programación orientada a objetos** y la **programación orientada a eventos**.

La programación orientada a eventos es un tipo de programación que responde a los eventos, en donde un evento, puede ser el presionar un *botón* en la pantalla, presionar una tecla, o cualquier situación que cambie el estado de la aplicación. Este tipo de programación se diferencia de la convencional en que no se cuenta con una aplicación formada de instrucciones secuenciales (una instrucción sigue a la otra). Cada *objeto* en la pantalla posee sus propios *métodos*, los cuales son disparados cuando el *evento* asociado a éstos es detectado. Por lo tanto es imposible utilizar una programación convencional para el diseño de una interfaz gráfica.

[7] En el ambiente UNIX,UNIX es el SO y X Windows pertenece al ambiente gráfico.
[8] Las API son la interfaz por medio de la cual el programador trabaja, mientras que GUI es la interfaz con la cual trabaja el usuario. Son el método por medio del cual los desarrolladores referencian los servicios disponibles. Son un conjunto de funciones que el programador utiliza para accesar los servicios, ejemplos de estas funciones son: CreateWindow, Access Resourse, AdjustWindowRectEx, etc.

Como ya se menciono, para la asistencia en la programación de las interfaces gráficas existen en el mercado una gran cantidad de API y herramientas de programación disponibles, pero la incompatibilidad entre éstas representa un problema muy grave para una empresa que posee una amplia gama de equipos de cómputo y sistemas operativos. En muchas ocasiones, resulta más fácil hacer las aplicaciones portables para distintos sistemas operativos que para diferentes interfaces gráficas. Es por esto que se busca la estandarización de las interfaces, el beneficio de esta estandarización es que se tendrá un ambiente común, lo cual permitirá a los usuarios disminuir el tiempo de aprendizaje en el uso de una aplicación.

Los recursos de hardware que demanda una GUI son elevados debido a que se requiere de una gran cantidad de proceso, afortunadamente los adaptadores de video, coprocesadores gráficos y monitores están siendo mejorados constantemente.

BASES DE DATOS

Las computadoras son diseñadas para manipular información, pero para una computadora los datos son unicamente un conjunto de .bits. Nosotros somos quienes damos estructura y significado a los datos almacenados en una computadora. Los datos son accesados para su explotación por aplicaciones como hojas de cálculo, procesadores de palabras y bases de datos.

Para una empresa el volumen de los datos almacenados y la complejidad de su organización aumentan a pasos agigantados. El adecuado aprovechamiento de esta gran cantidad de información exige el desarrollo de herramientas que permitan su administración, transmisión, acceso a los datos desde localidades remotas, consistencia, y en general que mantengan la calidad de la información. Por ello es necesario un manejador de base de datos, lo cual es nuestro objeto de estudio en este tema.

Los sistemas de bases de datos comenzaron aparecer durante la década de 1960; en los siguientes veinte años sufrieron grandes transformaciones en sus conceptos y en su tecnología. La tecnología de las bases de datos se ha descrito como una de las áreas de las ciencias de la computación y la información de más rápido desarrollo.

La idea básica de la implementación de una base de datos es la de que los mismos datos deben ser aprovechados para tantas aplicaciones como sea posible. Anterior al surgimiento de las bases de datos, existía una sorprendente cantidad de datos duplicados o redundantes. Una gran cantidad de datos eran simultáneamente almacenados en varios volúmenes con distintas finalidades y con diferentes fechas de actualización. Con el manejo de la base de datos se pretende eliminar esta redundancia. La base de datos ha sido definida como una colección no redundante de grupos de datos, pero en realidad, en muchos casos se admite cierta redundancia con objeto de reducir los tiempos de acceso o simplificar los métodos de direccionamiento, algunos registros se duplican para facilitar la reconstrucción de la base de datos en caso de daños. De modo que es preferible hablar de una redundancia controlada o mínima en lugar de no redundancia como criterio de diseño.

Una de las características más importantes de la mayoría de las bases de datos es la de permitir la evolución y crecimiento de los datos, siendo esto transparente al usuario. La base de datos debe prestar una facilidad de reestructuración, siempre que haya que agregarle nuevos tipos de datos. La reestructuración no debe causar la reprogramación de las aplicaciones .

En general, un sistema de base de datos mantiene unidas (organizadas) piezas o listas de información, las cuales están relacionadas con nuestro trabajo, o bien, con nuestra vida cotidiana. Provee una forma de almacenar y mantener la información.

La primera computadora comercial fue únicamente una máquina dedicada al control, almacenamiento y manipulación de la información de un censo. En nuestros días, una de las razones más comunes para comprar un equipo de cómputo, es la necesidad de controlar la información por medio de una base de datos.

Un sistema de base de datos consta de dos partes:

1. El sistema de control de la base de datos (DBMS, *DataBase Managament System*), el cual se encarga de organizar la información, y

2. Las aplicaciones de la base de datos, los cuales son programas que permiten desplegar, insertar, actualizar y borrar la información almacenada en DBMS.

Es común que las aplicaciones y el DBMS se encuentren en la misma computadora. Actualmente, la tendencia es la distribución de las aplicaciones y del DBMS, esto incrementa el poder de procesamiento del DBMS, debido a que las aplicaciones corren en una o más estaciones de trabajo (en muchos casos computadoras personales), comunicándose a través de una red con uno o más DBMS, los cuales se encuentran corriendo en otras computadoras. Esta arquitectura la analizaremos más detalladamente en el siguiente capítulo al estudiar **sistemas distribuidos**.

Los siguientes temas tienen como objetivo definir los términos que describen una base de datos, métodos de almacenamiento y manejo de la información.

Qué es un DBMS

Un DBMS es un sistema de control de datos, el cual los organiza y mantiene. Algunos de los términos más usados en este ámbito son:

POBLACIÓN. Son el conjunto de datos que se almacenan en la base de datos. Se forma de cualquier grupo o clase de objetos que podamos definir.

REGISTRO. La población debe ser almacenada en forma de registros; un registro es un conjunto de atributos, los cuales describen a cada uno de los elementos dentro de la población.

CAMPO. Un registro es dividido en campos, los cuales son atributos propios de los elementos que almacenamos, por ejemplo, si queremos controlar la información referente a los empleados de una empresa, los atributos podrían ser el nombre del empleado, la dirección y el teléfono.

Para el almacenamiento de la información en el disco, el DBMS debe proveer de servicios de definición de registros y campos en la base de datos, como también de comandos de inserción, actualización, borrado y despliegue de la información almacenada.

El DBMS también es responsable de mantener la integridad de la base de datos, de nada nos sirve contar con los datos, almacenados en el DBMS, si no podemos confiar en la validez de ellos. El DBMS es responsable de proveer servicios de integridad para garantizar la consistencia de los datos; fallas, como la falta de energía eléctrica o daños en el disco, deben ser considerados para prever los mecanismos de recuperación. Además, si hablamos de sistemas multiusuario, el DBMS debe ser capaz de mantener el control de los accesos concurrentes a la información.

En resumen, un DBMS proporciona los siguientes servicios:

- **Definición de datos**. Provee un método de definición y almacenamiento de datos.
- **Mantenimiento de los datos**. La información debe ser almacenada por medio de registros y campos, lo cual facilita su control .
- **Manipulación de los datos**. Proporciona servicios al usuario para insertar, actualizar, borrar y ordenar los datos.
- **Desplegado de los datos**. Brinda servicios de despliegue en computadoras personales, terminales o impresoras.
- **Integridad de los datos**. Provee uno o más métodos que garantizan la veracidad de la información.

La forma en que se almacena y mantiene la información depende del tipo de modelo de base de datos que se tenga. Estos modelos tienen como objetivo definir la estructura de datos utilizada para el almacenamiento de la información. A continuación se explica en forma breve las principales características de éstos.

Modelos de bases de datos

Los modelos de bases datos que actualmente existen son: sistema de archivos, jerárquico, de red y relacional. Cada uno de estos modelos posee sus propios métodos de manejo de datos. La evolución natural de estos sistemas ha sido del sistema de archivos, pasando por el jerárquico, el de red y finalmente el relacional.

Sistema de archivos

Es el modelo más fácil de comprender y el único que describe la forma en que los datos son almacenados en el disco. Cada campo es almacenado en forma secuencial en un solo archivo de datos. Para localizar un elemento en específico, se deberá iniciar una búsqueda desde el principio del archivo y verificar cada uno de los datos hasta encontrar la información deseada. Este modelo se caracteriza por lo siguiente:

- Fue el primer método utilizado para el almacenamiento de datos en un ambiente de cómputo.
- Cuenta con una gran cantidad de desventajas: no existe una relación clara entre los datos (el programador, y en muchas ocasiones el usuario, debe conocer exactamente la forma en que los datos son almacenados para poder manipularlos), esto implica un amplio rango de errores, debido a que cada aplicación debe controlar el acceso a los datos.
- Este tipo de sistemas tiene problemas muy graves con respecto a la integridad de los datos, para mantenerla es necesario verificar los datos antes de ser almacenados[9].

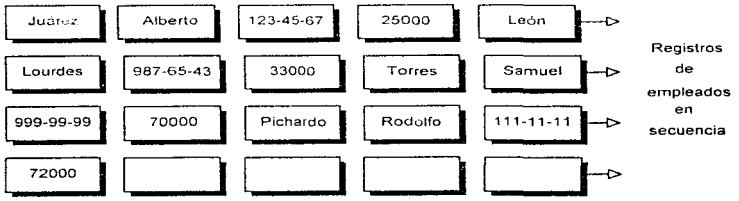
[9] Es necesario verificar que todas las relaciones existentes con el dato que se capturara existan previamente, por ejemplo, si se captura información referente al puesto que ocupa un empleado en cierta empresa, es necesario que dicho empleado exista antes de poder capturar información referente a su trabajo. Dependiendo del número de relaciones que contengan los datos, la verificación de la información podría llegar a tomar demasiado tiempo.

☞ No existen elementos para realizar búsquedas rápidas de información; toda búsqueda inicia desde el comienzo del archivo. Por lo anterior, para "ordenar" los datos será necesario leerlos completamente y reescribirlos.

☞ La mayor desventaja de un sistema de este tipo es que no permite en forma simple realizar cambios a la estructura de la base de datos. Por ejemplo, añadir un nuevo atributo implica generar un archivo temporal en donde se incluya y posteriormente borrar el archivo original, esto además significa invertir más tiempo, es un medio en el cual pueden generarse una gran cantidad de errores, y provocar probablemente con esto la corrupción de la base de datos.

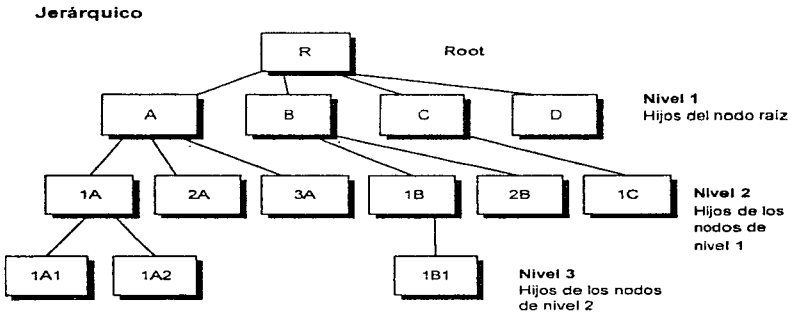
Las desventajas mencionadas y la necesidad de un mejor medio para describir las relaciones de uno-a-muchos, conducen a los desarrolladores al siguiente modelo de base de datos.

Sistema de archivos



Modelo jerárquico

En este modelo los datos son almacenados dentro de una estructura de árbol, que nace desde un *nodo* llamado raíz (*root*). Diferentes clases de datos se encuentran localizados en diferentes niveles . La mínima estructura de datos que es manejada en este modelo es llamado nodo, los últimos nodos de la estructura son llamados hojas.



Las principales características de este modelo son:

☞ Define una relación de padres a hijos.

☞ Una ventaja de este modelo es la capacidad de definir relaciones de uno a muchos, sin redundar en la información.

☞ Facilita y aumenta la velocidad de la búsqueda de información, ya que no es necesario que se busque en toda la base de datos la información, solamente es necesario seguir la ruta de la información requerida.

☞ Debe existir siempre un único nodo raíz, el cual pertenece a la DBMS.

☞ En este modelo, cada nodo puede tener un número ilimitado de hijos, pero solamente un *padre*, con esto se preserva la relación de uno-a-muchos.

☞ La estructura física de los datos no importa en este modelo, el DBMS almacena los datos como una lista ligada, con apuntadores que van del padre a los hijos, o bien, de los hijos a hijos, debido a esto resulta muy simple agregar un nuevo nodo a la estructura.

☞ Los cambios en la jerarquía de la estructura **implica rediseñar toda la estructura**.

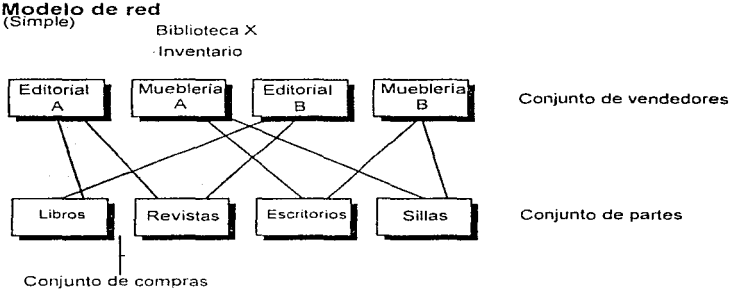
☞ No existe un método simple para la definición de relaciones de muchos-a-muchos.

La necesidad de las relaciones de muchos-a-muchos hace evolucionar este modelo hacia el modelo de red.

Modelo de red

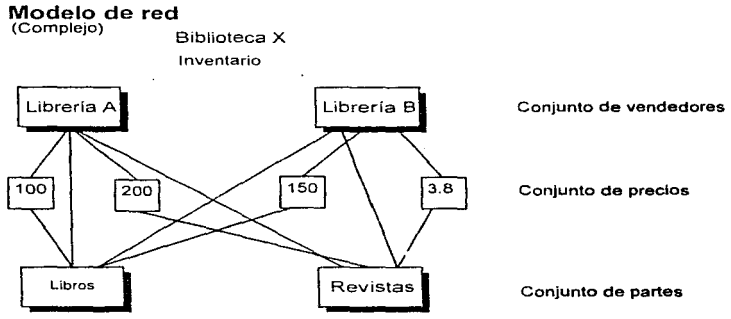
El nombre de red no tiene ninguna relación con el medio de comunicación por el cual se soporta la base de datos. El modelo de red describe una base de datos en la cual las relaciones de muchos a muchos existen (múltiples ligas de padres e hijos).

En este modelo las relaciones entre los datos son referenciadas como *conjuntos*, la estructura es mantenida por listas ligadas, lineales y circulares.



Las relaciones entre la información pueden llegar a ser extremadamente complicadas y difíciles de mapear, por lo que se requieren programas dedicados a mantener las rutas de accesos a los datos.

El modelo de red padece de los mismos problemas de estructura que el modelo anterior. Una vez que el diseño es configurado y aceptado resulta muy complicado realizar cambios, ya que se requerirá crear totalmente una nueva estructura.



Modelo relacional

El modelo relacional abandona el concepto de la relación padre e hijo, en vez de esto los datos son organizados en conjuntos matemáticos, a partir de una estructura tabular. En este modelo los campos son transformados a columnas en una tabla, y cada registro a un renglón en una *tabla*. Las relaciones entre las tablas son definidas a través de funciones matemáticas como **join, unión e intersección**.

Las principales características de este modelo son:

- ✓ Cuenta con una completa flexibilidad para la descripción de las relaciones entre los datos.
- ✓ El programador define la estructura de datos, creando tablas y decidiendo cuáles columnas de las tablas estarán relacionadas.
- ✓ El usuario puede efectuar consultas que relacionen columnas de una sola *tabla*, o bien, de diferentes *tablas*.
- ✓ Los cambios a la estructura son fáciles de realizar.
- ✓ Nuevas tablas pueden ser generadas, ya sea totalmente nuevas, o bien, crearse a partir de información previamente almacenada en otras tablas.
- ✓ No es necesario reconstruir toda la estructura, si se requieren cambios o adecuaciones al modelo.

✓ La información sobre la estructura de la base de datos (tablas, índices, vistas, sinónimos, etc.) se encuentra almacenada en forma de tablas; comúnmente, estas tablas son referenciadas como tablas de sistema o diccionario de la base de datos. El DBMS lasmaneja como cualquier otra tabla de datos, por lo que los usuarios pueden hacer uso de ellas para obtener información referente a la estructura de la base de datos.

✓ El principal objetivo del modelo relacional es preservar la integridad de los datos. El DBMS no debe permitir accesar la información por algún otro medio ajeno a la base de datos, por lo que el formato en que se almacena la información será propio del manejador de base de datos.

✓ El acceso a los datos no requiere del conocimiento de su ubicación en el disco o de la estructura de datos utilizada para su almacenamiento. El DBMS realizará las operaciones necesarias para la búsqueda y selección de la información.

✓ La desventaja de este modelo es su necesidad de una gran cantidad de procesamiento, lo que en ocasiones causa un *overhead* considerable. Actualmente, con el avance de la electrónica, la cual permite velocidad de más de 100mhz, este problema casi ha desaparecido.

INTEGRACIÓN DE APLICACIONES

Una vez que hemos conocido los principales componentes de una aplicación, resulta necesario mencionar los requerimientos actuales para la portabilidad e interoperabilidad de éstas. La tendencia actual es mantener un ambiente de trabajo heterogéneo en donde los usuarios y desarrolladores no dependan de un solo vendedor, y donde la diversidad de plataformas de hardware, sistemas operativos y herramientas de desarrollo hacen necesario el establecimiento de estándares para lograr la interoperabilidad de diferentes aplicaciones .

El siguiente tema tiene como objetivo el estudiar los principales requerimientos para el logro de la interoperabilidad.

Sistemas abiertos (Open Systems)

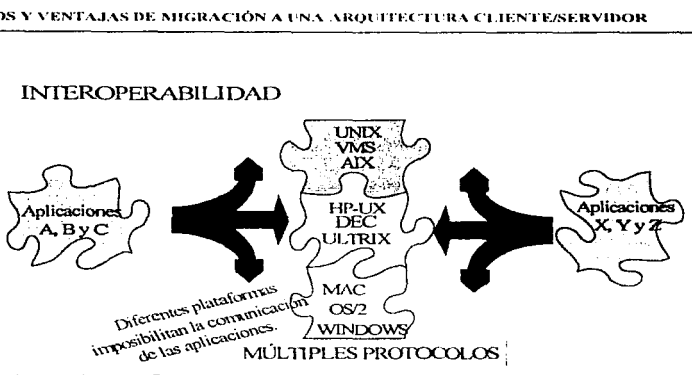
Una de las principales necesidades del mundo de la computación actual es la necesidad de interconectar equipo (hardware y software) de diferentes compañías y que éstos funcionen juntos dentro de un mismo ambiente. Algunas de las principales ventajas de esta posibilidad son: escalabilidad, reducción de costos y optimización de procesos. La clave para lograr esta interoperabilidad es la utilización de un **sistema abierto.**

Los sistemas abiertos permiten la integración de aplicaciones, el siguiente diagrama ilustra la imposibilidad de las aplicaciones A, B y C de comunicarse con las aplicaciones X, Y y Z; esto es debido a la incompatibilidad de hardware. Cuando el hardware y el software no pueden comunicarse se generan los siguientes resultados:

✗ Imposibilidad de acceso a la información

✗ Dificultades en la distribución de aplicaciones

✗ Dificultad en la administración del sistema.



Un sistema abierto (*Open System*) posee la capacidad para correr en una gran variedad de computadoras, así como la posibilidad de interconexión con otros equipos. Un sistema de este tipo cuenta con especificaciones basadas en estándares.

Existen dos tipos de estándares: De Facto Standard y De Jure Standard.

De Facto Standard

Es un estándar no oficial, pero la tendencia lo coloca como un patrón para seguirse.

De Jure Standard

Se encuentra soportado oficialmente por un cuerpo público y por leyes (pero no necesariamente se sigue).

De Jure Standard son creados por dependencias nacionales e internacionales y son producidos a través de procesos públicos en los cuales participan las empresas, las cuales realizan comentarios y diversas actividades. A menudo toma demasiado tiempo definir un estándar de esta manera. Ejemplo de esto son: el puerto de comunicación serial IEEE RS232 o la definición del lenguaje ANSI C.

De Facto Standard son creados como resultado de las tendencias del mercado cuando un producto basado en una nueva tecnología obtiene un dominio del mercado, debido a que es el primero en aparecer lo que le da una superioridad sobre sus rivales. En algunas ocasiones De Jure Standard están basados en un Facto Standard ya existente.

Compañías con innovaciones tecnológicas a menudo establecen precios y licencias con la especulación de que sus productos se tomen como base para un estándar. En otras ocasiones, se forman agrupaciones con el objetivo de tratar de obtener un estándar .

El uso de estándares permitirá:

✓ La portabilidad a través de una amplia gama de sistemas (mínimos cambios son permitidos).

✓ La interoperabilidad con otras aplicaciones en sistemas locales y remotos.

La necesidad de un sistema abierto es resultado en un 70% del requerimiento de correr una misma aplicación en diferentes plataformas, y en un 30% el evitar la dependencia hacia un solo vendedor.

Para hablar de un sistema abierto, éste deberá de contar con las siguientes características:

☞ **No tener un propietario.**

☞ **Interoperabilidad**. Los procesos distribuidos son la base de los sistemas de información remotos. Hardware y software deben trabajar conjuntamente para proveer soluciones.

☞ **Distribución**. El software debe encontrarse modularizado por lo que los componentes pueden distribuirse a través de la red a la ubicación donde más sentido tenga su existencia.

☞ **Escalabilidad**. La misma aplicación se encuentra disponible en todas las plataformas, esto da flexibilidad y reduce el mantenimiento y los costos de entrenamiento.

☞ **Portabilidad**. El software escrito para una plataforma de hardware en específico debe ser capaz de poder migrarse hacia una diferente plataforma con pequeñas modificaciones al código.

☞ **Extensible**. Las inversiones en software y hardware no deben perderse por el avance de la tecnología.

☞ **Disponibilidad**. Hardware y software estándares se encuentran disponibles por múltiples fuentes, esto tiene como consecuencia: selección, precios competitivos, portabilidad y sistemas de administración e interfaces de programación estándares.

☞ **Basados en estándares.**

☞ **Vendedores neutrales.**

Una pregunta que surge es la siguiente: ¿por qué los sistema abiertos son necesarios, si anteriormente no existían? En el pasado, las compañías realizaban un esfuerzo para crear un comité de vendedores, poco a poco se fueron juntando más de un solo vendedor con características comunes en sus interfaces y arquitecturas, por lo que no era posible que un producto que no cumpliera con las condiciones del grupo formara parte de éste. Esta situación no producía precios competitivos dentro del mercado, los usuarios se sentían bajo el control de un vendedor y un solo vendedor no siempre era capaz de proveer una aplicación específica, por lo que el usuario tenía que diseñarla. Esto ocasionaba gastos elevados en el desarrollo de software debido a que cada empresa tenía que contar con sus expertos de software para cada sistema en particular.

Cuando una determinada empresa compraba diferentes marcas de equipos tenía que cuidar la conectividad de éstos; en caso contrario, contaría con un sistema aislado por la incompatibilidad. En cualquier caso, para cada sistema por separado, los usuarios necesitan aprender a manejar nuevos tipos de interfaces y aplicaciones.

En un sistema abierto se contendrá hardware y software de diferentes vendedores, los cuales presentan interfaces comunes. Cuando se adquiere un nuevo producto se puede seleccionar de varios, con la misma funcionalidad, y éste puede ser integrado sin modificaciones dentro del actual sistema de información. Si el nuevo producto es un software de aplicación, éste podrá correr en cualquiera de las computadoras y todas las interfaces con el usuario pueden mantenerse sin cambio alguno.

De lo anterior, concluimos que los requerimientos para un sistema abierto son:

REQUERIMIENTO	AREAS
Estándares	• Portabilidad • Interoperabilidad • Escalabilidad
Extensión	• Integración total
Innovación tecnológica	• Administradores de sistemas • Administradores de redes • Internacionalidad • Seguridad
Puentes para los existentes sistemas	

Por lo que, para la existencia de un sistema abierto, es necesario el trabajo conjunto de diversas corporaciones.

Los estándares son requeridos para diversos productos de diferentes proveedores. Estos estándares pueden ser utilizados en la misma plataforma, lo que permite ver al sistema en forma global (total compatibilidad). Los estándares definen las interfaces entre los componentes de un sistema, a fin de lograr la portabilidad, interoperabilidad y escalabilidad. Algunos estándares existen únicamente con el objetivo de satisfacer esta necesidad.

Los sistemas abiertos benefician en diferentes formas:

✓ **Desarrolladores de software.** El desarrollo de software día con día incrementa su complejidad y la demanda por parte de los usuarios es mayor, dando como resultado aplicaciones con una complejidad creciente. En un sistema abierto los costos del desarrollo de software se abaten debido que se crean librerías y productos estándares.

✓ **Manufactura de hardware.** La gran motivación, por parte de los fabricantes de hardware, para emigrar a sistemas abiertos es el hecho de que los propios usuarios lo demandan. En un sistema abierto, las compañías pueden compartir costos de desarrollo y utilizar componentes preexistentes como partes que agregan valor a un producto determinado.

✓ **Usuarios.** Obtienen una mayor flexibilidad en sus sistemas y pueden integrar productos de diferentes vendedores, lo cual reduce los costos de adquisición, debido a que se estimula la competencia. Una ventaja de contar con estándares es que nos reduce el costo del personal encargado del desarrollo de software (interfaces, administradores de sistemas y programas de aplicación estándares).

Las ventajas de un sistema abierto se ven opacadas por las siguientes cuestiones (las cuales surgen de forma natural) :

✗ Rendimiento

¿Al estandarizar los procesos no se llegará a una reducción del rendimiento de una determinada aplicación?

✗ Costo

¿La migración hacia un sistema abierto significa una pérdida de capital y personal ?

✗ Progreso

¿Es posible que los estándares encierren tanto a los usuarios como a los distribuidores dentro de productos que rápidamente puedan llegar a la obsolescencia?

✗ Migración

¿El costo de migración hacia un sistema abierto no resulta ser demasiado elevado?

✗ Seguridad

¿En un sistema abierto se pierde la seguridad?

Las posibles contestaciones de los expertos en sistemas abiertos podrían ser:

☞ **Rendimiento**. Es verdad que en general los estándares en sistemas de computación no siempre resultan ser una solución óptima para ciertas aplicaciones. Cuando se compara un producto no-estándar existen ciertas condiciones que deben tomarse en cuenta, entre otras, considerar costos escondidos asociados con el equipo, cursos especializados de actualización e integración.

☞ **Costos**. Adoptar la filosofía de un sistema abierto no significa que una compañía pueda o deba desechar sus actuales arquitecturas. Una meta importante de los sistemas abiertos es lograr la interoperabilidad con las plataformas ya existentes.

☞ **Progreso**. Los estándares de calidad se fundamentan en la posibilidad de la innovación, es decir, que el estándar se desarrolle a la par de la tecnología.

☞ **Migración**. El precio competitivo da como resultado la posibilidad de múltiples fuentes para un determinado producto, esto permite que el costo de migración sea relativamente barato.

☞ **Escalabilidad**. Las plataformas de hardware, de muchas compañías, tienen una amplia variedad en rendimiento y características. Los usuarios demandan la posibilidad de elegir software para arquitecturas diferentes, tal como un procesador de palabras, el cual pueda correr en diferentes máquinas (distintos fabricantes y arquitecturas).

Las áreas de mayor importancia para el establecimiento de estándares para un ambiente cliente/servidor son:

Interfaces para los usuarios. Estandarizando las interfaces para los usuarios, se logra la mayor portabilidad. Los usuarios pueden navegar entre diferentes máquinas y aplicaciones en una forma más simple. Los principales patrones en esta rama son OSF Motif, Open Look, Next y Microsoft Windows.

Aplicaciones. Dentro del sistema de información de una compañía, la estandarización de las aplicaciones puede traer a la empresa una gran cantidad de beneficios y una reducción en los cursos de capacitación del personal. Si el software se encuentra disponible en el mercado para una gran variedad de plataformas, es entonces cuando un sistema abierto puede ser creado. La única desventaja es el hecho de contar con un sistema no compatible, ya que los costos de la portabilidad y soporte aumentan en gran medida (es posible que sobrepasen el precio de la aplicación).

Software. Una de las áreas de más reciente actividad en la estandarización la representa el software, que debe ser compatible a diferentes plataformas. Desafortunadamente, existen diversos factores que imposibilitan la estandarización: primero, la mayoría de las aplicaciones se encuentran referidas a arquitecturas que no forman parte de un estándar; segundo, muchos de los compiladores han sido desarrollados pensando en una arquitectura y tareas muy específicas.

Otras áreas en donde también es importante y necesario establecer estándares son: sistemas operativos, hardware e interconexión.

CONCLUSIONES

Actualmente, la necesidad de mayor capacidad de procesamiento está ocasionando que las compañías inviertan en sistemas de información más "poderosos", los cuales en muchas ocasiones involucran diferentes arquitecturas de hardware y software que, por lo general, no pueden ser cubiertas por un solo proveedor. Esta situación genera la necesidad de un ambiente multivendedor, es aquí donde comienza el paradigma de la arquitectura cliente/servidor.

El paradigma de la arquitectura cliente/servidor no podría ser posible sin la existencia de las siguientes tecnologías:

- Redes de computadoras.
- Interfaces gráficas.
- Bases de datos.

Pero, para lograr la interoperabilidad de los diferentes componentes del sistema de información, será necesario la existencia de estándares que definan los mecanismos y protocolos a seguir en la implatación de un sistema heterogéneo.

El prerrequisito para el éxito de una implementación cliente/servidor se basa, en gran medida, en el conocimiento de sus fundamentos, por lo que es importante estudiarlos y comprenderlos. En este capítulo estudiamos estos fundamentos, así como la importancia de los estándares que permiten implantar sistemas abiertos.

Como analizaremos más adelante, el concepto de la arquitectura cliente/servidor es muy simple, clientes que generan peticiones de servicio y servidores que responden. Debajo de la superficie, sin embargo, la computación cliente/servidor es enormemente compleja. Los desarrolladores (arquitectos, analistas, etc.) deben de navegar a través de un intrincado laberinto de difíciles decisiones. En los siguientes capítulos definiremos los conceptos necesarios para comprender este paradigma.

CAPÍTULO II
MODELOS DE CÓMPUTO

OBJETIVO

Analizar la evolución de las arquitecturas de cómputo.

INTRODUCCIÓN

El diseño de una aplicación debe prever la posibilidad de cambios en el ambiente de la empresa y tomar en cuenta la evolución del mismo sistema. Por lo que, en el diseño de una aplicación, deben cuidarse los siguientes aspectos:

- **Diseños que consideren el cambio**

  Debemos mantener la idea, cuando se realiza un desarrollo, de que la empresa a la cual se está diseñando la aplicación no es estática, sino que se encuentra en constante evolución. La forma en que el analista diseñe una aplicación tendrá un impacto directo en las futuras adaptaciones que requiera el sistema.

- **Diseño para una implementación flexible**

  Una aplicación debe ser instalada para funcionar adecuadamente, tanto para operaciones largas (pesadas) como cortas (simples). Se deben generar diseños con una apropiada capacidad de administración, por lo que se debe lograr un buen rendimiento en la ejecución de las operaciones y una facilidad en el mantenimiento del sistema para cualquier tipo de ambiente.

- **Diseños para una rápida incorporación de los avances al ambiente de trabajo de la empresa**

  El diseño debe permitir la incorporación de la funcionalidad del sistema lo más prontamente posible a las funciones que desempeña la empresa.

- **Diseño para la incorporación futura de nuevas tecnologías**

  Un uso apropiado debe hacerse con la nueva tecnología que emerge, de tal forma que se incorpore al diseño de la aplicación de una manera transparente, sin cambios significativos a las aplicaciones ya existentes.

Las empresas tienen un gran requerimiento de sistemas de cómputo, aplicaciones típicas como servicios a clientes, administración de contrataciones, nómina, requerimientos de intercambio de información (E-mail, conferencias, soporte, etc.) tienen que ser implementadas de una forma rápida y eficiente, el éxito del desarrollo de esta clase de aplicaciones dependerá en gran medida de la arquitectura[1] que se tenga, o bien, que se esté diseñando.

Por lo anterior, el objetivo de este capítulo es estudiar las principales arquitecturas de datos y procesamiento disponibles.

[1] Muchas de las barreras, que causan el fracaso de la implementación de una aplicación, se deben primordialmente a la utilización de una arquitectura de procesamiento débil. Por lo que una arquitectura poderosa colocará a una empresa en una posición ventajosa.

ARQUITECTURAS

La arquitectura define las directrices por las cuales las aplicaciones son diseñadas e implementadas. Los programas que componen una aplicación generalmente realizan tres funciones principales:

- **Servicios de presentación**. Forman la interfaz entre el programa y el medio externo.
- **Servicios de procesamiento/algoritmos**. Son los encargados de realizar las operaciones de la empresa.
- **Control de datos**. Se encargan del control, modificación, agregado y borrado de los registros almacenados.

No existe una teoría para la implementación de una arquitectura de procesamiento, pero en la práctica se han implantado diferentes modelos que analizaremos con objeto de distinguir sus atributos.

Procesamiento centralizado

En la arquitectura de procesamiento centralizado los recursos (interfaz con el usuario, capacidad de procesamiento y control de datos) se encuentran localizados en un solo nodo. En este modelo de cómputo, los usuarios se comunican hacia el procesador central por medio de terminales tontas, las cuales no tienen capacidad de procesamiento (*dump-terminal*) y el procesador central es el encargado de realizar las siguientes actividades:

☞ Control de la interfaz con el usuario

☞ Control de datos

☞ Ejecución de los procedimientos y funciones que definen las reglas del negocio

Los resultados generados son regresados al usuario para completar el ciclo de operación. Es importante notar que, en los últimos 15 años, el campo de la computación ha tenido avances en la capacidad de procesamiento y almacenamiento de datos, se ha mejorado el factor precio/desempeño y ha aumentado la relación geográfica de los usuarios con respecto a los recursos de cómputo, pero no se ha modificado en esencia el modelo de cómputo centralizado, la siguiente figura ilustra este hecho.

Ejemplos del procesamiento centralizado son los siguientes.

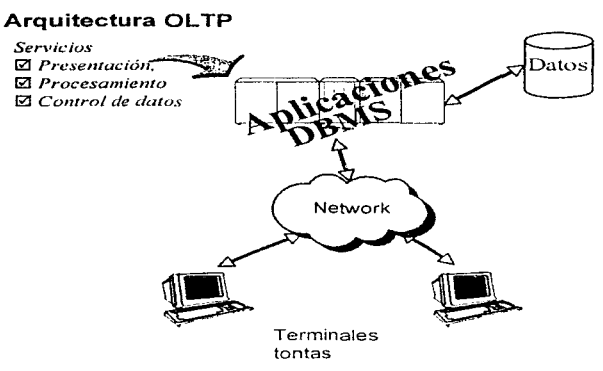
OLTP (ON LINE TRANSACTION PROCESSING)

Esta ha sido por muchos años la arquitectura utilizada por las empresas que requieren de una respuesta en línea, es decir, los resultados obtenidos de un proceso deben ser regresados al usuario (o a equipos de control) en fracción de minutos o segundos según sean los requerimientos de la aplicación. El procesamiento, control de la interfaz con el usuario y los datos se encuentran localizados en un procesador central, en muchas ocasiones la interfaz con el usuario es mínima, o bien, sólo existe un número muy reducido de usuarios que accesan al procesador. Los atributos de esta arquitectura son:

- **Uso de terminales tontas**. Estas terminales únicamente poseen la capacidad de despliegue y son utilizadas sólo para la conexión hacia el procesador central.

- **Uso de un DBMS para la manipulación de datos**. En muchas ocasiones la mayoría de los manejadores de base de datos utilizados son "propietarios", es decir, son específicos al equipo, o bien, a la "marca" del equipo; estas bases obtienen, en algunos casos, un alto grado de desempeño.

- **Uso de lenguajes de tercera y cuarta generación**. La programación es una combinación de un lenguaje de tercera y cuarta generación.



Arquitectura de procesamiento batch

Esta arquitectura es una derivación del procesamiento centralizado, con la diferencia de que no cuenta con terminales, es utilizada para "correr" programas muy largos en modo *batch*.

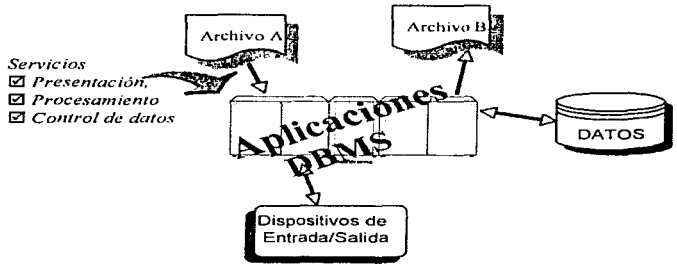
En este tipo de arquitectura los procesos son controlados por medio de un esquema de prioridades. Una prioridad elevada implica un proceso que corre en un tiempo más corto debido a que dispone de más tiempo de los recursos (memoria y CPU), a diferencia de un prioridad baja que son procesos con una duración de tiempo más larga ya que utilizan los recursos por menos tiempo. En esta clase de ambientes, los procesos se ejecutan periódicamente, o bien, en un tiempo específico.

Básicamente estos sistemas son utilizados para la ejecución de procesos muy largos, como procesamiento de imágenes, estadísticas, nóminas, etc. En general tareas que requieren de procesos matemáticos complejos o de un tiempo muy largo de procesamiento.

Sus principales atributos son:

- ✓ Por definición no es un ambiente interactivo. Debido a que es una arquitectura diseñada para la ejecución de procesos muy largos, la interacción con el usuario es mínima, la comunicación con las aplicaciones se realiza a través de archivos de entrada y los resultados son almacenados en archivos de salida.
- ✓ Uso de lenguajes de tercera generación.
- ✓ Uso de un DBMS.
- ✓ Utilización de un software de control de *JOBS*. Este software es el encargado de realizar la coordinación de los procesos, tiene a su cargo el control de las "colas" de batch.

Arquitectura HOST-BATCH



Arquitectura de tiempo compartido

Permite que múltiples usuarios (programadores y usuarios-finales) utilicen los recursos de cómputo simultáneamente. Estos usuarios se conectan al procesador a través del uso de terminales, las cuales se comunican al procesador central por medio de una red.

El sistema operativo que es utilizado en esta arquitectura deberá compartir los recursos entre los procesos, de tal forma que simule que cada proceso es dueño de los recursos, algunas de sus funciones principales son:

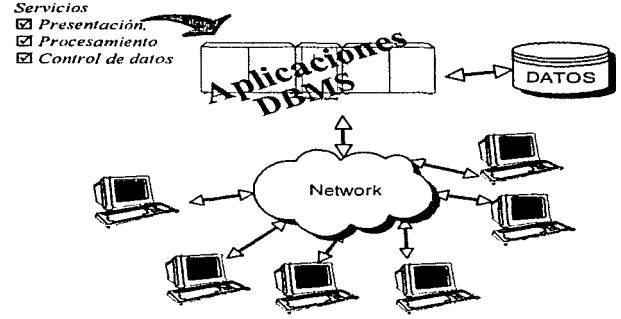
- Coordinar los accesos a la información, estableciendo para esto privilegios y derechos a los usuarios.
- Controlar las prioridades y capacidades de memoria asignadas a los usuarios.
- Mantener los procesos de red (monitoreo, levantamiento y desactivación de la red).

El DBMS que se utiliza en este tipo de arquitecturas debe garantizar la integridad de los datos y controlar el acceso a éstos mismos, ejemplos de estos *RDBMS* son: **ORACLE** y **Sybase**.

En general, en esta arquitectura se garantiza una respuesta adecuada a los usuarios, no tan rápida como la que se tendría en un sistema OLTP, pero si apropiada a los requerimientos.

El control de la interfaz con los usuarios, el procesamiento de información y la administración de los datos se mantienen coordinados por el procesador central, esto en muchas ocasiones produce una sobrecarga del sistema que se refleja en una degradación de la respuesta del sistema.

Arquitectura de tiempo compartido



Terminales tontas

Los atributos comunes de esta arquitectura son:

- Herramientas para usuarios finales. Son aplicaciones desarrolladas para la ejecución de las actividades de la empresa, corren bajo un ambiente de texto (no gráfico), lo cual permite su rápida ejecución, pero en ocasiones no son fáciles de manejar y comprender.
- El DBMS controla los datos. Los datos deben de residir en el nodo (máquina) local, en caso contrario la aplicación es responsable de localizar la ubicación de éstos.
- Se utilizan lenguajes de tercera, cuarta generación y de propósito específico. Las aplicaciones son monolíticas y se redunda en código en la programación de éstas.
- A menudo se encuentra basada en el uso de una minicomputadora. Existen algunos sistemas operativos que poseen la capacidad de creación de CLUSTER[2], lo que permite compartir discos, periféricos y mantener un administración centralizada de los diferentes procesadores; ejemplos de estos sistemas operativos son: **OPEN VMS** y **OSF/1**.

Arquitecturas mixtas

La necesidad de compartir información y distribuir la carga de proceso genera el surgimiento de arquitecturas mixtas, a continuación mencionamos en forma breve algunas de estas arquitecturas.

[2] El CLUSTER es un modelo que permite conectar varias máquinas (HOST) en donde una de éstas es la **maestra**, la cual contiene la lista de usuarios autorizados en el sistema, y las demás son **esclavas**. Los nodos **esclavos** inician su proceso de *boot* coordinado por el nodo **maestro**. Este tipo de modelo permite compartir espacio en disco, cpu y periféricos entre todos los usuarios participantes en el CLUSTER.

Arquitectura de emulación de terminales

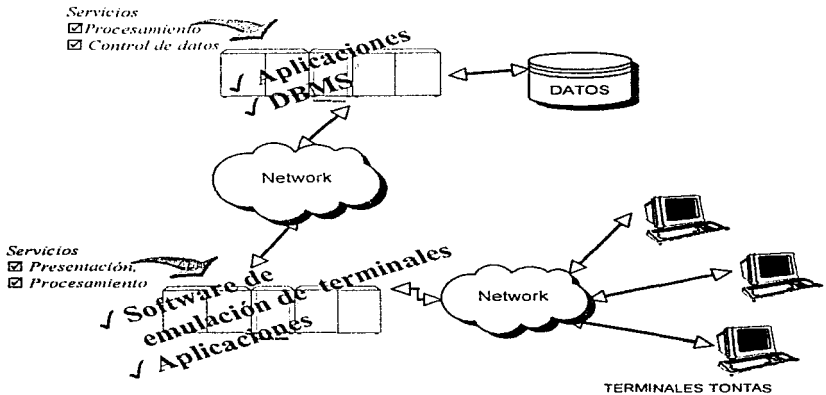
Este modelo de cómputo ha sido utilizado para que dos arquitecturas distintas trabajen conjuntamente, uno de los procesadores actúa como si se tratará de una terminal tonta. A menudo esta arquitectura es utilizada para que los procesadores departamentales se comuniquen con el *mainframe* en la ausencia de un protocolo *peer-to-peer*.

Un ejemplo clásico de esta arquitectura es el acceso a un sistema OLTP a través de la emulación de terminales por una arquitectura de tiempo compartido.

Las desventajas de este tipo de arquitecturas son:

- En caso de falla de las comunicaciones, los procesos de recuperación son complejos y tardados.
- Se genera una sobrecarga en el sistema debido a que el procesador da servicio a los procesos de red, usuarios y procesos locales.
- Complicados y deficientes esquemas de seguridad.

Arquitectura de Emulación de Terminales



Arquitectura de transferencia de archivos

Esta arquitectura es utilizada para permitir la transferencia de grandes volúmenes de información de un nodo a otro.

Un escenario típico es (ver siguiente figura) :

☞ En el procesador "A" se extraen los datos necesarios de la base de datos local.

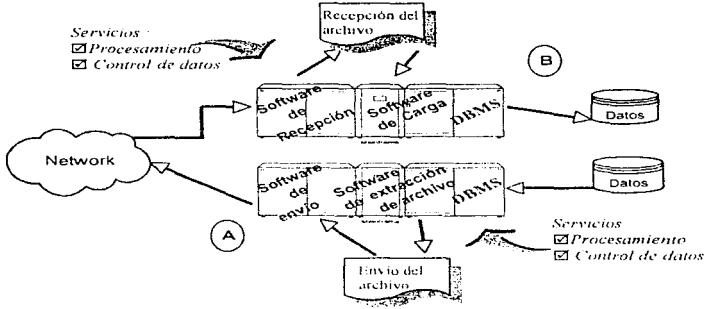
☞ El software de transferencia de archivos envía el archivo al procesador "B".

☞ El archivo recibido es cargado en la base de datos que se encuentra en el procesador "B".

Las desventajas de este modelo son:

✗ Un alto grado de duplicidad de información, debido a que la misma información existe en diferentes nodos.
✗ Inconsistencia en la información almacenada, porque no se cuenta con los procesos de "replicación" adecuados.

Arquitectura de Transferencia de Archivos



En los sistemas mencionados, las arquitecturas de datos que principalmente se han utilizadas son:

- **Sistema de archivos**. La información es almacenada en archivos planos. En el capitulo uno estudiamos las desventajas de esta forma de almacenamiento.
- **Bases de datos**. Estas bases son relacionales en su mayoría y se encuentran centralizadas.

Los sistemas anteriormente descritos son ideales para compartir porciones de información entre diferentes *hosts*, pero no permiten a los usuarios accesar datos que no se encuentran almacenados localmente. Los usuarios deben cambiar su conexión hacia un *host* distinto para accesar otra base de datos; para esto, se requiere de conocer en dónde se encuentra ubicada la base de datos por accesar. Esta forma de intercambio de información tiene serios obstáculos, los programadores y usuarios se enfrentan a problemas, tales como la duplicidad de datos y la falta de espacio en disco para almacenar información; también la sincronización de los procesos de actualización de los datos trae consigo otros problemas más complejos, como podrían ser la recuperación de la

información en caso de falla de la red. A través del **procesamiento distribuido**[3] es posible solucionar los problemas mencionados en las arquitecturas anteriores.

Procesamiento distribuido

El procesamiento distribuido dispersa una aplicación a través de múltiples procesadores (nodo), los cuales se encuentran conectados por medio de una red. En un ambiente distribuido, las aplicaciones son diseñadas con base en módulos, por lo que más de un nodo puede estar involucrado en la ejecución de un solo proceso, las principales características de este ambiente son:

☞ Dispersa la funcionalidad sobre la red.

☞ Posibilita un mejor uso de los recursos disponibles en la red.

Este tipo de procesamiento permite a los usuarios incrementar el acceso a los recursos que se encuentran distribuidos sobre la red, por ejemplo, servicios de archivos, bases de datos, etc.

En los sistemas de procesamiento distribuido, cuando un usuario realiza una petición de datos, si el nodo local determina que no cuenta con la información, se dispara un proceso que viaja a través de la red para conseguir la información en otro de los nodos. El proceso es totalmente transparente para el usuario, quien no sabe si la información se encuentra en el nodo local o en otro nodo de la red (es posible que cuando se realiza una petición de datos que no se encuentren en el nodo local se note una demora en la respuesta del sistema, pero esto es normal en un sistema distribuido).

En un sistema distribuido se abarcan los siguientes aspectos:

1. Funcionalidad

- ✓ Distintas funciones se encuentran en diferentes componentes, todo el código necesario para una función es escrito en un solo módulo.
- ✓ Una aplicación debe ser vista como una colección de módulos funcionales, y no como una secuencia de instrucciones individuales o como una colección de programas.
- ✓ Las aplicaciones distribuidas permiten una gran capacidad de crecimiento y flexibilidad.

2. Físicamente

- ✓ Los componentes de una aplicación, se encuentran localizados físicamente en diferentes sites.
- ✓ La distribución física de los componentes es posible únicamente donde exista un diseño modular de las funciones y datos.

[3] La decisión de implantar una aplicación distribuida dependerá de los resultados del análisis situacional de la empresa. La implementación de este tipo de sistemas puede en ocasiones, sino se realiza un buen análisis, sobrepasar el presupuesto y el tiempo esperado en su desarrollo.

3. Lógicamente

Las especificaciones de cada uno de los *host* que participan en la distribución se encuentran bajo un control diferente. Esto permite que los recursos estén distribuidos en plataformas con diferentes características técnicas.

4. Operacionalmente

Cada componente se encuentra bajo un control de operación separado. La instalación y control de los clientes están separados de la administración de los servidores.

5. Administración

En una sola aplicación, diferentes componentes pueden pertenecer a distintas organizaciones. En este nivel de distribución las aplicaciones son compartidas por más de una organización, lo que se conoce como *Enterprise Wide Distributed Integrated Systems*. La administración de este tipo de sistemas debe realizarse de una manera transparente, sin afectar las operaciones de los usuarios y sin importar las diferentes plataformas.

Los sistemas distribuidos arrojan beneficios en diferentes áreas como son:

☞ **PROCESAMIENTO**

- ✓ Trascienden las limitaciones de un solo CPU.
- ✓ Incrementan el poder de procesamiento del sistema.
- ✓ Optimizan el tráfico en la red.

☞ **DISPONIBILIDAD**

- ✓ Accesan a recursos que se encuentran físicamente distribuidos.
- ✓ Comparten información entre diferentes sites.
- ✓ Comparten funciones entre más de una aplicación.
- ✓ Extienden los sistemas existentes hacia nuevos usuarios.

☞ **DESARROLLO**

- ✓ Mejores estrategias. Los recursos y la información pueden ser instaladas donde se obtenga el mejor acceso para los clientes autorizados. Esto permite garantizar que la información "correcta" es recibida por las personas "correctas" en el menor tiempo posible.
- ✓ Agregan nuevas funciones a los sistemas existentes.
- ✓ Permiten el desarrollo en paralelo de pequeños módulos por medio de equipos de trabajo reducidos, o bien, por equipos de trabajo formados por personas de diferentes organizaciones.

☞ **ADMINISTRACIÓN**

- ✓ Proveen un rango de tolerancia a fallas.
- ✓ Facilitan el agregar nuevos módulos (funcionalidad).
- ✓ Permiten que las computadoras de escritorio tengan acceso a todo el sistema a través de la red.
- ✓ Escalabilidad. No existe una capacidad mínima, o una mitad o un máximo en un ambiente distribuido. El ambiente puede iniciar pequeño y crecer como sea necesario.

Una ventaja que sobresale de todas las mencionadas anteriormente y que es la principal propulsora hacia un ambiente distribuido es *el incremento en la competitividad.*

Un resultado de la distribución es el incremento en la competitividad. Esto se debe a que un sistema distribuido permite a una empresa :

✓ Reducir el costo en desarrollos, administración, mantenimiento y soporte, una vez que la tecnología ha sido totalmente dominada y comprendida.

✓ Incrementar la productividad de los usuarios.

✓ Optimizar la ubicación de los recursos.

✓ Mejorar el uso de los recursos (CPU, discos, etc).

✓ Incrementar la vida útil de los recursos existentes.

Un sistema distribuido puede llegar a tener un gran número de ventajas, pero es necesario cuidar los siguientes aspectos:

☞ La capacidad de transferir datos a diferentes nodos es una ventaja que puede verse disminuida por el problema de mantener un rendimiento óptimo.

☞ La ventaja de accesar y actualizar datos en sites remotos trae consigo problemas, como es el control de la concurrencia y de recuperación en caso de falla.

☞ Puede ser más costoso implementar un ambiente distribuido, debido a su grado de complejidad, así como también por el nivel de seguridad que se desee alcanzar.

☞ Un ambiente distribuido no es una tarea fácil de implementar debido a que puede requerirse de la interoperabilidad de múltiples protocolos y arquitecturas de redes, múltiples vendedores de hardware, y de software.

☞ En un ambiente distribuido los usuarios cuentan con estaciones de trabajo o computadoras personales, las cuales no resultan ser tan "seguras" como el ambiente de un sistema de tiempo compartido en el cual son utilizadas terminales tontas.

☞ En ocasiones, el problema principal de la implementación de un sistema distribuido es el convencer a los usuarios de compartir la información que controlan, es sabido que por lo general resulta más difícil lograr vencer las actitudes de resistencia al cambio que presentan las personas que la parte técnica del proyecto.

Es por esto que, sin una clara estrategia de implantación de un sistema distribuido, la complejidad del sistema puede sobrepasarnos fácilmente. Una empresa deberá tomar la decisión de implementar un ambiente de esta clase únicamente si se ha realizado de manera previa un análisis de factibilidad y las necesidades de la empresa así lo requieren.

Los programadores que desarrollen aplicaciones para un ambiente distribuido enfrentarán aspectos como los siguientes:

☞ **Identificación y localización de objetos**

Objetos como los archivos, servicios, y discos deben ser identificados por medio de un nombre único, el cual debe ser el mismo a través de todo el ambiente distribuido. Debe mantenerse un directorio global para la definición y ubicación de los objetos en la red.

☞ **Acceso global a los archivos**

Los archivos deben ser fácilmente accesados desde cualquier lugar de la red y el mismo nombre de archivo debe ser usado para cualquier proceso.

☞ **Seguridad global**

Mecanismos de seguridad, como identificación y autorización, deben operar consistentemente a través del ambiente distribuido.

☞ **Sicronización de tiempos**

Todas las máquinas en el ambiente distribuido deben tener una vista consistente del tiempo actual.

☞ **Paralelismo**

Los programadores deben ser capaces de ejecutar partes de una aplicación concurrentemente en múltiples procesadores con objeto de mejorar el desempeño de las aplicaciones.

☞ **Disponibilidad de los recursos**

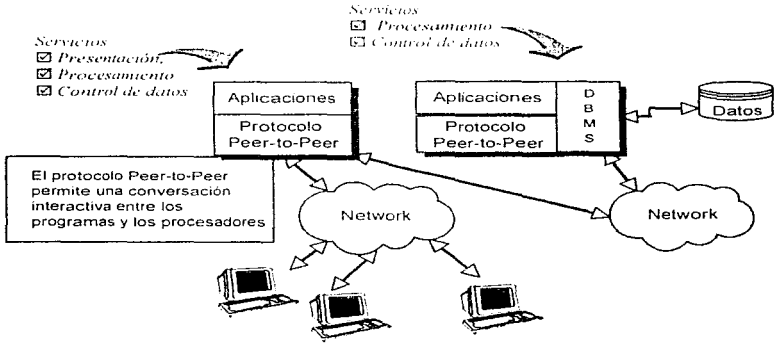
Debe ser posible accesar servicios, recursos y datos, incluso después de alguna falla en el sistema distribuido.

La evolución del sistema distribuido ha dado lugar a nuevas formas de arquitecturas, más complejas y que integran en una forma más íntima los recursos (datos y procesamiento), a continuación se explican estas nuevas formas de procesamiento distribuido.

Procesamiento cooperativo

En este tipo de procesamiento distribuido los elementos de una aplicación-presentación, procesamiento y datos-se encuentran distribuidos a través de diferentes sistemas y se caracterizan por un alto grado de integración entre ellos. Cada uno de esto sistemas "coopera" para alcanzar un objetivo común. Los protocolos cooperativos definen las reglas y la sintaxis para la interacción dinámica de las aplicaciones, ejemplos de estos protocolos son: LU 6. 2 y TCP/IP

Arquitectura de Procesamiento Cooperativo



Algunas de las técnicas utilizadas para el procesamiento cooperativo son:

Pipes. Esta técnica es muy utilizada en los sistemas con ambientes UNIX. Consiste en el paso de información de un procesador a otro por medio de un canal de conexión. Una analogía de esta forma de comunicación es la de un tubo en donde el agua fluye de un extremo a otro; en este caso, el agua son nuestros datos. Los procesos que se comunican puede ser que corran bajo diferentes sistemas operativos y los detalles del soporte a la comunicación es transparente al usuario.

Los protocolos utilizados en esta técnica son muy simples y tienen pocas restricciones en los formatos utilizados por los usuarios, básicamente proveen medios para indicar los límites de los mensajes, identificación del proceso que envía y verificación de la recepción de los mensajes. Un ejemplo de esta técnica es IBM con *Advanced Program to Program Communications (APPC).*

RPC (*Remote Procedure Call*) Es un mecanismo por medio del cual un proceso puede ejecutar otros procesos (subrutinas) que residen en un diferente sistema (por lo general en forma remota), el sistema operativo en el cual corren los procesos puede ser diferente, los detalles del proceso de comunicación son transparentes a los usuarios. El principal requerimiento para el éxito de una implementación RPC, es la habilidad de que el solicitante de la función localice el lugar donde se encuentra el proceso que se ejecutará. Una forma de controlar esto es creando una base de datos con la información correspondiente al nombre del proceso y servidor donde se localiza. Ejemplos de RPC son: *OSF RPC y IBM OS/2 Remote Procedure Link.*

SQL (Structured Query Language). Es una técnica con la cual se realizan peticiones de datos a un servidor. En este caso, el servidor es un servidor de base de datos relacional.

A diferencia de los PIPES y los RPC, los cuales son transparentes al usuario, SQL impone protocolos y formatos en la sintaxis que los usuarios deben utilizar. La desventaja de esta técnica es que sólo puede utilizarse en aplicaciones que involucren un RDBMS.

Procesamiento PEER-TO-PEER

Este tipo de procesamiento divide la aplicación en piezas, las cuales corren en procesadores separados conectados por medio de una red. Los componentes de procesamiento son considerados iguales y pueden intercambiar roles, un cliente puede convertirse en servidor y un servidor en cliente, por lo que no existe la distinción entre clientes y servidores. El procesamiento puede realizarse en cualquier parte en donde los recursos de cómputo (CPU, memoria y periféricos) se encuentren disponibles.

Una meta del ambiente de procesamiento peer-to-peer es el soportar bases de datos distribuidas, en donde los usuarios accesan información de bases de datos heterogéneas que se encuentran en diferentes plataformas (hardware y software).

Independientemente de la arquitectura de procesamiento distribuido a utilizar, el control de los datos es una actividad crítica, la cual (la mayoría de los casos) se realiza por medio de un manejador de base de datos, por lo general relacional (RDBMS[4]). Para un ambiente distribuido el RDBMS debe ser capaz de realizar, de entre otras, las siguientes funciones:

✓ Controlar la seguridad.

✓ Administrar a los usuarios.

✓ Coordinar los procesos concurrentes.

✓ Recuperar las transacciones en caso de falla.

Por lo que es muy importante el definir y comprender el funcionamiento y características de una base de datos distribuida.

Bases de datos distribuidas

El procesamiento distribuido puede involucrar diversas bases de datos ubicadas en diferentes zonas. Pero esto no implica un ambiente de bases de datos distribuidas, para que un sistema pueda ser catalogado de esta forma se requiere que la ubicación de los datos sea transparente al usuario, en muchas ocasiones las empresas dicen contar con un ambiente distribuido pero en realidad sólo cuentan con datos que se encuentran localizados en diferentes bases de datos, ya sea en forma local o bien en diferentes zonas geográficas.

[4] Relational Database Management System.

Para considerar una base de datos distribuida deben cumplirse los siguientes requisitos:

☞ Localización transparente.

☞ Replicación transparente.

☞ Fragmentación transparente.

☞ Ilusión de la existencia de una sola base de datos, por lo cual cada site posee todo lo que requiere para ser autónomo (ilusión de ser dueño de la información).

En un ambiente de base de datos distribuida se debe contar con elementos que garantizen que los datos se encuentren en un estado consistente antes y después de ser realizada una transacción a través de la red, así como en caso de falla del hardware o software. Un proceso que mantiene esta consistencia es el TWO-PHASE COMMiT.

Requerimientos para una base de datos distribuida

☞ ***Localización transparente***

Los usuarios y programadores no necesitan conocer la ubicación de los datos. La base de datos se encarga de mantener las rutas de acceso a la información; por lo tanto, no existe ninguna diferencia si la *tabla* se encuentra local o remotamente.

Si los datos no se encuentran en la máquina local, el sistema puede actuar de la siguiente manera: Se puede realizar el proceso en el site remoto y enviar los resultados, a través de la red, a la máquina local, o bien, se pueden transferir los datos a la máquina local para su procesamiento (es posible optar por una combinación de estas dos opciones).

☞ ***Replicación transparente***

Una replicación de datos significa que un mismo *objeto* (de la base de datos) puede ser almacenado en diferentes sites. Esto aumenta el "rendimiento" y la "disponibilidad". El rendimiento es maximizado debido a que los datos se encuentran localmente a los usuarios y la disponibilidad se incrementa debido a que si falla la red, o bien un nodo es dado de baja, el usuario puede continuar accesando los datos requeridos del site remoto.

La replicación transparente significa que todos los procesos referidos con la localización y mantenimiento de las réplicas son soportadas por el sistema y no por el usuario.

Debido a que los accesos se efectúan de manera directamente de la copia de los datos que se encuentran remotamente, la replicación puede ocasionar problemas debido a que las actualizaciones de la información deben realizarse en todas sus copias; como consecuencia, es necesario contar con los medios adecuados para mantener la consistencia de los datos en caso de que el site o la red fallen.

☞ ***Fragmentación transparente***

Los datos pueden ser fragmentados[5] (los fragmentos son porciones de información), por lo que la base de datos relacional puede ser dividida en pequeños segmentos, cada uno de los cuales son tratados como datos separados que podrían ser almacenados en distintos nodos en la red.

Los datos son presentados al usuario como una "vista" de los fragmentos combinados, y utilizando el modelo relacional pueden ser fragmentados[6] y distribuidos de diferentes formas.

Otros trequerimientos.

☞ ***Transparencia en el uso de la red***

Implica que el usuario no debe percatarse de la existencia de una red, ni mucho menos de la existencia de múltiples protocolos.

☞ ***Transparencia en el rendimiento***

Existirá un optimizador que se encargará de la determinación del plan más óptimo para la ejecución de los comandos.

Optimizar un *query* es un factor crítico para el rendimiento, un mismo query puede tomar segundos o bien horas dependiendo de la forma en que es ejecutado.

☞ ***Transparencia en las transacciones***

Implica que un usuario puede ejecutar un proceso que actualice los datos en cualquier número de sites y la operación será aceptada (**commit**) o será abortada (**rollback**) en "todos" los nodos, no existiendo estados intermedios.

Esta función es realizada a través de los procedimientos de TWO-PHASE COMMIT.

Protocolo two-phase commit

El objetivo del protocolo two-phase commit es garantizar que los participantes en una transacción "se dirijan hacia el mismo sentido", es decir, todos acepten un commit o ejecuten un rollback. Two-phase es un proceso complejo, expondremos únicamente su concepto.

Como su nombre lo especifica implica dos fases, a continuación se dará una breve descripción de éstas, será una explicación genérica debido a que cada vendedor define su propia forma de ejecutarlas.

[5] El único modelo de base de datos que permite esta característica es el relacional.
[6] La información puede ser fragmentada en forma vertical, horizontalmente, o en una combinación

Un componente llamado el *coordinador* es el encargado de controlar estas dos fases. Las acciones inician cuando el *coordinador* recibe el "reconocimiento" de actualización.

Primera fase

Primeramente, el *coordinador* escribe un registro en el archivo de *LOG*, de inicio de operación de commit, posteriormente envía un mensaje de "preparación" a las computadoras, participantes en la transacción, en la red. El mensaje coloca a los sites en un estado de commit , o bien, de rollback. Cuando un participante recibe el mensaje de "preparación" éste determina si puede o no realizar commit de las transacciones realizadas. Si es posible realizar el commit; se escribe un registro de "listo" en el LOG y se envía un mensaje de *OK* hacia el coordinador. Sino se puede; se enviará un mensaje de *NOT OK*. El proceso es automático y es realizado por el software instalado en cada site y no por los programadores, operadores o administradores de la base de datos.

Segunda fase

Después de que el coordinador ha recibido el mensaje de los participantes, decidirá realizar el aceptar (*COMMIT*) o el abortar la operación (*ROLLBACK*). Si alguno de los participantes envía el mensaje de *NOT OK*, el coordinador abortará las transacciones por completo enviando una señal de rollback a todos los participantes, en esta condición los participantes utilizarán sus archivos de LOG para restaurar los datos a su forma anterior. Si todos los participantes envían una señal de OK, el coordinador enviará el mensaje de *COMMIT*, y en este caso los participantes realizarán un commit en las transacciones realizadas.

En ocasiones, para evitar el proceso de TWO-PHASE COMMIT, se opta por realizar las actualizaciones en dos pasos. El primer paso consiste actualizar solamente un site "primario" con las transacciones realizadas en línea, los demás sites serán actualizados por medio de procesos en *batch*, que correrán generalmente por las noches o en horas de poca actividad. Por lo que a la mañana siguiente los sites se encontraran sincronizados. Para muchos tipos de aplicaciones es un proceso que es permitido, evitando el problema que los sites participantes en una transacción, no se encuentren disponibles simultaneamente.

Reglas de un ambiente distribuido

De acuerdo con nuestra experiencia y con lo descrito anteriormente, se definirán algunas de las reglas que un ambiente de bases de datos distribuidas debe cumplir:

- **Autonomía**

  Cada site en el ambiente distribuido deberá ser independiente uno del otro.

- **Evitar la existencia de un site central**

  Las operaciones en un ambiente distribuido deben evitar la existencia de sites críticos, debido a que si éstos fallarán se suspenderían las operaciones en los nodos dependientes.

- **Operación continua**

  Una base de datos distribuida nunca debe requerir ser "bajada", debe ser capaz de mantener una operación constante. Deben existir comandos que ejecuten respaldos en línea (*HOT BACKUP*) y recuperación en caso de fallas..

- **Localización e independencia transparente**

  Los usuarios no necesitan conocer la ubicación de los datos, los cuales simulan su existencia local, en caso de ser remotos.

- **Fragmentación**

  Las tablas pueden ser divididas en fragmentos que son almacenados en diferentes ubicaciones, siendo ésto transparente al usuario.

- **Replicación**

  Los datos pueden ser replicados en las computadoras conectadas a la red.

- **Independencia de la red**

  La base de datos distribuida debe ser capaz de operar con una gran variedad de protocolos de red y topologías.

- **Capacidad de procesamiento de consultas distribuidas**

  El rendimiento de los *queries* deberá ser independiente del site en el cual se ejecuten. Esto permite mantener un balance en la carga de trabajo de los nodos, evitando la existencia de nodos saturados, mientras que otros no registren proceso.

- **Administrador de transacciones distribuidas**

  El sistema debe soportar transacciones "atómicas" (la atomicidad implica que la transacción es ejecutada como un todo), ésto se refiere a la capacidad de realizar commit o rollback a una transacción, no permitiendo la existencia de estados intermedios.

- **Independencia del hardware**

  Los datos deberán ser accesibles desde cualquier tipo de plataforma, o por lo menos, de una gran variedad de hardware. Por lo que la implementación del sistema deberá regirse por estándares de comunicación y procesamiento (sistemas abiertos).

- **Independencia del sistema operativo**

  La base de datos distribuida debe ser capaz de operar con distintos sistemas operativos.

- **Independencia en el manejador de la base de datos**

  Un sistema distribuido ideal debe soportar una interoperabilidad entre diferentes tipos de DBMS; éstos corriendo en diferentes nodos y sistemas operativos.

Uno de los principales problemas que surgen en un ambiente de bases de datos distribuidas, lo representa el diseño de estrategias que garanticen:

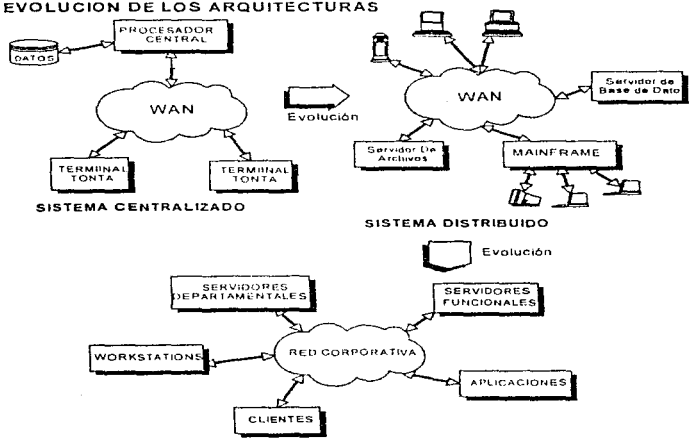
- Respaldos
- Recuperación en caso de falla del sistema
- Cambio en la administración y en las versiones del producto
- Creación y borrado de los objetos en la base de datos

También es necesario considerar que la seguridad (acceso a los recursos, entrada al sistema, etc.) es más complicada de controlar en un ambiente distribuido que en uno centralizado.

La migración a un sistema distribuido no es una tarea fácil debido a aspectos como:

- El tiempo disponible de migración, por lo general es muy crítico.
- Demasiado riesgo.
- Procesos rígidos, si la empresa está soportada por una estructura rígida en la programación de sus aplicaciones, entonces los cambios en sus aplicaciones significará, generalmente, cambios en sus políticas.

EVOLUCION DE LOS ARQUITECTURAS



Arquitectura multicapas

Actualmente, las empresas han iniciado la migración hacia ambientes distribuidos soportándose de un modelo de múltiples capas (multitiered), el cual tiene las siguientes ventajas:

✓ Mantener, en lo posible, las inversiones existentes en software y hardware.

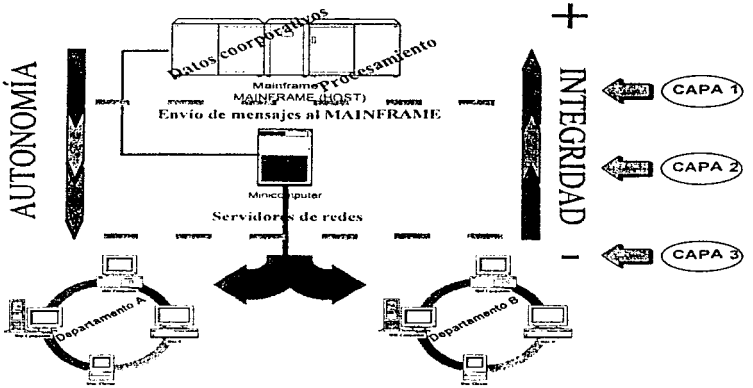
✓ Integrar la información que se encuentra en diferentes plataformas.

✓ Facilitar la evolución de la organización.

✓ Compartir procesos e información a través de los diferentes componentes de la empresa.

✓ Evitar la redundancia en los procesos.

ARQUITECTURA MULTI-CAPAS



Un modelo básico de esta arquitectura se encuentra formada por tres capas. En la capa superior, se ubicará el equipo con mayor capacidad (mainframe), el cual anteriormente conformaba el sistema centralizado. Esta capa tiene a su cargo:

☞ Llevar acabo los procesos que requieren de mayor capacidad de recursos (memoria, espacio en disco, velocidad).

☞ Almacenar la información corporativa.

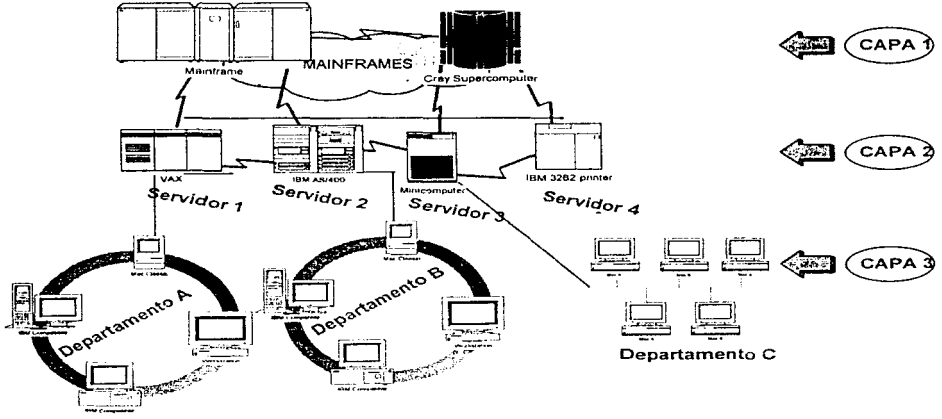
La segunda capa está formada por servidores de redes, que tienen a su cargo:

- Resolver las peticiones de las PC y estaciones de trabajo, que están en la tercera capa.
- Realizar peticiones de datos o procesamiento hacia el mainframe.

La tercera capa de este modelo se encuentra formada por redes de área local, cuyos servidores se encuentran en la segunda capa, tiene a su cargo compartir periféricos y ser la interfaz con los usuarios.

Esta arquitectura, es posible que crezca horizontalmente, agregando mainframes en la capa superior, servidores de red en la segunda y redes de área local en la tercera capa. El resultado es un sistema más complejo donde se incrementan los problemas de rendimiento, integridad y seguridad, pero por otra parte se tendrá un sistema escalable sin la necesidad de reprogramar las aplicaciones y/o reconstruir la red, lo que permite que la empresa se extienda fácilmente.

ARQUITECTURA MULTICAPAS CON CRECIMIENTO HORIZONTAL



El modelo multicapas se presenta como una opción razonablemente flexible para ser implantada. En principio es una aplicación distribuida en forma cooperativa. En la práctica su implementación dependerá de las respuestas que se den a las siguientes preguntas:

- ¿En qué capa (o capas) deben colocarse los datos?
- ¿En qué capa (o capas) deben colocarse las aplicaciones del negocio?
- ¿En qué capa (o capas) deben colocarse la interfaz con los usuarios?

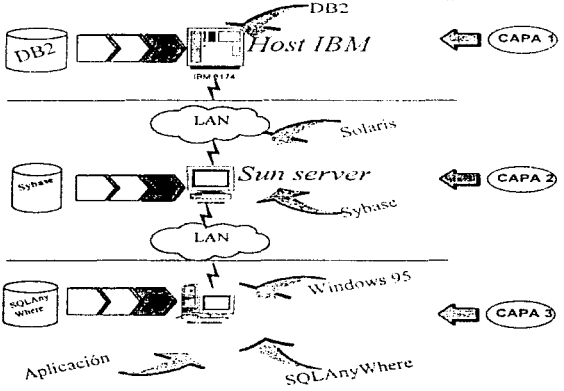
A continuación explicaremos en qué capa se deben colocar estos componentes de una aplicación.

En un ambiente centralizado estos tres elementos residen en un solo nodo (en el mismo sistema) y se encuentran en un sólo programa ejecutable. Para un ambiente distribuido, en el cual se utiliza la arquitectura de multicapas no existe un lugar único para cada uno de estos componentes, algunas recomendaciones son:

- En general la interfaz con el usuario es colocada en el cliente, los cuales son típicamente colocados en la capa más baja (PC y workstation).
- Debido a que los clientes cuentan con capacidad de procesamiento, es posible colocar en éstos parte de una aplicación.
- Si los datos que accesan los clientes, son casi estáticos, o bien sólo tiene pocas actualizaciones durante un período de tiempo considerable, entonces parte de los datos pueden ser colocados en los propios clientes (copias de información).
- Los datos coorporativos, los cuales se comparten entre grupos de trabajo, deben residir en los servidores.
- Los datos que muy frecuentemente son actualizados, agregados o borrados deben residir en la capa superior, en el mainframe.

Lo anterior hace posible observar que cada capa puede poseer su propia base de datos, por lo que los programadores se enfrentan al problema de acceso a bases de datos heterogéneas. Debemos recordar que los datos coorporativos realmente existirán en los mainframes, y de ahí se replicarán hacia los servidores o hacia los clientes, o inversa, de los clientes a los servidores y de los servidores a el mainframe, no importa la dirección de la replicación, pero si el hecho que la base de datos del mainframe debe siempre mantener un estado consistente con respecto a todas las capas. La siguiente figura muestra este hecho.

Multi-Niveles con acceso a bases de datos heterogéneas



Para ejemplificar los problemas potenciales de una distribución de datos en un sistema distribuido de tres capas. Consideremos el siguiente escenario.

Una compañía de seguros de vida organizará su sistema de información bajo un ambiente distribuido de tres capas, de la siguiente manera:

- Las oficinas centrales mantienen la información financiera y libros de control de la empresa.
- Los centros de servicios regionales se encargan del control de los servicios al cliente de una región en especifico.
- Los agentes de seguros son quienes le dan servicio a los clientes.

La compañía decidió equipar a los agentes con computadoras portátiles, capaces de correr diferentes aplicaciones de seguros (registro, actualizaciones, etc.) y periódicamente los agentes deben conectarse a los servidores regionales para cargar sus computadoras con nueva información (nuevos clientes, cambios en las políticas de la empresa, etc.), la cual se obtiene de los servidores regionales y de las oficinas centrales.

Podemos estructurar esta arquitectura en tres capas, en donde las *notebooks* de los agentes forman la tercera capa, los servidores departamentales la segunda capa y el mainframe de las oficinas centrales la primera capa. Para obtener éxito en la implantación de esta arquitectura será necesario resolver los siguientes puntos:

- Los agentes llevan sus notebooks con los clientes, por lo que es necesario que en la notebook exista la información referente al cliente, esto es, una parte de la información debe residir en las computadoras portátiles, las cuales tienen instalado Windows 95 y SQL AnyWhere.
- Los servidores regionales mantienen la información referente a los agentes asignados a una región en particular, también se almacena información necesaria para la toma de decisiones por parte de los directores de las regiones. En el servidor se cuenta con UNIX como sistema operativo y Sybase como manejador de base de datos.
- En las oficinas centrales se almacenan los datos de los libros coorporativos y la información completa respecto a los clientes. El mainframe es IBM con sistema operativo MVS y como base de datos DB2.
- Las actualizaciones que los agentes realicen a la información; que se encuentra en su notebook, debe propagarse hacia los servidores departamentales y hacia las oficinas centrales. Estas transacciones deben cuidar su integridad, en caso de falla deberá realizarse un *rollback* en las tres capas.
- Cada DBMS, en esta arquitectura, utiliza sus propio lenguaje de manipulación de datos.

Este ejemplo muestra algunos de los problemas a los cuales debemos enfrentarnos al diseñar un sistema distribuido. En el siguiente capítulo estudiaremos cómo la arquitectura cliente/servidor puede ayudar en la implantación de esta clase de sistemas.

CONCLUSIONES

La arquitectura de cómputo define las directrices por las cuales las aplicaciones son diseñadas e implementadas. La utilización de una arquitectura de procesamiento débil predispone el fracaso de las aplicaciones, es decir, las capacidades de facilidad de mantenimiento, extensibilidad, escalabilidad, flexibilidad e interoperabilidad de las aplicaciones serán muy difíciles, o tal vez, imposibles de realizar, debido a que las aplicaciones heredan las características de la arquitectura de procesamiento en la que son construidas.

Las arquitecturas han evolucionado de acuerdo con la necesidad de procesamiento de las empresas; actualmente, el ambiente de las empresas se encuentra en constante cambio, por ejemplo, la globalización. Por lo que las empresas requieren ser más competitivas y más dinámicas. El tiempo de comercialización, toma de decisiones y obtención de beneficios son las claves para alcanzar el éxito, ya que que la disponibilidad de datos y la capacidad de procesamiento deben encontrarse al alcance de los usuarios lo más rápidamente posible. Las arquitecturas de cómputo centralizadas ya no son suficientes para cubrir estas necesidades de procesamiento, por lo cual la migración a sistemas distribuidos (o diseño de nuevos sistemas distribuidos) resulta ser una necesidad para las empresas que desean mantener o aumentar su grado de competitividad.

La filosofía de la arquitectura cliente/servidor no es la de reemplazar los mainframes con PC, sino de distribuir el procesamiento de tal forma que un proceso es resuelto por una red de computadoras y no por un procesador central. Esto permite, por una parte, obtener el máximo beneficio de cada una de las plataformas de hardware y software presentes en una empresa y, por otra, mantener una capacidad de procesamiento acorde con las necesidades de la misma, esto es cuando se requiera de una mayor capacidad de procesamiento bastará con agregar más computadoras (escalabilidad horizontal) a la infraestructura informática existente.

Sin una clara estrategia de implantación de un sistema distribuido, la complejidad del sistema puede sobrepasarnos fácilmente. Una empresa deberá tomar la decisión de implantar un ambiente distribuido únicamente si previamente ha realizado un análisis de factibilidad y las necesidades de la empresa lo requieren.

Arquitectura Cliente/Servidor

CAPÍTULO III
AMBIENTE CLIENTE/SERVIDOR

OBJETIVO

Definir la arquitectura cliente/servidor.

INTRODUCCIÓN

Dos décadas atrás, el procesamiento de datos se basaba en el uso de poderosos mainframes, el acceso a éstos se llevaba a cabo por medio de rudimentarias terminales que sólo permitían la comunicación con el procesador central. Los métodos para el procesamiento de la información resultaban ser arduos y complejos debido a la centralización de los recursos y a las hostiles interfaces de usuarios. Así que cuando aparecieron las primeras computadoras personales, cerca de los años ochenta, su crecimiento fue inmediato debido a su facilidad al acceso de los datos, así como por sus interfaces amigables.

Sin embargo, las PC siguieron siendo incapaces de manejar el procesamiento de datos necesarios para las grandes empresas. Las redes de área local (LAN, *Local Area Network*) dieron una solución a este problema interconectando PC y mainframe. Esto es, las PC son utilizadas como terminales inteligentes (amigables), las cuales facilitan la conexión del usuario con el procesador central (MAINFRAME).

Actualmente, tanto las PC como las redes de área local son usadas en casi la totalidad de las empresas; las PC permiten contar con una variedad de procesadores de texto, hojas de cálculo y software de base de datos, mientras que el uso de redes de área local permite compartir archivos y periféricos (impresoras, modems, unidades de respaldo, etc.). Sin embargo, la mayoría de las aplicaciones de "misión crítica" permanecen en "manos" de los mainframes (y minicomputadoras). El objetivo de la industria de redes y en particular la industria de los servidores de base de datos se centra en "bajar" las aplicaciones existentes en los mainframes hacia servidores PC y LAN, usando para ello la arquitectura cliente/servidor. Esto tiene como consecuencia el desarrollo de nuevos sistemas operativos enfocados hacia redes de computadoras y desarrollo de manejadores de base de datos capaces de soportar el esquema cliente/servidor.

El término cliente/servidor se utiliza frecuentemente como sinónimo del procesamiento cooperativo o distribuido. Pero el procesamiento distribuido no es nuevo, desde los años 70 existen sistemas de cómputo que pueden considerarse distribuidos, pero se desarrollaban de tal manera que la distribución de las funciones y los datos quedaba implícita en el diseño y código del programa, como también en las comunicaciones entre las aplicaciones. Esto dio como resultado aplicaciones distribuidas de forma cooperativa con una estructura muy rígida. Los programas y datos se encontraban asociados biunívocamente, por lo que los datos no se accesaban de una manera global desde las diferentes aplicaciones, sino como parte integral de éstas mismas.

Actualmente, las computadoras personales y los paquetes de software de aplicaciones proliferan comercialmente. Las estaciones de trabajo están conectadas a redes de área local (LAN) mediante las cuales se comparten aplicaciones y datos. Las nuevas tecnologías de distribución de funciones y datos en una red permiten desarrollar aplicaciones distribuidas de una manera transparente, de forma que múltiples procesadores puedan ejecutar partes distintas de una aplicación. Si se elige una adecuada infraestructura de sistemas distribuidos y herramientas de desarrollo, las aplicaciones resultantes podrán transladarse a plataformas de diferentes proveedores.

El proceso distribuido se reconoce actualmente como el nuevo paradigma de los sistemas de información. Este cambio ha surgido fundamentalmente como consecuencia de importantes factores (negocio, tecnología, proveedores), y se apoya en la existencia de una gran variedad de aplicaciones estándares y herramientas de desarrollo fáciles de usar que soportan un entorno informático distribuido. Esta infraestructura no debe ser implementada sólo por razones tecnológicas o de moda, deberá utilizarse para desarrollar o rediseñar aplicaciones que soporten los objetivos de la empresa. Sin embargo, la migración de aplicaciones ya existentes sin modificar su funcionalidad, puede acarrear costos substanciales y no producir los resultados deseados.

Las condiciones que sugieren la implantación del ambiente cliente/servidor en una empresa son:

- Cambios estructurales y organizativos.
- Cambios en los organigramas, con mayor delegación en personas y departamentos.
- Respuesta a la dinámica del mercado.
- Cambios en los procesos de negocio.

La situación del mercado está cambiando, de una época en la cual se tenía una masiva producción industrial, se está pasando a otra de ajustada adaptación a la demanda, en otras palabras, nos encontramos en una época en la cual es necesario contar con la capacidad de aproximación de los productos y servicios a la medida de las necesidades del cliente, y es necesario producirlos y suministrarlos con rapidez y costos mínimos.

Las razones que impulsan el crecimiento de las aplicaciones cliente/servidor son:

- La demanda de sistemas más fáciles de usar, que contribuyan a una mayor productividad y calidad.
- La relación precio/rendimiento de las estaciones de trabajo y de los servidores se mejora constantemente.
- La creciente necesidad de acceso a la información para la toma de decisiones y el soporte de los procesos mediante aplicaciones más ajustadas a la estructura organizativa de la empresa.
- La utilización de nuevas tecnologías y herramientas de alta productividad más aptas para la dinámica del mercado.

Enfocado correctamente, el modelo cliente/servidor permite ejecutar las aplicaciones en la plataforma informática más adecuada para maximizar el beneficio operativo. También significa la posibilidad de reubicar funciones y datos, es decir, modificar los procesos de negocios sin alterar la lógica de la aplicación. La selección de herramientas adecuadas conduce a un desarrollo acelerado de las aplicaciones y mediante la reutilización de servidores puede reducirse notablemente la tarea de mantenimiento.

En los años 70 y principios de los 80 era frecuente construir el organigrama de la empresa y de los procesos en función de la capacidad de procesamiento de sus recursos de cómputo. Los mainframes eran costosos y el proceso de transacciones estaba soportado básicamente por terminales no programables. Con la llegada de las estaciones de trabajo programables y la gran disponibilidad de software de productividad personal, las funciones que antes se situaban en el mainframe ahora pueden realizarse donde tienen lugar los procesos de negocio que los utilizan. Adicionalmente, se crea una tendencia a utilizar soluciones de sistemas abiertos. La mayoría de los proveedores anuncian que sus productos son abiertos, que permiten accesar datos y aplicaciones independientemente del sistema operativo y del protocolo de red, soportando sistemas heterogéneos de diferentes proveedores.

En el mundo de sistemas propietarios que ha existido hasta hace poco, los proveedores definían las arquitecturas y los productos que los soportaban. La responsabilidad del departamento de sistemas de información era diseñar y desarrollar aplicaciones basadas en arquitecturas y productos suministrados por el proveedor. Los usuarios eran los receptores de las aplicaciones desarrolladas por los profesionales informáticos y, frecuentemente, estaban a merced de ellos pocas veces tenían posibilidades de influir en su desarrollo.

Hoy la situación ha cambiado drásticamente. La nueva generación de profesionales informáticos está abierta al uso de nuevas herramientas de desarrollo y lenguajes que se presten a la implantación de aplicaciones cliente/servidor, con frecuencia olvidando las metodologías de desarrollo más formales y estructuradas del pasado. Están disponibles sofisticadas herramientas que soportan el desarrollo de interfaces gráficas de usuario (GUI) y lenguajes orientados a objetos. El diseño de prototipos, que se desarrolla en colaboración con los usuarios, se ha convertido ya en parte integral del desarrollo de aplicaciones cliente/servidor.

Por otra parte, los departamentos y grupos de usuarios eligen aplicaciones estándares disponibles comercialmente. Debido a que los usuarios normalmente se centran en las aplicaciones, los aspectos de plataforma de sistemas, integración de aplicaciones y administración de sistemas se descuidan muy frecuentemente.

Como resultado, el departamento de sistemas de información tiene que hacer frente a tareas que anteriormente fueron responsabilidad de los proveedores de software. Actualmente, los gerentes del área de sistemas deben asumir las siguientes responsabilidades, entre otras:

Soporte de las gestiones empresariales. El departamento de sistemas de información debe estudiar el modelo cliente/servidor y la forma de implantarlo eficazmente. Generalmente, poseen conocimientos necesarios para apoyar la gestión de la empresa, formulando objetivos del negocio y proponiendo tecnologías de información que los soporten.

Selección de estándares y tecnologías de información. Como los estándares de sistemas abiertos no están sólidamente establecidos, la responsabilidad de su selección se ha desplazado de los proveedores hacia los departamentos de sistemas de información.

Creación de una infraestructura cliente/servidor. El departamento de sistemas de información debe asumir la responsabilidad de definir la infraestructura cliente/servidor que incluya:

- Plataformas operativas.
- El entorno de desarrollo de aplicaciones.
- La gestión de sistemas distribuidos.
- Establecer los principios de diseño que garanticen aplicaciones portables, interoperables y distribuidas.

Desarrollo de aplicaciones corporativas e integración de aplicaciones. La integración de las aplicaciones desarrolladas en la empresa con aplicaciones estándares es fundamental. Muchas de las aplicaciones estándares suponen normas propias de instalación, parametrización y soporte del *middleware* cliente/servidor.

El convencer a los directores del área de sistemas que migren a un ambiente cliente/servidor, con frecuencia no es una tarea fácil, porque piensan que es un área arriesgada y todavía inmadura. El peligro real es que fácilmente pueden quedarse atrás respecto a otras empresas competidoras que están rediseñando sus procesos de negocios y adoptando el modelo cliente/servidor y sistemas abiertos. Los departamentos de sistemas de información que no entren en la dinámica perderán el control, por lo que el objetivo del este capítulo es describir las principales características de la arquitectura cliente/servidor.

NECESIDAD DE LA EXISTENCIA DE LOS AMBIENTES CLIENTE/SERVIDOR

Qué ha ocurrido para que la arquitectura cliente/servidor sea posible en la década de los noventas, la respuesta a esta pregunta se genera de la convergencia de seis factores.

I La necesidad de subsistir
II La necesidad de cambio y adaptabilidad
III Cambios en la economía
IV División de labores
V Existencia de la tecnología adecuada
VI Desarrollo de nuevas tecnologías

A continuación se analizan cada uno de estos factores:

I. NECESIDAD DE SUBSISTIR

La arquitectura de cómputo en la cual se encuentra sostenida la empresa define su grado de competencia y de estancia en un mercado altamente competitivo en la actualidad.

II. CAMBIO Y ADAPTABILIDAD

El uso de una ambiente de cómputo rígido, tiene como consecuencia:

- Sistemas rígidos
- Altos costos de mantenimiento
- Sistemas frágiles
- Insatisfacciones de parte de los usuarios
- Sistemas monolíticos
- Obsolescencia en la tecnología
- Resistencia al cambio
- Incongruencia en datos
- Inmovilidad

III. CAMBIOS EN LA ECONOMÍA

La relación costo/rendimiento de las estaciones de trabajo y PC se mejora de manera constante; actualmente en el mercado existen PC con capacidades de procesamiento semejantes a un mainframe (250M de memoria RAM, procesadores de más de 130 Mhz, discos duros de 1G o mayores), esta condición está provocando el *downsizing*, es decir, las aplicaciones que antes sólo podían ser ejecutadas por los mainframes se están rediseñando para ser ejecutadas por estaciones de trabajo o PC. Sin embargo, esto no implica la desaparición de los mainframes, sino al contrario, esta situación permite mejorar el desempeño y utilización de éstos. La arquitectura cliente/servidor incrementa la utilización de los recursos de cómputo de una empresa, debido a que permite distribuir la carga de trabajo, localizando los procesos y recursos en la plataforma más óptima.

IV. DIVISIÓN DE LABORES

Los beneficios que son obtenidos con el uso de PC y *workstations* abarcan áreas que no podrían ser cubiertas por la computación centralizada, por ejemplo el uso de interfaces gráficas, soporte a multimedia (voz, video e imagen) y procesadores de palabras WYSIWYG (*what you see if what you get*). Pero existen otras cualidades que no pueden ser sustituidas, como son:

- Manejo de datos coorporativos
- Operaciones de 24 * 7
- Manejo de procesos en *batch*
- Seguridad

Por lo que es necesario dividir las labores con respecto a su razón de características/funcionalidad, colocándolas en donde se optimice su trabajo.

La computación cliente/servidor provee de la arquitectura necesaria para obtener el máximo beneficio de las diferentes plataformas.

V. EXISTENCIA DE LA TECNOLOGÍA ADECUADA

La implantación de la arquitectura cliente/servidor no sería posible sin la existencia de las siguientes tecnologías:

- Sistemas abiertos
- SQL (*Structure Query Language*)
- GUI (*Graphical User Interface*)
- Protocolos de procesamiento cooperativo
- Redes
- *Bonding software* (*middleware*)

Protocolos de procesamiento cooperativo
Estos protocolos proveen la capacidad de *PEER-to-PEER* que permiten que el *middleware* sea construido sobre él. Ejemplos de estos protocolos son: LU 6.2 y TCP/IP.

Bonding software / MIDDLEWARE
Provee de las API requeridas para las aplicaciones. El *bonding software* simplifica el ambiente cliente/servidor para el usuario, ya que oculta los procesos asociados con la red. Ejemplos de esto son TUXEDO, SQL-NET y Sybase Open Server.

VI. DESARROLLO DE NUEVAS TECNOLOGÍAS

La arquitectura cliente/servidor fomenta el desarrollo de nuevas tecnologías, como por ejemplo;

- ✓ **Multimedia**. Manejo de voz, video e imágenes, el uso de la multimedia permite eficientar el manejo de documentos, disminuyendo el "papeleo" existente en una empresa.
- ✓ **Bases de datos distribuidas**.
- ✓ **Servidores especializados**. De una manera relativamente sencilla es posible agregar a la red servidores para tareas especializadas, los cuales mejoran la relación costo/rendimiento.
- ✓ **Inteligencia artificial**.
- ✓ **CASE** (*Computer Aided Software Engineering*).
- ✓ **WORK-FLOW Management Software**. Permite la planeación y ruteo de documentos entre un grupo de trabajo; esto mejora la productividad, permitiendo procesos concurrentes y asegurando la terminación de cada una de las etapas del trabajo. Este tipo de productos son diseñados para ambientes de PC y estaciones de trabajo.

Concluimos que la reducción de los costos en el equipo de cómputo y el aumento de sus capacidades, junto con el fracaso de los sistemas tradicionales dan lugar a un nuevo escenario en donde las computadoras personales y workstations juegan un papel muy importante. En este nuevo escenario el trabajo es dividido entre clientes y servidores, no se trata de una nueva moda en el campo de la computación, sino de una necesidad de evolución y lo que es más importante de la necesidad de competitividad. La computación cliente/servidor cubre la necesidad fundamental de una empresa: **la necesidad de competitividad.**

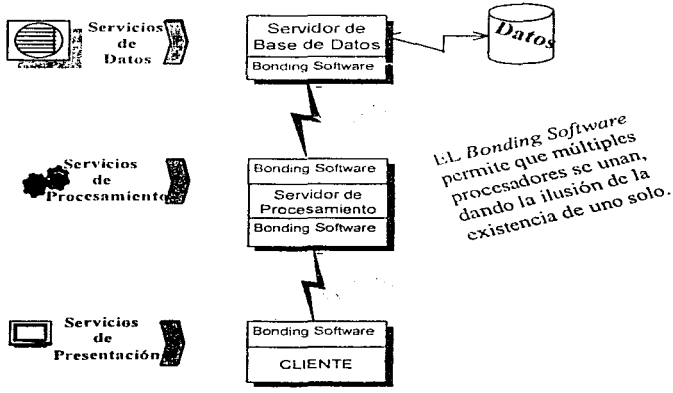
En los 90 nos enfrentamos con un gran poder de cómputo, por ejemplo poderosas estaciones de trabajo con un soporte total a multimedia. Datos y procesamiento se encuentran distribuidos a través del uso de redes de alta velocidad, y la arquitectura de cómputo de la empresa debe ajustarse lo más rápidamente posible a los estímulos del mundo exterior.

DEFINICIÓN DE CLIENTE/SERVIDOR

La computación cliente/servidor es una arquitectura de procesamiento en la cual una aplicación es particionada a través de múltiples procesadores, los cuales coperan de una manera unificada para completar un proceso como una sola tarea. Una definición más rigurosa es:

La computación cliente/servidor es un modelo de cómputo en el cual una aplicación es particionada entre múltiples procesadores (front-end y el back-end), los procesadores coperan (transparente al usuario) entre sí para completar el procesamiento como una sola tarea unificada. Los productos para este ambiente permiten administrar a los procesadores para que juntos realicen una misma tarea dando la ilusión de un solo sistema. Los recursos compartidos se encuentran colocados como servidores, los cuales ofrecen uno o más servicios. Las aplicaciones son colocadas como clientes, los cuales accesan los servicios autorizados. La arquitectura es recursiva, esto es, que un servidor puede transformarse en cliente y realizar peticiones de servicio hacia otro servidor en la red.

ARQUITECTURA CLIENTE/SERVIDOR



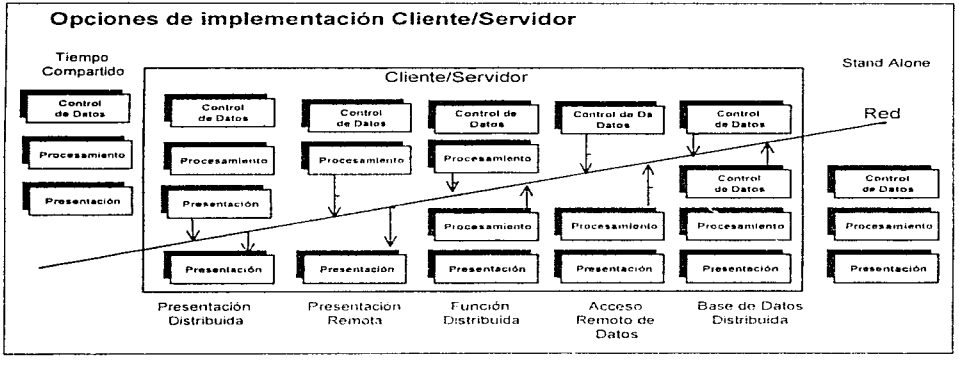
Características generales de cliente/servidor

En el capítulo anterior analizamos diferentes arquitecturas de procesamiento, cada una de éstas ofrecen distintos beneficios para ciertas situaciones, y algunas son claramente mejores que otras, a continuación se enlistan las características que hacen a la arquitectura cliente/servidor más poderosa que otras.

Mantenibilidad
La descomposición de sistemas monolíticos y rígidos en módulos independientes facilitan el mantenimiento de las aplicaciones.

Modularidad
La arquitectura cliente/servidor es construida por medio de módulos conectables entre sí. Cada cliente y servidor es un módulo, el cual es independiente y puede ser reemplazado. Nuevas funciones pueden ser agregadas escalando los módulos existentes, o bien, creando nuevos módulos.

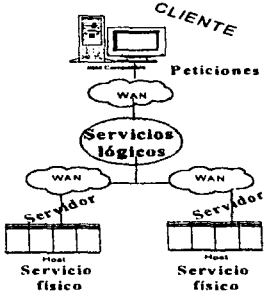
Adaptabilidad
Como se muestra en la siguiente figura, la arquitectura cliente/servidor puede distribuir de diferentes formas la lógica de presentación, procesamiento (lógica del negocio) y datos entre el cliente y el servidor. El tipo de distribución por utilizar dependerá de los requerimientos de volumen de datos, número de transacciones, manejo de datos y ubicación geográfica de los usuarios, entre otros.



Dependiendo de la evolución del sistema, los componentes de una aplicación pueden ser relocalizados con el objeto de mantener una respuesta óptima.

La adaptabilidad se refuerza con la capacidad de recursión de la arquitectura cliente/servidor (un servidor puede convertirse en un cliente y realizar peticiones). La recursión permite crear servicios lógicos en donde un proceso es ejecutado por más de un servidor, y el acceso a los servicios físicos es realizado de una forma indirecta, lo que oculta a los usuarios la estructura de la arquitectura, la cual puede ser reconfigurada sin afectar la imagen que el usuario tiene del sistema.

SERVICIOS LÓGICOS



Escalabilidad
Junto con las capacidades de modularidad, uso de estándares y adaptabilidad, la arquitectura cliente/servidor puede ser escalada para abarcar los cambios en las necesidades del negocio.

Portabilidad
El poder de procesamiento puede encontrarse en diferentes plataformas, desde computadoras portátiles hasta poderosos mainframes. Construir un ambiente cliente/servidor basado en estándares permite colocar los componentes de una aplicación en la plataforma de cómputo en donde se obtenga su mayor rendimiento.

Estándares/Sistemas abiertos
El uso de estándares crea ambientes heterogéneos en donde existen distintos protocolos de red, distintas plataformas de hardware, diversas configuraciones de redes y sistemas operativos.

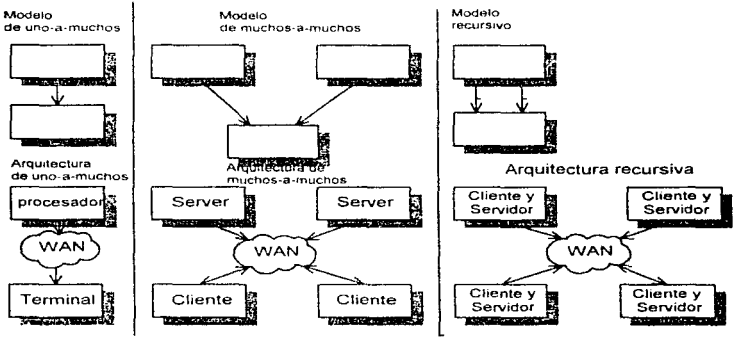
La siguiente tabla muestra algunos de los estándares más utilizados, actualmente, en ambientes cliente/servidor, junto con el estándar se especifican el consorcio que lo promueve y el impacto en su uso.

Organización	Estándar	Importancia en un ambiente cliente/servidor.
Consorcio Xwindows	X.11.3	Permite contar con una interfaz gráfica consistente entre diferentes procesadores (heterogéneos).
ANSI	Lenguaje de Programación C:ANSI X.3JIIC Lenguaje de Programación COBOL: ANSI X.323 SQL definición, manipulación y administración: ANSI X3.135 Intercambio electrónico de datos (Electronic Data Interchange EDI):ANSI X.12 Acceso remoto de datos	Permite la portabilidad y la escalabilidad.
Militar	TCP-IP estandard:1777-78	Permite la interoperabilidad.
IEEE	Sistema operativo POSIX	Permite la portabilidad, escalabilidad y la interoperabilidad.
ISO	Modelo OSI de 7 capas	Permite la Interoperabilidad.

Autonomía
Los clientes pueden poseer mínimas configuraciones de cómputo como podrían ser: computadoras sin disco duro o terminales gráficas, o bien, totalmente configuradas para realizar algún tipo de procesamiento. Con una adecuada configuración, cada cliente y servidor puede trabajar en forma independiente.

Flexibilidad
La arquitectura cliente/servidor permite modelar las relaciones más complejas entre usuarios y equipo de cómputo, la siguiente figura muestra las relaciones que pueden ser construidas.

ARQUITECTURA DE MUCHOS-A-



La arquitectura cliente/servidor puede reducir el costo de mantenimiento debido a que las aplicaciones heredan los atributos de la arquitectura sobre la cual está construida. Por lo que las aplicaciones construidas sobre un ambiente cliente/servidor son: escalables, portables, adaptables, modulares, etc. Estas capacidades facilitan el mantenimiento de las aplicaciones. También se puede incrementar la productividad del software, eliminando o minimizando la necesidad del desarrollo de nuevo; esto es debido a la existencia de librerías compartidas, por lo que al aumentar el número de librerías el tiempo de construcción de nuevas aplicaciones decrece[1].

Las anteriores propiedades hacen al ambiente cliente/servidor la arquitectura de procesamiento más poderosa que puede ser implantada para cubrir las necesidades del negocio.

En la arquitectura cliente/servidor un cliente es un proceso que interactúa con el usuario y cuenta con las siguientes características:

[A] Presenta una interfaz al usuario (UI, *User Interface*), la cual tiene como objetivo enviar y recibir información del servidor. Generalmente, el cliente presenta una interfaz gráfica al usuario.

Debido a que una arquitectura cliente/servidor puede consistir de múltiples clientes, es posible que múltiples interfaces puedan correr simultáneamente, cada cliente cuenta con su propia interfaz por

[1] Por si sola la arquitectura cliente/servidor no resuelve todos los problemas asociados con el desarrollo de aplicaciones, es necesario utilizar otras estrategias, como son: CASE y la Programación Orientada a Objetos

lo que se puede presentar una interfaz tipo Presentation Manager y Microsoft Windows simultáneamente.

[B] Crea uno o más *queries* en un lenguaje predefinido para ser enviados al servidor. El cliente y el servidor pueden comunicarse a través de un lenguaje estándar como lo es SQL, o bien, por algún lenguaje particular.

[C] Se comunica hacia el servidor a través de un proceso de comunicación. Idealmente este proceso es transparente al usuario, es decir, el usuario nunca se dará cuenta de este proceso.

La característica [B] es un proceso escondido que es realizado en las computadoras conectadas al host, ya que éstas poseen medios inteligentes y capacidades de procesamiento.

Un servidor es un proceso, o un conjunto de procesos, que debe existir en una máquina y su objetivo será el de proveer un servicio a uno o más clientes, posee las siguientes características:

[A] Un servidor da un servicio a los clientes

Un servicio otorgado por un servidor puede requerir de un apoyo mínimo por parte de éste (ejemplo de esto un *print server* o un *file server*) o bien de un cálculo intenso como podrían ser servidores de base de datos o servidores de procesamiento de imágenes.

[B] Un servidor ideal esconde su estructura al cliente y al usuario, es decir, se establece una comunicación transparente entre el usuario y el servidor sin importar la plataforma del servidor (hardware y software), así como también de la tecnología de comunicación que se utilice. Por ejemplo, un cliente DOS puede ser habilitado para poder comunicarse con un servidor UNIX o uno OS/2 de la misma manera, sin importar el sistema operativo en el servidor y la tecnología LAN que conecta al cliente con el servidor.

En un ambiente multiservidor, los servidores que atienden a un determinado número de clientes se comunican entre sí transparentemente, sin que el cliente se dé cuenta de esto, por lo que, en un ambiente de procesamiento distribuido, el cliente no podrá ser capaz de localizar al servidor o servidores que los atienden.

De este modo, una arquitectura cliente/servidor divide una aplicación en procesos independientes, los cuales son procesados en máquinas separadas que se encuentran conectadas a través de una red. Entre más avanzado sea el sistema operativo de red más se reducirán, en código, el tamaño de los procesos por ejecutar. Por ejemplo, Microsoft LAN Manager da una amplia gama de funciones para la implementación de un sistema cliente/servidor, es decir, el sistema corre soportado por las funciones dadas por Microsoft LAN Manager , lo cual reduce el tiempo de desarrollo. Si el mismo sistema se implementara en un sistema operativo de red que únicamente provee de las posibilidades de compartir archivos y periféricos, el código de las aplicaciones sería fácilmente el triple.

Los servidores pueden ser computadoras de propósito general, especializadas (para eficientar ciertas tareas), o bien, computadoras con la capacidad de *multitasking*. Cualquier actividad que puede ser catalogada de uso general o reutilizable (comunicaciones, cálculos numéricos intensos, mail/fax, gráficos, periféricos, etc.) es un candidato para ser servidor.

Las siguientes figuras ilustran los conceptos de la arquitectura cliente/servidor, mostrando sus principales atributos:

1. Múltiples clientes pueden accesar múltiples servidores.

2. Múltiples servidores pueden ser accesados por múltiples clientes.

3. Un servidor puede proveer múltiples servicios.

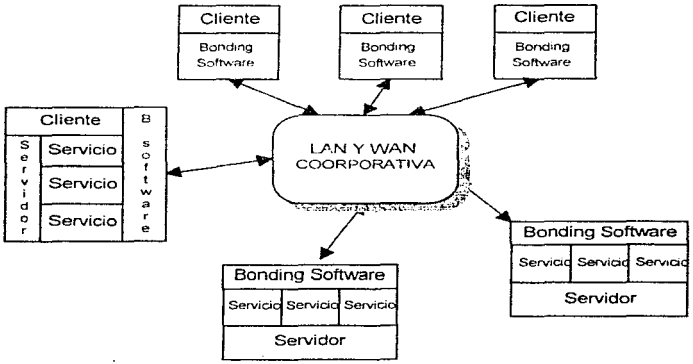
4. Un servicio puede ser ofrecido por múltiples servidores.

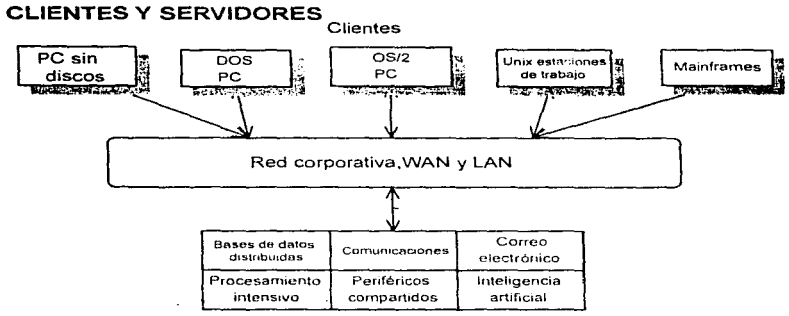
5. La arquitectura es recursiva, los servidores pueden actuar como clientes y realizar peticiones de servicio a otros servidores; por lo anterior, la arquitectura es denominada de "muchos a muchos", la cual se distingue de la arquitectura de "uno a muchos" del tradicional ambiente centralizado.

6. Redes de área local y extendida son usadas para conectar los clientes y los servidores que se encuentran ubicados en diferentes zonas geográficas.

7. Una capa de software llamada *bonding software* reside en los clientes y en los servidores, y es la encargada de controlar el ruteo de mensajes entre los nodos.

ARQUITECTURA C/S : MÚLTIPLES CLIENTES Y SERVIDORES





El *bonding software* es el medio por el cual se crea la ilusión de la existencia de un solo sistema, y es comúnmente referenciado como el *middleware*.

COSTOS Y BENEFICIOS DE CLIENTE/SERVIDOR

Los costos de la implementación de soluciones cliente/servidor no deben contemplarse sólo en términos absolutos, sino que deben medirse en función del beneficio que reporten los nuevos desarrollos. Como el precio de las computadoras personales se ha reducido en los últimos años, con frecuencia se comete el error de pensar que las soluciones cliente/servidor son económicas, o más económicas que las basadas en mainframes o minicomputadoras. Algunos estudios muestran que el costo del hardware y software en un periodo de cinco años representa solamente el 20% de los costos totales y el 80% restante son costos de infraestructura y gastos de explotación.

Algunos de los factores que son necesarios de tomar en cuenta en un ambiente cliente/servidor y que en muchas ocasiones se olvidan son:

- Aumento de la complejidad debido a la conexión de múltiples sistemas heterogéneos.
- Diversidad de fuentes de datos.
- Nuevas plataformas (sistemas operativos) y operaciones.
- Inversión inicial en infraestructura.
- Formación y creación de especialistas en nuevas herramientas y metodologías.
- Arquitecturas de aplicaciones y diseño para un entorno distribuido.
- Nuevas guías de procedimientos operativos.
- Soporte técnico a los usuarios.

La inversión en un sistema cliente/servidor tiene las siguientes ventajas:

- ✓ Utilizar los adelantos en el campo de la computación de una forma más rápida. Actualmente, las estaciones de trabajo poseen un considerable poder de cómputo que anteriormente sólo se encontraba en los mainframes o minicomputadoras.

- ✓ Permitir que el procesamiento resida cerca de la fuente de los datos, por lo que el tráfico en la red y el tiempo de respuesta pueden ser reducidos significativamente.

- ✓ Facilitar el uso de interfaces gráficas, las cuales son fáciles de aprender y permiten el rápido desarrollo de aplicaciones atractivas y funcionales.

- ✓ Permitir la heterogeniedad en software y hardware, diferentes arquitecturas con distintos sistemas operativos pueden ser utilizados para el procesamiento. Esta heterogeneidad es lograda a través del uso de estándares.

- ✓ Impulsar la aceptación de sistemas abiertos.

Beneficios de la arquitectura cliente/servidor

Los beneficios obtenidos por la implementación de un sistema cliente/servidor pueden ubicarse en alguna de las siguientes categorías.

- ✓ Mayor productividad de los usuarios, debido al uso de interfaces gráficas que permiten accesar e integrar aplicaciones de una forma intuitiva.

- ✓ Incremento en la disponibilidad de software comercial, como procesadores de texto, hojas de cálculo, etc.

- ✓ Cercanía del usuario a las aplicaciones y los datos que son necesarios para su actividad, compartiendo servicios.

- ✓ Mayor disponibilidad de herramientas de desarrollo fáciles de usar, reduciendo con esto la dependencia del departamento de sistemas.

- ✓ Reducción del costo de mantenimiento.

- ✓ Mejoramiento de la utilización de los recursos de cómputo.

- ✓ Incremento de la portabilidad del software.

- ✓ Mejora del rendimiento de las redes existentes en una empresa.

- ✓ Reducción en el tiempo de desarrollo de las aplicaciones.

- ✓ Aumento en la productividad de los programadores.

- ✓ Permite a la empresa adaptarse, muy rápidamente, a los cambios en las políticas internas de la compañía, como a los avances de la tecnología, lo cual da a la empresa una ventaja competitiva sobre sus contrincantes.

Entre los beneficios percibidos por la alta dirección, seguramente estarán los siguientes:

- ✓ Un mejor ajuste del sistema de información a la organización y a los procesos del negocio. Cada tarea se puede ubicar en la plataforma que sea más eficaz, y los recursos informáticos se pueden ampliar progresivamente, esto es, funciones y datos pueden ser ubicados en donde sea necesario para las operaciones diarias, sin modificar las aplicaciones.

- ✓ Adición de nuevas funciones, cambiando solamente las partes afectadas, sin modificar el conjunto (aplicaciones incrementales).

- ✓ Acceso a la información donde y cuando la necesitan los usuarios. Este es, probablemente, la mayor ventaja del sistema.

- ✓ Disponibilidad de las aplicaciones estándares (comprar en vez de desarrollar).

- ✓ Libertad para migrar a plataformas de sistemas alternativos y usar servidores especializados.

- ✓ Una respuesta más rápida a las necesidades del negocio gracias a la disponibilidad de software de productividad personal y de herramientas de desarrollo fáciles de usar.

- ✓ Un entorno de utilización más sencillo, que proporciona una mayor productividad. A largo plazo, las GUI reducen los costos asociados a la educación y formación de los usuarios.

Los beneficios que se derivan de una aplicación cliente/servidor dependen, en gran medida, de las necesidades del negocio.

Pero la arquitectura cliente/servidor no es perfecta, ni tampoco es la solución a toda clase de problemas; existen algunas desventajas que deben ser mencionadas, como son:

- ☞ Si el servidor realiza la mayor parte del procesamiento de las aplicaciones, éste puede saturarse y provocar cuellos de botella (*bottleneck*) al aumentar el número de usuarios.
- ☞ La administración de esta clase de sistemas es más compleja y delicada, debido a que se trabaja en un ambiente distribuido.
- ☞ La curva de aprendizaje puede llevar demasiado tiempo.
- ☞ Sin una clara definición de las metas y objetivos del sistema se puede llegar a incurrir en costos y tiempos que sobrepasen por mucho a los estimados.
- ☞ Las aplicaciones distribuidas, especialmente las diseñadas para un ambiente cooperativo, son más complejas de desarrollar, implementar y administrar que los sistemas centralizados.

PROCESAMIENTO COOPERATIVO

La computación cliente/servidor se fundamenta en el procesamiento cooperativo (también llamado arquitectura *peer-to-peer*). Una arquitectura de procesamiento cooperativo utiliza un protocolo de comunicación peer-to-peer para realizar una conversación interactiva. El middleware simplifica la programación de las aplicaciones y esconde los fundamentos del protocolo tanto en el cliente como en el servidor. Las aplicaciones accesan al middleware por medio de API (*Application Program Interface*); con el uso de éstas, se crea la ilusión de un solo sistema. Las técnicas más comúnmente utilizadas son RPC (*Remote Procedure Call*) y SQL (*Structured Query Language*). El RPC oculta al cliente y al servidor los procesos que se realizan a través de la red, de hecho, se ocultan las localizaciones físicas de éstos. La siguiente figura ilustra el acceso a los datos en un ambiente cliente/servidor utilizando SQL.

1. En una computadora personal, el usuario realiza operaciones utilizando una interfaz gráfica (GUI).

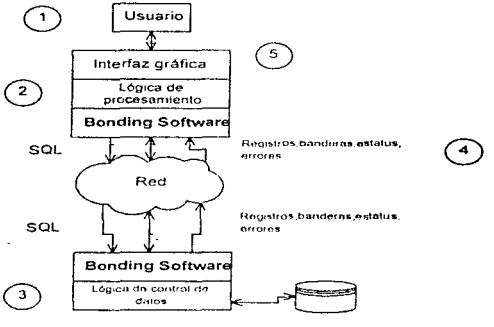
2. Cuando es requerido, la capa de procesamiento de la aplicación lanza peticiones de SQL para datos, los cuales son procesados transparentemente a través del software de middleware por la red hacia el servidor.

3. El servidor ejecuta la petición.

4. Registros, banderas, errores y estatus són regresados al cliente.

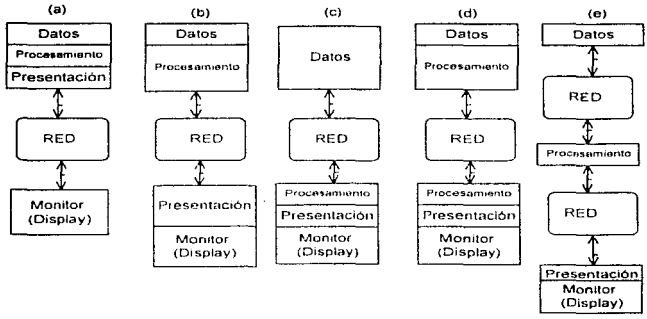
5. El resultado es desplegado al usuario en su computadora personal.

SQL:Interacción Cliente/Servidor



En un ambiente cliente/servidor el trabajo es dividido a través de procesadores cooperativos. La siguiente figura ilustra cómo los clientes y servidores pueden dividir los servicios de presentación, procesamiento y de datos a través de los procesadores participantes.

ALTERNATIVAS DE PARTICIÓN EN UN AMBIENTE CLIENTE/SERVIDOR

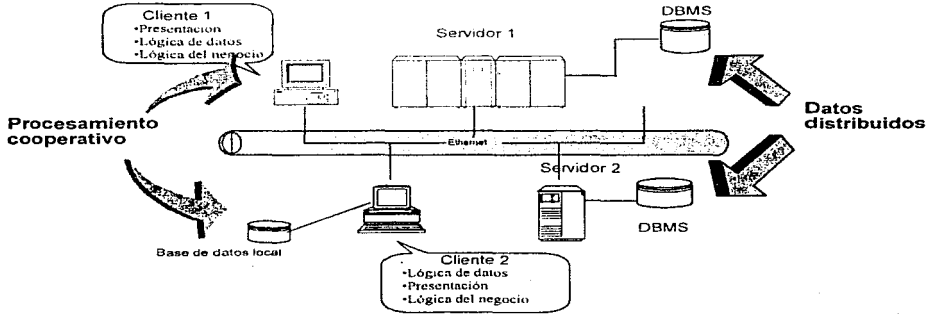


Los clientes pueden utilizar los recursos compartidos que se encuentran localizados en diferentes servidores para completar su trabajo.

La publicidad de los ambientes cliente/servidor enfatiza únicamente la opción de distribución de los datos, pero en un verdadero ambiente de este tipo se abarca más que la pura distribución de datos:

- Distribución de la interfaz con el usuario (lógica de presentación).
- Distribución del procesamiento (transacciones distribuidas).
- Utilización de sistemas abiertos.

PROCESAMIENTO DISTRIBUIDO Y COOPERATIVO



La distribución de las transacciones y de la interfaz de usuario se logra a través del uso del procesamiento cooperativo, como ya mencionamos es el fundamento de la arquitectura cliente/servidor. La característica principal de este tipo de procesamiento es su alto grado de integración entre los diferentes componentes de un sistema, en un ambiente cliente/servidor esta integración es la que existe entre las peticiones del cliente y los servicios ofrecidos por el servidor. Para entender esta interacción definamos los componentes en que se divide una aplicación:

☞ **Lógica de presentación**. Esta parte de la aplicación es responsable de interactuar con el usuario a través de dispositivos, tales como terminales o workstations. Se realizan tareas como las siguientes:

- El formateo de la pantalla.
- Lectura y escritura de información en el monitor.
- Manejo del ambiente gráfico.
- Verificación de los tipos de datos y rangos de los datos introducidos.
- Control en el acceso (seguridad).
- Manejo del mouse y teclado.

☞ **Lógica del negocio**. Esta parte de la aplicación no tiene una relación directa con el usuario ni con las operaciones de entrada/salida de la base de datos. En vez de esto, en esta capa residen los procesos y funciones particulares a una empresa, se encuentra formada por los "programas" que automatizan las actividades del negocio. Esta es la parte de la aplicación que utiliza los datos de entrada (introducidos por medio de la capa de interfaz con el usuario) para realizar las tareas del negocio. Típicamente se utilizan lenguajes de tercera generación, como los son: C, COBOL, PL/1.

☞ **Lógica de datos**. Típicamente se divide en dos componentes.

- Procesamiento de datos. Esta es la parte de la aplicación que se encarga de la manipulación de los datos, los cuales son controlados a través de un manejador de base de datos (DBMS, *Database Manager System*).

- Procesamiento del manejador de base de datos. Lo comprende el procesamiento de los datos que están relacionados directamente con las peticiones generadas a través de DML (*Data Manipulation Language*), como son: operaciones de entrada/salida, manejo del *buffer* de la base, control de candados, control de la concurrencia, etc. Este procesamiento es realizado por el DBMS (*Database Management System*), y desde el punto de vista del usuario es transparente, pero para la arquitectura es una parte esencial de la aplicación.

En un ambiente centralizado estos elementos radican en un solo nodo, por lo que la aplicación se encuentra restringida por los recursos disponibles en la plataforma en la cual corre. Con el uso del procesamiento distribuido surgen nuevas oportunidades que permiten el desarrollo de sistema escalables y portables que corren en un ambiente de sistemas abiertos (multivendedores).

La distribución implica dividir los recursos de cómputo en fragmentos y dispersarlos a través de la red, la pregunta que surge es ¿qué recursos deben ser distribuidos? uno de los principales componentes de una aplicación lo representan los datos, los cuales pueden localizarse en donde se obtenga su mayor disponibilidad y rendimiento en el acceso, además de los datos el "procesamiento" puede ser distribuido a través de la red, mejorando con esto el factor de costo/rendimiento.

Una vez que los componentes de una aplicación son divididos, éstos deben cooperar para realizar el procesamiento como una sola tarea unificada.

El cómo distribuir los componentes de una aplicación entre los clientes y el servidor dependerá del tipo de aplicación por implementar, algunas recomendaciones generales son:

✓ La interfaz con el usuario (lógica de presentación) es colocada en los clientes, y estos clientes se ubican en la capa más baja de un ambiente de multicapas (ver capítulo II).

✓ Debido al poder de cómputo de las workstations (clientes), y al hecho de que la interfaz con el usuario reside en el cliente, es posible colocar parte del procesamiento de las aplicaciones en el cliente.

✓ Si el procesamiento que realizan los clientes utiliza sólo datos con muy pocas modificaciones, o bien de sólo lectura, es posible mantener una base de datos local al cliente.

✓ Los datos que son compartidos por la mayoría de los clientes deben ser colocados en los servidores (datos coorporativos).

Aunque la interfaz con el usuario es un elemento importante, el control y acceso de los datos como también los algoritmos propios del negocio (lógica del negocio) son las partes fundamentales de una aplicación, a la unión de la *lógica del negocio* y la *lógica de datos* se le denomina *procesamiento de la aplicación.*

En un ambiente cliente/servidor la lógica de presentación (interfaz con el usuario) es colocada por lo general en el cliente, pero la ubicación de la lógica del negocio y la lógica de datos es un factor crítico en el diseño de una arquitectura de este tipo, obviamente existen tres posibles opciones:

☞ Colocarlos totalmente en el cliente.

☞ Ubicarlos en el servidor.

☞ Dividirlos, distribuyendo los fragmentos entre el cliente y el servidor.

Cada una de estas soluciones cuenta con ventajas y desventajas, las cuales mencionaremos a continuación.

Caso 1. El procesamiento de la aplicación reside en el cliente. Existen diferentes razones para seleccionar esta opción, algunas de ellas son:

✓ La relación costo/beneficio de las estaciones de trabajo en la actualidad mejora el obtenido en un mainframe o minicomputadora, recordemos que en la arquitectura cliente/servidor de multicapas la segunda capa se encuentra formada por microcomputadoras que son servidores de redes de área local (LAN), las cuales operan como servidores de los clientes que se encuentran en la tercera capa, y como clientes de la primera capa.

✓ El tráfico en la red es significativamente reducido.

✓ No es necesario una sincronización entre los componentes de la aplicación, debido a que todo reside en el cliente.

Algunas desventajas son:

✗ La necesidad de mantener copias de las mismas aplicaciones en todos los clientes es un problema muy grave para la administración de un sistema.

✗ La ventaja de mantener módulos compartidos (librerías) no es posible de realizarla.

✗ El poder de cómputo del servidor no es explotado en su totalidad y los clientes pueden llegar a sobrecargarse, lo que implicará un *upgrade*.

Caso 2. Residencia en el servidor, las razones para tomar esta decisión son:

✓ Colocar los módulos comunes (librerías) en el servidor central, eliminando con esto redundancia y simplificando el mantenimiento.

✓ Colocar la lógica del negocio en el servidor donde se encuentra la base de datos reduce el tráfico entre la aplicación y los datos.

✓ No es necesaria la sincronización de los procesos.

Las desventajas de esta opción son:

✗ Colocar totalmente el procesamiento de las aplicaciones en el servidor puede generar una sobrecarga de los recursos, esto comúnmente sucede cuando el número de aplicaciones y usuarios es incrementado.

✗ La interacción entre la interfaz del usuario y el procesamiento puede generar un alto grado de tráfico en la red, decrementando con esto el tiempo de respuesta.

✗ El poder de cómputo del cliente posiblemente sea subutilizado.

Caso 3. El procesamiento de la aplicación es divido entre el cliente y el servidor.

En esta opción se combinan las ventajas de los anteriores casos y se disminuyen sus desventajas, los módulos comunes son colocados en el servidor de tal forma que se comparten entre los diferentes clientes, otros componentes pueden ser colocados en los clientes cerca de la interfaz con el usuario, con esta opción:

☞ La redundancia es eliminada.

☞ Se optimiza el tráfico en la red.

☞ Los recursos son ubicados en donde se obtenga su mayor rendimiento.

La distribución y la sincronización de los procesos de los componentes de una aplicación es una de las tareas más críticas en el diseño de una arquitectura cliente/servidor, los clientes interactúan con los servidores de una forma cooperativa, intercambiando peticiones y respuestas. La comunicación se realiza de una forma sincronizada, en donde el cliente envía una petición y espera el resultado del servidor antes de poder enviar la siguiente.

Las preguntas que un diseñador de ambientes cliente/servidor debe contestar son:

- *¿Cuál es la distribución más adecuada de los componentes de una aplicación en una arquitectura cliente/servidor? ¿Dónde deben encontrarse cada uno de estos componentes (lógica de presentación, lógica del negocio y la lógica de procesamiento de los datos)?*
- *¿Cómo deben ser distribuidos los datos, accesos y administración?*
- *¿Cómo puede ser implementada la interacción entre los componentes?*
- *¿Cómo puede ser implementada la distribución y administración del software en un ambiente distribuido?*
- *¿Cuáles son los roles y requerimientos para la administración de transacciones distribuidas?*
- *¿Cómo se controla la seguridad en un ambiente cliente/servidor?*

Las respuestas a estas preguntas representan la parte central de la arquitectura cliente/servidor.

Recordemos que el procesamiento cooperativo es un caso especial del procesamiento distribuido, en el cual las funciones y componentes de una aplicación -lógica de presentación, lógica del negocio, lógica de datos- se encuentran distribuidos en dos o más sistemas de cómputo, y se caracterizan por un alto grado de interacción entre ellos. En un ambiente cliente/servidor, esta interacción toma la forma de peticiones del cliente y reacciones de los servidores a las peticiones.

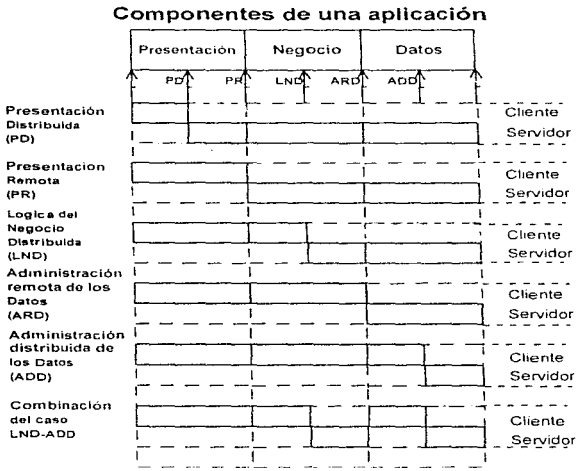
Para disponer de los recursos de cómputo de una forma óptima los componentes de una aplicación deben ser distribuidos de tal forma que el procesamiento cooperativo entre ellos sea posible de realizar, por lo que la computación cliente/servidor utiliza el procesamiento cooperativo para:

- Distribuir el procesamiento de una aplicación entre el cliente y el servidor.
- Soportar un alto de grado de cohesión entre los clientes y los servidores de una forma cooperativa.

La distribución de los componentes de una aplicación puede realizarse en diferentes estilos de procesamiento cooperativo, como son :

- ✓ Presentación distribuida (PD).
- ✓ Presentación remota (PR).
- ✓ Lógica del negocio distribuida (LND).
- ✓ Administración distribuida de los datos (ADD).
- ✓ Administración remota de los datos (ARD).

DISTRIBUCIÓN DE LOS COMPONENTES DE UNA APLICACIÓN



Cada uno de los estilos anteriores pueden ser implementados en una arquitectura cliente/servidor a través del soporte de una o más técnicas de procesamiento cooperativo (RPC, SQL, pipes).

A continuación explicaremos cada una de estos estilos de procesamiento cooperativo.

Lógica de presentación

Desde el punto de vista del procesamiento cooperativo, la forma en que se divide la lógica de presentación del resto de los componentes de la aplicación determinan dos estilos de presentación cooperativa: *presentación distribuida* y *presentación remota*.

Presentación distribuida

La presentación distribuida corresponde a la división PD en la figura anterior. Esto es, las funciones de presentación son distribuidas cuando parte de la lógica de presentación es dividida entre dos o más nodos en la red. La siguiente figura ilustra la presentación distribuida en donde parte de la interfaz con el usuario se localiza en un nodo, mientras que el resto de la lógica de presentación junto con los demás componentes de la aplicación, se localizan en otro.

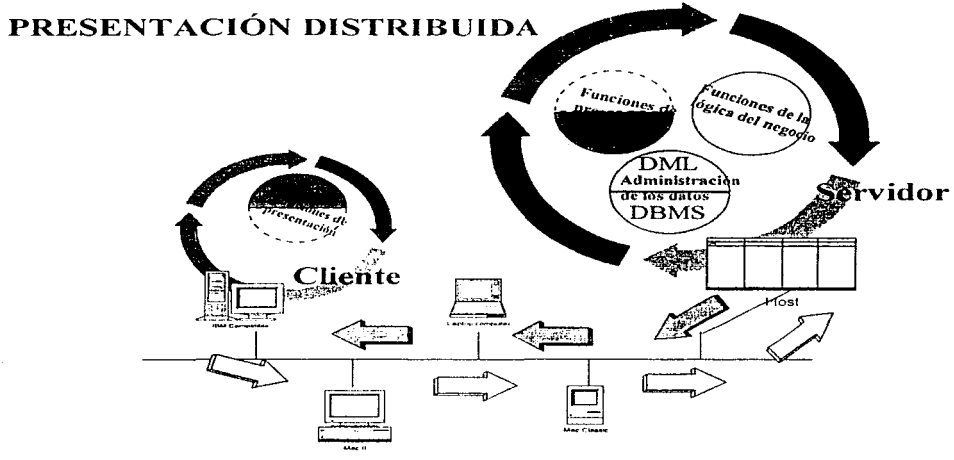
Un modelo típico de presentación distribuida consiste de dos componentes: el *front-end* y el *back-end*. El front-end se encarga de controlar la parte física de la interfaz con el usuario -despliegue en la pantalla, interfaz gráfica, administración de las ventanas, colores, tamaños y tipos de letras, manejo del mouse y el teclado- por lo que el front-end se localiza en el dispositivo que el usuario utiliza para comunicarse -terminal, computadora personal o estación de trabajo-, en la terminología de la arquitectura cliente/servidor, el front-end reside en el cliente.

El componente de *back-end* reside en un nodo diferente al del front-end, y tiene a su cargo las funciones comunes a los clientes (front-end), en la terminología del ambiente cliente/servidor el back-end reside en el servidor.

Una característica importante de la presentación distribuida es el hecho de que se realiza un procesamiento cooperativo entre el front-end y el back-end.

La presentación distribuida es especialmente utilizada cuando se utilizan computadoras personales como interfaz para las aplicaciones de un mainframe.

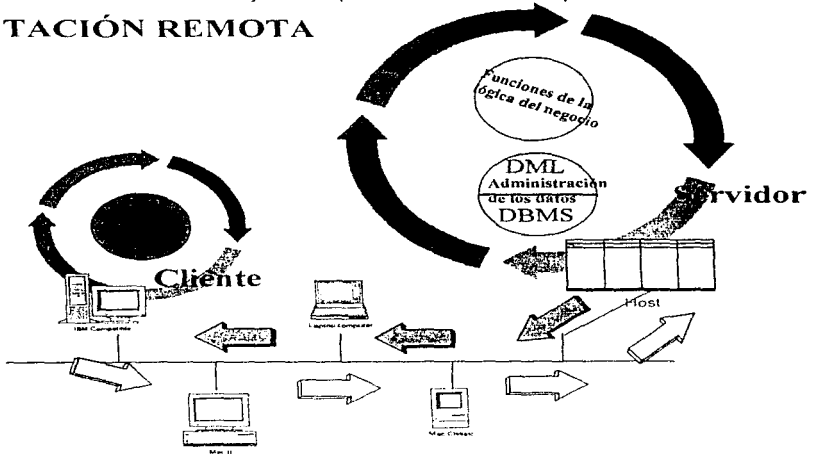
X windows y OS/2 presentation manager son ejemplos típicos de este tipo de distribución.



Presentación remota

La presentación se dice que es remota (de la lógica del negocio) cuando el código de la presentación se encuentra totalmente en un solo nodo, mientras el resto de la aplicación es localizada en otro nodo distinto, este tipo de procesamiento cooperativo es utilizado principalmente para aplicaciones en donde las interacción con el usuario se encuentra predefinida.

Las funciones de la lógica de presentación se encuentran limitadas al envío de simples mensajes del nodo del usuario hacia los demás componentes de la aplicación que se encuentran en otro nodo, y la respuesta consiste de un mensaje de aceptación o rechazo de la petición.



El procesamiento en este tipo de sistema se realiza en forma cooperativa entre la lógica de presentación y los demás componentes de una aplicación. Este modelo de procesamiento cooperativo puede ser soportado por medio de **RPC** (*Remote Procedure Call*) o por alguna forma de comunicación programa-hacia-programa.

Lógica de negocio

La lógica del negocio es la parte de la aplicación que es diseñada para cubrir las actividades del negocio. En el más simple de los ambientes distribuidos, la lógica de aplicación se encuentra contenida en un solo sistema -un solo nodo en la red-. Los inconvenientes de este sistema centralizado son:

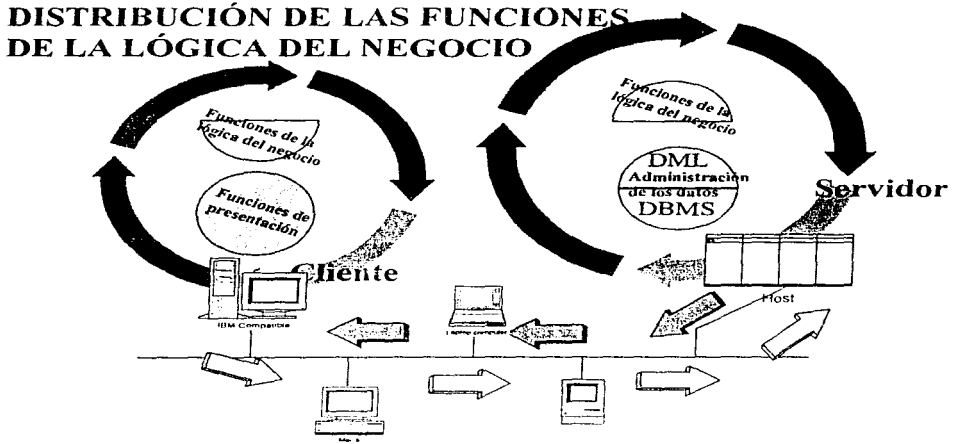
- ✗ La ubicación centralizada puede causar cuellos de botella (*bottleneck*), especialmente en aquellas aplicaciones que requieren de un alto grado de procesamiento.
- ✗ La existencia de nodos críticos, si uno de estos nodo falla el sistema en su totalidad deja de funcionar.

- Los recursos de cómputo, que se encuentran dispersos a través de la red, son escasamente utilizados y los nodos donde residen las aplicaciones constantemente deben ser actualizados en hardware para poder mantener los incrementos en la carga de proceso.

La forma de resolver estos problemas es a través del uso de aplicaciones distribuidas en el red.

Lógica del negocio distribuida

En el contexto de una estructura cooperativa, la lógica distribuida del negocio corresponde a la división LND de una aplicación. La lógica del negocio es dividida, por lo que puede ser ubicada en diferentes nodos. De esta manera, la ejecución de la transacciones es realizada en una forma cooperativa entre todos los componentes de la lógica del negocio con la participación cooperativa, tanto de la presentación como del manejo de los datos.



Este modelo proporciona la máxima flexibilidad y permite a los departamentos de desarrollo una total flexibilidad sobre dónde situar las funciones en la red. Es necesario evitar que la distribución de las funciones de la aplicación quede fija por diseño, cuando esto sucede para la redistribución de funciones y datos casi siempre son necesarias importantes modificaciones, o bien, rediseñar la aplicación.

En este caso, un servidor puede realizar funciones relativas al negocio y accesar archivos o bases de datos, devolviendo el resultado al proceso cliente que lo ha invocado. El cliente recibe el resultado sin importar la ubicación del servidor, el cual se puede encontrar local o remotamente. Un servidor de aplicaciones puede, a su vez, actuar como cliente, solicitando funciones de otros servidores en la red.

Ejemplos de productos middleware, que permiten desarrollar funciones distribuidas, son:

- Customer Information Control System (CICS).
- LAN Distributed Platform (LANDP).
- Distributed Application Environment (DAE).
- MQSeries.
- OSF Distributed Computing Environment (DCE).

Una división típica de la lógica del negocio se ha realizado de la forma de front-end/back-end, en donde el front-end inicia las interacciones y el back-end reacciona a las peticiones del front-end. Dentro de este escenario, los componentes del front-end son colocados en los nodos clientes, mientras que los componentes de back-end residen en nodo servidor.

La distribución de la lógica del negocio es particularmente útil en donde existen sistemas complejos o de alta interacción y de un elevado grado de operaciones de entrada/salida de la base de datos.

La tecnología que es utilizada para soportar la lógica distribuida del negocio incluye la implementación de **RPC**, como también de mecanismos de comunicación program-to-program. La distribución de la lógica del negocio es el procesamiento cooperativo más complicado de diseñar y desarrollar, y su complejidad se incrementa de acuerdo al número de nodos y programas que participan en el procesamiento, como de la posibilidad de utilizar distintos tipos de bases de datos (recordemos la arquitectura en multicapas, en la cual existen bases de datos locales).

Lógica de la administración de datos

La información oportuna y veraz dentro de una empresa permite obtener beneficios como:

✓ Mejorar la posición del negocio.

✓ Facilitar y anticipar el cambio de estrategias.

✓ Reducir el tiempo de comercialización para nuevos productos y servicios.

✓ Ofrecer una significativa ventaja competitiva.

Por lo anterior, la lógica de datos -*administración y uso de la información*- tiene un rol importante dentro de una aplicación.

En un ambiente de cómputo, los datos pueden ser almacenados en dos clases de medios distintos: *archivos* y *bases de datos*. Las bases de datos permiten tener una independencia entre las aplicaciones y los datos, ofrecen un medio consistente de definición y manejo de la información. Un DBMS provee de las capacidades de independencia de datos, consistencia, integridad, seguridad y recuperación, al igual que es la interfaz entre el software de la red y el código del negocio, permitiendo el desarrollo de sistemas heterogéneos.

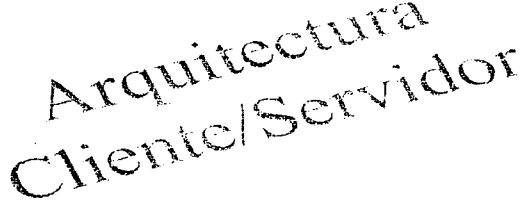
Una arquitectura de procesamiento cooperativo distribuido se caracteriza por los siguientes elementos:

☞ Los datos son distribuidos a través de diferentes sistemas de cómputo.

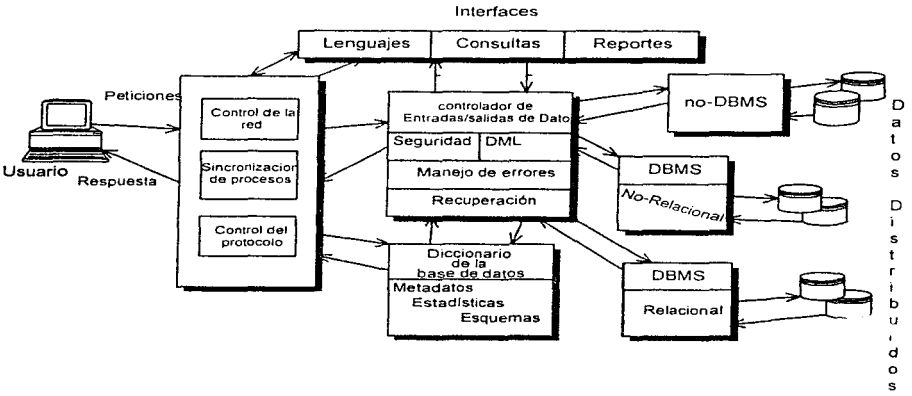
☞ El procesamiento es distribuido en diferentes nodos y es controlado por medio de un administrador de transacciones distribuidas (control de la red, coordinación de procesos, sincronización, etc.).

☞ Los sistemas de administración de datos son distribuidos junto con los datos.

La siguiente figura representa la vista conceptual de una arquitectura distribuida, los datos remotos pueden ser distribuidos en diferentes servidores, y los datos locales pueden residir en los clientes.



ARQUITECTURA DISTRIBUIDA (ANSI)



La distribución de los datos es una fase importante en el diseño de un sistema distribuido, no únicamente porque permite colocar los datos cerca de la fuente, sino también porque provee de un alto grado de disponibilidad de la información; esto es, si múltiples copias de los datos más importantes se encuentran en diferentes ubicaciones, no existirán nodos críticos.

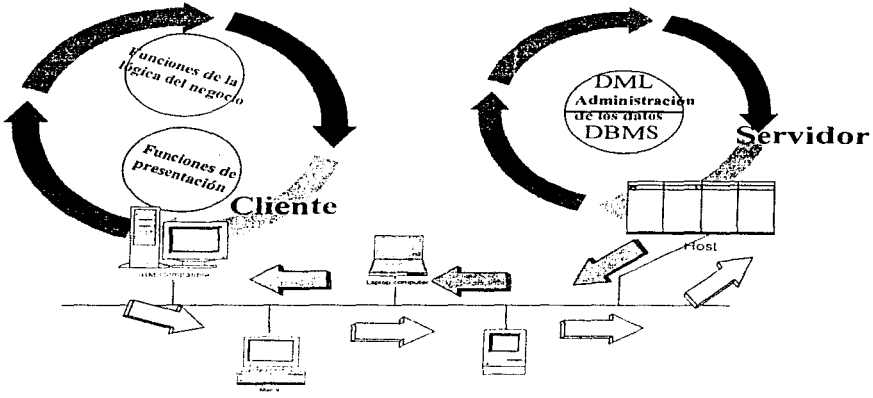
Cuando los datos son ubicados en diferentes nodos (datos distribuidos), o bien, en uno solo (datos remotos), todo o parte de la distribución de la administración de datos debe acompañarlos.

Dos estilos de administración de datos pueden existir en un ambiente cooperativo distribuido: administración remota y administración distribuida. El estilo que se utilizará dependerá de si los datos son distribuidos o se encuentran en forma remota.

Administración remota de los datos

Dentro del contexto de la estructura de una aplicación cooperativa, la administración remota de los datos corresponde a la división de ARD de la figura.

ADMINISTRACIÓN REMOTA DE LOS DATOS



Recordemos que la lógica de administración de datos consiste de dos componentes: la lógica de procesamiento de datos (parte del código de la aplicación que se encarga de manipular los datos dentro de una aplicación) y el procesamiento de la base de datos (es el procesamiento realizado por la base de datos resultante de las peticiones de la lógica de procesamiento de datos).

Cuando todos los datos (lógica de procesamiento de datos) y el DBMS residen en un solo sistema separado del nodo de la lógica del negocio, es considerado un sistema con administración remota de los datos.

A nivel arquitectura, la administración remota de los datos es implementada por medio del modelo de procesamiento front-end/back-end, en donde el front-end contiene la lógica del negocio, mientras que el back-end contiene la base de datos y el DBMS. En un ambiente de un servidor y multiples-clientes, se asume que los datos, no se encuentran distribuidos debido a que existe una administración remota de los datos, por lo que, en esencia, se cuenta con un diseño de una base de datos centralizada. Muchas de las bases de datos comerciales son capaces de soportar la administración remota de los datos,por ejemplo. DB2, Oracle, SyBase, Informix, Ingres y SQLBase, por mencionar algunas.

En el caso de una base de datos relacional (RDBMS), la administración remota de los datos es implementada por medio de un tipo de técnica cooperativa especial llamada SQL (*Structured Query Language*).

El procesamiento cooperativo dentro de la administración remota es relativamente simple de implementar, debido a que sólo existe una sola localidad donde se encuentran los datos; por lo tanto, la consistencia e integridad es controlada localmente.

La administración remota provee, en muchas ocasiones, una total transparencia en la localización y acceso de los datos, lo cual es ideal para la portabilidad de datos y aplicaciones. Pero también se cuentan con algunas desventajas, como son:

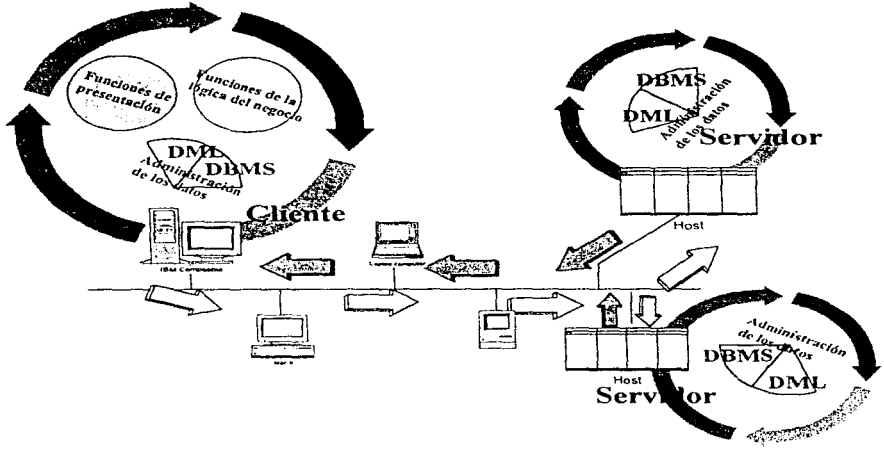
- ✗ Debido a que los datos residen en un solo nodo, esto produce una sobrecarga en los recursos de cómputo.
- ✗ Se requiere que todas las peticiones de datos y respuestas sean transmitidas por medio de la red entre el servidor y la aplicación.
- ✗ El servidor de base de datos se encuentra limitado en su funcionalidad al procesamiento de las peticiones enviadas por el front-end (DML, *Data Manipulation Language*).
- ✗ La existencia de una sola fuente de los datos crea nodos críticos.

Las desventajas de esta arquitectura son resueltas con el uso de la administración distribuida de los datos.

Administración distribuida de datos

El estilo de procesamiento cooperativo de administración distribuida de los datos trata de eliminar los problemas existentes en una administración remota. Dentro del contexto de una aplicación cooperativa, la administración distribuida de los datos corresponde a la división ADD y trata con la distribución de los datos a través de múltiples nodos. Dicha distribución permite colocar los datos cerca de su fuente y proporcionar un alto grado de disponibilidad, colocando múltiples copias de los datos críticos en diferentes ubicaciones.

ADMINISTRACIÓN DISTRIBUIDA
DE LOS DATOS



Cuando los datos son distribuidos sobre varios nodos, todo o parte de la administración de los datos deben ser distribuidos también junto con los datos. Esto es, por lo menos alguna parte del procesamiento de la base de datos deben residir en el mismo sistema que el de la base de datos.

Un ambiente de administración distribuida se caracteriza por dos formas de distribución:

1. Los datos y el DBMS son distribuidos a través de múltiples nodos, incluyendo el nodo donde radica la lógica de la aplicación.

2. Las funciones de administración se encuentran distribuidas entre el front-end y el back-end.

Esta arquitectura puede reducir el tráfico en la red, debido a que las peticiones de datos son analizadas en el nodo local por medio de la lógica de procesamiento de datos, realizando una verificación inicial de la sintaxis, parsing, compilación y determinación de la localización de los datos requeridos. Si los datos son locales, la petición de datos es satisfecha sin enviar señales al servidor, de igual forma la petición mal formadas es rechazada antes de ser enviada a la red.

Esta arquitectura utiliza técnicas de procesamiento cooperativo como son:

- RPC (Remote Procedure Call).
- Program-to-program comunication.
- Mecanismos especiales como el uso de "procesamiento almacenados" (*stored procedures*) y disparadores (*triggers*).

La implementación de sistemas de este tipo permiten:

- ✓ Localización transparente de los datos hacia los usuarios.
- ✓ Integridad de datos.
- ✓ Consistencia.
- ✓ Veracidad de datos.
- ✓ Rapidez en el acceso de los datos.

Sin embargo, para lograr una implementación exitosa, se requiere de considerar los siguientes factores:

- La forma en que los datos serán distribuidos.
- La sincronización de los procesos.
- Candados en los datos.
- Integridad.
- Transparencia.
- Facilidad de administración.

Actualmente, las empresas se encuentran más y más interesadas en implementar bases de datos distribuidas, algunas de las razones son:

☞ Reducción de los costos de operaciones y descentralización de operaciones, de tal forma de ser más competitivos y responder más rápidamente a las demandas de los clientes.

☞ Ubicación de los datos en las zonas donde existe mayor demanda.

☞ Independencia de las áreas regionales.

Una base de datos distribuida ofrece al usuario la ilusión de la existencia de una sola base de datos, pero en realidad es un conjunto de bases de datos almacenadas en diferentes computadoras. La información almacenada en las diferentes bases de datos puede ser accesada simultáneamente y cada una de éstas es controlada por medio de un manejador local (DBMS). Cada uno de los servidores de base de datos coopera para mantener la consistencia de la información en forma global, cada uno de los nodos participantes en la distribución es autónomo en su administración, lo cual permite obtener los siguientes beneficios:

✓ Los datos locales son controlados por el administrador local, por lo que se tiene un ambiente más controlado con responsabilidades compartidas.

✓ Independencia en las fallas.

✓ La recuperación de fallas es realizada en cada uno de los nodos en donde se generaron.

✓ Existencia de un diccionario de datos para cada uno de los nodos.

✓ La actualización en el software de los nodos se realiza en forma independiente.

La ubicación de los datos en un ambiente distribuido se encuentra definida por dos diferentes factores:

☞ **Factores de negocio**

-*Dueños de la información* Los datos deben ser ubicados donde se realizan los procesos que hacen uso de ellos, por lo que el departamento que es dueño de la información posee el mayor nivel de acceso a los datos, mientras que los sites que requieren del acceso pueden poseer una copia de los datos, o bien, establecer un acceso remoto.

-*Tipos de datos* Los datos operacionales y la información que es utilizada como soporte para la toma de decisión (DSS, *Decision Support System*) tienen distintas características y deben ser manejadas en forma diferente (por ejemplo, la ubicación de almacenamiento).

☞ **Factores tecnológicos**

-*Ambiente*. La base de datos y el servidor en donde reside cuenta con restricciones de diseño, por lo que siempre debemos preguntarnos cuestiones como:

¿Existen algunas limitaciones (memoria, espacio en disco, etc.) impuestas por el servidor de base de datos?

¿Puede el servidor de base de datos almacenar el volumen de datos requerido?

¿Puede la base de datos controlar el nivel de volúmenes de datos?, etc.

-Rendimiento. Es necesario tomar en cuenta los requerimientos de tiempo de respuesta del sistema, número de transacciones y el número de usuarios concurrentes que deben ser soportados por el host.

-Disponibilidad. Dependiendo de la necesidad de una operación continua, del plan de contingencias y la facilidad de recuperación en caso de desastre, se decidirá la ubicación de los datos y las técnicas de respaldo a utilizar.

-Control. La ubicación de las aplicaciones también dependerá del nivel de seguridad que se requiera, así como del grado de entrenamiento de los usuarios que accesan el nodo.

La principal ventaja de la distribución es la habilidad de accesar datos que se encuentran localizados en múltiples sites de una manera transparente y, mantener la mayor cantidad de datos locales al site. La arquitectura cliente/servidor ofrece una solución ideal a los requerimientos de una administración de datos distribuidos.

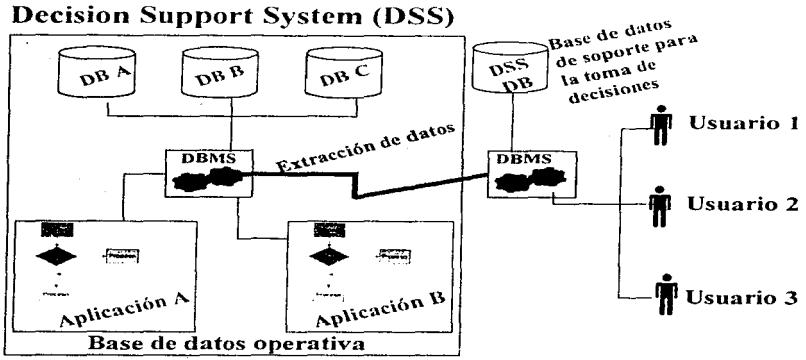
Como mencionamos anteriormente, la ubicación de los datos depende también de la distinción entre los datos operativos y los históricos (los cuales permiten tomar decisiones), por lo que el procesamiento dentro de un arquitectura cliente/servidor puede ser dividido en dos categorías:

Operativo. Es el procesamiento que afecta las operaciones diarias que son realizadas en un negocio, por ejemplo: ¿Cuál es el último pago realizado en una cuenta? ¿Cómo se encuentra las cotizaciones de un producto en el mercado? algunas de sus características principales son:

- Se trabaja con información que es verdadera en el momento exacto en que se está utilizando.
- El desarrollo del sistema es realizado por una aplicación a un tiempo.
- La manipulación de la información se realiza a un detalle de registro a registro.
- Se archivan datos con un alto grado de probabilidad de acceso y de relativo bajo volumen.
- La información generada en este ambiente es utilizada para realizar decisiones rápidas.

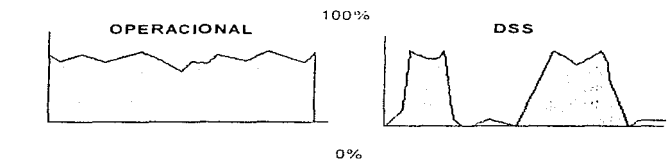
Sistema de soporte para la toma de decisiones (DSS, *Decision Support System*). Es utilizado para el soporte de la toma de decisiones por parte de los gerentes, cuenta con las siguientes características :

- Opera con datos de sólo lectura, es decir, una vez que los datos son almacenados no pueden ser actualizados.
- El procesamiento es realizado sobre datos "integrados", a diferencia del ambiente operativo que los datos se encuentran dispersos.
- Es utilizado para la toma de decisiones a largo plazo.



La distinción entre estos dos tipos de procesamiento es esencial para el éxito de un ambiente cliente/servidor. Cuando el diseñador no entiende la diferencia el resultado será un ambiente desarmonizado.

La utilización del hardware en cada uno de estos tipos de procesamiento es una diferencia notable, la siguiente figura muestra los patrones de utilización del hardware:



La diferenciación entre el ambiente operacional y el DSS es de suma importancia en la construcción de un ambiente cliente/servidor. Debe existir una separación -lógica y física- del procesamiento operacional y el procesamiento del DSS, la mezcla de estos dos ambientes causa:

- ✗ Cuellos de botella en el mantenimiento.
- ✗ Degradación en el rendimiento de las aplicaciones.
- ✗ Eleva la complejidad en las aplicaciones.

Si sólo existe un ambiente, las ventajas de la arquitectura cliente/servidor no serán obtenidas (mantenibilidad, performance, etc.).

Estructura de datos

La estructura de datos es importante en cualquier tipo de ambiente de procesamiento, pero en un sistema cliente/servidor su importancia es mayor, y es necesario considerar los siguientes aspectos:

☞ Qué datos son públicos (corporativos) y cuáles son privados.

☞ La división de los datos entre el ambiente operativo y el DSS.

☞ Almacenamiento de los datos del ambiente DSS (lo que se denomina *Data Warehouse*). En un ambiente cliente/servidor el *Data Warehouse* es el repositorio de datos para el ambiente DSS, sus principales características son:

- Contener la información más representativa de la corporación.
- La información es generada de las aplicaciones.
- Es no volátil, se presenta como una colección de fotografías de los datos.
- Cambia con el tiempo, cada fotografía contiene información sobre el momento de su ejecución.

Dependiendo de las necesidades de acceso a los datos distribuidos, se definen diferentes métodos de distribución. A continuación se explican los métodos más utilizados en el diseño de bases de datos distribuidas.

Métodos de distribución

La distribución de los datos a través de múltiples sites ofrece los siguientes beneficios:

✓ Coloca los datos cerca de su fuente.

✓ Ofrece una alta disponibilidad por medio de múltiples copias de los datos críticos en diferentes ubicaciones, esto elimina la existencia de nodos críticos.

✓ Un acceso más eficiente a los datos, lo cual mejora el rendimiento de las aplicaciones.

Existen diferentes métodos de distribución, a continuación mencionaremos los principales.

Snapshots

Un *snapshot* es una copia total o parcial de una tabla o un conjunto de tablas. El nodo que contiene la tabla o tablas generadoras del snapshot, se denomina como nodo maestro. Sus características principales son:

☞ Típicamente está limitado a sólo lectura.

☞ Se actualiza automáticamente durante un determinado periodo de tiempo.

☞ Son diseñados para la distribución de la información estática o donde la frecuencia de cambio es muy larga.

Replicación

Para las aplicaciones en que los datos distribuidos pueden ser actualizados en múltiples ubicaciones y las actualizaciones deben realizarse simultáneamente, y en el momento en el cual ocurren, los snapshots no son suficientes debido a que se requiere mantener copias de las mismas tablas en múltiples ubicaciones actualizadas.

A fin de mantener las copias de la información actualizadas, el manejador de base de datos debe ser capaz de soportar el método de distribución por replicación, esto permite:

- Crear y mantener copias (réplicas) de una misma tabla en múltiples ubicaciones.
- Mantener la consistencia de los datos a través de todas las réplicas.
- Realizar las actualizaciones en el momento en que se generan, las cuales pueden realizarse en cualquiera de las réplicas y los cambios deben de propagarse a las demás réplicas de la información.

El manejador de base de datos deberá mantener la sincronización de las actualizaciones entre todas las réplicas, así como proporcionar una transparencia al acceso de éstas.

Deberán existir mecanismos de recuperación de fallas y de control de consistencia de las transacciones, por ejemplo ***TWO-PHASE COMMIT*** (explicado en el capítulo II), así como de las utilerías necesarias para la configuración de los servicios de acceso remoto a los datos.

Resulta importante recordar que a través de la arquitectura cliente/servidor es posible la distribución de los datos y el acceso transparente a éstos. ***Pero, si se carece de una acertada administración de datos y de una adecuada técnica de ingeniería de sistemas, es posible que se cree una masa de información inmanejable.***

Arquitecturas de aplicaciones de dos y tres niveles

Las aplicaciones cliente/servidor pueden clasificarse en aplicaciones de dos y tres niveles. En esencia, es una diferenciación basada en la forma en que las aplicaciones se distribuyen entre el cliente y el servidor.

Mantengamos en mente que ésta es una filosofía de diseño, y que los dos o tres niveles describen estructuras lógicas, no físicas. El middleware y las herramientas de desarrollo utilizadas determinan el tipo de distribución y el nivel de flexibilidad, así como los procesadores soportados.

Aplicaciones de dos niveles

Las aplicaciones se dividen en partes que se ejecutan en el cliente y en un servidor. Los servicios de presentación y la lógica de aplicación y acceso a los datos es, normalmente, un bloque monolítico que se ejecuta en la estación cliente, mientras la base de datos está situada físicamente en el servidor.

Estos desarrollos se implementan sobre sistemas de servidores de bases de datos. El formato del mensaje entre cliente y servidor es, invariablemente, algún tipo de SQL.

Algunos manejadores de bases de datos ofrecen *stored procedures* y *triggers* que sirven para desarrollar la lógica de la aplicación en el servidor. Su utilización típica es la creación de sistemas de soporte de decisiones.

Este tipo de arquitectura también es adecuada para nuevas aplicaciones departamentales. Permite que los departamentos de usuarios creen sus propias aplicaciones utilizando herramientas gráficas.

Su ventaja potencial es su rápido desarrollo de nuevas aplicaciones en un entorno de alta productividad.

Aplicaciones de tres niveles

Las funciones de la aplicación están divididas en lógica de presentación, lógica de aplicación o de negocio, y la lógica de datos. Cada una de estas partes puede distribuirse entre los múltiples procesadores de la red. Probablemente, los servicios y la lógica de presentación se ejecutarán en la estación cliente, mientras que la lógica de la aplicación y la lógica de datos puede repartirse entre diferentes procesadores. Separando la lógica de aplicación, puede lograrse un nivel de flexibilidad que no es posible con las aplicaciones de dos niveles.

Como resultado pueden existir servidores de aplicaciones o de transacciones que ejecuten funciones accesibles por muchos procesos clientes con distintas formas posibles de interacción con el usuario.

Las utilizaciones típicas son:

- Muy aptas para aplicaciones corporativas que soportan procesos de negocio que pueden sufrir modificaciones.
- Potentes y adecuadas para aplicaciones grandes, dinámicas o complejas.

Sus ventajas son maximizar el nivel de flexibilidad en términos de distribución de funciones y datos. Las aplicaciones resultantes son escalares, flexibles, reutilizables y abiertas. Pueden seleccionarse las herramientas y lenguajes más adecuados para cada una de las partes de la aplicación.

Su desventajas potenciales son que se tiene una mayor complejidad de desarrollo y se requiere de una inversión inicial en el diseño del sistema. Para la implementación será necesario la utilización de **gestores de transacciones**.

GESTIÓN DE TRANSACCIONES

En un entorno cliente/servidor, el control de la aplicación se desplaza hacia el usuario. Los procesos cliente interaccionan con el usuario, quien a través de ellos solicita servicios que pueden estar situados en el mismo procesador o en otro dentro de la red. La lógica del negocio está implementada como servidores de aplicación, algunos de ellos con acceso a recursos externos, por ejemplo: archivos planos, archivos jerárquicos, bases de datos relacionales, impresora u otros dispositivos. Cada gestor de recursos es responsable de la integridad de su recurso, incluyendo la recuperación física o los daños lógicos. Sin embargo, la integridad de los datos también debe garantizarse a través de todos los recursos utilizados mientras se ejecuta una transacción de negocio. Esto requiere que la transacción que se está protegiendo tenga un punto de inicio lógico y un punto final definidos con claridad, entre los cuales se protejan y se controlen los accesos a los recursos.

Las funciones de gestión de transacciones son vitales en un proceso distribuido. Normalmente las transacciones se inician y finalizan en la estación de trabajo del usuario. Un soporte de "two-phase commit" como parte de un servidor de base de datos puede no ser suficiente, debido a que la base de datos no conoce todos los recursos involucrados en la ejecución de la transacción del negocio.

Servicios de aplicación

En todas las aplicaciones existen funciones que no deben desarrollarse como parte de la lógica del negocio o de datos, debido a que tendrían un impacto negativo sobre la portabilidad y flexibilidad para la distribución de las funciones y datos.

Los aspectos, que a continuación se explican, deben desarrollarse como parte de un grupo distinto de la aplicación, denominado *servicios de aplicación*:

Funciones específicas del entorno operativo

- La programación que dependa del sistema operativo no debe incluirse en la lógica de presentación, de negocios o de datos, debido a que en caso contrario el código resultante no podrá migrarse a una plataforma distinta.

Servicios comunes a múltiples aplicaciones

- Funciones operativas, como el arranque y cierre de estaciones de trabajo y servidores, proporcionan un acceso único para todas las aplicaciones.
- Funciones relacionadas con la administración del sistema, que incluye el respaldo, almacenamiento y restauración de datos locales. El medio donde se realiza el respaldo y el almacenamiento, así como los procedimientos correspondientes pueden pertenecer al entorno del sistema específico.
- Servicios de importación y exportación de datos, que ofrecen copias de la base de datos y todas las funciones requeridas para sincronizar la base de datos.

Control de sucesos y flujos de trabajo de los procesos de negocio

- Los controles necesarios para el flujo de procesos del negocio (aplicación, transacción), el control de flujo no debe de incluirse en la lógica de presentación, de negocio o de datos.
- Las acciones que se tomarán, a nivel de aplicación, cuando no está disponible un servidor. Normalmente el proceso no puede terminarse si un componente o servidor está ocupado en otra tarea. Frecuentemente se necesitan de funciones de proceso específicas de la aplicación para afrontar estas condiciones especiales.

Funciones de monitoreo

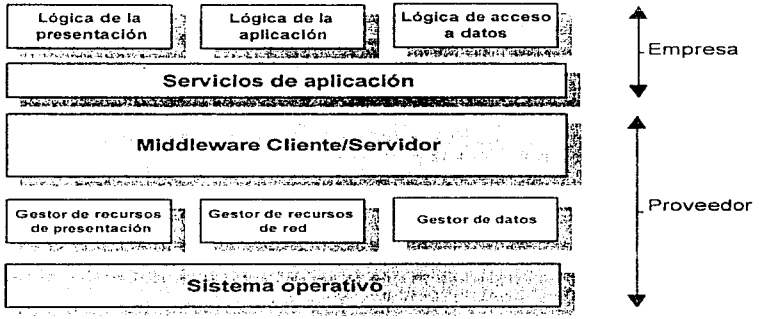
- Control de las tareas y balance de la carga de proceso.
- Control de errores.

Los *servicios de aplicación* aíslan la lógica de presentación, de negocio y de datos del entorno operativo. Los cambios de este entorno suponen cambios sólo en un lugar, y no en los múltiples componentes que pueden verse afectados.

Cuando se eligen la arquitectura de las aplicaciones y los principios de diseño, una de las decisiones básicas es la de si se debe permitir a los procesos cliente y servidor que invoquen directamente a los gestores de recursos, o bien, tener todas esas API en un solo lugar, es decir, en los *servicios de aplicación*. La ventaja de este enfoque es que las API pueden cambiarse más fácilmente, debido a que las modificaciones se realizan en un único lugar (los servicios de aplicación) en lugar de hacerlo en los diferentes módulos funcionales.

Las funciones de la lógica de presentación, de negocio y de datos, si se desarrollan con lenguajes o herramientas adecuadas, son portables y pueden situarse en el procesador más adecuado para su ejecución. El middleware aisla los elementos de la aplicación de los aspectos de distribución relativos al acceso de los recursos locales y remotos, la siguiente figura ilustra este hecho.

Plataforma Operativa



Gestores de recursos

Al principio de la era informática, las aplicaciones se escribían directamente sobre el sistema operativo y los gestores de recursos estaban estrechamente ligados a éste para soportar recursos, tales como dispositivos o archivos. Para conseguir más flexibilidad, en la actualidad los gestores de recursos se programan independientes de la plataforma de sistema operativo que opera sobre éste y pueden ser suministrados por cualquier proveedor.

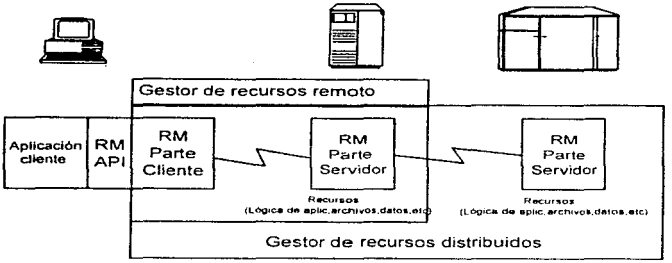
Conceptos

Los gestores de recursos son librerías que accesan los recursos y proporcionan un conjunto de API. Un recurso puede ser un dispositivo, un archivo, una base de datos, un programa o un objeto.

Definiremos tres clases de gestores:

- Gestor de recursos local.
- Gestor de recursos remoto.
- Gestor de recursos distribuidos.

Concepto de gestor de recursos



Gestor de recursos local. Opera sobre el sistema operativo local, proporcionando servicios que son realizados por el sistema operativo local. Ejemplo de esto es el controlador de memoria.

Gestor de recurso remoto. Opera en un sistema remoto, pero es accesado como si se encontrara localmente. Lo componen dos elementos:

- La parte cliente que soporta la API utilizada por el usuario (aplicación) para solicitar el servicio.
- La parte servidor que suministra el servicio y gestiona el recurso físico.

Gestor de recursos distribuido. Gestiona recursos distribuidos situados en cualquier lugar de la red. Es también un gestor de recursos remoto si proporciona soporte al cliente; el recurso distribuido en este caso, aparece ante el cliente como único recurso lógico. Ejemplos de productos que incorporan gestores de recursos distribuidos son:

☞ **Oracle** y **Sybase** que configuran bases de datos distribuidas.

☞ La familia de productos **CICS**, los cuales administran sistemas de archivos distribuidos en un entorno de aplicaciones distribuidas.

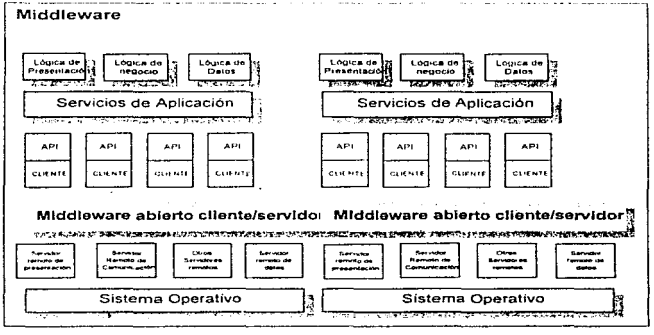
Los gestores de recursos remotos y distribuidos proporcionan acceso transparente a los recursos. Los productos que proporcionan este tipo de soporte son adecuados para desarrollar soluciones cliente/servidor que soporten *presentación distribuida* o un *modelo de datos distribuidos*. El propósito de un gestor de recursos remoto o distribuido es facilitar a los clientes el acceso transparente a los recursos que controla. El cliente no sabe dónde está el recurso y puede interaccionar con el gestor de recursos en la forma más adecuada (por ejemplo, SQL para accesar bases de datos relacionales).

La figura anterior muestra que los gestores de recursos se accesan a través de API. El tipo de API soportada por el gestor de recursos correspondiente determina si la aplicación es de naturaleza abierta o propietaria. Si se utiliza una API estándar, se proporciona un mayor nivel de flexibilidad e independencia, así como una solución más abierta. De la misma forma, la comunicación entre la parte cliente y la servidora del gestor de recursos puede realizarse de forma propietaria, o utilizando estándares de comunicación.

Middleware

Así como los servicios de aplicación aíslan las aplicaciones del entorno operativo y de los detalles de la configuración, el middleware separa las aplicaciones de los aspectos de distribución. Se trata de un software que proporciona un conjunto de servicios para conseguir transparencia en la ubicación de los recursos distribuidos. Con ello se libera al programador de la complejidad de las comunicaciones.

A través de los servicios del middleware, los recursos de la red pueden ser accesados como si fueran locales. Las aplicaciones solicitan servicios a través de una API; el middleware redirecciona la solicitud, determinando la dirección física del gestor del recurso que se puede encontrar localmente o remotamente.



Con objeto de obtener la transparencia en los accesos a los recursos, el **middleware** deberá proporcionar los siguientes servicios:

- Directorios y nominación.
- Comunicaciones.
- Seguridad.
- Gestor de transacciones.

Normalmente no existe un solo producto que proporcione todas estas funciones. Se consiguen mediante la combinación de diferentes componentes de software que se complementan entre sí.

Directorios y nominación

Estos servicios son necesarios para asociar el nombre lógico de un recurso con su nombre físico. Cuando la topología de la red cambia, las modificaciones pueden reflejarse en un directorio en vez de hacerlo en las aplicaciones.

Los directorios y la nominación identifican y proporcionan información sobre los recursos de una red, por ejemplo: servidores, archivos, discos y aplicaciones.

Sus características principales son:

☞ El servicio de directorio proporciona información sobre los recursos de la red. Mantiene información sobre las características de un recurso, como su nombre, dirección en la red, fecha de creación, etc.

☞ El servicio de nominación dispone de los medios necesarios para referenciar los recursos de la red. El uso de un modelo de nominación consistente permite accesar a los recursos por su nombre, incluso si se cambia alguna de sus características como, por ejemplo, su ubicación.

Casi todos los productos middleware incorporan su propio directorio; debido a que es posible que existan varios productos middleware simultáneamente en un mismo entorno operativo, deberán mantenerse varios directorios.

Comunicaciones

Estos servicios son necesarios entre la parte cliente y la servidora del gestor de recursos remotos o distribuidos. Los siguientes tres modelos están aceptados por la industria:

- ✓ Conversacional.
- ✓ Llamadas a procedimientos a remotos (RPC).
- ✓ Cola de mensajes.

Estos modelos permiten al programador codificar únicamente la interfaz de la función, en lugar de tener que realizar toda la estructura de comunicaciones necesaria.

☞ ***Conversacional***

En este modelo fluyen una serie de mensajes síncronos entre el remitente y el receptor, permaneciendo ambos activos durante toda la conversación. Los participantes pueden monitorear el estado, envío y recuperación de los datos.

El modelo conversacional es esencialmente un modelo orientado a sesiones, debido a que se establece una sesión al momento de iniciar una conversación entre los sistemas que contienen las aplicaciones.

El modelo conversacional está basado en la comunicación de igual a igual (peer-to-peer).

☞ ***Llamadas a procedimientos remotos***

En el modelo RPC, la aplicación servidora se encarga de procesar las peticiones y retornar los resultados a los clientes. Es un proceso síncrono, es decir, el programa solicitante no continúa hasta que se ha completado su petición[1].

Utilizando el modelo RPC, un servidor se convierte en un recurso general de la red con un conjunto de funciones definidas que proporcionan servicio a cualquier cliente autorizado. El proceso invocante no conoce los programas que están involucrados ni el mecanismo de las funciones servidoras. Existen muchos distintos desarrollos de RPC, pero el que se encuentra considerado como el estándar de facto es el modelo de RPC de ***Distributed Computing Environment (DCE)***.

[1] En ocasiones es posible realizar operaciones asíncronas.

Cola de Mensajes

En este modelo las peticiones se sitúan en una cola en alguna parte de la red para que sean procesadas por un servicio que está monitoreando la cola. La entrega de los mensajes a la cola están garantizados, por lo que no es necesario establecer una sesión para el control de la comunicación.

Una plataforma de sistema cliente/servidor debe soportar los tres modelos de comunicación citados anteriormente. La utilización de un método u otro dependerá de las necesidades específicas de la aplicación y del sistema.

Ejemplos de productos middleware cliente/servidor son:

- ✓ *DCE*, que proporciona el modelo RPC.
- ✓ *MQSeries*, que proporciona un mecanismo de cola de mensajes.
- ✓ *Customer Information Control System* (CICS), el cual tiene los tres modelos de comunicación.
- ✓ *LAN Distributed Platform* (LANDP), con un mecanismo tipo RPC.
- ✓ *Distributed Application Environment* (DAE), con un mecanismo tipo cola de mensajes.

Los productos middleware deben ser independientes del protocolo de red.

Seguridad

Estos servicios son necesarios para controlar el acceso a los recursos locales y remotos. En los anteriores sistemas centralizados, el sistema operativo validaba la identidad de los usuarios y autorizaba el acceso a los recursos. En un ambiente distribuido[2], las operaciones de seguridad deben realizarse mediante un conjunto de servicios independientes, por lo que la mayoría de los productos middleware incorporan sus propios sistemas de seguridad.

Esto tiene un impacto negativo, tanto para el usuario, que tiene que conectarse a varios sistemas con distintas *usernames* y *password*, como para los administradores. Para evitar este problema, se utilizan los servicios de seguridad del *Distributed Computing Environment*, que incorpora y expande las especificación *Kerberos* del MIT (Massachusetts Institute of Technology).

Gestión de transacciones

Estos servicios son necesarios para garantizar la integridad de todos los recursos críticos.

Una transacción puede comprender muchas interacciones con diversos gestores de recursos a través de la red. Todas las interacciones de una transacción deben estar terminadas antes de realizar cambios de forma definitiva. Si se produce un fallo en medio de la transacción, deberán anularse todos los efectos de ella. Un determinado gestor de recursos es responsable de la integridad de su recurso, incluyendo la recuperación del daño físico o lógico, la anulación de los cambios incompletos, el reintento de las operaciones, etc. El gestor de transacciones proporciona servicios de sincronización para que los distintos gestores de recursos puedan actuar en concierto, garantizando que los recursos gestionados por separado sigan siendo consistentes para una determinada transacción.

[2] En un ambiente distribuido las estaciones de trabajo son independientes, por lo que su seguridad es menor que en un ambiente centralizado.

ENTORNO DE DESARROLLO DE APLICACIONES

Respecto a esta importante área, no existe una única solución válida para todas las problemáticas posibles, debido a que las situaciones son muy diferentes en cada empresa; distintos sistemas operativos, distintos modelos de distribución cliente/servidor o empleo de diferentes tecnologías. Algunas compañías optan por trabajar con software ya probado, mientras que otras prefieren utilizar el software más reciente. Hay quienes han realizado grandes inversiones en herramientas y no están dispuestos a abandonarlas.

El propósito de este tema es ayudar en la selección de las herramientas para un entorno determinado, por lo cual se explica la relación entre la plataforma operativa y el entorno de desarrollo de aplicaciones.

Relación con las plataformas operativas

El entorno de desarrollo de aplicaciones y la plataforma operativa están estrechamente ligados. No se pueden considerar de forma independiente por las siguientes razones :

☞ Una plataforma operativa para la que no existen herramientas de desarrollo, difícilmente puede ser útil.

☞ Algunas herramientas de desarrollo llevan integradas partes de la plataforma operativa o incluso pueden llegar a predeterminar una. Esto puede conducir a un sistema muy cerrado y propietario si se produce alguna de las siguientes situaciones:

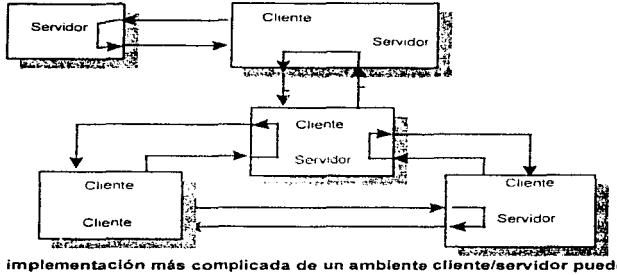
- La comunicación entre los procesos cliente y servidor se basa en protocolos de transporte de bajo nivel. En este caso, el producto puede imponer el protocolo que ha de utilizarse y los sistemas que deben interconectarse.

- La interfaz entre los procesos cliente no es externa, en este caso, un proceso servidor no puede ser invocado por un proceso cliente construido con una herramienta diferente.

☞ Muchas herramientas de desarrollo soportan únicamente un subconjunto de los distintos modelos de distribución cliente/servidor o solamente las soluciones cliente/servidor que hemos denominado de dos niveles.

En un entorno cliente/servidor las aplicaciones tradicionales ya no existen como tales. Las funciones base definidas durante el proceso de diseño de la aplicación se traducirán en procesos cliente y servidor. El resultado será un entorno de aplicación como el que se muestra en la siguiente figura, el cual contiene componentes distribuibles.

Modelo de Interacción cliente/servidor



La implementación más complicada de un ambiente cliente/servidor puede ser comparada con un equipo de trabajo que tiene diferentes actividades. En este equipo, cada miembro tiene sus tareas específicas y las desempeñan realizando peticiones hacia los otros miembros. Cada persona es libre de utilizar cualquier método, siempre y cuando los resultados sean los correctos.

Los componentes distribuibles pueden ser clientes que hacen de interfaz con el mundo exterior o servidores que proporcionan servicios a los clientes y que pueden solicitar servicios de otros servidores, todos distribuidos a través de diversos procesadores en la red. Los servidores pueden ser funciones de negocio encapsuladas, desarrolladas por la empresa o aplicaciones estándares disponibles en el mercado.

En este ambiente el incorporar una nueva aplicación implica:

- Añadir un nuevo proceso cliente.
- El nuevo cliente llama a muchos servidores ya existentes.
- Adicionar un nuevo servidor que también podrá ser utilizado en otras aplicaciones en el futuro.
- Es previsible que se cambien algunos de los procesos-cliente existentes para que se beneficien de los nuevos servidores de aplicación.

Debido a estos atributos, el entorno también puede denominarse ***entorno de aplicaciones incrementales.***

Cuando se define una infraestructura cliente/servidor el objetivo es seleccionar plataformas y herramientas que proporcionen flexibilidad para los distintos modelos de distribución y soporten un entorno de aplicaciones incremental.

El proceso de desarrollo

Existen dos conceptos básicos para conseguir los beneficios derivados de la arquitectura cliente/servidor:

- ✓ *Estructura de la aplicación.* Uno de los objetivos de la separación de los componentes de la lógica de presentación, lógica del negocio, y lógica de datos es que puedan distribuirse en una red. Esto implica que los componentes se desarrollen y se prueben como entidades distintas.

✓ *Entorno de aplicaciones*. Al implantar un entorno de aplicaciones incremental, los procesos cliente y servidor se desarrollan como entidades separadas.

Una correcta infraestructura cliente/servidor proporcionará el marco adecuado para poder usar herramientas de desarrollo de diferentes proveedores.

Para aprovechar las ventajas que ofrece el modelo cliente/servidor es necesario, en la mayoría de los casos, utilizar diferentes herramientas o lenguajes para desarrollar las distintas partes de una aplicación. Seleccionar las herramientas adecuadas no garantizará que la aplicación se diseñe correctamentamente, será necesario contar con una metodología apropiada de desarrollo que asegure la atomicidad de los módulos que formarán la aplicación.

Proceso en cascada. El enfoque tradicional para el desarrollo de aplicaciones se ha descrito como proceso en cascada. Los pasos de definición de necesidades, análisis, diseño, desarrollo y prueba aparecen en una determinada secuencia; un paso sólo puede comenzar una vez terminado el anterior.

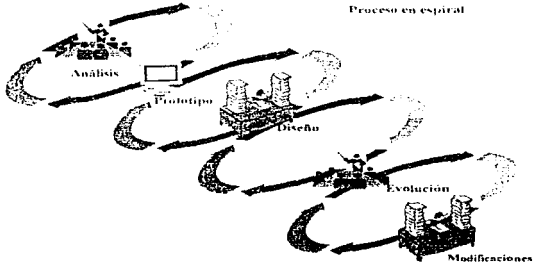
En este tipo de proceso los requerimientos de la aplicación permanecen fijos, pasadas las primeras fases del proceso de desarrollo. Sin embargo, los requerimientos del negocio son mucho más dinámicos.

Cuando una aplicación tarda dos o tres años en desarrollarse, lo más probable es que al final no cubra las necesidades existentes. Por lo que comienza el mantenimiento.

Una de las razones de la existencia de la arquitectura cliente/servidor es la necesidad de una inmediata adaptación al mundo de los negocios, por lo que el proceso en cascada no es el apropiado, es demasiado rígido y lento para responder al creciente ritmo de cambio de la empresa actual.

Una gran cantidad de estudios han demostrado que más del 70% del tiempo de desarrollo y de los recursos se invierten en mantener las aplicaciones existentes. Esto no incluye solamente corregir los fallos o defectos, sino añadir funciones que deberían haber estado desde el comienzo o funciones que cubran posibles modificaciones en los requerimientos.

Proceso interactivo. La nueva tendencia es utilizar un diseño interactivo, también conocido como *proceso en espiral*. Las herramientas cliente/servidor y la metodología de programación orientada a objetos son muy adecuadas para este tipo de proceso, en el que, una vez terminada la fase de análisis, los programadores trabajan directamente con el usuario, utilizando prototipos que se generan muy rápidamente.



El diseño interactivo permite que el desarrollo de las aplicaciones evolucione incrementalmente de un paso al siguiente, volviendo a comprobar los resultados. La ventaja derivada es una mayor participación del usuario, con el consiguiente aumento de rapidez en la puesta en producción de la aplicación, además de una mayor riqueza funcional. Este proceso permite dar una respuesta más rápida a las necesidades del negocio.

La construcción de prototipos se ha convertido en una de los pasos más importantes en el proceso de diseño de aplicaciones, hasta el punto de que estos componentes no se consideran desechables al final. Por lo que deben utilizarse las mismas herramientas para los prototipos que para el desarrollo de las aplicaciones de producción, y no se debe descuidar el trabajo de diseño y arquitectura de la aplicación. Uno de los retos de los prototipos es convencer al usuario de que el proceso no se termina una vez que ha visto el prototipo del "front-end" de la aplicación. Aquí se aplica la regla 80/20, es decir, que el 80% del trabajo se realiza en el 20% del tiempo, mientras que el 20% de trabajo restante requiere del 80% del tiempo.

Cliente/servidor y orientación a objetos

Cliente/servidor y orientación a objetos (OO) no son elementos necesariamente ligados, las aplicaciones cliente/servidor se pueden diseñar con características que no son orientadas a objetos; de la misma manera, una aplicación orientada a objetos puede no tener ningún atributo cliente/servidor. Sin embargo, los dos conceptos funcionan bien conjuntamente. Actualmente, el desarrollo de aplicaciones cliente/servidor involucra la utilización de la metodología de programación a objetos.

Dentro de la metodología de programación orientada a objetos se identifican varias disciplinas que son:

- Interfaces de usuario (OOUI, *Object Oriented User Interface*) .
- Diseño orientado a objetos.
- Programación orientada a objetos.
- Bases de datos orientadas a objetos.

Interfaces de usuario orientas a objetos (OOUI)

Una OOUI es una clase especial de GUI (*Graphical User Interface*) basada en el concepto de objeto-acción e incorpora las siguientes características:

- El usuario selecciona objetos mediante un mouse y los arrastra hacia el objetivo (drag and drop)
- El usuario obtiene una respuesta inmediata a las acciones seleccionadas.
- Los objetos se presentan a los usuarios en la forma de **WYSIWYG** (del inglés "What you see is what you get").

Como ejemplo de una sintaxis objeto-acción, consideremos la supresión de un archivo, la instrucción típica para el borrado del archivo, en un sistema tradicional, sería *delete <nombre del archivo>*. Primero se especifica la acción y luego el objeto. Con una interfaz de usuario orientada a objetos el usuario borraría el archivo seleccionando el objeto y arrastrándolo hasta un icono con el aspecto de un cubo de basura.

Nótese que una **GUI** no está orientada a objetos por definición. Las primeras aplicaciones que incorporaron la utilización de una **GUI** estaban más orientadas a tarea que a objetos. Una **OOUI** es más intuitiva y productiva para los usuarios. Uno de los criterios de selección de una herramienta **GUI** debería ser si soporta o no, la construcción de una **OOUI**.

Diseño orientado a objetos y programación orientada a objetos (OOP)

La unidad básica de organización en la programación orientada a objetos es un objeto, el cual se compone de propiedades y métodos, es decir, las acciones que se van a realizar sobre los datos. El método se invoca enviando un mensaje al objeto

Lo más importante de la programación orientada a objetos, lo cual es la causa de su utilización en el ambiente cliente/servidor, es su característica de *encapsulamiento*, que permite realizar una programación modular, en donde los módulos son cien por ciento independientes unos de otros.

Debido a la característica de *encapsulamiento* la única forma en que se pueden comunicar los módulos de una aplicación (métodos) es a través del envío de mensajes a los objetos.

Servidores de aplicaciones como objetos

Como se ha explicado anteriormente, una de las características básicas de las aplicaciones cliente/servidor es la encapsulación de los servicios. Un servidor de aplicación puede considerarse un objeto a nivel aplicación aunque puede ser desarrollado con un lenguaje procedural.

No hay necesidad de codificar los objetos del negocio utilizando OOP (*Object Oriented Programming*), sino que pueden desarrollarse utilizando lenguajes 3GL o 4GL.

Aparece de esta forma, una vía de migración natural: comenzar el desarrollo de servidores de negocio encapsulados de una forma tradicional y cuando sea necesario evolucionar hacia herramientas de programación y técnicas orientadas a objetos.

El uso de este enfoque hace posible la migración gradual hacia un entorno orientado a objetos; los componentes de la aplicación construidos con tecnologías procedural y de orientado a objetos pueden coexistir y trabajar conjuntamente en el mismo entorno del sistema.

Herramientas de desarrollo de aplicaciones

Características

La clasificación de las herramientas de desarrollo

La clasificación que es frecuentemente utilizada por los consultores independientes de la industria es:

- 3GL
- Generadores 3GL
- 4GL enlazados con bases de datos
- 4GL independientes

Algunas de las herramientas 4GL tienen atributos de orientación a objetos o programación visual y, a veces, se consideran como herramientas 5GL.

3GL Cobol, C o C++ son ejemplos típicos de lenguajes de programación en esta categoría. Proporcionan buena portabilidad e independencia del proveedor. La desventaja es que su productividad es relativamente baja.

Generadores 3GL ofrecen generalmente una buena productividad, pero pueden requerir algo de programación de bajo nivel en otros lenguajes. Como la mayoría de ellos generan código en COBOL o C estándar, representa una gran independencia del proveedor.

4GL enlazados con bases de datos, ejemplos de esta categoría son ***SQL*Forms de Oracle, APT Workbench de Sybase, Windows 4GL de ingres,*** o ***Progress*** *de progress*. Sus puntos fuertes son su facilidad de uso y de generación de prototipos. El modelo de aplicación que soportan se suele denominar centrado en los recursos (*resource centric*). Estos modelos dependen totalmente de los gestores de recursos remotos para la distribución (desde la perspectiva de la aplicación, la lógica de la aplicación reside en el cliente). Las aplicaciones centradas en los recursos son de dos capas en donde el recurso remoto accedido es una base de datos relacional.

Algunas de estas herramientas soportan únicamente una determinada base de datos, al igual, que en su mayoría, incorporan su propio middleware, existiendo un riesgo potencial de que el usuario quede atado al suministrador o a la base de datos.

4GL Independiente. Algunos productos de esta categoría son VisualAge de IBM, Powerbuilder de Powersoft o Choreographer de Guidance Technology. Combinan la facilidad de uso con la flexibilidad para acceder a los DBMS de distintos proveedores. Las desventaja es que son lenguajes propietarios por lo que existe un riesgo de quedar ligado al suministrador. El uso de estas herramientas puede generar una aplicación en dos capas.

Tendencias

La tendencia es la utilización de la metodología de la programación orientada a objetos. Aunque la tecnología orientada a objetos ya está siendo utilizada por algunos de los productos disponibles en el mercado, sigue siendo relativamente nueva para el desarrollo de aplicaciones de soporte del negocio. Desde la perspectiva de los lenguajes de programación, se puede considerar que la tecnología orientada a objetos tiene raíces en los lenguajes de 3GL y 4GL. C++ evolucionó a partir de C, y el COBOL orientado a objetos evolucionó a partir de COBOL. Ambos están posicionados como los lenguajes de orientación a objetos preferidos por aquellas compañías que están comprometidas con un entorno de tipo 3GL.

De forma similar, *smalltalk* se coloca como un lenguaje preferido por aquellas organizaciones que necesitan un entorno de desarrollo tipo 4GL muy integrado. Así, la tecnología de orientación a objetos ofrece lenguajes de programación que incluyen características tradicionales, tanto de 3GL como de 4GL.

Sistemas distribuidos orientados a objetos

En la actualidad existe una gran cantidad de productos que incorporan la tecnología de orientación a objetos, desafortunadamente cada uno de estos productos controla de forma diferente los objetos, por lo que se imposibilita la interoperabilidad entre ellos. Esto llevará a una situación en la que las ventajas de OO (*Object Oriented*) quedan limitadas. La potencial reutilización de un objeto quedará frustrada si pertenece a un régimen OO distinto del que se quiere reutilizar.

Veamos algunos ejemplos, de entornos, como smalltalk, que presuponen un entorno completo de ejecución. La interacción con programas externos a ese entorno se aparta del modelo puro de objetos, y pueden perderse muchas de las ventajas de esta programación. Otros, como C++, pueden producir unidades fijas entre clientes y objetos. Esto conduce a la necesidad de recompilar cuando se realizan pequeños cambios en los objetos.

Y ¿qué decir sobre la interacción entre un objeto smalltalk y otro de C++?, o de algo tan vital dentro de la filosofía cliente/servidor como puede ser la interacción de un objeto en una plataforma con otro objeto de otra plataforma

Esta situación, es obviamente, poco deseable. Sería mejor disponer de un único sistema orientado a objetos que abarcara todas las posibilidades de OO en el sistema, incluso si éste es distribuido. Un sistema que abarcara un gran número de computadoras conectadas, con objetos, mensajes y métodos funcionando conjuntamente sin las limitaciones impuestas por la infraestructura de orientación a objetos.

El ámbito debería traspasar los límites de los entornos de programación, de los subsistemas y de los sistemas. Esto significa, por ejemplo, que un objeto podría enviar un mensaje a otro objeto que estuviese en una máquina diferente, y que los objetos podrían interoperar, incluso si estuvieran escritos en diferentes lenguajes de programación OO

SOM (System Object Model) de IBM, proporciona un entorno transparente al lenguaje y que trata de resolver estos problemas. Igualmente existe DSOM (Distributed System Object Model). El objetivo es que los objetos empaquetados puedan ser accedidos desde cualquier punto de la red y por cualquier lenguaje de programación.Tanto SOM como DSOM fueron desarrollados cumpliendo las especificaciones CORBA (Common Objectc Request Broker Architecture).

Selección de las herramientas de desarrollo

Tecnología

Tecnología probada a nuevas, éste es un criterio básico de decisión. Aunque la programación orientada a objetos es muy prometedora, no todas las empresas están preparadas para utilizarla. Educar a los programadores en la tecnología orientada a objetos requerirá de una sustancial inversión. Al mismo tiempo, existe una presión por construir aplicaciones cliente/servidor con una interfaz gráfica de usuario. Esto también requiere nuevos conocimientos por parte de los programadores. Debe considerarse seriamente su formación con herramientas GUI orientadas a objetos:

☞ Estas herramientas son muy adecuadas para desarrollar interfaces gráficas de usuario. No sólo aíslan al programador de la complejidad de diseñar una GUI, sino que proporcionan un entorno de desarrollo integrado que incluye un conjunto de herramientas complementarias, tales como editores, depuradores y de utilerías gráficas para el diseño de las "pantallas". Algunas de ellas permiten una construcción visual a partir de los componentes, lo que redunda en una mayor productividad.

☞ Cuando la arquitectura y el diseño de las aplicaciones cumplen con la separación entre la lógica de presentación, de negocio y de datos. La lógica de datos y del negocio puede desarrollarse de una forma más tradicional, por lo que solamente los programadores responsables de desarrollar la GUI tendrán que ser formados en las nuevas herramientas.

Por lo tanto, la tecnología orientada a objetos debe introducirse de un modo evolutivo

Componentes de tiempo de ejecución

Una gran cantidad de herramientas ofrecen funciones que se encuentran soportadas por el middleware. Si éste es el caso, se deberá tener cuidado en los siguientes aspectos:

- ¿Es el middleware propietario o se utilizan servicios de distribución que cumplen estándares de la industria?
- ¿Qué modelos de distribución cliente/servidor están soportados?
- ¿Permite la herramienta crear aplicaciones de dos y tres niveles?
- ¿Cuál es el nivel de integración soportado? o ¿se trata de una herramienta abierta para acoger nuevas tecnologías, por ejemplo multimedia, programación orientada a objetos?

Elección de proveedores

Comprometerse con una sola herramienta crea dependencia de un único proveedor. En cuanto se empiece a utilizar será muy difícil salir de ella debido a la gran inversión realizada. Sería aconsejable, por lo tanto, utilizar un conjunto de las mejores herramientas para cada propósito específico.

En cualquier caso la elección de un proveedor debe basarse en los criterios:

- Aspectos técnicos.
- Presencia y soporte en el país.
- Aceptación en el mercado.
- Posibilidad de continuidad en el futuro.

Desarrollo en grupos de trabajo

La mayoría de las aplicaciones están desarrolladas por un equipo en lugar de una sola persona. Las herramientas utilizadas deben permitir que más de una persona trabaje al mismo tiempo en la misma aplicación. Ello también implica que deba existir algún tipo de librería de desarrollo, en la que sea posible almacenar la entrada y salida de la herramientas de desarrollo. Esta librería necesita tener funciones de versión. Todos estos aspectos se dan por supuesto cuando se desarrollan aplicaciones tradicionales, pero tienden a ignorarse cuando se trata de un ambiente cliente/servidor.

Apertura frente a productividad

Uno de los dilemas de la elección de herramientas estriba en que la mayoría de las que ofrecen una elevada productividad son también de carácter propietario. Hay una relación inversa entre apertura y los desarrollos propietarios. Probablemente cuando más productiva sea la herramienta, más cerrado y propietario será su carácter y, por lo tanto, mayor la dependencia del proveedor. El remedio no es recurrir a API de bajo nivel para conseguir el máximo nivel de apertura. Algunos factores a tener en cuenta son:

- Una plataforma de sistemas y aplicaciones claramente definida.
- Definición de estándares, formatos y protocolos soportados.

☞ Una arquitectura de aplicación que también incluya: plataforma operativa y desarrollo de aplicaciones, delimitando el nivel permitido de características de tipo propietario.

☞ Principios de diseño de aplicaciones que definan cómo se diseñan y desarrollan las aplicaciones cliente/servidor.

No utilizando explícitamente funciones específicas del producto y desarrollando conscientemente ciertas funciones en **servicios de aplicación**, puede llegarse a un compromiso para conseguir un alto nivel de productividad sin crear dependencia de un determinado proveedor.

SISTEMAS HEREDADOS (LEGACY SYSTEMS)

Como la mayoría de las empresa han realizado una importante inversión en soluciones informáticas, muy raramente se pueden construir nuevos sistemas abandonando los ya existentes. Las inversiones se contemplan generalmente en términos de hardware instalado, pero es probable que sean más importante las inversiones realizadas en :

☞ Software de sistemas

☞ Datos disponibles en la empresa

☞ Aplicaciones.

☞ Conocimientos en el departamento de sistemas.

☞ Conocimientos de los usuarios.

Todos éstos son activos fundamentales que frecuentemente se olvidan o se deprecian. Cliente/servidor posibilita una migración paso a paso, es decir, la generación de nuevas aplicaciones que coexistan con las ya existentes, protegiendo de esta forma las inversiones realizadas por la empresa.

Trabajando con datos heredados

Existen tres opciones para tratar a los datos ya existentes:

✓ Convertir todos los datos hacia el nuevo sistema.
✓ Mantener todos los datos en el sistema heredado, mientras se crean las aplicaciones en el nuevo sistema.
✓ Utilizar una combinación de datos del sistema heredado y del nuevo sistema para la creación de las aplicaciones

Estrategias de conversión de datos heredados

☞ **Datos homogéneos**

- *Conversión de datos*. Es necesaria la conversión física de los datos del sistema heredado hacia el nuevo sistema . Todos los datos residen en el nuevo sistema.

☞ **Datos heterogéneos**

- *Gateways*. Los datos residen exclusivamente en el sistema heredado. Las aplicaciones son procesadas en el cliente y los datos son accesado vía gateways.

- *Sincronización por medio de procesos en batch.* Creación de datos en el nuevo sistema, manteniendo los datos en el sistema heredado. En este caso los datos residen en sistema heredado como en el nuevo sistema, y la información común es "refrescada" en intervalos específicos de tiempo.

- *Sincronización en tiempo real.* Creación de datos en el nuevo sistema, manteniendo datos en el sistema heredado. Al igual que en caso anterior, los datos residen tanto en el sistema nuevo como en el heredado, pero los datos comunes son mantenidos en tiempo real.

Factores a considerar para la toma de decisión en la migración

Los principales factores a considerar en la migración de un sistema hacia un ambiente cliente/servidor son:

- Autonomía
- Sistemas heredados
- Necesidades del negocio
- Costo de transformación
- Facilidad de la conversión
- Temporabilidad de la información
- Impacto de la transformación
- Necesidad de los datos heredados

A partir de los factores anteriores se crea un *patrón de estrategia de migración.* Este patrón de estrategia es un conjunto de modelos basados en un escenario "ideal" para cada una de las estrategias de migración, es una guía por lo que no es definitiva, es decir, se deberá adecuar a cada empresa en forma específica. A continuación se explican los criterios de evaluación y su significado a partir de un promedio que va del valor de -5 a 5.

Criterio de Evaluación	Rango de -5 a 0	Rango de 0 a 5
Autonomía	La consolidación de datos es el propósito de la implementación del ambiente cliente/servidor.	La autonomía es el propósito de la implementación del ambiente cliente/servidor.
Sistemas heredados	Existe una considerable inversión en el sistema heredado.	No existe una inversión a recuperar por parte del sistema heredado.
Necesidades del negocio	La arquitectura cliente/servidor inicialmente sólo agrega nueva funcionalidad a l sistema existente.	La arquitectura cliente/servidor es originada por nuevos requerimientos de información.
Costo	Las necesidades del negocio son mayores a las restricciones de costo.	El costo del sistema frena la implementación del ambiente Cliente/Servidor.
Transformación	Existe una fuerte relación entre los tipos de datos existentes y la estructura orgánica existente y las nuevas necesidades.	No existe o es muy escasa la relación entre los tipos de datos y estructura existente y los nuevos requerimientos.
Temporabilidad de los datos	Los datos son estáticos o las necesidades del negocio no requieren del manejo de información.	Los datos son dinámicos y las necesidades del negocio requieren del uso de la información.
Impacto de la transformación	Cambios en los datos almacenados en el sistema heredado no impactan en los datos almacenados en el nuevo sistema.	Los cambios en los datos heredados afectan directamente a los datos almacenados en el nuevo sistema.
Necesidad de los datos heredados	Las aplicaciones pueden continuar ejecutándose sin los datos heredados.	Las aplicaciones requieren de los datos heredados para poder operar.

El análisis de los criterios de evaluación da como resultado distintas estrategias de migración, en el siguiente tema se establecen las características de las estrategias posibles de implantar

Modelos de estrategias de migración

Modelo de conversión de datos

Este modelo de migración especifica que el sistema heredado no es esencial para el nuevo ambiente cliente/servidor, pero en cambio los datos deben ser migrados totalmente al nuevo ambiente.

El perfil que recomienda esta técnica es como se muestra en la tabla 3.1.

Criterio de Evaluación	Grado	Consideraciones
Autonomía	+5	La autonomía es un elemento crítico para el éxito de la implementación del ambiente cliente/servidor
Sistemas heredados	+5	No existe una inversión recuperable por parte del sistema heredado.
Necesidades del negocio	+5	El ambiente cliente/servidor involucra nuevos requerimientos de datos.
Costo	+2	El costo del sistema es considerable como una restricción en la implementación.
Transformación	+3	Existe muy poca relación entre los nuevos datos y los ya existentes.
Temporabilidad de los datos	+5	Los datos deben encontrarse disponibles en todo momento.
Impacto de la transformación	+5	Los datos deben mantenerse actuales en todo momento.
Necesidad de existencia	+5	Sin los datos heredados,el sistema no es capaz de funcionar.

Tabla 3.1 Modelo de conversión de datos

Modelo gateway

Cuando se ha realizado una importante inversión (tiempo, dinero, entrenamiento, personal) en el sistema heredado y el ambiente cliente/servidor únicamente ofrece mayor funcionalidad a la empresa, no será posible prescindir del sistema heredado por lo que deberá mantenerse una liga con los datos heredados.

La tabla 3.2 muestra el perfil que recomienda la implantación de esta opción.

Criterio de evaluación	Grado	Consideraciones
Autonomía	-5	La consolidación de los datos es el factor crítico para el éxito de la implementación del ambiente cliente/servidor.
Sistemas heredados	-5	Existe una considerable inversión realizada sobre el sistema heredado.
Necesidades del negocio	-5	La implementación del ambiente cliente/servidor inicialmente sólo agrega funcionalidad a los datos existentes.
Costo	-3	Las necesidades del negocio son mayores a cualquier restricción de costo.
Transformación	-2	Es mediana la necesidad de relación de las aplicaciones y datos ya existentes y los nuevos requerimientos.
Temporabilidad	+5	Los datos se deben encontrar disponibles en todo momento.
Impacto	+5	Los datos deben encontrarse actualizados en todo momento.
Necesidad de existencia	+5	Sin los datos heredados el sistema fracasará.

Tabla 3.2 Modelo gateway

Modelo de sincronización en batch

Cuando existe una importante inversión realizada sobre el sistema heredado y el nuevo ambiente cliente/servidor incorpora nuevos requerimientos de datos, entonces será necesario mantener copias de los datos heredados en el nueva ambiente. La sincronización de la información se mantiene a través de un proceso batch, que se ejecute periódicamente (diario, semanal, mensual).

La tabla 3.3 muestra el perfil recomendado para el uso de esta técnica.

Criterio de Evaluación	Grado	Consideraciones
Autonomía	+5	La necesidad principal es la autonomía del negocio.
Sistema heredados	-5	Existe una considerable inversión realizada sobre el sistema heredado.
Necesidades del negocio	+2	El ambiente cliente/servidor inicialmente agrega nueva funcionalidad a los datos existentes, pero incorpora nuevos requerimientos de datos.
Costo	+2	El costo representa una leve restricción.
Transformación	-2	Existe algo de relación entre los datos y aplicaciones ya existentes y los nuevos requerimientos.
Temporabilidad	-5	Las necesidades del negocio no requieren de la manipulación de datos.
Impacto	-5	Cambios en los datos heredados no tienen impacto en los datos almacenados en el nuevo sistema.
Necesidad	-5	El nuevo sistema puede operar sin la necesidad de los datos heredados.

Tabla 3.3 Modelo de sincronización en batch

Modelo de sincronización en tiempo real

Este modelo es semejante al de sincronización e batch, su única diferencia es que las copias de los datos heredados se mantienen sincronizadas en tiempo real.

La tabla 3.4 muestra el perfil que recomienda la utilización de esta técnica.

Criterio de evaluación	Grado	Consideraciones
Autonomía	+5	La necesidad es la autonomía del negocio.
Sistema heredado	-5	Existe una considerable inversión en el sistema heredado.
Necesidades del negocio	+2	Inicialmente el ambiente cliente/servidor incluye una nueva funcionalidad para los datos existentes y agrega nuevos requerimientos de datos.
Costo	+5	El costo del ambiente es un factor crítico.
Transformación	-2	Existe una mínima relación de las aplicaciones y datos existentes y los nuevos requerimientos.
Temporabilidad	+5	Las necesidades del negocio requieren que los datos se encuentren actualizados en todo momento.
Impacto	+5	Los cambios en los datos heredados tienen un significativo impacto en los datos almacenados en el nuevo sistema.
Necesidad de existencia	-2	El nuevo sistema requiere de parte de los datos heredados.

Tabla 3.4 Modelo de sincronización en tiempo real

PASOS PARA UNA MIGRACIÓN EXITOSA HACIA UN AMBIENTE CLIENTE/SERVIDOR

Primer paso. *Identificación de los riesgos*

Los principales desafíos a los que se enfrentan los directores de departamento de sistemas al migrar a un ambiente cliente/servidor podrían caer en alguno de los siguientes aspectos:

- ✓ Demostración del valor del sistema para el negocio
- ✓ Control de los riesgos de la migración
- ✓ Hacer frente a las complejidades
- ✓ Mantener la autonomía
- ✓ Entender el plan del negocio

A continuación se explicará cada uno de estos puntos y se recomendarán estrategias para resolver dichos desafíos.

❑ *Demostrar el valor del sistema para el negocio*

Para poder realizar esta demostración es necesario tomar en cuenta los siguientes puntos:

☞ Validar la decisión de la migración hacia un sistema cliente/servidor.

☞ Demostrar resultados lo más rápidamente posible.

☞ Continuar entregando de una manera oportuna resultados de acuerdo al avance del proyecto.

Algunas de las estrategias factibles para lograrlo son:

☞ Ligar la tecnología con el plan del negocio.
- La tecnología utilizada debe cubrir completamente las necesidades del negocio.

☞ En cada una de las fases del proyecto deben establecerse objetivos en términos de las necesidades del negocio, dichos objetivos deben ser establecidos antes del inicio del proyecto.

❑ *Controlar los riesgos de la migración*

Los principales riesgos a enfrentar son:

☞ Evolución de la tecnología.

☞ Complejidad de las tareas de administración (interconectividad, configuración de equipos, etc.).

☞ Curva de aprendizaje de la tecnología cliente/servidor.

Las estrategias para el control de estos aspectos podrían ser las siguientes:

Desarrollar arquitecturas de negocio y tecnología razonables, al desarrollar estas arquitecturas debemos mantener en mente que:

☞ La simplicidad es el principal determinante para el éxito.

☞ Para cada una de las arquitecturas será necesario estimar los tiempos y costos.

☞ El contexto del negocio es el principal determinante de la tecnología a utilizar.

A partir de los sistemas existentes (si es que existen) es posible:

☞ Realizar la planeación de las fases de migración hacia un ambiente cliente/servidor y los beneficios de negocio que se deriven de esta implementación determinarán la prioridad de la migración.

☐ *Hacer frente a las complejidades*

Las complejidades por considerar son:

✓ Los cambios en productos y vendedores que existen en el mercado.
✓ Organización de personal, de procesos y tecnología.

Las estrategias para hacer frente a estas problemas podrían ser :

☞ Crear equipo de trabajo formados por expertos, para esto se recomienda:

- Entrenamiento para las personas que formarán los equipos.
- Contratar calificados consultores externos.
- Mantener los equipos de trabajo tanto tiempo como sea necesario.

☞ Utilizar métodos prácticos, ya probados, para esto será necesario basarse en la experiencia de otros.

☐ *Mantener la autonomía*

El mantener la autonomía implica.

☞ Desarrollo de aplicaciones por parte del personal de la compañía.

☞ Entrenamiento del departamento de sistemas.

☞ Políticas de responsabilidad interdepartamentales.

Posibles estrategias que permiten la existencia de la autonomía:

✓ Diseño de un plan de entrenamiento. Es necesario ofrecer entrenamiento con el fin de obtener las habilidades y conocimientos necesarios para la realización de las tareas dentro del negocio.

✓ Crear un perfil de habilidades para la contratación y reclutamiento de nuevos empleados.

Segundo paso. *Entender el plan del negocio*

Para la implementación de un ambiente cliente/servidor es necesario realizar un documento detallado sobre el plan del negocio. El objetivo es documentar la misión, funciones y organización del negocio. Es necesario el desarrollo de un anteproyecto del negocio, en donde se muestren las perspectivas, roles, procesos, actividades, medidas de rendimiento y necesidades de información. A partir de este análisis, se decidirá si es necesario (o no) rediseñar la organización del negocio.

A través del anteproyecto, los directivos de una empresa se encontrarán mejor preparados para:

- Priorizar las iniciativas de reingeniería.
- Identificar áreas en donde se tenga un rápido mejoramiento al realizarse un proceso de reingeniería.
- Identificar las áreas con mayores índices de recuperación de inversión.

Tercer paso. *Desarrollar una clara definición de la estructura de la empresa*

El prerrequisito para la definición de la estructura de la empresa es un análisis completo de sus procesos, y el objetivo es convertir las estrategias y procesos del papel hacia una implementación de un ambiente cliente/servidor.

El análisis de la estructura permite identificar:

- ¿Qué es lo que necesitamos?
- ¿Cuándo lo necesitamos?
- ¿Cómo lo lograremos?

Un buen análisis de la estructura de la empresa es importante para evitar sobrepasar los costos establecidos. Existen cuatro áreas interrelacionadas:

- **Datos**

  Organiza la fuente y almacenamiento de la información del negocio.

- **Aplicaciones**

  Desarrollo del software requerido para cubrir la funcionalidad del negocio.

- **Tecnología**

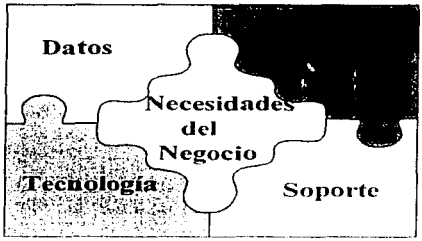
  La infraestructura de cómputo utilizada para el control de los datos y aplicaciones.

- **Soporte**

  Es la interfaz con la empresa, incluye mantenimiento del sistema, soporte técnico, planes de crecimiento y de migración, etc.

Estas áreas cubren las necesidades de la empresa.

ÁREAS



Basándose en los requerimientos del negocio, la estructura provee los siguientes elementos:

- Todas las actividades que deben ser realizadas.
- Dependencias e interrelaciones.
- Toda la información que requiere ser capturada.
- Planes de soporte para aplicaciones corporativas, incluyendo a los sistemas heredados como a los nuevos sistemas.

Los beneficios de contar con una estructura son, por mencionar algunos:

- ✓ Permite adaptaciones de acuerdo con los cambios en los requerimientos del negocio y de la tecnología.
- ✓ Facilita el desarrollo ordenado de una infraestructura.
- ✓ Alienta al desarrollo sistemático y por etapas de las aplicaciones.
- ✓ Permite identificar las ineficiencias entre los departamentos u organizaciones.

Fases en el desarrollo de la estructura

Para el desarrollo de las áreas es necesario realizar un proceso que consta de cinco fases, a continuación se definen cada una de éstas, así como también se especifican las tareas por realizar en cada una de ellas.

Fase I
Diagnóstico de la estructura actual de la empresa

El propósito es revisar y validar los requerimientos del negocio en términos de la estructura existente. El documento resultante de este análisis debe de incorporar :

- La documentación del alcance y objetivos de la empresa.
- Funciones del negocio.
- Restricciones.
- Determinación de los factores críticos para el éxito.

☞ Estimar duración y nivel de esfuerzo.

☞ Valorar los componentes existentes.

☞ Identificar las aplicaciones.

En el momento de la documentación es necesario tomar en cuenta los siguientes aspectos:

✓ Estabilidad de los requerimientos de la empresa.
✓ Madurez de los procesos existentes.
✓ Dinámica de la empresa.
✓ Riesgos y costos escondidos.

Las tareas que son necesarias para la creación de las áreas se mencionan a continuación.

Sistemas de datos existentes

Tareas

❑ Documentar el volumen de datos existentes.
❑ Documentar la localización.
- Datos almacenados en forma magnética o bien en papel.

❑ Analizar el volumen actual de datos.
❑ Valorar la integridad actual de los datos.
- Documentar cualquier problema resultante de la pérdida de datos o bien de datos incorrectos.

Aplicaciones existentes

Tareas

❑ Revisar las aplicaciones.
- Identificar las principales aplicaciones en el actual sistema.
- Identificar los procesos de negocio soportados por dichas aplicaciones.

❑ Documentar las dependencias críticas e interfaces.
- Documentar cómo el tiempo y el flujo de información afecta a estos procesos.

❑ Determinar cuándo y dónde la entrada de datos se encuentra disponible.

❑ Determinar cuándo y dónde las salidas son producidas.

❑ Analizar el funcionamiento de las aplicaciones
- Documentar dónde corren las aplicaciones.
  - En el mainframe
  - Localmente
- Documentar servicios de aplicación de terceros.

❑ Analizar las características de rendimiento de las aplicaciones
- Documentar la respuesta en horas pico y en horas de actividad normal en las transacciones más críticas para cada aplicación.

- ❑ Valorar el actual nivel de servicio.
  - Documentar la disponibilidad y estabilidad del actual sistema.
  - Identificar el ambiente de soporte de las aplicaciones.

Tecnología existente

Tareas

- ❑ Documentar los componentes del sistema.
  - Identificar los componentes de hardware y software del actual sistema.
  - Documentar las características fundamentales del sistema.
    - Capacidad
    - Tolerancia a fallas
    - Seguridad
- ❑ Valorar el nivel de servicio del actual sistema.
- ❑ Documentar los problemas en la infraestructura.
  - Documentar cualquier problema existente con la tecnología actual; por ejemplo, antigüedad, costo de mantenimiento y limitantes de capacidad, por mencionar algunas.

Soporte existente

Tareas

- ❑ Revisar los roles del departamento de soporte
  - Determinar la estructura actual para el soporte del sistema.
  - Determinar las principales funciones de soporte.

- ❑ Documentar las funciones de administración del sistema.
  - Determinar el procedimiento utilizado para la distribución del software, tareas de configuración, administración de la red y soporte al software.
  - Documentar cualquier problema causado por un soporte inadecuado.

- ❑ Valorar el alcance del actual soporte.
  - Documentar los tiempos para cada una de las funciones principales de soporte.

Fase II
Establecer los requerimientos de la estructura de la empresa

El propósito de esta fase es convertir los requerimientos del negocio en requerimientos de la estructura, el resultado de esta etapa entrega:

☞ Documentación de la descripción de los procesos de la empresa.

☞ Determinación de los requerimientos de la estructura de la empresa.

Este documento permite iniciar una investigación sobre tecnologías que cubran los requerimientos de la empresa.

Las consideraciones importantes en esta fase son:

☞ Considerar la capacidad de la infraestructura.

☞ Independencia en la tecnología.

☞ Interacción entre los procesos y datos.

☞ Soporte y mantenimiento.

☞ Duración y nivel de esfuerzo.

Las tareas necesarias para cada una de las áreas se muestran a continuación.

Requerimientos de datos

Tareas

❑ Documentar el modelo de datos.
- Describir la estructura de datos y las relaciones.
- Definir cualquier nuevo requerimiento.

❑ Documentar la ubicación de los datos.
- Determinar la probable ubicación del almacenamiento de los datos.

❑ Documentar el volumen de datos esperados.
- Estimar cuántos datos existirán.
- Estimar la razón de cambio de los datos.

❑ Documentar los requerimientos de sincronización.
- Determinar con qué frecuencia los datos deben ser sincronizados.
- Determinar qué técnica será utilizada para la sincronización.

❑ Definir los requerimientos de la integridad de datos.

Requerimientos de las aplicaciones

Tareas

❑ Documentar la naturaleza de la estructura de las aplicaciones.
- Describir los aspectos críticos.
  - Procesamiento de las transacciones.
  - Ambiente de toma de decisiones (DSS).
  - Control de procesos de tiempo real.
  - Interactividad - multimedia.

❑ Definir las dependencias críticas.
- Identificar cualquier requerimiento de tiempo.
- Documentar los requerimientos del flujo de información.
- Describir las interfaces de usuario críticas.

- ❑ Documentar los requerimientos de procesamiento.
  - Determinar dónde correrán las aplicaciones.
    - En el mainframe
    - Localmente

- ❑ Documentar los requerimientos de rendimiento de las aplicaciones.
  - Estimar el tiempo de respuesta en horas pico y de actividad normal para las transacciones más críticas en cada aplicación.

- ❑ Definir el nivel de servicio esperado.
  - Determinar los requerimientos de disponibilidad y estabilidad del sistema.
  - Determinar el ambiente de desarrollo más óptimo para la programación de las aplicaciones.

Requerimientos tecnológicos

Tareas

- ❑ Identificar los principales componentes del sistema.
  - Determinar los requerimientos de hardware y software necesarios para los clientes, servidores y la red.
  - Identificar cualquier servicio especial que el sistema pueda requerir, por ejemplo:
    - Alta capacidad de almacenamiento.
    - Procesamiento de voz e imagen.
    - Tolerancia a fallas.
  - Identificar plataformas que soportan el volumen de datos requerido y el número de transacciones en las horas pico.

- ❑ Revisar la disponibilidad de productos en el mercado que cumplan con los requerimientos.
  - Determinar la existencia y disponibilidad de la tecnología necesaria para el sistema.
  - Documentar la tolerancia a fallas de los componentes (y recuperación en caso de fallas).
  - Documentar aspectos de compatibilidad.

- ❑ Valorar la factibilidad de los requerimientos de la estructura.
  - Basado en las investigaciones de mercado, determinar si los requerimientos de la estructura son demasiado "agresivos", complejos, o bien, costosos de cubrir.
  - Determinar si la estructura puede ser implementada dentro del tiempo requerido.

Requerimientos de soporte

Tareas

- ❑ Definir los roles de soporte.
  - Identificar los principales roles de soporte, requeridos para la nueva estructura.
  - Determinar las habilidades necesarias para cubrir los roles.
  - Determinar el entrenamiento requerido para el equipo de soporte.

- ❑ Documentar los requerimientos de la administración del sistema.
  - Documentar los principales requerimientos de soporte para el nuevo sistema.
  - Determinar cualquier habilidad especial necesaria para el mantenimiento del sistema.
  - Describir cualquier hardware (especial) o requerimiento de software para mantener funcionando el sistema.

Fase III
Estructura lógica

El propósito de esta fase es crear un modelo preliminar que muestre la tecnología seleccionada de acuerdo con los requerimientos y objetivos del negocio.

Los resultados de esta etapa deben determinar y documentar los siguientes aspectos:

- Modelo lógico.
- Estrategias de migración.
- Secuencia de implementación.
- Anteproyecto del sistema.
- Duración y nivel de esfuerzo de migración.

Las consideraciones importantes en esta fase son:

☞ La escalabilidad y expansibilidad del sistema.

A continuación se establecerán las tareas necesarias para cada una de las áreas.

Datos

Tareas

- ❑ Presentar un modelo lógico de datos detallado.
  - Definir un modelo de "alto-nivel" basado en los procesos del negocio para:
    - Datos
    - Relación entre los datos

- ❑ Diseñar un modelo físico de datos detallado.
  - Definir una estrategia de almacenamiento de datos.
  - Diseñar una estrategia de sincronización de datos.

- ❑ Definir detalladamente los datos.
  - Diseño de la base de datos.

- ❑ Definir los mecanismos para la preservación de la integridad de los datos.
  - Describir cómo el sistema garantiza que los datos sean correctos y completos.

Aplicaciones

Tareas

- ❑ Diseño del modelo de procesos para la aplicación.
  - Incluyendo nombres, funciones, características, dependencias, entradas y salidas.
  - Definir niveles de seguridad.
  - Diseñar una estrategia para la introducción sisternática de las aplicaciones.
- ❑ Desarrollar una descripción detallada de cada aplicación.
- ❑ Definir una estrategia de migración de las apiicaciones.
  - Restricciones específicas del diseño.
- ❑ Definir una estrategia para considerar los aspectos de "rendimiento" de las aplicaciones.
  - El diseño debe considerar los requerimientos esperados de tiempo de respuesta y volúmenes de transacción.
- ❑ Diseñar una estrategia de servicio.

Tecnología

Tareas

- ❑ Describir el modelo de la infraestructura del sistema.
  - Funciones y características del hardware y software.
  - Determinar la carga esperada en la red.
- ❑ Diseñar la estrategia de servicio.
  - Recursos
  - Datos centrales
  - Personal

Soporte

Tareas

- ❑ Diseñar estrategias de administración de sistemas.
  - Especificar procedimientos para la distribución del software, monitoreo del rendimiento, control de fallas, y recuperación en caso de desastre (pérdida de la información).
  - Especificar procedimientos de control del ambiente de producción y niveles de seguridad.
  - Diseñar manuales de referencia y manuales de operación.
- ❑ Desarrollar planes de entrenamiento y reclutamiento de personal.

Fase IV
Representación de la estructura física

El objetivo es la construcción de la estructura. Para esto se mapean los componentes de la estructura lógica hacia los componentes, bases de datos y soporte específicos para su implementación.

Los resultados de esta fase determinan y documentan:

☞ Un prototipo.

☞ Una metodología de desarrollo de aplicaciones.

☞ La implementación del ambiente inicial del tipo cliente/servidor.

Las consideraciones importantes son:

☞ Seleccionar aplicaciones para probar la áreas.

☞ Seleccionar aplicaciones con resultados significativos.

Datos

Tareas

- ❑ Producir el modelo físico.
  - Mapear el modelo lógico hacia el modelo físico requerido para soportar las aplicaciones.
- ❑ Producir una fragmentación del esquema.
  - Mapear las funciones de la aplicación contra los componentes del modelo de datos.
- ❑ Producir un esquema de distribución de espacio requerido.
  - Mapear tablas y columnas del modelo lógico contra los componentes de la arquitectura tecnológica.
- ❑ Producir la definición física de la base de datos.
  - Definir llaves, índices y *constraints* para la integridad referencial.
- ❑ Definir estrategias de migración de datos.
  - Definir una estrategia para la migración de datos entre el nuevo ambiente de bases de datos y el sistema de archivos o bases de datos existente.

Tecnología

Tareas

- ❑ Producir la configuración de red.
  - Definir la distribución final de nodos clientes y servidores.
  - Especificar mecanismos para garantizar la seguridad y disponibilidad de los nodos.

- ❑ Configuración de la plataforma del cliente y del servidor.
  - Asignar configuraciones de hardware y software.

- ❑ Configuraciones de los servidores dentro de un esquema de replicación.
  - Determinar el número de servidores en la replicación.
  - Especificar la configuración física final.

Soporte

Tareas

- ❑ Programar procedimientos para la administración.
  - Bibliotecas de procedimientos.
  - Distribución de software, control de versiones y licencias.
  - Monitoreo de rendimiento.
  - Tolerancia a fallas.
  - Recuperación en caso de desastre.

- ❑ Producir un plan de soporte.
  - Producir un plan detallado para el soporte ofrecido por el staff de soporte.
  - Desarrollar planes de entrenamiento.

Fase V
Implementación

Estructura física final

Esta es la fase donde la implementación da inicio.

Tareas

- ❑ Examinar el modelo de datos para garantizar que representa totalmente el dominio de la empresa.

- ❑ Ajustar detalles del diseño de la estructura de la empresa
  - Confirmar la exactitud del volumen de datos esperado para las principales tablas.
  - Probar que los datos replicados se encuentren correctamente sincronizados.
  - Documentar cualquier posible causa de generación de inconsistencias en la base de datos.

Para la puesta en producción es necesario:

- ❑ Entrega de la estructura en forma incremental.

- ❑ Montar dos líneas de entrega.
  - Entrega de la estructura.
  - Entrega de las aplicaciones.

- ❑ Los ciclos de entrega deben basarse en la evolución del ciclo anterior.
  - Cada ciclo debe basarse en la evaluación de qué tanto la estructura cumple los requerimientos.

- ❑ Proveer de soporte a nuevas aplicaciones.
  - La estructura deberá soportar el desarrollo de nuevas aplicaciones sin la necesidad de realizar cambios.

HERRAMIENTAS DE AYUDA PARA LA IMPLANTACIÓN DEL AMBIENTE CLIENTE/SERVIDOR

Comprender la arquitectura cliente/servidor es simple, clientes que envían peticiones de servicio y servidores que responden. Pero, debajo de la superficie, la arquitectura es enormemente compleja, como ya lo vimos. Los desarrolladores (analistas, programadores, etc.) deben navegar a través de un intrincado laberinto de difíciles decisiones, por lo que a continuación se dan dos herramientas: *client/sever infrastructure road map* y la *autoevaluación*, ambas son un intento de asistencia para simplificar la complejidad del ambiente.

El *road map* ofrece una vista global de la arquitectura y junto con esto se da un grupo de posibles alternativas de tecnologías para cada uno de los elementos que forman la arquitectura.

La *autoevaluación* es un medio para determinar las áreas de entrenamiento y de riesgo en una implementación cliente/servidor. Las habilidades y conocimientos identificados para la implementación de ambiente clliente/servidor son divididas en seis áreas, las cuales se explican en la autoevaluación del ambiente cliente/servidor.

CONCLUSIONES

El ambiente actual de las empresas se encuentra en constante cambio, por lo que éstas deben contar con sistemas de información capaces de adaptarse de la forma más rápida posible. En otras palabras, nos encontramos en una época en la cual es necesario tener la capacidad de aproximación de los productos y servicios a la medida de las necesidades del cliente, produciéndolos y suministrándolos con rapidez y mínimos costos.

La capacidad de cómputo de las empresas no debe limitar las posibilidades de crecimiento de la misma, es decir, a la par que crecen las necesidades del negocio es necesario aumentar la capacidad de procesamiento de su plataforma de información. En los años setenta y principios de la década de los ochentas era frecuente construir el organigrama de la empresa y de los procesos en función de la capacidad de procesamiento de sus recursos de cómputo. Actualmente, sucede lo contrario (o debería); los recursos de cómputo deben ajustarse a las necesidades de la empresa. La utilización de poderosas estaciones de trabajo y software de alta productividad (interfaces gráficas y amigables) hacen posible que las funciones que antes se situaban en el mainframe puedan ahora realizarse donde tienen lugar los procesos de negocio que las utilizan. Adicionalmente, se crea una tendencia a utilizar soluciones de sistemas abiertos. La mayoría de los proveedores anuncian que sus productos son abiertos, que permiten accesar a datos y aplicaciones independientemente del sistema operativo y del protocolo de red, esto da lugar a la implantación de ambientes multivendedores.

El modelo cliente/servidor permite una clara separación de las funciones basándose en el concepto de servicio, posibilita situar las aplicaciones en las plataformas más adecuadas, dentro de un entorno de sistemas abiertos y heterogéneos, optimizando así los beneficios de la empresa. Además, se podrán redistribuir las funciones y datos en caso de que cambie la organización o los procesos de negocio.

A continuación se enlistan los objetivos más importantes de cliente/servidor:

Localización transparente
El servidor es un proceso que puede residir en la misma máquina que la del proceso cliente o en otra diferente de la red. El usuario no necesita saber dónde se encuentran situados los servicios, las aplicaciones o los datos. Del mismo modo, la aplicación cliente se codifica con independencia de la localización de los servidores. Es a través del **middleware cliente/servidor** que los clientes obtienen los servicios necesarios para localizar e interoperar con los servidores.

Recursos compartidos
Un servicio puede servir a muchos clientes al mismo tiempo, controlando, además, el acceso a los recursos compartidos.

Escalabilidad de las aplicaciones
El objetivo es lograr que las aplicaciones puedan ser escaladas horizontal o verticalmente. Escalabilidad horizontal significa añadir o suprimir estaciones de trabajo/procesadores. Escalabilidad vertical significa migrar a servidores mayores o menores, o de un tipo diferente. Esto puede ser necesario por cuestiones de cápacidad, o porque las funciones y datos deban de ser redistribuidas a procesadores distintos, o bien, debido a un rediseño de los procesos de negocio. Para conseguir esto es necesario que el código de la aplicación sea portable. La elección del hardware para ejecutar la aplicación, o parte de ella, debe poder realizarse en tiempo de ejecución, y no en el momento del diseño e implantación.

Interoperabilidad
Permite que los usuarios trabajen con aplicaciones y datos distribuidos en distintos procesadores de una red que pueden no tener una arquitectura similar. El middleware ideal es independiente de las plataformas de hardware o de los sistemas operativos. El usuario debe poder combinar distintos procesos cliente y servidor en diferentes plataformas que interactúen entre sí, ya sean del mismo o de distintos proveedores.

En la actualidad, los sistemas de información tienden a distribuir el procesamiento a través del uso de una red, en la cual se encuentren conectadas las computadoras de un empresa -mainframes, minicomputadoras y computadoras personales-. Las aplicaciones y los datos están distribuidos más allá de una red de área local. El objetivo es tener aplicaciones que operen a través de múltiples plataformas de red, con un acceso transparente a funciones, datos y recursos del sistema, donde quiera que se encuentren.

Para lograr lo anterior, existen dos arquitecturas: la arquitectura de aplicaciones de dos niveles, basada principalmente en sistemas de gestión de base de datos, y la arquitectura de tres niveles, que implica servicio de gestión de transacciones; las cuales estudiaremos más adelante.

El **middleware** cliente/servidor conecta los procesos que componen la aplicación. Este término, ya muy extendido, se utiliza para describir aquellas funciones que son necesarias para proporcionar una localización transparente de los recursos.

El **middleware** agrupa una serie de funciones y servicios no incluidos en las aplicaciones ni en el sistema operativo, que aíslan al programador de tener que ocuparse de los aspectos de bajo nivel propios de sistemas distribuidos.

Mientras que el **middleware** define las plataformas y el nivel de transparencia de la ubicación de los recursos, las verdaderas ventajas de implementar una ambiente cliente/servidor sólo se pueden obtener mediante un adecuado diseño del sistema, de las aplicaciones y con una acertada elección de los elementos de desarrollo (herramientas y lenguajes). El objetivo es conseguir una total flexibilidad para distribuir las distintas partes de una aplicación entre varios procesadores, sin que ésta tenga que modificarse. Para poder lograr esto es necesario que una aplicación cliente/servidor disponga de los siguientes atributos:

Separación de funciones
Para lograr la distribución de los componentes de una aplicación, la lógica debe dividirse en los siguientes módulos funcionales:

- Lógica de presentación.
- Lógica de negocio.
- Lógica de datos.

Encapsulación de servicios
Con objeto de que el acceso a los módulos funcionales sea conocido y accesible por clientes y servidores, las funciones deben encapsularse, es decir, una función debe realizar todas sus funciones dentro de un solo módulo y no depender de otro.

Internamente la función puede cambiarse o mejorarse, incluso utilizando diferentes herramientas o lenguajes sin que los clientes tengan que ser notificados.

Un servidor puede ser considerado como un objeto. Este objeto, si se diseña correctamente, puede ser compartido por muchos procesos de distintos clientes y diferentes aplicaciones.

Portabilidad
Los módulos que forman una aplicación es probable que se requiera colocarlos en diferentes procesadores dentro de una red. Para que esto sea posible, es necesario que la programación de los módulos sea portable lo cual permita su ubicación en sistemas de diferentes arquitecturas.

Operación síncrona y asíncrona
Múltiples clientes pueden accesar a múltiples servidores. Los clientes siempre inician la petición y los servidores esperan pasivamente las peticiones de los clientes. Un cliente puede iniciar una petición en modo síncrono o asíncrono. El proceso síncrono significa que el cliente espera hasta que el servidor ha cumplido la petición o hasta que haya expirado el plazo determinado.

En el modo asíncrono, el proceso cliente continúa sin esperar la terminación del proceso en el servidor; una vez terminado el proceso en el servidor se notifica al cliente. El modo asíncrono es útil para procesos largos.

Una de las promesas más importantes del modelo cliente/servidor es que no sólo se puede lograr la integración de las aplicaciones a nivel de datos, sino también al nivel de funciones. Sin embargo, esto sólo se conseguirá si las aplicaciones se diseñan adecuadamente. El modelo cliente/servidor pondrá fin a las aplicaciones monolíticas en las que los datos y el código que se utilizan están situados en un mismo lugar. La distribución de las funciones de la aplicación y la adopción de la metodología de programación orientada a objetos tienen un importante impacto en la estructura y el diseño de dichas aplicaciones.

En un entorno cliente/servidor, las aplicaciones se componen de partes de desarrollo propio y aplicaciones disponibles comercialmente. La integración se logra a través de la interfaz gráfica de usuario, compartiendo datos comunes o mediante comunicación directa e intercambio entre aplicaciones. Estas dos últimas posibilidades ocultan al usuario todos los aspectos relativos al sistema y proporcionan un acceso transparente y un fácil uso de las aplicaciones de negocio. Las funciones de la aplicación se distribuyen entre los procesadores de la red.

Uno de los retos más importantes de los departamentos de sistemas de información es materializar una infraestructura de tecnología informática que permita desarrollar este tipo de entorno de aplicaciones. Para conseguirlo, será necesario:

☞ Seleccionar los productos de middleware más adecuados.

☞ Seleccionar las herramientas de desarrollo más apropiadas a la empresa.

☞ Diseñar las aplicaciones para un entorno distribuido.

En un entorno de proceso distribuido, el control de la aplicación pasa a la estación de trabajo y al usuario. Típicamente, la estación de trabajo opera con múltiples aplicaciones y el usuario controla su ejecución manipulando iconos de la interfaz gráfica. Desde ahí se accesan los servicios locales de la red departamental o a las servicios remotos de una WAN. Los procesadores que forman parte de la LAN o WAN se convierten en proveedores de servicios para las aplicaciones de las estaciones de trabajo.

En el ambiente cliente/servidor las funciones (servidores) se constituyen como entidades autónomas, que el usuario puede seleccionar de manera aleatoria. Dado que los procesos de negocio exigen una cierta secuencia en su ejecución, es necesario que esas funciones independientes se liguen en esa misma secuencia. Por ejemplo, el proceso de aprobación de un préstamo en un banco requiere que se sigan procedimientos estrictos. No se puede dar el dinero antes de reunir la información necesaria y de que la persona responsable haya autorizado la petición. Es muy probable que las autorizaciones dependan de la cantidad en cuestión. Grandes cantidades de dinero quizá deban ser autorizadas por la oficina regional o por la central, mientras que las cantidades pequeñas podrían ser autorizadas por el personal encargado de la oficina local.

El código que controle la ejecución secuencial del proceso de negocio debe estar fuera de la lógica de las funciones, evitando así que los cambios en los procesos del negocio obliguen a la recodificación de las funciones.

Pero la implantación de una plataforma cliente/servidor no es una tarea fácil de realizar. Existen barreras tecnológicas que deben enfrentarse y en caso de no comprenderlas totalmente es posible que el sistema resultante no cumpla con las expectativas establecidas, incluso podría tenerse una arquitectura inoperable; pero si la implantación tiene éxito, se contará con un sistema totalmente flexible y capaz de cubrir todas las necesidades presentes y futuras de la empresa. Es por esto que en muchas ocasiones el convencer a los directores del área de sistemas de migrar a sistemas cliente/servidor no es una tarea fácil. Pero el peligro real es que pueden quedarse atrás respecto a otras empresas competidoras que están rediseñando sus procesos de negocios y adoptando el modelo cliente/servidor y sistemas abiertos.

La arquitectura cliente/servidor no establece que una empresa deba de desechar sus actuales plataformas de hardware y software, sino al contrario lo que sugiere es la posibilidad de la distribución de los procesos y recursos, esto implica combinar las actuales plataformas con la implantación de una nueva arquitectura de procesamiento. Las características de esta nueva arquitectura serán:

- Mantenibilidad
- Modularidad
- Adaptabilidad
- Escalabilidad
- Portabilidad
- Estándares/sistemas abiertos
- Autonomía
- Flexibilidad

La combinación de los anteriores atributos da como resultado la capacidad de **maniobrabilidad** requerida por una empresa.

El concepto de la arquitectura cliente/servidor es muy simple: clientes que realizan peticiones de servicio y servidores que responden a dichas peticiones. Pero debajo de la superficie, la computación cliente/servidor es extremadamente complicada. Los arquitectos deben de navegar en un intrincado laberinto de difíciles decisiones. Es por esto que se requiere de la utilización de las herramientas adecuadas para el desarrollo de este tipo de ambientes.

Básicamente, los principales elementos en que se divide la computación cliente/servidor son:

- Clientes
- Redes
- Servidor

Estos son los tres principales elementos, pero es necesario considerar la administración, comunicación y herramientas de desarrollo; por lo que estas tres áreas se expanden en las siguientes, las cuales forman la columna vértebral de la arquitectura cliente/servidor:

- Network Management
  - System Management
  - Network Management Protocols
  - Network Managers
- Networking
  - Protocols
  - Networks
  - Wiring
  - Inter-Network
- Middleware
  - Remote Procedure Call
  - Message Oriented Middleware (MOM)
  - Object Request Broker
  - Conversational
  - Remote Database Access
  - Message Routing Services
  - Directory Services
  - Time Services
  - Terminal Services
  - Security Services

Básicamente las áreas anteriores son los retos tecnológicos que deben ser enfrentados en la construcción de un ambiente cliente/servidor.

Para iniciar la migración (o la implementación) a un ambiente cliente/servidor es necesario tener siempre en mente los siguientes aspectos, los cuales en muchas ocasiones no son tomados con la seriedad necesaria:

- Aumento de la complejidad debido a la conexión de múltiples sistemas heterogéneos.
- Diversidad de fuentes de datos.
- Nuevas plataformas (sistemas operativos) y operaciones.
- Inversión inicial en infraestructura.
- Formación y creación de especialistas en nuevas herramientas y metodologías .
- Arquitecturas de aplicaciones y diseño para un entorno distribuido.
- Nuevas guías de procedimientos operativos.
- Soporte técnico a los usuarios.

La implantación de la arquitectura cliente/servidor es un proceso que depende en específico de la empresa en que se esté implementado, pero en general siempre deberá de darse contestación a las siguientes preguntas antes de iniciar la migración o implantación:

- *¿Cuál es la distribución más adecuada de los componentes de una aplicación en una arquitectura cliente/servidor? ¿Dónde deben encontrarse cada uno de estos componentes (lógica de presentación, lógica del negocio y la lógica de procesamiento de los datos)?*
- *¿Cómo deben ser distribuidos los datos, accesos y administración?*
- *¿Cómo puede ser implementada la interacción entre los componentes?*
- *¿Cómo puede ser implementada la distribución y administración del software en un ambiente distribuido?*
- *¿Cuáles son los roles y requerimientos para la administración de transacciones distribuidas?*
- *¿Cómo se controla la seguridad en un ambiente cliente/servidor?*

La distribución de los componentes de una aplicación puede realizarse en diferentes estilos de procesamiento cooperativo, el decidir cuál será el que se utilizará dependerá directamente de las necesidades de la empresa y de la tecnología con la cual se cuente.

Actualmente, las empresas se encuentran más y más interesadas en la implementación de este tipo de sistemas, algunas de las razones son:

- Reducción de los costos de operaciones y descentralización de operaciones, de tal forma de ser más competitivos y responder más rápidamente a las demandas de los clientes.
- Ubicación de los datos en las zonas donde existe mayor demanda.
- Independencia de las áreas regionales.

La siguiente figura ilustra lo que la arquitectura cliente/servidor implica para una empresa.

Necesidades comerciales en continua evolución

La tecnología informática debe ser capaz de responder con rapidez a las cambiantes necesidades del negocio. Las aplicaciones tienen que desarrollarse rápidamente y adaptarse a procesos de negocios cambiantes.

Infraestructura abierta cliente/servidor

El departamento de sistemas de información tiene que proporcionar una infraestructura que permita distribuir funciones y datos de forma flexible y se adapte a los procesos de negocio.

Nuevos roles de sistemas de información y de usuarios

El rápido desarrollo de aplicaciones exige que los usuarios participen activamente en el proceso de desarrollo. Los usuarios tienen responsabilidades en las funciones, procesos de negocio y facilidad de uso. La responsabilidad del departamento de sistemas de información se centra en proporcionar infraestructuras, aplicaciones integradas y definir estándares coorporativos para las tecnologías.

Nuevas herramientas de desarrollo

Las aplicaciones se desarrollan con base en "prototipos", construyendo nuevas funciones a partir de una combinación de componentes de desarrollo propio y aplicaciones estándares disponibles comercialmente.

Nuevo proceso de desarrollo

Se necesita un nuevo proceso de desarrollo iterativo que aborte desarrollos basados en prototipos, en los que puedan participar los usuarios conjuntamente con los profesionales de sistemas de información.

CLIENT/SERVER
INFRASTRUCTURE
ROAD
MAP
CLIENTE
GUI
OS
HW
Word Processing
Spreadsheets
Drawing & Painting
Communications
NETWORK
MIDDLEWARE
NETWORKING
NETWORK MANAGEMENT
RDBMS
DB2
GROUPS
TRANSACTION MANAGERS
IMAGE/WORK FLOW
SERVIDOR
OS
HW
NETWORK
MANAGEMENT
NETWORKING
MIDDLEWARE

AUTOEVALUACIÓN DEL AMBIENTE CLIENTE/SERVIDOR

La siguiente autoevaluación abarca las habilidades y conocimientos requeridos para la planeación, implementación y soporte de un sistema cliente/servidor, y su propósito es:

- ✓ Ser una evaluación personal sobre el nivel de conocimientos y habilidades en las áreas más importantes para la computación cliente/servidor.
- ✓ Proveer de la información necesaria para lograr una transición .

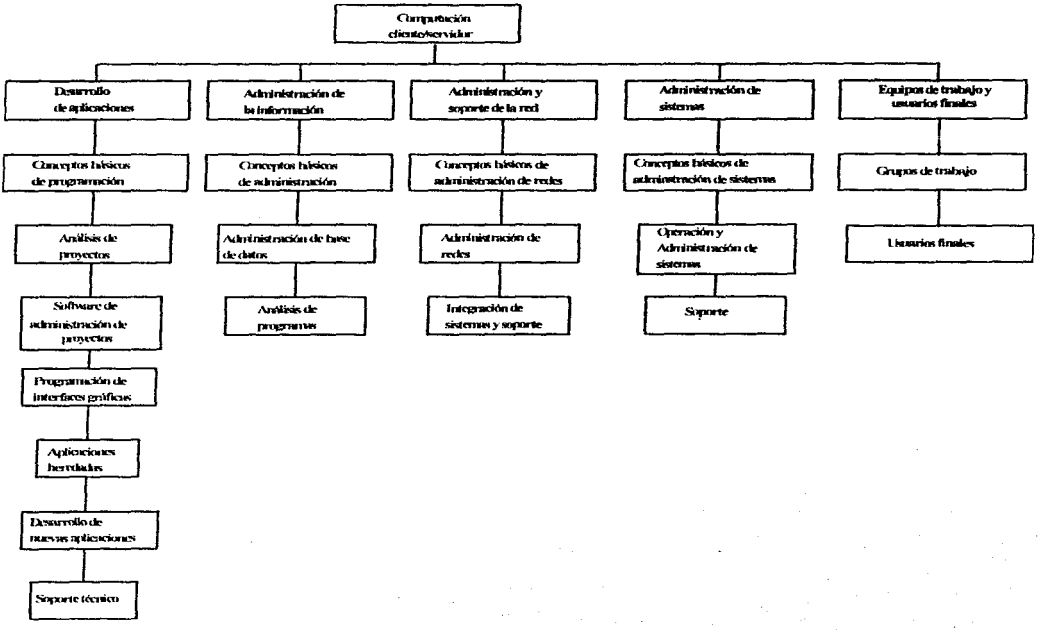
Los resultados obtenidos de esta evaluación pueden ser utilizados por los gerentes y expertos para establecer:

- ✓ Planes de entrenamiento y desarrollo.
- ✓ Actividades a desarrollar por los programadores, analistas y líderes de proyecto.

Las habilidades y conocimientos identificados para la implementación de un ambiente cliente/servidor son divididas en cinco categorías (áreas de competencia), las cuales se describen a continuación:

Categoría	Descripción
Nivel básico para cliente/servidor	Conocimientos básicos que son necesarios para la implementación de un ambiente cliente/servidor.
Desarrollo de aplicaciones	Conocimientos y habilidades que son requeridos en el desarrollo de aplicaciones dentro de un ambiente cliente/servidor.
Administración de los datos	Conocimientos necesarios para la administración de la información dentro de un ambiente cliente/servidor.
Administración de redes	Conocimientos necesarios para las funciones de administración de la red dentro de un ambiente cliente/servidor.
Administración de sistemas	Conocimientos que son requeridos para la administración de sistemas en un ambiente cliente/servidor.

Para cada área de competencia se definen subáreas y los conocimientos requeridos para cada una de estas subáreas son abarcados por la autoevaluación, la siguiente figura muestra la descomposición de las áreas de competencia.



Instrucciones para la autoevaluación

La siguiente información ayudará a familiarizarse con este documento y proveerá de la información sobre cómo se utiliza esta autoevaluación.

☞ Seleccione el área o áreas de competencia deseada para la autoevaluación.

☞ Para cada uno de los puntos indique el nivel de conocimientos **requerido** en su trabajo para el futuro.

Utilice la siguiente escala y marque con un **círculo** la opción:

1. No se requieren habilidades o conocimientos.
2. Se requerirán conocimientos generales.
3. Se requiere de estos conocimientos, pero **con ayuda** de consultores.
4. Se requiere de aplicar estos conocimientos **sin** ayuda.
5. Se requiere ser un experto.

☞ Para cada uno de los puntos indique su **nivel actual** de conocimiento.

Utilice la siguiente escala y marque con una **X** su nivel.

1. No se tiene conocimientos.
2. Se encuentra familiarizado, pero no se tiene el conocimiento necesario para aplicarlo en el trabajo.
3. Puede aplicarlo en su trabajo pero con ayuda.
4. Puede aplicarlo en su trabajo sin ayuda.
5. Es un experto en esta área.

Finalmente, dibuje una línea conectando todas las X y otra línea conectando los círculos, las áreas que requieran de una mayor atención serán aquéllas en donde se tenga una gran separación entre la línea que une los círculos y la que une las X.

Por ejemplo, sus opciones podrían verse como las del siguiente diagrama.

Conocimiento	Requerido (O)/Actual (X)				
Comprender las características, funciones y beneficios de la arquitectura cliente/servidor.	1	2	X	(4)	5
Comprender los modelos de cómputo distribuidos.	1	2	3	X	5
Comprender los componentes del cliente, técnicas de acceso a datos y herramientas.	1	2	X	4	(5)
Conocimientos de la configuración de un ambiente cliente/servidor.	1	X	3	(4)	5

Todos los conocimientos son importantes, pero el nivel actual es menor para el ítem 4. Las categorías en donde existe la mayor separación indica las principales áreas de entrenamiento o de riesgo.

Conocimientos básicos del ambiente cliente/servidor

Esta sección describe las habilidades y conocimientos básicos requeridos para todas las áreas de competencia del ambiente cliente/servidor.

Conceptos de la arquitectura cliente/servidor

Conocimiento de las características, funciones y beneficios de una arquitectura cliente/servidor.	1	2	3	4	5
Conocimiento de los conceptos de un ambiente de computación distribuido (DCE, Distributed Computing Environment).	1	2	3	4	5
Conocimiento de las arquitecturas de procesamiento (centralizado, remoto, distribuido y cliente/servidor).	1	2	3	4	5
Conocimientos de los conceptos de *downsizing*.	1	2	3	4	5
Conocimiento de los roles de un cliente, servidor y la red.	1	2	3	4	5
Conocimiento de los conceptos de Intercomunicación de procesos (IPC *InterProcess Comunication*).	1	2	3	4	5
Conocimiento de las barreras y riesgos de la migración hacia una tecnología cliente/servidor (culturales, organizacionales, financieros).	1	2	3	4	5

Instalación de equipo PC

Conocimiento de cableado básico de LAN.	1	2	3	4	5

Conocimiento de los conceptos básicos de hardware

Conocimiento de los componentes de hardware y de sus funciones, por ejemplo CPU, discos, monitores e impresoras.	1	2	3	4	5

Conocimiento de los conceptos y operación de interfaces gráficas

Manejo del teclado y del Mouse.	1	2	3	4	5
Control de ventanas (tamaño, scroll, cambios de ubicación, minimización, maximización).	1	2	3	4	5
Uso de aplicaciones en un ambiente gráfico.	1	2	3	4	5
Manejo de ventanas de diálogo a través del mouse y el teclado.	1	2	3	4	5

Conocimiento de los conceptos de seguridad básica

Conocimiento de los principales aspectos de seguridad (manejo de grupo, protección de archivos, directorios, dispositivos, etc.).	1	2	3	4	5
Manejo de utilerías para el control de la seguridad.	1	2	3	4	5

Conocimientos en el desarrollo de aplicaciones cliente/servidor

Esta sección describe los conocimientos requeridos para el desarrollo de aplicaciones en un ambiente cliente/servidor.

Conceptos básicos del desarrollo de aplicaciones cliente/servidor

Conocimiento de los conceptos básicos del desarrollo de aplicaciones cliente/servidor

Conocimiento de las características, funciones y beneficios de la arquitectura cliente/servidor.	1	2	3	4	5
Conocimiento de los problemas potenciales en la administración del sistema.	1	2	3	4	5
Conocimiento de los conceptos de administración del sistema y cómo se comparan con la administración de un sistema centralizado.	1	2	3	4	5
Conocimiento de los modelos de cómputo distribuidos.	1	2	3	4	5
Manejo de configuraciones y métodos de desarrollo.	1	2	3	4	5
Conocimiento de los componentes del cliente, técnicas de acceso de datos y herramientas de programación.	1	2	3	4	5
Conocimiento de los componentes del servidor.	1	2	3	4	5

Planeación de las estrategias y herramientas

Manejo del ambiente y herramientas para:					
• Interfaces gráficas.	1	2	3	4	5
• Base de datos.	1	2	3	4	5
Manejo de los ambientes y herramientas para:					
• Computer Aided Software Engineering (CASE).	1	2	3	4	5
• System Integration (SI).	1	2	3	4	5
Conocimiento de las estrategias y rutas de migración para el ambiente cliente/servidor.	1	2	3	4	5

Conocimiento del proceso de desarrollo de aplicaciones

Conocimiento del modelado de los procesos del negocio en un ambiente cliente/servidor.	1	2	3	4	5
Conocimiento en la técnica de programación orientada a objetos.	1	2	3	4	5
Ciclo de vida de un desarrollo.	1	2	3	4	5

Análisis del *performance* del ambiente cliente/servidor

Análisis del ambiente actual.	1	2	3	4	5
Determinación de los problemas actuales.	1	2	3	4	5
Determinación de las aplicaciones más adecuadas para el contexto del ambiente y los requerimientos del negocio.	1	2	3	4	5

Planeación de actividades

Conocimiento en el desarrollo de aplicaciones para un ambiente gráfico.	1	2	3	4	5
Conocimiento de lenguajes de tercera generación para el desarrollo de aplicaciones, por ejemplo C++, Fortran, COBOL, Ada y basic.	1	2	3	4	5
Conocimiento de lenguajes de cuarta generación (4GL), por ejemplo power builder.	1	2	3	4	5
Manejo y conocimiento de estándares.	1	2	3	4	5
Utilización de métodos de análisis y diseño orientado a objetos.	1	2	3	4	5
Programación en C, C++, Windows SDK, Visual Basic, COBOL, Ada, Fortran y BASIC.	1	2	3	4	5
Conocimiento y manejo de los conceptos de bases de datos relacionales, por ejemplo. ORACLE, SYBASE, RDB e INFORMIX.	1	2	3	4	5
Utilización de herramientas cliente/servidor, por ejemplo, Forté, Gupta, DECset y Uniface.	1	2	3	4	5

Realización de mantenimientos y pruebas

Conocimiento en el mantenimiento y pruebas de aplicaciones cliente/servidor.	1	2	3	4	5
Desarrollo de estrategias efectivas de pruebas.	1	2	3	4	5

Soporte técnico

Entender y aplicar los procesos requeridos para la realización de cambios, por ejemplo, configuración y control de la documentación.	1	2	3	4	5
Conocimiento y manejo de estrategias para garantizar la seguridad y la integridad de datos.	1	2	3	4	5
Comprender y utilizar herramientas de soporte, por ejemplo, CASE.	1	2	3	4	5

Dirección de las actividades de implementación de un ambiente cliente/servidor

Conocimiento de las áreas problemáticas y proposición de soluciones.	1	2	3	4	5
Realización de simulaciones del negocio (prototipos).	1	2	3	4	5

Análisis del proyectos

Aplicación de los conceptos de distribución y "downsizing"

Análisis del ambiente actual para determinar los problemas existentes.	1	2	3	4	5
Establecimiento de planes para el logro de las metas.	1	2	3	4	5

Conocimiento de las aplicaciones cliente/servidor

Conocimiento de los tipos de aplicaciones en un ambiente distribuido y cómo interoperan.	1	2	3	4	5
Conocimiento en la configuración de los clientes y de los servidores.	1	2	3	4	5

Calificando las soluciones cliente/servidor

Estimación de las necesidades del negocio.	1	2	3	4	5
Conocimientos técnicos de la infraestructura.	1	2	3	4	5
Conocimientos de las políticas de intercambio de información.	1	2	3	4	5
Estimación del costo de la solución integral, incluyendo productos, entrenamiento, implementación, soporte y mantenimiento.	1	2	3	4	5

Reingeniería y *Downsizing*

Creación de una estrategia para la distribución de los recursos de cómputo.	1	2	3	4	5
Determinación de la forma en que las aplicaciones requieren de ser distribuidas.	1	2	3	4	5
Conocimientos en los mecanismos de interfaz entre los usuarios, aplicaciones y datos.	1	2	3	4	5

Desarrollo de propuestas

Desarrollo de soluciones prototipo, donde se incluya una estimación de alcances, calendarización de actividades y costos.	1	2	3	4	5

Documentos de requerimientos

Información detallada sobre el diseño y operación de la solución propuesta.	1	2	3	4	5

Administración de proyectos

Desarrollo de un plan

Determinación de recursos y costos.	1	2	3	4	5
Determinación de las fases del proyecto por medio de herramientas y técnicas de administración de proyectos.	1	2	3	4	5
Determinación de los riesgos y cómo controlarlos.	1	2	3	4	5

Administración de los recursos del proyecto

Realización de un análisis de riesgos.	1	2	3	4	5
Seleccionar, monitorear y administrar los recursos en base a los requerimientos del proyecto.	1	2	3	4	5
Creación de equipos de trabajo de acuerdo con las metas y objetivos.	1	2	3	4	5
Mantener a los clientes informados del estatus de los aspectos críticos del proyecto.	1	2	3	4	5
Utilización de técnicas efectivas de comunicación (ejemplo Mail).	1	2	3	4	5

Programación de interfaces gráficas

Creación de GUI para las existentes aplicaciones de bases de datos

Posibilidad de utilizar lenguajes para la programación de front-end, como Visual Basic.	1	2	3	4	5
Posibilidad del uso de lenguajes tradicionales de tercera generación, como lo es C.	1	2	3	4	5
Posibilidad de utilizar lenguajes orientados a objetos como C++.	1	2	3	4	5
Posibilidad de utilización de herramientas GUI-FE, tales como PowerBuilder, SQL Windows, Lotus Notes, EASEL Workbench, y Paradox.	1	2	3	4	5

Programación de aplicaciones de interfaz con el usuario

Diseño de interfaces.	1	2	3	4	5
Conocimiento en ambientes gráficos, como son X-Windows System, Microsoft Windows, Windows NT, Mac, y OS/2.	1	2	3	4	5
Uso del SDK ,X/Motif Toolkits, Mac, y Otros.	1	2	3	4	5
Programación de interfaces con herramientas de orientación a objetos, por ejemplo, Visual C++.	1	2	3	4	5

Utilización de sistemas operativos de red

Programación en ambientes de redes de área local (LAN), por ejemplo NetWare, PATHWORKS, y LAN Manager.	1	2	3	4	5
Programación en un ambiente distribuido.	1	2	3	4	5

Sistemas heredados (Legacy Wrapping)

Comprender las aplicaciones distribuidas o multiprocesos

Uso de lenguajes de tercera generación (3GLS), por ejemplo, C, C++.	1	2	3	4	5
Conocimientos de las técnicas de procesamiento paralelo.	1	2	3	4	5
Conocimiento en la programación de RPC, threads y directorios de servicios.	1	2	3	4	5

Conocimientos en la configuración de controladores (drivers) y programación de *internals*

Programación de controladores (*drivers*) sobre el sistema operativo seleccionado.	1	2	3	4	5
Conocimientos del lenguaje con el cual son programados los controladores.	1	2	3	4	5
Conocimiento de los *internals* del sistema operativo.	1	2	3	4	5
Conocimiento de los *internals* del hardware.	1	2	3	4	5

Conocimientos en la integración de componentes

Conocimiento de las funciones provistas por los procesos de intercomunicación (IPC).	1	2	3	4	5
Evaluación y comparación de protocolos.	1	2	3	4	5

Desarrollo de nuevas aplicaciones

Análisis y diseño

Conocimientos de los conceptos básicos de análisis y diseño.	1	2	3	4	5

Programación orientada a objetos

Conocimientos de los principios y beneficios de la programación orientada a objetos.	1	2	3	4	5
Conocimiento de la metodología de desarrollo en espiral.	1	2	3	4	5
Conocimiento de los métodos de análisis y diseño orientado a objetos.	1	2	3	4	5

Análisis estructurado

Conocimiento de los principios y beneficios de la programación estructurada.	1	2	3	4	5

Desarrollo de aplicaciones

Conocimientos en el manejo de API (*Application Programing Interface).*	1	2	3	4	5
Conocimientos y uso de lenguajes de tercera generación como C o bien de lenguajes de programación orientada a objetos como C++.	1	2	3	4	5
Uso de software de depuración (*debuggers*).	1	2	3	4	5
Manejo de las librerías del sistema.	1	2	3	4	5
Uso de herramientas de desarrollo para de un ambiente cliente/servidor como por ejemplo Sybase Gain Momentum, Forté, SQLWindows.	1	2	3	4	5
Conocimiento de los conceptos básicos de bases de datos.	1	2	3	4	5
Conocimientos el programación SQL (ORACLE, INFORMIX o SYBASE.	1	2	3	4	5
Conocimientos del sistema operativos del servidor, por ejemplo, UNIX, OpenVMS o WNT.	1	2	3	4	5
Desarrollo de aplicaciones que se comuniquen a una base de datos relacional, como ORACLE, INFORMIX, INGRES o SYBASE.	1	2	3	4	5

Soporte técnico

Instalación y administración de software

Instalación de aplicaciones heterogéneas (ambiente multivendedores) en un ambiente cliente/servidor.	1	2	3	4	5
Conocimientos en actualizaciones de software, interoperabilidad y compatibilidad.	1	2	3	4	5
Conocimientos de administración del sistema operativo, aplicaciones y bases de datos	1	2	3	4	5

Administración de la información en un ambiente cliente/servidor

Esta sección describe los conocimientos y habilidades requeridas para la administración de los datos en un ambiente cliente/servidor.

Conocimientos básicos de administración de datos

Comprender los conceptos básicos de cliente/servidor

Comprender la computación cliente/servidor.	1	2	3	4	5
Conocimiento del rol del cliente.	1	2	3	4	5
Conocimiento del rol del servidor.	1	2	3	4	5
Operación de clientes con diferentes arquitecturas (ambiente multivendedor).	1	2	3	4	5
Comprender el rol del middleware, por ejemplo, uso de API o de RPC.	1	2	3	4	5

Modelado de datos

Conocimiento de los conceptos de bases de datos relacionales.	1	2	3	4	5
Conocimiento de los principios de la programación orientada a objetos.	1	2	3	4	5

Servidores de bases de datos

Identificación de los diferentes tipos de servidores (archivos, aplicaciones, datos, etc.).	1	2	3	4	5
Comprender las diferencias y fundamentos entre las bases de datos de diversos proveedores, como ORACLE y SYBASE.	1	2	3	4	5

Administración de base de datos

Prerrequisitos para la administración de la base de datos

Tamaño de la base de datos.	1	2	3	4	5
Uso de las utilerías para la carga y reorganización de los datos.	1	2	3	4	5
Configuración de las base de datos multivendedor.	1	2	3	4	5
Soporte a las procesamiento de transacciones.	1	2	3	4	5
Conocimentos en análisis de crecimiento de la base de datos.	1	2	3	4	5
Implementación de estrategias de recuperación para un ambiente multivendedor de base de datos.	1	2	3	4	5
Integración de las bases de datos en un ambiente multivendedor.	1	2	3	4	5

Seguridad

Establecer la seguridad de datos y los derechos de los usuarios.	1	2	3	4	5
Definir los requerimientos de seguridad de datos para las aplicaciones.	1	2	3	4	5
Descripción de los comandos necesarios para la implantación de la seguridad.	1	2	3	4	5
Establecer los niveles de seguridad para el acceso de los datos remotos.	1	2	3	4	5
Evaluación global de la seguridad en un ambiente multivendedor.	1	2	3	4	5

Procesamiento de las transacciones

Control de las peticiones y transacciones remotas	1	2	3	4	5
Control de las peticiones y transacciones distribuidas.	1	2	3	4	5

Auditoría

Mantenimiento de la información de auditoria.	1	2	3	4	5

Bases de datos distribuidas

Configuración de las bases de datos distribuidas, especificar las estructuras de distribución necesarias para los datos.	1	2	3	4	5

Consultas (Queries)

Ejecución de los queries distribuidos.	1	2	3	4	5
Identificación de las nuevas funciones o extensiones existentes del manejador de base de datos (RDBMS) para el control de los datos almacenados en múltiples bases de datos	1	2	3	4	5

Habilidades básicas de análisis por parte de los programadores

Programación

Desarrollo de aplicaciones utilizando un lenguaje de tercera generación (3GL).	1	2	3	4	5
Desarrollo de aplicaciones utilizando un lenguaje de cuarta generación (4GL).	1	2	3	4	5
Desarrollo de aplicaciones utilizando herramientas de desarrollo de front-end.	1	2	3	4	5
Uso de herramientas de definición de datos (Artisan, DB manager o sqldba).	1	2	3	4	5
Acceso a servidores de bases de datos, en un ambiente multivendedor, para la consulta o mantenimiento de los datos usando una interfaz de SQL.	1	2	3	4	5
Identificar la forma en que en un ambiente multivendedor las tablas son organizadas e interrelacionadas.	1	2	3	4	5

Realizar pruebas que garanticen la funcionalidad y calidad de las aplicaciones

Documentación de los procedimientos de pruebas.	1	2	3	4	5
Realización de pruebas de integración y pruebas de módulos independientes.	1	2	3	4	5
Corrección de los problemas identificados durante la fase de pruebas.	1	2	3	4	5

Conocimientos en redes para un ambiente cliente/servidor

Esta sección describe las habilidades y conocimientos requeridos para la administración de la red en un ambiente cliente/servidor.

Conocimientos básicos del control y soporte de redes

Conectividad

Conocimientos en redes de área local (LAN), redes de área extendida (WAN) y conceptos de *cluster*.	1	2	3	4	5
Conocimientos de las funciones realizadas por los componentes de la red, por ejemplo, puentes, ruteadores, repetidores, etc.	1	2	3	4	5
Comprender y describir tecnologías de comunicación, incluyendo sus características y limitaciones, por ejemplo, *Ethernet, token ring, Fiber Distributed Data Interface* (FDDI) y comunicación sin cables.	1	2	3	4	5

Arquitecturas de red y estándares

Identificar y describir las capas del modelo OSI *(Open Systems Interconnection Model)*.	1	2	3	4	5

Administración de redes

Instalación

Configuración de los dispositivos de redes.	1	2	3	4	5
Instalación y configuración del software de red y del sistema operativo.	1	2	3	4	5
Realizar registro de los nodos participantes en la red.	1	2	3	4	5
Instalar y configurar software de comunicación.	1	2	3	4	5

Configuración

Entender e implementar el esquema de direccionamiento adecuado, por ejemplo, *Domain Name System* (DNS).	1	2	3	4	5
Comprender los parámetro de la base de datos y de las estructuras de datos requerida para la configuración de la base de datos.	1	2	3	4	5

Rendimiento

Comprender el proceso utilizado para mejor el rendimiento del ruteo de datos a través la selección de la ruta más óptima, balanceo de la carga y control del tráfico.	1	2	3	4	5
Comprender las operaciones de acceso a los archivos compartidos.	1	2	3	4	5
Comprender el impacto de restringir el acceso a objetos en la red.	1	2	3	4	5
Administración de la memoria y control de la memoria cache para optimizar el desempeño.	1	2	3	4	5
Comprender cómo realizar la afinación de los procesos en la red para optimizar el tiempo de respuesta.	1	2	3	4	5

Control de problemas

Comprender las técnicas y procesos utilizados para aislar los problemas en la red, como, segmentación y dispositivos de tolerancia a fallas.	1	2	3	4	5
Utilización de protocolos de mantenimiento, por ejemplo, *Simple Network Management Protocol* (SNMP), *Maintenance Operations Service Protocol* (MOP), *Network Information and Control Exchange* (NICE).	1	2	3	4	5
Comprender e interpretar los protocolos de la capa de sesión y de transporte, por ejemplo, *Network Service Protocol* (NSP), *Transmission Control Protocol* (TCP), SPX y ADSP.	1	2	3	4	5
Comprender e interpretar los protocolos de ruteo, por ejemplo, DECnet OSI, IP, IPX, etc.	1	2	3	4	5

Seguridad

Comprender los aspectos de seguridad, en un ambiente de redes, y utilerías de protección disponibles.	1	2	3	4	5
Conocer las utilerías de seguridad, por ejemplo, encriptación, protección de archivos y control de accesos.	1	2	3	4	5

Hardware

Comprender las diferencias entre banda base(*baseband*) y banda amplia (*broadband*), incluyendo ventajas y desventajas.	1	2	3	4	5
Configuración de redes de acuerdo con las reglas de configuración para la tecnología que se implementa, por ejemplo, ethernet, token ring, FDDI.	1	2	3	4	5
Crear e interpretar mapas de topología.	1	2	3	4	5
Integrar los sistemas heredados a la red.	1	2	3	4	5
Utilizar protocolos de mantenimiento para aislar las fallas de hardware en la red, por ejemplo *Network Control Program* (NCP), *Network control Language* (NCL), SNMP y MOP.	1	2	3	4	5

Administración de un ambiente cliente/servidor

Esta sección incluye las funciones que un administrador de sistemas deberá conocer en un ambiente cliente/servidor.

Conocimientos básicos de administración

Conocimientos básicos

Conocimiento del ambiente cliente/servidor, comprender las áreas de problemas potenciales (ambiente multivendedor, administración, tiempo de implantación, etc.) en la arquitectura.	1	2	3	4	5
Conocimientos en hardware, comprendiendo las diferentes arquitecturas de PC, así como sus capacidades entre éstas.	1	2	3	4	5

Comprender las relaciones entre los clientes (en este caso cliente son las personas que requieren de algún servicio)

Escuchar los problemas del cliente y generar una relación de confianza.	1	2	3	4	5
Establecer acciones y recomendaciones a las necesidades inmediatas de los clientes.	1	2	3	4	5
Presentarse ante el cliente de una forma profesional y responsable.	1	2	3	4	5

Habilidades para la operación

Conocer el ambiente cliente/servidor

Comprender los conceptos de administración en un ambiente cliente/servidor y cómo estos conceptos se comparan con la administración de un ambiente centralizado.	1	2	3	4	5

Conocimientos en el hardware

Instalación de tarjetas de red, discos ,etc.	1	2	3	4	5
Instalar y configurar múltiples servidores en un ambiente heterogéneo.	1	2	3	4	5
Instalar y soportar periféricos compartidos, como son impresoras, escáners, líneas de comunicación, módem, y CD.	1	2	3	4	5

Conocer el ambiente del usuario

Instalación de interfaces gráficas (GUI)	1	2	3	4	5
Creación de procedimientos de comandos (COM)	1	2	3	4	5
Alta y control de cuentas de usuarios.	1	2	3	4	5
Instalación de software para la conectividad en un ambiente multivendedor.	1	2	3	4	5
Establecer la seguridad en un ambiente cliente/servidor.	1	2	3	4	5
Uso de herramientas para la administración y control de la seguridad.	1	2	3	4	5
Cambios de passwords.	1	2	3	4	5
Comprender e implementar la seguridad de las aplicaciones.	1	2	3	4	5

Administración de archivos

Uso de los comandos de manipulación de archivos en los servidores.	1	2	3	4	5
Conocer las diferencias en las convenciones y reglas para los nombres de los archivos en las diferentes tecnologías de los servidores.	1	2	3	4	5
Configuración de las colas de impresión compartidas.	1	2	3	4	5
Comprender los diferentes formatos de archivos que existen en los servidores y sus relaciones con los formatos manejados en los clientes.	1	2	3	4	5
Conocer e instalar convertidores de formatos.	1	2	3	4	5
Mantenimiento de la seguridad y establecimiento de áreas públicas.	1	2	3	4	5
Manejo de la auditoría e identificación de los "huecos" en la seguridad.	1	2	3	4	5

Comprender el ambiente operativo

Instalación de los sistemas operativos tanto del cliente como del servidor.	1	2	3	4	5
Conocimiento de la forma de realizar la afinación del servidor para obtener su máximo desempeño.	1	2	3	4	5
Comprender las diferencias entre los distintos sistemas operativos en un ambiente heterogéneo.	1	2	3	4	5
Instalar, configurar y mantener el software en el servidor para el uso de los clientes.	1	2	3	4	5
Conocer los archivos de inicialización.	1	2	3	4	5
Realizar actualizaciones en el servidor.	1	2	3	4	5
Comprender y controlar las implicaciones de ejecutar múltiples versiones de una aplicación en el servidor.	1	2	3	4	5
Establecer los procedimientos necesarios para la protección de VIRUS e identificar áreas de riesgo potenciales.	1	2	3	4	5
Comprender y utilizar utilerías para la administración de la memoria.	1	2	3	4	5
Comprender los requerimientos de las aplicaciones y capacidades de rendimiento en relación con la memoria disponible en los sistemas.	1	2	3	4	5

Administración de discos

Configuración de las cuentas de usuario tomando consideraciones de privilegios, cuotas y recursos de necesarios.	1	2	3	4	5
Establecer respaldos diarios y estrategias de recuperación tanto para los clientes como para los servidores.	1	2	3	4	5
Comprender los procesos de inicialización de discos y de configuración, y conocer las técnicas de balanceo de discos.	1	2	3	4	5

Administración de la comunicación

Configurar e instalar productos de correo electrónico.	1	2	3	4	5
Instalación, configuración y software de soporte para la emulación.	1	2	3	4	5
Realizar la configuración de *módem*, como también conocer de su mantenimiento.	1	2	3	4	5

Conocimientos de redes

Comprender los conceptos y terminología de LAN y WAN, así como el impacto que tienen en la administración del sistema.	1	2	3	4	5
Conocer las tecnologías de redes, como por ejemplo, ethernet y token ring.	1	2	3	4	5
Comprender las diferencias de capacidades en los diferentes protocolos de transporte, como son DECnet, TCP/IP, NetBEUI e IPX.	1	2	3	4	5
Instalación de redes, software, sistemas operativos de red (NOS) y registro de nodos.	1	2	3	4	5
Configuración del mapeo de teclas y emulación de terminales.	1	2	3	4	5
Comprender y realizar la administración del sistema operativo de red (NOS).	1	2	3	4	5

Comprender portabilidad

Comprender los aspectos de seguridad.	1	2	3	4	5
Uso de utilerías de seguridad.	1	2	3	4	5
Cuidar de la seguridad relacionada con los sistemas de escritorio.	1	2	3	4	5
Instalar software para el control de la seguridad.	1	2	3	4	5
Comprender la instalación y el upgrade del hardware.	1	2	3	4	5
Conexión a la red.	1	2	3	4	5
Comprender las capacidades y limitaciones.	1	2	3	4	5
Identificar y rectificar problemas de comunicación.	1	2	3	4	5
Instalar y configurar el software de comunicación, e implantación del ambiente del usuario.	1	2	3	4	5

Soporte

Hardware

Conocimientos básicos de la conexión de redes.	1	2	3	4	5

Comprender el ambiente del usuario

Personalizar el ambiente de interfaz gráfica del usuario.	1	2	3	4	5

Administración de archivos

Renombrado, cambios de ubicación, borrado, respaldo y recuperación de archivos.	1	2	3	4	5
Importación y exportación de archivos.	1	2	3	4	5
Comprender la estructura de directorios, cómo navegar dentro y entre los directorios, y cómo mostrar la localización de los archivos.	1	2	3	4	5
Personalizar el ambiente.	1	2	3	4	5
Identificar y resolver problemas en las colas de impresión.	1	2	3	4	5
Comprender los diferentes formatos de archivos y las extensiones en el sistema.	1	2	3	4	5
Distinguir los diferentes formatos de archivos.	1	2	3	4	5
Identificar y resolver los problemas de formatos de archivos.	1	2	3	4	5
Establecer seguridad en los archivos.	1	2	3	4	5

Comprender el ambiente operativo

Instalación del sistema operativo en los clientes.	1	2	3	4	5
Instalación del sistema operativo de los servidores.	1	2	3	4	5
Instalación de productos.	1	2	3	4	5
Comprender los archivos de inicialización (*.ini).	1	2	3	4	5
Identificar y resolver problemas en la instalación.	1	2	3	4	5
Actualización de las aplicaciones de los clientes.	1	2	3	4	5
Comprender lo que es un VIRUS, qué daños puede provocar ,y cómo protegerse contra ellos.	1	2	3	4	5
Comprender la relación entre la administración de la memoria y las aplicaciones y el sistema.	1	2	3	4	5
Conocer la diferencia entre memoria real, extendida y expandida.	1	2	3	4	5
Determinación de la memoria disponible en los clientes	1	2	3	4	5

Administración de discos

Determinación del espacio disponible en el disco.	1	2	3	4	5
Realización de compresión de archivos.	1	2	3	4	5
Reconocimiento y solución de fallas en los discos.	1	2	3	4	5
Manejo de file systems.	1	2	3	4	5

Administración de la comunicación

Conocimientos en el manejo de mail.	1	2	3	4	5
Direccionamiento del correo.	1	2	3	4	5
Conocimientos en formatos de archivos y convertidores.	1	2	3	4	5
Reconocer problemas con el *mail*.	1	2	3	4	5
Comprender, instalar y utilizar software de emulación.	1	2	3	4	5
Comprender las conexiones asíncronas.	1	2	3	4	5

Administración de documentos

Conversión de documentos.	1	2	3	4	5
Transferencia de un documento de una PC a otra.	1	2	3	4	5
Administración de archivos compartidos.	1	2	3	4	5

Redes

Comprender las características y funciones de una red en un ambiente cliente/servidor.	1	2	3	4	5
Conocimientos en la instalación de redes.	1	2	3	4	5
Conocimientos en la configuración de la red y del ambiente de los usuarios.	1	2	3	4	5

Grupos de trabajo y usuarios finales

Esta sección describe los conocimientos requeridos por los usuarios finales en un ambiente cliente/servidor.

Grupos de trabajo

Equipos

Establecimiento de metas y misión del equipo.	1	2	3	4	5
Seguimiento de los procesos tal como son definidos por el equipo de trabajo.	1	2	3	4	5
Comparte información con los miembros del equipo.	1	2	3	4	5
Respuestas honestas hacia y de los miembros del equipo de trabajo.	1	2	3	4	5

Usuarios finales

Administración de archivos

Mover, renombrar, borrar y respaldar los archivos en la PC.	1	2	3	4	5
Importar y exportar archivos sobre la red.	1	2	3	4	5
Comprender el manejo de directorios, cómo navegar entre los directorios y cómo mostrar la ubicación de los archivos.	1	2	3	4	5
Comprender las condiciones sobre los nombres de los archivos.	1	2	3	4	5
Impresión de archivos.	1	2	3	4	5
Manejo de colas de batch.	1	2	3	4	5
Comprender los diferentes formatos y extensiones de archivos existentes en el sistema local.	1	2	3	4	5
Manejo de protecciones sobre los archivos.	1	2	3	4	5

Ambiente operativo

Comprender lo que es un VIRUS, qué daño puede ocasionar y cómo protegerse de éstos.	1	2	3	4	5

Administración de discos

Determinar el espacio disponible en el disco.	1	2	3	4	5
Realizar compresión de archivos.	1	2	3	4	5
Realizar respaldos de discos duros.	1	2	3	4	5
Realizar formateo de discos duros y flexibles.	1	2	3	4	5

Comunicaciones

Manejo del e-mail.	1	2	3	4	5
Formatos de archivos.	1	2	3	4	5
Comprender, instalar y utilizar software de emulación de terminales.	1	2	3	4	5

Administración de documentos

Transferencia de documentos.	1	2	3	4	5
Conversión de formatos de archivos.	1	2	3	4	5
Uso de áreas compartidas, manejo de archivos públicos.	1	2	3	4	5
Comprender el uso de versiones, candados y seguridad de archivos.	1	2	3	4	5

Redes

Comprender las características y funciones de la red en un ambiente cliente/servidor.	1	2	3	4	5
Conocimientos básicos de cableado.	1	2	3	4	5
Conocimiento de los métodos de acceso a los recursos compartidos en la red.	1	2	3	4	5

Arquitectura Cliente/Servidor

CAPÍTULO IV
ESTRATEGIAS DE MERCADO

OBJETIVO

Comprender la importancia que tiene para una empresa la implantación de un ambiente cliente/servidor.

INTRODUCCIÓN

La computación cliente/servidor representa una innovadora, distinta y ventajosa capacidad que permite que nuevas y mejores soluciones de cómputo sean aplicadas a las prácticas de la empresa. La computación cliente/servidor permite que la voz, procesamiento y datos se unifiquen entre sí, y si se reacciona con la estrategia apropiada, se obtendrán importantes ventajas, en términos de mejorar la posición competitiva de la empresa.

¿Por qué no tratar a la computación cliente/servidor sólo como otra estrategia y dejar su introducción como un proceso normal de transición?

¿Por qué no tratarla de la misma forma que una hoja de cálculo o procesador de palabras o como cualquier otro producto nuevo?

¿Qué la hace tan especial que requiere una planeación estratégica?

Las preguntas que arriba aparecen serán el caso de estudio de este capítulo, asimismo se explicará por qué es importante ligar la tecnología de información con el negocio, en vez de ligar el negocio con la tecnología de información.

PLANEACIÓN ESTRATÉGICA

Se refiere a la disciplina por medio de la cual las metas y estrategias son desarrolladas para guiar a largo plazo la dirección de la compañía. Los aspectos de las estrategias son administrados por el consejo ejecutivo de la empresa.

La figura 4.1 muestra una vista inicial del proceso de planeación estratégica. Esta figura puede ser entendida como a continuación se explica:

1. Los datos estratégicos son reunidos para su análisis interno (sobre la compañía) y externo (el ambiente que impacta a la compañía).

Las áreas estratégicas de mayor representación son: competencia, mercado, recursos humanos, productos, métodos de venta y administración de sistemas. Diferentes métodos analíticos como el análisis de oportunidades/amenazas, análisis de los factores críticos para el éxito y el análisis de la cinco fuerzas son utilizados para interpretar estos datos.

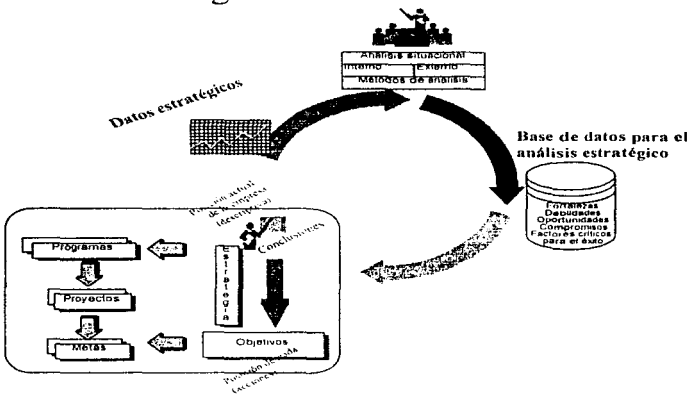


Figura 4.1 Planeación estratégica

Al proceso de colección y análisis de datos estratégicos, se le llama **análisis situacional.** El análisis situacional permite crear una base de datos que opera como repositorio para la toma de decisiones. La siguiente tabla muestra alguna de las áreas que son consideradas durante el análisis situacional.

Area estratégica	Análisis Interno	Análisis Externo
Productos/servicios	✓	
Mercado	✓	✓
Clientes	✓	✓
Tecnología	✓	
Métodos de producción	✓	
Métodos de distribución	✓	
Recursos naturales	✓	✓
Proveedores		✓
Recursos humanos	✓	✓
Sistemas de información	✓	
Finanzas	✓	
Competencia		✓

2. Las conclusiones son establecidas para aquellas opciones que requieren de una respuesta estratégica o iniciativa. Éstas describen el estado actual de las áreas estratégicas, permitiendo determinar cuáles son los aspectos que requieren de mayor atención.

3. Se definen las metas que identifican en dónde necesita estar la empresa ¿Qué es lo que debe estar terminado al final de este periodo de planeación? Las metas son específicas, medibles y fechadas.

Las metas permiten a la empresa obtener una nueva posición de competencia.

4. Se generan las estrategias, las cuales son acciones determinadas y coherentes, necesarias para alcanzar las metas. Las estrategias son el medio por el cual se moverá la empresa de la posición en la que se encuentra (las conclusiones) a donde queremos estar (las metas). Las estrategias se distinguen de las conclusiones en el hecho de que las conclusiones son descriptivas mientras que las estrategias son acciones pot realizar.

5. Las estrategias se implementan a través de programas.

La computación cliente/servidor requiere de una **planeación estratégica**, es decir, es una estrategia que necesita ser entendida, manejada e implementada dentro del contexto de la figura 4.1.

PERSPECTIVA ADMINISTRATIVA

La figura 4.2 ilustra la estructura de la empresa y las relaciones del IM&M[1] con ésta.

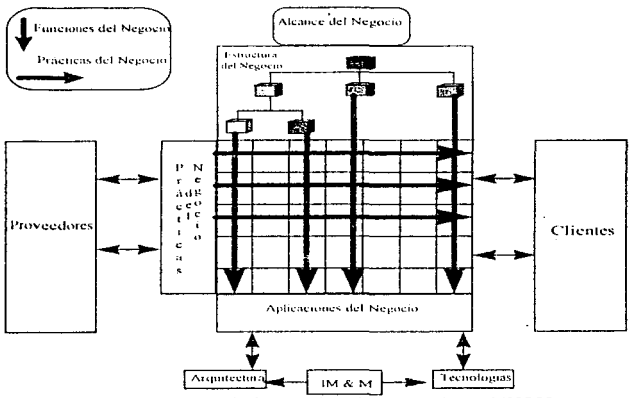


Figura 4.2 Perspectiva administrativa del negocio y el IM&M

[1] IM&M (*Information Movement and Managment*) es la preparación, colección, transporte, obtención, almacenamiento, acceso, presentación y transformación de la información en todas sus formas (voz, gráficos, texto, video e imágenes). El intercambio de información puede generarse entre personas, personas y computadoras, y computadoras. El control asegura la selección apropiada, administración, operación, mantenimiento y evolución de los objetivos del sistema de acuerdo con los objetivos y metas de la empresa.

La figura 4.2 puede ser entendida de la siguiente forma:

1. La empresa tiene un ámbito que define sus principales atributos y límites. Un ambiente típico se ilustra en la siguiente tabla.

Atributo	Definición
Misión	El propósito del negocio
Valores	En lo que cree el negocio
Clientes/mercados	¿A quién se le venderá?
Geografía	¿Dondé se venderá?
Productos/Servicios	¿Qué se vende?
Estrategia	¿Cuál es la posición final que se desea obtener a largo plazo?
¿Cuál es el soporte de la ventaja competitiva (SCA, *Sustainable Competitive Advantage*) ?	¿Qué es lo que atrae a los clientes y disuade a sus competidores?

2. La empresa se organiza en unidades funcionales para realizar su misión.

3. Una unidad realiza una o más funciones. La empresa arroja productos o servicios y opera ejecutando las prácticas del negocio. Las prácticas trabajan para alcanzar los objetivos de la corporación.

Algunas de las prácticas del negocio son esenciales y son conocidas como **prácticas críticas del negocio**, éstas son las prácticas que deben ser realizadas en forma eficiente si se desea competir exitosamente.

4. Las prácticas del negocio son automatizadas como aplicaciones, las cuales utilizan las herramientas de software apropiadas para obtener el máximo de ventajas en su ejecución. Estas ventajas pueden ser reducción de tiempo, económicas, de capacidad, de volumen, nivel de servicio, etc. Mientras que los requerimientos de capacidad de una práctica del negocio pueden permanecer constantes al paso del tiempo, el método de solución es dinámico debido a la evolución de la tecnología, lo cual es la fuente de la ventaja.

Las prácticas del negocio requieren de ser automatizadas utilizando la tecnología más adecuada para ello, recordemos que no todas las prácticas son iguales, también notemos que normalmente para cubrir completamente una necesidad del negocio se requiere de la integración de más de una tecnología.

La perspectiva administrativa de la relación entre el negocio y el IM&M (tecnología de información) es de la siguiente manera:

- ✓ El objetivo del negocio es realizar su misión.
- ✓ El negocio se organiza funcionalmente para lograr sus objetivos.
- ✓ Las prácticas del negocio son el medio por el cual se relacionan las unidades funcionales de la empresa para ofrecer un producto o servicio.
- ✓ Seleccionando la solución de IM&M apropiada, las prácticas del negocio son automatizadas como aplicaciones.
- ✓ **El propósito del IM&M es permitir (habilitar) las prácticas del negocio.**

La perspectiva administrativa ve al IM&M como otro recurso, que puede ser utilizado para alcanzar la misión de la compañía. Por lo que es importante considerar al IM&M dentro del contexto de la planeación estratégica.

Ahora que ya se ha comprendió por qué es importante ver al IM&M desde la perspectiva administrativa, la siguiente pregunta por responder sería ¿por qué es tan importante comprender y utilizar la estrategia de la computación cliente/servidor? En la sección *Los problemas del negocio* se presentarán evidencias que muestran que actualmente con la utilización de las herramientas comunes para la construcción de las aplicaciones existen graves problemas que impiden la explotación efectiva del sistema de información para lograr una ventaja competitiva. Un gran porcentaje de estos problemas radica en la utilización de arquitecturas "débiles", que no ofrecen la posibilidad de crecimiento y flexibilidad necesaria para las prácticas del negocio. El uso de una arquitectura "poderosa" predispone el éxito de las aplicaciones debido a que éstas heredan las ventajas de la arquitectura (flexibilidad, escalabilidad, modularidad, mantenibilidad, etc.). Este es un punto crítico ya que la arquitectura de cómputo en la cual se soportan las aplicaciones de la empresa define el grado de competitividad de la misma.

Las conclusiones de acuerdo con los argumentos vistos en esta sección son:

- La planeación estratégica provee del medio para determinar las actividades de importancia crítica a largo plazo necesarias para mantener la competitividad de la empresa.
- La razón de la existencia del IM&M es la de establecer la mejor solución de cómputo posible para la automatización de las prácticas del negocio, es decir, **el valor de la computación radica en las aplicaciones y no en los microprocesadores, dispositivos de almacenamiento, etc.**
- Debido a que la arquitectura cliente/servidor es una arquitectura "poderosa" (como lo probamos en los capítulos anteriores) deberá ser colocada como el fundamento de las aplicaciones del negocio; esto permitirá heredar sus atributos hacia las aplicaciones, lo cual es de importancia crítica para la competitividad a largo plazo de la empresa.

En resumen, las soluciones del IM&M deben ser ligadas con las prácticas del negocio con el propósito de obtener mayores rendimientos, es decir, las prácticas del negocio vienen primero.

COMPETENCIA

El propósito de esta sección es desarrollar un modelo a través del cual se pueda evaluar la competitividad de una empresa. El ambiente operativo normal para la mayoría de la compañías, excepto aquéllas que poseen un estatus de monopolio, es la competencia.

Las ventajas competitivas usadas efectivamente contribuyen a mejorar la competitividad de la empresa, mientras que las ventajas no utilizadas a favor de la empresa drenan y ponen en peligro la competitividad con las consecuencias de la decadencia y "muerte" de ésta.

Se usará un modelo conocido como "el modelo de las cinco fuerzas" (*Five Forces Model*) para evaluar la competitividad de una empresa. Este modelo es una metodología utilizada para entender la dinámica de las fuerzas que afectan a una empresa, las cuales se combinan para determinar la situación y estado de una empresa.

Modelo de las cinco fuerzas

Este modelo establece que el estado de competencia en una empresa está en función de la dinámica de cinco fuerzas:

Proveedores	El poder de los distribuidores de la industria para controlar precios, calidad y condiciones globales de compra de bienes y servicios.
Clientes	El poder de lo clientes, los cuales influyen en precios y calidad.
Nuevas compañías	El grado de probabilidad de que nuevos competidores entren al mercado.
Productos sustitutos	La disponibilidad y atractivo de productos sustitutos para los clientes.
Rivalidad entre los competidores existentes	La intensidad de las "maniobras para conseguir una posición" entre los competidores.

La posición de una empresa relativa a estas cinco fuerzas determina en cualquier momento su grado de competencia. La figura 4.3 ilustra la relación de la empresa con las cinco fuerzas anteriormente mencionadas.

La empresa se encuentra en una constante batalla por mejorar o por lo menos sostener su posición de competencia. Para mantenerse en el mercado se emprenderán iniciativas para mejorar su posición con respecto a algunas de las fuerzas. Igualmente, es posible en cualquier momento, reaccionar a las iniciativas de otros quienes están tratando de mejorar su posición.

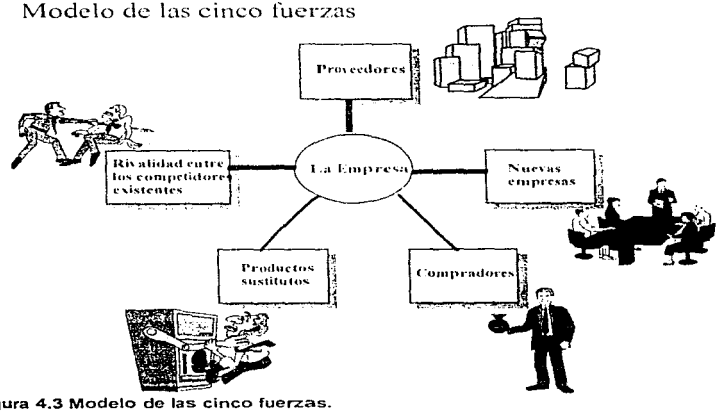


Figura 4.3 Modelo de las cinco fuerzas.

Con esta explicación básica del modelo de las cinco fuerzas, podemos evaluar el grado de competitividad de una empresa. En cualquier momento, las cinco fuerzas impactarán en la empresa de la siguiente forma:

- **No existe impacto** — Los factores del modelo están inactivos.
- **Acciones realizadas por otros** — Un distribuidor, cliente, rival, etc., toman acciones para mejorar su posición, es necesario reaccionar prontamente o existe un riesgo potencial de perder mercado.
- **Acciones realizadas por la empresa** — La empresa inicia una acción para mejorar su posición a través del modelo de las cinco fuerzas, a costa de los distribuidores, clientes, competidores potenciales, etc.
- **Acción externa** — Un evento ocurre fuera del modelo de las cinco fuerzas (guerra, fallas de corriente eléctrica, crisis económicas, etc.), por lo que se requiere una respuesta por parte de la empresa.

Lo que el modelo de las cinco fuerzas realmente expresa es la necesidad primaria de cambios en las acciones de la empresa, con objeto de mantener su competitividad. Los clientes, diferenciación de productos, lealtad, ventaja del precio, etc., son ventajas transitorias que se encuentran en constante cambio. La verdadera ventaja radica en la habilidad de adaptación, actuación y reacción en el momento que se requiera, por lo que la única ventaja competitiva real sobre los competidores es la capacidad de reaccionar, cambiar y adaptarse en una forma más rápida y eficiente que los contrincantes.

La conclusión de este análisis nos lleva a determinar cuales deben ser los atributos de lo que denominaremos una arquitectura IM&M "poderosa", si el objetivo es el cambio y la adaptabilidad, la arquitectura, en la que se soporten las aplicaciones del negocio, debe poseer los siguientes atributos.

- **Mantenibilidad** — La facilidad para el mantenimiento de la arquitectura.
- **Modularidad** — La habilidad de agregar, modificar y quitar piezas de la arquitectura.
- **Escalabilidad** — Es la habilidad para escalar la arquitectura debido al aumento del volumen de las transacciones, almacenamiento de datos y usuarios.
- **Adaptabilidad** — La facilidad para el cambio.
- **Portabilidad** — Es la habilidad para mudar las aplicaciones en forma total o parcialmente, hacia un lugar diferente dentro de la arquitectura.
- **Abierto/soportado por estándares** — El uso de estándares permite la interoperabilidad de las aplicaciones.
- **Flexibilidad** — La habilidad para crecer y contraer la arquitectura como sea requerida.
- **Autonomía** — Es la habilidad de operar en forma individual, o bien, como parte de un todo.

• **Accesibilidad de datos**	La habilidad para accesar datos locales o remotos de manera transparente.
• **Interoperabilidad**	Es la habilidad de migrar transacciones y datos corporativos a través de un ambiente heterogéneo.
• **Conectividad de aplicaciones**	Es la habilidad de conectar una gran variedad de aplicaciones a la arquitectura.

La suma de estos atributos genera la capacidad de **maniobrabilidad**. La maniobrabilidad es el requerimiento primario que las prácticas del negocio impone al *IM&M*.

ASPECTOS CRÍTICOS DE LOS SISTEMAS DE INFORMACIÓN

El propósito de esta sección es analizar los aspectos críticos de los sistemas de información. El grado en el cual cualquier arquitectura permite o bloquea el éxito de los objetivos de las aplicaciones es otro aspecto para la evaluación de la utilidad de la arquitectura.

Aspectos críticos

A continuación se enlistan los principales aspectos que definen el éxito de la implantación de un sistema de información en una empresa:

Aspecto 1
Reconfiguración de las prácticas del negocio a través de la tecnología de información

Definición: Este aspecto está relacionado directamente con la reingeniería de los procesos de la empresa. La tecnología informática debe conducir las acciones futuras de la empresa.

Aspecto 2
Alinear las metas de los sistemas de información junto con las metas de la empresa.

Definición: Las metas de la empresa conducen los objetivos del sistema de información y ambos deben de coincidir en sus fines.

Comentarios: Recordemos "La perspectiva administrativa" la cual afirma que el único propósito del sistema de información es permitir (habilitar) las prácticas del negocio.

Aspecto 3
Sistema de funciones cruzadas

Definición: Este punto reconoce que las prácticas del negocio están organizadas con una funcionalidad cruzada (las unidades que forman a la empresa se relacionan entre sí) por lo que las aplicaciones deben tomar esto en cuenta para poder brindar servicios eficientes a la empresa.

Comentario: Se reconoce que no existe, algo así, como un sistema único del negocio. Sino que realmente existe un proceso continuo formado por subprocesos del negocio interconectados entre sí y que debido a su complejidad, tecnología, necesidad, etc., es particionado en unidades.

Aspecto 4
Estimular la productividad del software

Definición: Este aspecto enfatiza la importancia de mejorar continuamente la productividad del software. Esto permite la creación y mejoramiento de las aplicaciones de una forma más rápida. Productividad, en este contexto, debe ser entendida en la funcionalidad ofrecida por el software

Aspecto 5
Utilización de datos

Definición: Este aspecto reconoce la importancia ascendente de los datos como un recurso corporativo estratégico.

Comentario: En este aspecto se enfatiza la mayor importancia del software (datos) sobre el hardware.

Aspecto 6
Desarrollo de un plan estratégico para la implantación de los sistemas de información

Definición: Este aspecto reconoce la importancia del plan estratégico del sistema de información para soportar el plan estratégico del negocio.

Comentario: Se requiere que el plan tecnológico sea congruente con el plan del negocio, otra vez el negocio es primero. La decisión estratégica de mayor importancia que el departamento de sistemas de información debe realizar, es la de definir la arquitectura de cómputo en la que se soportará el sistema. Esta arquitectura no sólo debe ser capaz de resolver las necesidades actuales del negocio, sino también debe ser capaz de crear el contexto necesario para nuevas estrategias del negocio. Lo que el consejo directivo desea del sistema es la posibilidad de cambiar el curso del negocio y obtener resultados lo más rápidamente posible.

Aspecto 7
Mejorando la calidad del desarrollo del software

Definición: Este aspecto balancea al punto 4 "Estimulando la productividad del software", mientras que el punto 4 tiene que ver con realizar las aplicaciones en la forma más rápida posible, este punto tiene que ver con construir las aplicaciones con la calidad requerida, claro está que ambas características son necesarias.

Comentarios: Este aspecto involucra la necesidad de igualar la tecnología con los requerimientos específicos de las prácticas del negocio.

Aspecto 8
Creación de una arquitectura de información

Definición: Este punto reconoce la necesidad de prevenir la "entropía" (caos) del sistema, lo cual se logra por medio de la construcciones de aplicaciones dentro de un contexto de una infraestructura consistente, formada por estándares, arquitecturas de datos y arquitecturas de procesamiento.

Comentario: La figura 4.4, "Óptima arquitectura de IM&M" provee de los elementos básicos para una arquitectura de información completa, la cual requiere de un conjunto de estándares y prácticas apropiadas.

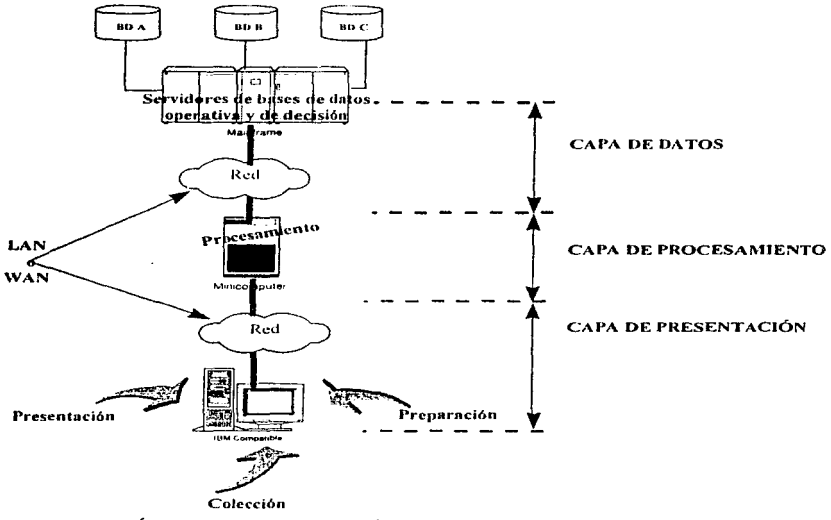


figura 4.4 Óptima arquitectura de IM&M

Aspecto 9
Integrando los sistemas de información

Definición: Este aspecto reconoce los requerimientos necesarios para la interoperabilidad de los sistemas a través de un ambiente heterogéneo de hardware y software.

Comentarios: La arquitectura cliente/servidor posee los atributos de mantenibilidad, modularidad, escalabilidad, adaptabilidad, portabilidad, basado en estandares, etc. Estos atributos hacen de la arquitectura cliente/servidor el medio óptimo para la implantación de sistemas heterogéneos.

Aspecto 10
Reducción de los costos de los sistemas de información

Definición: Este aspecto está relacionado con el continuo mantenimiento o mejoramiento de los servicios del IM&M simultáneamente con una reducción de gastos.

Comentario: Si vemos la relación ingresos/gastos de IM&M como una medida financiera importante, el IM&M no debe ser visto como un gasto necesario y costoso, sino como un componente que agrega valor a los productos y servicios de una empresa, por lo que el IM&M debe ser tomado como una ventaja y no como un gasto.

Aspecto 11
Utilizando los sistemas de información como una ventaja competitividad

Definición: Este aspecto es el opuesto al del punto 10, se enfatiza el uso de los sistemas como un medio para mejorar en forma radical los procesos y capacidades del negocio.

Comentarios: La ventaja más importante que otorgan los sistemas de información a una empresa es la de permitirle evolucionar continuamente de acuerdo con sus necesidades y al ambiente que la rodea.

Aspecto 12
Mejorando los recursos humanos del departamento de sistemas

Definición: Este aspecto está relacionado con el continuo crecimiento y mejoramiento del "staff" del IM&M.

Aspecto 13
Entrenamiento a los ejecutivos de la empresa en tópicos de sistemas de información

Definición: Este aspecto está relacionado con el adiestramiento de los directores, tomando en cuenta que los sistemas de información permiten alcanzar los objetivos de la empresa de una forma más rápida y eficiente.

Aspecto 14
Conectándose con los clientes y proveedores

Definición: Este aspecto está relacionado con el mejoramiento de la productividad por medio de enlaces electrónicos con los clientes y proveedores. Se reconoce que el mejoramiento en la productividad es posible por medio de mejorar el flujo de la información hacia los socios del negocio.

Comentarios: Este aspecto tiene algunas implicaciones muy importantes, la figura 4.2 ilustra cómo una arquitectura IM&M poderosa puede ser la base de las aplicaciones y por lo tanto los fundamentos de las prácticas del negocio. La figura 4.4 es la misma que la figura 4.2, pero enfatiza el punto de vista del cliente y de los proveedores. Existe un gran flujo de información entre la empresa, clientes y proveedores, las prácticas no son discretas ni separadas, sino continuas. La figura 4.5 ilustra la estrategia a seguir.

La integración de las prácticas del negocio con los clientes y proveedores genera:

- Junto con los distribuidores se aumenta la productividad.
- Se mejora los servicios hacia el usuario.

Conectando a los clientes y distribuidores se logra un escalamiento horizontal de la economía, lo cual es beneficioso para ambos, pero no lo es así para los nuevos competidores.

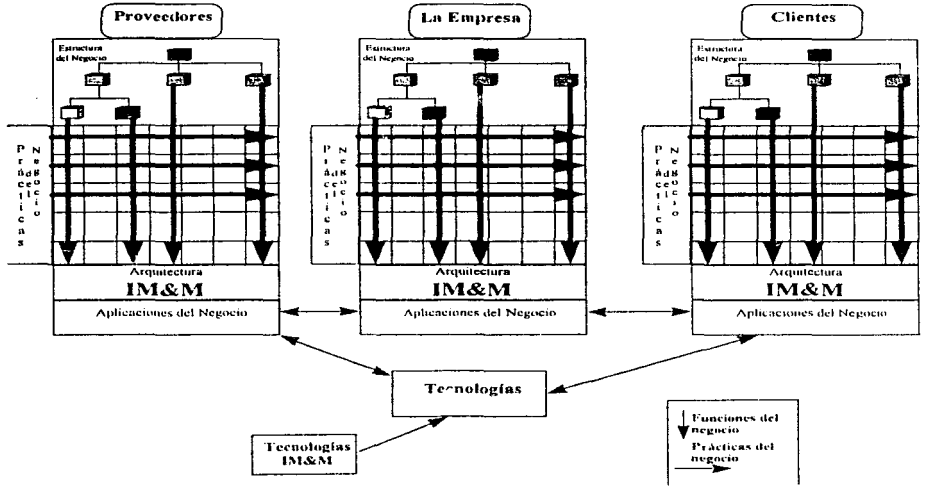


Figura 4.4 Relación entre clientes y proveedores

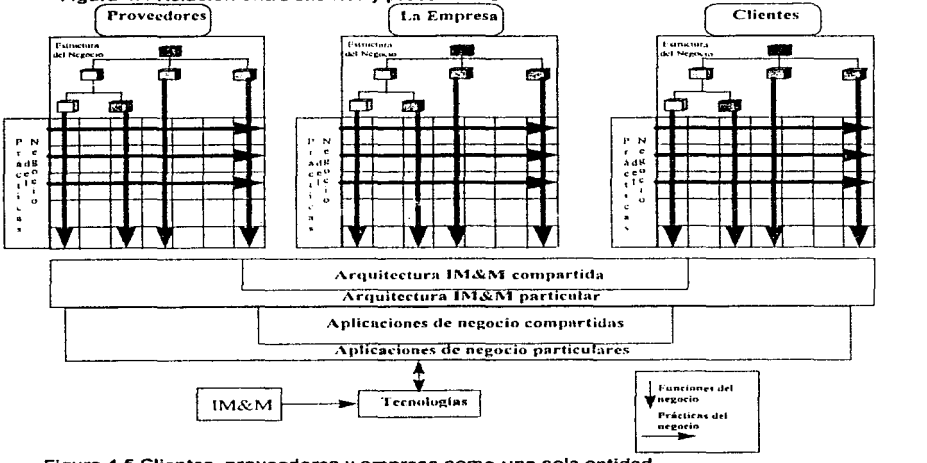


Figura 4.5 Clientes, proveedores y empresa como una sola entidad

Aspecto 15
Controlando los cambios originados por la tecnología

Definición: Este aspecto se refiere al hecho de que la introducción de nuevas tecnologías modifica la forma de trabajar, por lo que es necesario prevenir estos problemas, anticipándolos e incorporándolos en el plan de implantación.

Aspecto 16
Promover la funcionalidad de los sistemas de información

Definición: Este aspecto se refiere a mantener a los "directores" conscientes de la contribución que el IM&M hace al negocio.

Aspecto 17
Determinación del valor de los sistemas de información

Definición: Este aspecto tiene que ver con la relación costo/beneficio del IM&M.

Comentario: Este es un aspecto interesante porque implica que el IM&M tiene un valor intrínseco. El IM&M es valioso únicamente cuando permite la realización de las prácticas del negocio. Un excelente ejemplo de esto es la proliferación de las redes de área local (LAN).

Aspecto 18
Administración de sistemas dispersos

Definición: Este aspecto está relacionado en cómo los sistemas tradicionales de información pueden soportar la gran cantidad de procesamiento dispersado a través de la empresa. Es reconocido que una enorme redistribución del poder de cómputo está ocurriendo y está relacionado con el control de las inversiones. Algunos consultores estiman que entre el 70 u 80% del total de los recursos de cómputo de una empresa están localizados fuera del área central.

Aspecto 19
Inversión en tecnologías

Definición: Este aspecto tiene que ver con la identificación e incorporación de nuevas tecnologías a la empresa, y está relacionado estrechamente con la reingeniería aplicada a la empresa.

En este tema se han revisado algunos de los factores críticos que deben ser considerados en el diseño de un sistema de información. Utilizaremos lo visto aquí en la siguiente sección en donde se analizarán las repercusiones que tiene sobre la empresa el no tomar en cuenta estos factores al momento de implantar un sistema de información.

EL PROBLEMA DEL NEGOCIO

El objetivo es analizar los problemas que impiden obtener las ventajas del uso de los sistemas de información para resolver los problemas del negocio.

El uso del **modelo médico** como un medio para diagnosticar los problemas asociados con el IM&M es una herramienta muy poderosa. No sólo porque permite conocer el problema en una forma profunda y estructurada, sino también permite entender exactamente qué componente del IM&M es el que está fallando.

Modelo médico

Este modelo divide la enfermedad en tres principales componentes: valoración, diagnóstico y tratamiento. Se particiona el problema para la concentración de los esfuerzos, lo cual permite múltiples tratamientos, diferenciando entre el **síntoma**, la **patología** y la **etiología** de la enfermedad.

La enfermedad (problemas) puede ser dividida en tres componentes:

- **Síntoma** Es la manifestación externa de la enfermedad; son las señales visibles de que algo está mal.
- **Patología** La patología es el elemento específico que está enfermo, es el órgano cuyo mal funcionamiento está causando los síntomas.
- **Etiología** La etiología son las "raíces" de la enfermedad (problema), identifica el "agente" que ocasiona la patología.

Este es un modelo analítico muy poderoso, utilizado en el diagnóstico de un problema y distingue entre el problema superficial, el componente que no funciona y las verdaderas raíces que causan el problema.

Cuando en una empresa se establecen soluciones que sólo atacan los "síntomas" se generan costos excesivos, debido a que la enfermedad es recurrente, ya que exclusivamente se atacan los síntomas. El modelo médico permite prescribir la "curación" óptima, dependiendo de qué componente (síntoma, patología, etiología) del diagnóstico está siendo tratado.

Habiendo analizado el sistema de información a través del modelo médico, se estará en posición de demostrar más adelante, la eficiencia de la computación cliente/servidor no solamente en su capacidad de "aliviar" los síntomas del problema, sino también de atacar la patología y la etiología.

¿Cuáles son algunos de estos síntomas? Actualmente algunos de los síntomas de los sistemas de información son: rigidez/inflexibilidad, altos costos de mantenimiento, dispersión de datos, insatisfacción por parte de los usuarios, fragilidad de las aplicaciones, soluciones propietarias o fijas, resistencia al cambio, sistemas monolíticos y dependencia en tecnología obsoleta, por mencionar algunos.

La siguiente tabla analiza los síntomas relativos a los criterios tradicionales de desventaja como son costo, tiempo y funcionalidad. La **P** indica que se trata de una desventaja primaria y la **S** indica una desventaja secundaria.

Síntomas	Costos	Tiempo	Funcionalidad
Sistemas rígidos/inflexibles		P	S
Altos costos de mantenimiento	P	S	
Inconsistencia de datos	P	S	
Escasa documentación	P	S	
Insatisfacción de los usuarios	S		P
Dependencia en expertos	S	P	
Fragilidad de las aplicaciones	P	S	
Soluciones propietarias	S		P
Soluciones fijas y difíciles de escalar		S	P
Resistencia al cambio		S	P
sistemas monolíticos		P	S
Dependencia en tecnología obsoleta	S		P
Inmovilidad	P	P	P

A continuación se analizarán las principales enfermedades de los sistemas de información (estas enfermedades generan los síntomas anteriormente mencionados), para describirlas se utilizará la estructura de enfermedad, patología y etiología.

Enfermedad

Incapacidad de la estructura de procesamiento.

Definición

Incapacidad de explotar la arquitectura de cómputo para derivar ventajas competitivas para la empresa.

Síntomas

- Sistemas rígidos e inflexibles.
- Altos costos de mantenimiento.
- Fragilidad de los sistemas.
- Insatisfacciones por parte de los usuarios.
- Sistemas monolíticos.
- Dependencia en tecnología obsoleta.

Patología

La debilidad de la arquitectura se refleja en las aplicaciones.

Etiología

Uso de una tecnología emergente e inmadura. La mayoría de lo que se ha hecho hasta ahora en el área de cómputo, ya sea en arquitecturas de procesamiento, arquitecturas de datos, o desarrollo de software ha tenido la característica en común de tener una etapa de inestabilidad (inmadurez) en donde el sentido común y la intuición son los orígenes de las soluciones informáticas.

Enfermedad	Incapacidad del software.
Definición	La inflexibilidad y costo del mantenimiento del software imposibilita la construcción de nuevas aplicaciones que ofrezcan ventajas al negocio.
Síntomas	• Altos costos de mantenimiento. • Escasa documentación. • Desatisfacciones por parte de los usuarios. • Resistencia al cambio. • Dependencia en las herramientas.
Patología	Uso de herramientas de bajo nivel, escasez de procesos estructurados.
Etiología	Disciplinas inmaduras, escasez de herramientas y trabajo artesanal. El desarrollo de software debe construirse dentro de un modelo basado en técnicas de ingeniería y no artesanales.

Enfermedad	Incapacidad de los datos.
Definición	Es la incapacidad de explotar y administrar los datos.
Síntomas	• Altos costos de mantenimiento. • Insatisfacciones de los usuarios. • Dependencia en tecnologías obsoletas. • Dispersión de datos.
Patología	Manejo de archivos y arquitecturas de bases de datos cerradas, como fundamento para la administración de los datos.
Etiología	Uso de disciplinas inmaduras que no permiten (o se niegan) a adaptarse a los principios de ingeniería de software. De todas las disciplinas de la ingeniería de IM&M, la ingeniería de datos ha realizado el mayor progreso, estableciendo principios, procesos de análisis, modelado y administración de datos, y en la mayoría de las compañías, simplemente, no lo implantan.
Enfermedad	Incapacidad de la organización.
Definición	Involucra la imposibilidad de la tecnología de ser aplicada exitosamente sobre los recursos de la empresa.
Síntomas	• Insatisfacción por parte de los usuarios. • Soluciones propietarias. • Soluciones inflexibles. • Resistencia al cambio.
Patología	Carencia de vínculos entre la tecnología y las necesidades del negocio.
Etiología	Una tecnología opuesta a los objetivos del negocio.

Lo que es particularmente interesante sobre estas "enfermedades", es que generalmente se presentan juntas, la mayoría de las compañías experimentan todas, no solamente una o dos; por lo que existe una clase de enfermedad compuesta, la cual se describe de la siguiente forma.

Enfermedad	Incapacidad del IM&M.
Definición	Esto puede ser descrito como el efecto negativo colectivo de una pobre arquitectura de procesamiento, administración de datos y desarrollo de software, que afecta al sistema de información y evita obtener las ventajas competitivas deseadas.
Síntomas	Inmovilidad.
Patología	• Frágil arquitectura de procesamiento. • No existe una ingeniería de software. • Mala administración de los datos, por lo que no es posible obtener ventajas.
Etiología	• Disciplinas inmaduras. • Carencia de herramientas. • Mala administración de los datos, por lo que no se obtienen ventajas.

En esta sección se ha analizado los problemas asociados con el IM&M, utilizando el **modelo médico** como la herramienta analítica. Se demostró que los problemas asociados con el IM&M derivan en cuatro enfermedades, que combinadas limitan la movilidad de la organización.

El impacto de las enfermedades es severo, debido a que imposibilitan la competitividad de la empresa. La siguiente tabla muestra las acciones necesarias para resolver las enfermedades del IM&M. así como la relación entre los aspectos críticos del sistema de información y la enfermedad asociada.

Aspectos	Incapacidad de la arquitectura de procesamiento	Incapacidad del software	Incapacidad de los datos	Incapacidad de la organización
Redefinir las prácticas del negocio a través de la tecnología de información.	✓	✓	✓	✓
Alinear las metas de los sistemas de información junto con las metas de la empresa.	✓			
Implementación de un sistema modular, con funciones cruzadas (dependencias).	✓		✓	
Estimular la productividad del software.	✓			
Desarrollo de un plan estratégico para la área de sistemas.	✓			
Mejorar la calidad del desarrollo de aplicaciones.	✓		✓	
Creación de una arquitectura de información.	✓	✓	✓	✓

Aspectos	Incapacidad de la arquitectura de procesamiento	Incapacidad del software	Incapacidad de los datos	Incapacidad de la organización
Integración de los sistemas de información.	✓			
Reducción del los costos de desarrollo.	✓	✓	✓	✓
Mejorar los recursos humanos del departamento de sistemas.				✓
Conectando clientes y proveedores.	✓	✓		
Controlando los cambios causados por la introducción de la tecnologia de información.				
Promover la función de los sistemas de información.	✓	✓	✓	✓
Determinación del valor de los sistemas de información.				
Control de los sistemas dispersos.	✓	✓	✓	✓

Es así como el IM&M es afectado por cuatro "enfermedades" (que se relacionan entre sí), las cuales son:

- Incapacidad de la estructura de procesamiento.
- Incapacidad del software.
- Incapacidad de los datos.
- Incapacidad de la organización.

Estas enfermedades, por lo general, se presentan juntas y crean una nueva enfermedad, la cual denominaremos:

- Incapacidad del IM&M.

La consecuencia final de dichas enfermedades, en su conjunto, es la **inmovilidad,** es decir, la incapacidad de la empresa de responder a la competencia.

La estrategia por seguir para el tratamiento y curación de dichas enfermedades es la migración o implantación de un ambiente cliente/servidor, en la siguiente sección justificaremos esto.

Ventaja competitiva

El propósito de esta sección es comprender las ventajas otorgadas por la computación cliente/servidor, se analizará su eficacia para resolver las enfermedades (problemas) del IM&M.

Para cada una de las enfermedades mencionadas en la sección anterior, se explicarán los siguientes tópicos:

☞ Enfermedad.

☞ Efecto terapéutico de la computación cliente/servidor.

☞ Comentarios.

Comencemos, la computación cliente/servidor ofrece el siguiente alivio.

Enfermedad

Incapacidad de la arquitectura de procesamiento

Impacto. Resuelve la patología de una arquitectura débil.

Comentario. La arquitectura cliente/servidor ofrece a la empresa los atributos de:

- Mantenibilidad
- Modularidad
- Adaptabilidad
- Escalabilidad
- Portabilidad
- Sistemas abiertos/estándares
- Autonomía
- Flexibilidad
- Accesibilidad de datos
- Interoperabilidad

Cada uno de estos factores fueron analizados con detalle en capítulos anteriores. La combinación de ellos permite a la empresa maniobrar y obtener resultados lo más rápidamente posible.

Incapacidad del software

Impacto. Alivia los síntomas.

Comentarios: La arquitectura cliente/servidor permite aliviar la mayoría de los síntomas de esta enfermedad:

- Reducir el costo de mantenimiento debido a que las aplicaciones heredan los atributos de la arquitectura sobre la cual son construidas. Las aplicaciones son por lo tanto escalables, portables, modulares, etc. Todos estos atributos hacen que sea más fácil modificar las aplicaciones para que respondan a las necesidades del negocio.

- Reducir los costos de desarrollo debido a que permite contar con un ambiente de desarrollo escalable, diferente al ambiente de producción.

- Mejorar la productividad del software eliminando o minimizando la necesidad de desarrollar nuevo software; esto se debe en gran medida a que la programación se realiza de una forma modular, lo cual permite reutilizar módulos en diferentes aplicaciones o inclusive realizar nuevas aplicaciones, utilizando únicamente los módulos ya existentes.

Nota importante:
En este caso la computación cliente/servidor sólo alivia los síntomas, por lo que no elimina la patología como tampoco la etiología. Otras estrategias como son CASE, orientación a objetos, etc., requieren de ser adoptadas para atacar este problema.

Incapacidad de los datos

Impacto. Tratamiento parcial a la patología.

Comentarios. La computación cliente/servidor es un requisito para la creación del ambiente operativo, como también del ambiente de toma de decisiones (DSS, *Decisión Support System*). Antes de esto, aunque se desee crear la infraestructura de base de datos, no existe una arquitectura de procesamiento que permita el acceso de las aplicaciones a los datos.

Nota importante:
En este caso la computación cliente/servidor sólo ayuda y no se resuelve la etiología del problema. La ausencia de una apropiada administración de base de datos creará una masa de datos inconsistentes.

Incapacidad de la organización

Impacto: Podría ser peor.

Comentario: Un ambiente cliente/servidor para tener éxito requiere:

- Administración de datos.
- Estándares en redes.
- Arquitectura de datos.
- Interfaces estándares (API)
- Metodologías estándares de distribución del software.

Nota importante:
A menos que se tenga una clara definición de los objetivos, metas y la capacidad técnica se logrará mejorar la posición competitiva de la empresa a través del uso del ambiente cliente/servidor, en caso contrario, es probable que los problemas se aumenten en vez de reducirse.

Conclusión: La computación cliente/servidor mejora la "salud" del IM&M de la empresa. No es una cura milagrosa para todas las enfermedades pero, como se resume en la siguiente tabla, provee de un tratamiento significativo para tres de ellas.

Enfermedad	Impacto
Incapacidad de la arquitectura	Cura la patología.
Incapacidad del software	Elimina los síntomas.
Incapacidad de los datos	Tratamiento parcial sobre la patología.
Incapacidad de la organización	Resalta los síntomas, posiblemente forzará la creación de una solución.

Impacto del uso de la arquitectura cliente/servidor sobre los aspectos críticos de los sistemas de información

El propósito de esta sección es comprender cómo la computación cliente/servidor permite lograr los objetivos de los aspectos críticos de los sistemas de información, para los cuales, se presentará la siguiente información:

- Aspecto
  - ☞ Principal enfermedad que lo afecta.
  - ☞ Impacto cliente/servidor.
  - ☞ Comentarios.

Aspecto 1	
Reconfiguración de las prácticas del negocio a través de la tecnología de información.	☞ Todas. ☞ Positivo. ☞ El ambiente cliente/servidor provee de la arquitectura maniobrable necesaria para la reingeniería de los procesos de la empresa.
Aspecto 2	
Alinear las metas de los sistemas de información junto con las metas de la empresa.	☞ Incapacidad de la organización. ☞ Positivo. ☞ La computación cliente/servidor permite mejorar los accesos a los datos, compartir recursos y adopción de nuevas tecnologías.
Aspecto 3	
Sistema de funciones cruzadas.	☞ Incapacidad de la arquitectura de procesamiento e incapacidad de los datos. ☞ Positivo. ☞ La CCS (computación cliente/servidor) provee de la arquitectura necesaria para que múltiples aplicaciones accesan/compartan la base de datos operativa y la de decisión.
Aspecto 4	
Estimular la productividad del software.	☞ Incapacidad del software. ☞ Positivo. ☞ La CCS mejora la productividad del software debido a la creación de módulos reutilizables, provee de una plataforma de desarrollo escalable y flexible, ya que se heredan los atributos de la plataforma hacia las aplicaciones que se construyen sobre ella.
Aspecto 5 Utilización de datos.	☞ Incapacidad de los datos. ☞ Positivo. ☞ La CCS provee de la arquitectura necesaria para compartir los datos operativos como los del ambiente de decisión.
Aspecto 6 Desarrollo de un plan estratégico para la implementación de los sistemas de información.	☞ Incapacidad de la organización. ☞ No aplica.

Aspecto 7	
Mejorando la calidad del desarrollo del software.	☞ Incapacidad de la arquitectura de procesamiento e incapacidad del software. ☞ Positivo. ☞ Los atributos de la arquitectura cliente/servidor, como son adaptabilidad, flexibilidad, modularidad, accesibilidad de datos permiten a la empresa desarrollar aplicaciones en una forma más rápida y eficiente, tomando módulos reutilizables.
Aspecto 8	
Creación de una arquitectura de información.	☞ Todas. ☞ Positivo. ☞ La CCS provee de los componentes necesarios.
Aspecto 9	
Integración de sistemas de información.	☞ Incapacidad de la arquitectura de procesamiento. ☞ Positivo. ☞ La interoperabilidad es uno de los atributos básicos de la arquitectura cliente/servidor.
Aspecto 10	
Reducción de los costos de los sistemas de información.	☞ Todas. ☞ Positivo. ☞ La flexibilidad de la CCS minimiza los costos de mantenimiento y aumenta la reutilización.
Aspecto 11	
Utilización de los sistemas de información para el mejoramiento de la competitividad.	☞ Todas. ☞ Positivo. ☞ La arquitectura cliente/servidor permite una rápida adaptación a las necesidades del negocio a través de las nuevas tecnologías.
Aspecto 12	
Mejorando los recursos humanos del departamento de sistemas.	☞ Incapacidad de los datos e incapacidad del software. ☞ Neutral.
Aspecto 13	
Entrenamiento a los ejecutivos de la empresa en tópicos de sistemas de información.	

Aspecto 14

Conectándose con los clientes y proveedores.

- ☞ Incapacidad de la arquitectura de procesamiento e incapacidad de datos.
- ☞ Positivo.
- ☞ La arquitectura cliente/servidor permite construir la plataforma necesaria para la comunicación entre clientes y proveedores.

Aspecto 15

Controlando los cambios originados por la tecnología.

Aspecto 16

Promover la funcionalidad de los sistemas de información.

- ☞ Todas.
- ☞ Positivo.
- ☞ En un ambiente cliente/servidor la empresa tiene la capacidad de reaccionar rápidamente a los cambios, lo que permite mantener o mejorar su posición de competencia en el mercado.

Aspecto 17

Determinación del valor de los sistemas de información.

Aspecto 18

Administración de sistemas dispersos.

- ☞ Todas.
- ☞ Positivo.
- ☞ La CCS provee de la arquitectura necesaria para interconectar las "islas" de datos y procesamiento que se encuentran dentro de la empresa.

Aspecto 19

Inversión en tecnologías.

- ☞ Incapacidad de la arquitectura de procesamiento.
- ☞ Positivo.
- ☞ La CCS permite crecer la arquitectura en una forma incremental, agregando nueva tecnología a la ya existente.

Responsabilidades del departamento de sistemas de información

No existen soluciones estándares cliente/servidor (aunque pueden existir bloques que se puedan reutilizar), ni siquiera se puede asegurar, de forma general, que un modelo de distribución, una herramienta o plataforma sean las mejores. La solución correcta depende, en gran medida, de:

- Los objetivos de negocio de la empresa.
- Del grado de aceptación de las nuevas tecnologías.
- Los recursos disponibles actualmente de hardware, software y conocimientos.

El levantamiento de una infraestructura técnica que se aparte de las necesidades del negocio está condenada al fracaso.

Los factores que determinan la solución cliente/servidor son:

- *Las necesidades del negocio y cómo las tecnologías informáticas pueden soportarlas para conseguir los objetivos comerciales.*
- *El marco de tiempo disponible para la puesta en operación.*
- *El estudio económico (planeación estratégica).*
- *El impacto sobre la organización y los conocimientos requeridos.*
- *Las arquitecturas y estándares ya adoptados por la compañía.*

Pocas empresas pueden permitirse sustituir los sistemas y las aplicaciones existentes en un solo paso. Normalmente es necesario un proceso gradual de implantación de las nuevas aplicaciones, lo cual exige:

- Una infraestructura cliente/servidor capaz de acomodar las aplicaciones existentes junto con las nuevas aplicaciones cliente/servidor.
- Estándares corporativos de tecnología informática.
- Una arquitectura de aplicaciones que permita desarrollar y poner en servicio nuevas aplicaciones, mientras siguen funcionando las existentes, preferentemente sin tener que modificarlas.

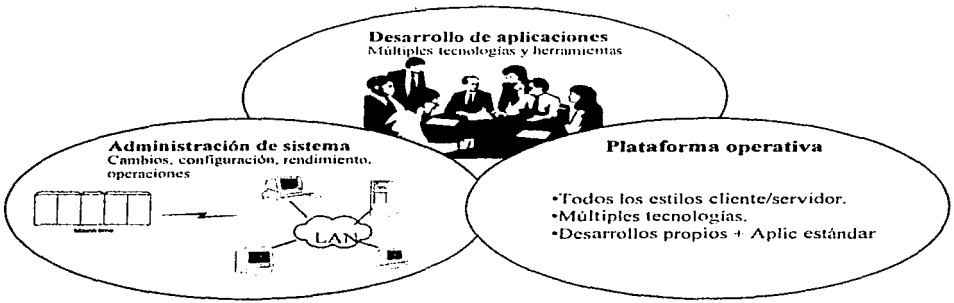


Figura 4.6 Componentes de una infraestructura cliente/servidor

La figura 4.6 muestra los componentes esenciales de una infraestructura cliente/servidor, todos ellos de igual importancia y estrechamente ligados, a continuación se explica cada uno de éstos.

Plataforma operativa. La plataforma deberá soportar todos los modelos de distribución cliente/servidor, así como la mayoría de los servicios de comunicación y utilizar preferentemente componentes estándares de la industria para los servicios de sistemas distribuidos. Los desarrollos propios deben coexistir con las aplicaciones estándares y su integración deberá ser imperceptible para el usuario, igualmente podrán integrarse programas escritos, utilizando diferentes tecnologías y herramientas.

Entorno de desarrollo de aplicaciones. Debe elegirse después de la plataforma operativa. Aunque es conveniente evitar una proliferación de herramientas de desarrollo, se garantizará que el enlace entre éstas y el middleware no sea excesivamente rígido. Será posible utilizar diferentes herramientas para desarrollar partes de una aplicación. Un entorno de aplicaciones incrementales debe posibilitar la coexistencia de procesos del cliente y servidor desarrollados con distintos lenguajes de programación y herramientas, así como utilizar distintas tecnologías (por ejemplo, lenguaje procedural, lenguaje orientado a objetos, multimedia).

Administración de sistemas. Estas funciones aumentan considerablemente el costo de una solución, pero no se pueden evitar. Siempre deben adaptarse a las necesidades de la organización y al decidir la plataforma operativa y el entorno de desarrollo, es decir, las primeras fases de la solución, es necesario considerar los siguientes aspectos:
¿Qué se requiere administrar?
¿Dónde estarán situados los procesadores y estaciones de trabajo?
¿Qué plataformas se soportarán?
¿Qué tipo de soporte es necesario y quién lo proporciona?

Cómo definir una infraestructura cliente/servidor

La plataforma operativa, el middleware y el entorno de desarrollo de aplicaciones están relacionados entre sí. Las necesidades de la apertura pueden condicionar la elección de la plataforma o del middleware, de igual manera que lo condiciona una determinada herramienta de desarrollo. El software de aplicación puede influir en la plataforma del sistema, y el tiempo disponible para la primera aplicación puede implicar algún tipo de compromiso. Por lo que es necesario establecer una metodología de infraestructura para sistemas distribuidos que permita definir una infraestructura para el ambiente cliente/servidor y evalúe la puesta en marcha del proyecto sobre una base racional.

Las actividades para la definición de la infraestructura son:

- Captación de las necesidades. Definir, analizar y evaluar, relacionando los requerimientos del negocio con las aportaciones de la tecnología.
- Diseño conceptual en el que se sitúen los principales bloques funcionales y de datos del sistema, mostrando su relación y comunicación entre éstos.
- Detalle de los principales componentes funcionales, selección de procesos, determinando los principios que deben aplicarse a la selección del software o del diseño de los módulos.
- Definición de la plataforma operativa, de los productos y componentes necesarios para la implementación.

Consideraciones sobre la implementación. Existen tres formas de abordar la implementación de soluciones cliente/servidor, éstas son:

- Construir la infraestructura y descuidar los aspectos del negocio a corto plazo (opción A).
- Concentrarse en soluciones del negocio a corto plazo e ignorar la infraestructura (opción B).

- Diseñar la infraestructura e implementarla incrementalmente a medida que se desarrollan las soluciones del negocio (opción C).

La primera opción puede acarrear dos problemas: la contratación de servicios a proveedores externos a la empresa, y la implantación incontrolada de soluciones por parte de los usuarios. Aunque puede ser una solución razonable a corto plazo, provocará a mediano y largo plazo caos e incompatibilidad entre sistemas y soluciones distintas, debido a la falta de una estructura corporativa cliente/servidor.

La opción B tiene como peligro que las plataformas, el middleware y las herramientas pueden cubrir las necesidades de las primeras aplicaciones abordadas, pero pueden no ser adecuados para las siguientes y, por lo tanto, resultar una mala inversión.

La opción C exige cierto trabajo inicial de diseño y arquitectura, pero garantizará una sólida infraestructura, a corto y mediano plazo, debido a que toma en cuenta las necesidades generales, aunque sólo se desarrollan los componentes a medida que se necesitan. Esta opción es obviamente la más adecuada, ya que combina las ventajas de las opciones A y B.

Proceso de planeación de sistemas de información

Necesidades de un nuevo proceso. Hay que revisar y adaptar el proceso de planeación informática. El rápido cambio de las tecnologías acorta el ciclo de vida de los productos. Ya no son posibles los ciclos de planificación de dos o tres años. El tiempo transcurrido desde la concepción de un nuevo producto o servicio hasta su puesta en el mercado se acorta cada vez más. El departamento de sistemas de información tiene que adaptarse rápidamente a los cambios en la tecnología. La infraestructura informática y el proceso de desarrollo tendrán que posibilitar la rápida introducción de nuevas aplicaciones, debido a que en caso contrario la empresa corre el riesgo de perder mercado.

Cartera de aplicaciones. Deben examinarse detenidamente las aplicaciones en cartera. Es muy probable que las necesidades formuladas hace uno o dos años ya no sean válidas. Hay que establecer nuevas prioridades con base en los objetivos de negocio de la empresa. El departamento de sistemas de información seguirá teniendo la responsabilidad del desarrollo de aplicaciones corporativas, así como la administración de los datos de la empresa. Sin embargo, una infraestructura cliente/servidor definida adecuadamente permitirá utilizar aplicaciones estándares seleccionadas por el departamento de informática o por los departamentos de usuario.

Aplicaciones tradicionales. Los sistemas y aplicaciones existentes juegan un papel importante en el procesos de migración hacia sistemas distribuidos, ya que frecuentemente no pueden sustituirse de modo inmediato y rentable, teniendo que coexistir con las nuevas aplicaciones cliente/servidor. Las consideraciones sobre migración e integración son responsabilidad del departamento de sistemas de la compañía.

Definición de responsabilidades

Propiedad. Es necesaria una clara definición de la "línea de confianza" que incluya los propietarios de las aplicaciones y datos corporativos, departamentales y de usuarios. Esta definición tiene una gran trascendencia debido a que regula hasta qué punto los departamentos y usuarios pueden agregar sus propias aplicaciones y datos. Esto afecta a la definición de configuraciones (procesador, memoria, almacenamiento), y a la forma en que las estaciones de trabajo y servidores están configurados, incluyendo criterios de disponibilidad y tiempo de respuesta. No es razonable traspasar al departamento de sistemas la responsabilidad de garantizar la disponibilidad y tiempo de respuesta, mientras se disponga de una total libertad para ejecutar las aplicaciones en las estaciones de trabajo y servidores.

Adquisición de hardware, software y aplicaciones estándar. Es razonable suponer que el departamento de sistemas defina las configuraciones estándares para los distintos tipos de estaciones de trabajo y servidores, y adquiera el software necesario. En grandes instalaciones, es ventajoso incluir los paquetes de productividad personal en el proceso de prueba, distribución y puesta en explotación.

Aunque es deseable cierto grado de libertad de elección por parte de los usuarios, es casi inevitable una normatividad sobre los estándares corporativos que definan las plataformas que se soportarán.

Evolución en el desarrollo de aplicaciones

Participación del usuario. En el pasado, el departamento de sistemas de información desarrollaba aplicaciones de acuerdo con las especificaciones aportadas por los usuarios. TÍpicamente, ambos grupos sólo se reunían de tarde en tarde, una vez fijada las especificaciones, y sólo para discutir sobre quién tenía la responsabilidad de los retrasos, del mal funcionamiento, etc. No es extraño que los profesionales de hoy en día sean escépticos sobre el grado de interés y la productividad de las organizaciones de desarrollo de aplicaciones que dependen de sistemas de información. Dentro de la tendencia de una creciente participación del usuario en el desarrollo de nuevas aplicaciones es fundamental que los usuarios y el departamento de sistemas de información puedan colaborar más estrechamente, de forma que las aplicaciones puedan ser construidas por equipos que trabajen con objetivos comunes de eficiencia, rapidez, rendimiento y calidad. Normalmente estos equipos estarán integrados por los propios usuarios y por personas del departamento informático. El papel del departamento de información será ahora el de definir y mantener una estructura corporativa (incluyendo la evaluación y selección de herramientas de desarrollo y aplicaciones) y proporcionar un conjunto de conocimientos especializados para dar servicios a los usuarios.

Prototipos. Como se ha explicado, en el proceso en espiral o iterativo, la generación de prototipos se ha convertido en la nueva forma de desarrollo de aplicaciones. Los departamentos de desarrollo trabajan directamente con los usuarios, que son capaces de verificar inmediatamente la funcionalidad y el aspecto de las aplicaciones. Pero los prototipos van mas allá de la construcción de la interfaz gráfica de usuario. En un entorno de sistemas distribuidos suele ser difícil evaluar las implicaciones sobre la distribución de datos y funciones. Es mucho más fácil construir un esqueleto de sistema durante las primeras etapas que refleje la distribución, pero que no contenga todo el código de aplicación. Las funciones y datos de la aplicación, que todavía no se han codificado ni probado, simplemente se simulan.

Aspectos organizativos

La introducción de nuevos métodos, herramientas y procesos de desarrollo conducen a nuevas profesiones especializadas en:

- Arquitectos de sistemas cuya responsabilidad es desarrollar una arquitectura de aplicaciones corporativa y establecer principios de diseño.
- Profesionales de desarrollo que se ocupan del diseño e implantación de la lógica de presentación, basándose en interfaces gráficas de usuario, para garantizar la consistencia, facilidad de uso y rendimiento.
- Especialistas en la definición e implantación de servicios de aplicación para garantizar que tienen el nivel adecuado de granularidad y se adaptan a la necesidad de ser compartidos por distintos procesos cliente.
- Si se utiliza tecnología de orientación a objetos, se necesitan expertos en la definición y desarrollo de aplicaciones u objetos de negocio.

El desarrollo de aplicaciones lo realizan equipos compuestos por personal de desarrollo y usuarios finales que ahora comparten la responsabilidad con el departamento de sistemas de la empresa. Junto a esta estrecha cooperación, es necesaria una estructura de dirección que permita tomar decisiones de forma compartida.

El entorno de aplicaciones incremental genera la necesidad de nuevos procedimientos de prueba, debido a que cada componente tiene ahora que probarse aisladamente y como parte de la aplicación; asimismo, el empaquetado del software de aplicación para su distribución tiene que ser incluido en el proceso de prueba y distribución.

CONCLUSIONES

Con objeto de mantener su competitividad, una empresa debe realizar cambios en sus acciones, clientes, diferenciación de productos, lealtad, precio, etc. Son ventajas transitorias que se encuentran en constante cambio. La verdadera ventaja radica en la habilidad de adaptación, actuación y reacción en el momento que es requerido, por lo que la única ventaja real sobre los competidores es la capacidad de reaccionar, cambiar y adaptarse en una forma más rápida y eficiente que los contrincantes.

El negocio debe organizarse para lograr su objetivo y para alcanzarlo se definen prácticas, las cuales son automatizadas a través de aplicaciones. El ambiente de la empresa se encuentra en constante cambio ya que, para mantener su competitividad, debe adaptarse lo más prontamente posible a dichos cambios; por lo tanto las aplicaciones en las que se soportan sus prácticas deben ser construidas sobre una arquitectura de cómputo que permita la maniobrabilidad requerida por la empresa. La arquitectura cliente/servidor ofrece a la empresa la posibilidad de adaptación y crecimiento requeridos para mantener, o bien aumentar, su grado de competencia.

La implantación de la arquitectura cliente/servidor dentro de una empresa hace posible la construcción de aplicaciones que poseen las características de mantenibilidad, extensibilidad, escalabilidad, flexibilidad, etc., debido a que éstas heredan las características de la plataforma en la que se construye; dichas propiedades permiten al sistema de información adaptarse a las necesidades del negocio de acuerdo con los requerimientos. Además la arquitectura cliente/servidor resuelve, o por lo menos, reduce muchos de los actuales problemas de los sistemas de información. Con una clara estrategia de implantación es posible lograr un sistema que sea capaz de crecer con base en los requerimientos de procesamiento del negocio; pero, para lograr esto, es necesario contar con distintas técnicas, como son la programación orientada a objetos, CASE, metodologías de implantación y, en general, un conjunto de conocimientos necesarios para la puesta en operación de un ambiente de este tipo. Si esto no se logra, el sistema resultante no será lo que se esperaba y probablemente se tengan más problemas de los que se tenían anteriormente.

Podemos concluir:

- Las consecuencias del cambio de paradigma que supone cliente/servidor tienen un fuerte impacto en el cometido de la organización de sistemas de información.
- No hay soluciones estándares cliente/servidor; se debe utilizar un enfoque estructurado para definir una solución específica para la empresa.
- Se necesita una sólida infraestructura cliente/servidor, una buena arquitectura de aplicaciones y principios de diseño, así como estándares corporativos de tecnología informática para lograr el éxito esperado.
- La infraestructura cliente/servidor debe diseñarse sobre el papel e implementarse según sea necesario y lo requieran las aplicaciones.
- Será necesario involucrar a los usuarios en las etapas iniciales e invitarles a que tomen parte del proceso de desarrollo.
- La computación cliente/servidor debe ser entendida dentro del contexto de la planeación estratégica de la empresa.
- **La arquitectura de cómputo en la cual se soporta la empresa define su grado de competitividad.**

GLOSARIO

ANSI — *American National Standards Institute.* Organización dedicada al establecimiento voluntario de estándares de cómputo.

API — *Application Program Interface.*

APPC — *Advanced Program to Program Communications.*

ASCII — *American National Standard Code for Information Interchange.* Código desarrollado por ANSI para el intercambio de la información entre sistemas de procesamiento de datos y comunicación. Un carácter ASCII consiste de un conjunto de siete bits más un bit de paridad.

Base de datos — Conjunto de datos organizados y relaciones entre sí.

Batch — Proceso que es ejecutado sin la necesidad de la interacción con el usuario, los resultados como los datos de entrada necesarios para la ejecución del proceso se almacenan en archivos. Para la ejecución de los procesos en batch es necesario que el sistema operativo cuente con el manejo de colas de batch y control de prioridades.

Commit — Cuando una transacción es aceptada, es decir, termina sin errores, es sólo entonces cuando se puede reflejar los cambios realizados hacia los datos en disco, a este proceso de aceptación se le conoce como proceso de COMMIT.

Constraint — Las reglas del negocio imponen restricciones (*constraints*) que los datos deben cumplir; estas restricciones pueden ser implementadas a nivel del manejador de base de datos.

DBMS — *DataBase Management System* (manejador de base de datos). Software que controla y administra los datos, con objeto de controlar la redundancia de datos y garantizar, entre otras características, la integridad, consistencia y disponibilidad de la información.

DCE — *Distributed Computing Environment.* Ambiente estándar desarrollado por la *Open Software Foundation* (OSF), el cual permite la interoperabilidad y portabilidad a través de un sistema distribuido heterogéneo.

DDL — *Data Definition Language.* Comando de SQL que permite la definición de los datos, es decir, creación de tablas y sus correspondientes modificaciones (*create table, alter table*).

Diccionario de datos — Descripción de los elementos de una base de datos y cómo están estructurados.

DML — *Data Manipulation Language.* Comandos de SQL que permiten manipular los datos (*select, update, delete e insert*).

Downsizing — Es la actividad de "bajar" las aplicaciones que se encuentran en los mainframes hacia redes de área local o extendida, la idea consiste en que una red de computadoras resuelva un proceso en vez de un solo procesador.

E-mail	Correo electrónico.
Estándares	Especificaciones técnicas u otros criterios precisos para ser utilizados por los integrantes de una organización, tales como normas, guías o definiciones de características, para asegurar que un sistema cumple con el propósito para el que fue creado. Los estándares internacionales contribuyen a simplificar el desarrollo de sistemas e incrementan su integridad y efectividad.
Estudio de factibilidad tecnológica y económica.	Documento formal para evaluar la problemática que gira en torno a un sistema. Su objetivo es presentar alternativas de solución para la elaboración de un sistema, que justifique el aspecto económico con base en una buena relación costo beneficio.
Host	Computadora que, por lo general, posee características que permiten ejecutar las aplicaciones en un menor periodo de tiempo.
HUB	Punto de conexión en una red, donde numerosos circuitos son conectados.
LAN	*Local Area Network.* Red de área local.
LU 6.2	*Logic Unit 6.2.* Protocolo peer-to-peer que permite que los nodos de una red se comuniquen directamente, no existe una relación primario-secundario, los nodos se comportan como iguales, algunas veces es referenciado como APPC (*Advanced Program to Program Communication*).
MAU	*Multistation Acces Unit.* Hub central utilizado en una LAN tipo *token ring.*
Multithread	Modo de procesamiento en el cual un proceso es dividido en subprocesos, los cuales son ejecutados independientemente uno del otro.
Overhead	Sobrecarga del sistema.
Paradigma	Modelo o patrón usado para el desarrollo de software o diseño de hardware.
Peer-to-peer communications	Método de comunicación en el cual cualquier parte puede iniciar la sesión. Ningún dispositivo tiene que jugar el rol de estación primaria.
POSIX	*Portable Operating System Interface* para el ambiente UNIX. Especificación generada por la IEEE, la cual define el lenguaje de interfaz entre las aplicaciones y el sistema operativo UNIX.
RDBMS	*Relational Database Management System.*
Road Map	Vista global de la arquitectura cliente/servidor, ofrece una vista conceptual de la arquitectura, mostrando posibles tecnologías que pueden ser utilizadas para cada uno de los elementos que la forman.
Rollback	Cuando una transacción falla, es necesario regresar los datos modificados a su estado original, a este proceso se le conoce como ROLLBACK.
SMTP	*Simple Mail Transfer Protocol.* Protocolo de E-Mail usado en redes TCP/IP.

Sistemas abiertos	Es un sistema de información basado en estándares, lo cual permite su interoperabilidad con distintos proveedores, en ocasiones es referido como un ambiente multivendedor.
Stored procedure	Conjunto de comandos de SQL que se encuentran precompilados y almacenados en la base de datos, se referencia a través de un nombre. Se ejecuta cuando es referenciado por una aplicación o bien directamente.
Tabla	Un RDBMS almacena la información en la forma de columnas y renglones; al conjunto de columnas y renglones se le denomina tabla.
Trigger	Es semejante al *stored procedure*, su diferencia radica en que se asocia a una tabla y a un evento; el evento puede ser de borrado, insertado o actualización de la información.
WAN	*Wide Area Network.* Red de área amplia.
Windows NT	*Windows New Technology*, de Microsoft Corp.

BIBLIOGRAFÍA

John E.McNamara. *Local Area Networks (An introduction to the technology).* Digital Press

Bernard H.Boar. *Implementing Client/Server Computing.* Mc Graw Hill

W.H. Inmon. *Developing Client/Server Applications.* Wiley-QED

Barbara Bochenski. *implementing production-quality client/server systems.* Wiley

Alex Berson. *Client/Server Architecture.* McGraw-Hill Series on Computer Communications.

Joe Salemi. *Guide to Client/Server Databases.* PCMagazine press

Guide to Building Client/Server Solutions . DIGITAL press

Client/Server Basis. Centro Europeo de Soluciones Cliente/Servidor de IBM. COMPUTERWORLD

Yourdon, E . *Análisis Estructurado Moderno.* PRENTICE HALL.

Client/Server Architecture. Instructor GUIDE. Sybase

Computerworld CLIENT/SERVER JOURNAL.

Michael C. R. *The Data Modeling Handbook. A best proactice approach to building quality data models.* Wiley

Thomas S. Ligon. A Guide for the Applications Developer. McGraw Hill Series on Client/Server Computing.