Estudio y Evaluacion del Modelo Cliente-Servidor en la Migracion de Redes y Aplicaciones en las Empresas, Dentro del Marco Definido por el Concepto Middleware.

Tesis

Maestria en Administracion de Sistemas de Informacion

Instituto Tecnologico y de Estudios, Superiores de Monterrey,
Campus Guadalajara

Por

Juan Roberto Mendez Aranda

Junio de 1999

Estudio y evaluación del Modelo Cliente/Servidor en la migración de redes y aplicaciones en las empresas, dentro del marco definido por el concepto Middleware.

Tesis
Maestría en Administración de Sistemas de Información.

Instituto Tecnológico y de Estudios Superiores de Monterrey,
Campus Guadalajara.

Por

Juan Roberto Méndez Aranda

Junio de 1999.

ESTUDIO Y EVALUACION DEL MODELO CLIENTE/SERVIDOR EN LA MIGRACION DE REDES Y APLICACIONES EN LAS EMPRESAS, DENTRO DEL MARCO DEFINIDO POR EL CONCEPTO MIDDLEWARE

POR

JUAN ROBERTO MENDEZ ARANDA

TESIS

Presentada a la División de Graduados e Investigación

Este Trabajo es Requisito Parcial para Obtener el Grado de Maestro en Administración de Sistemas de Información

INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

JUNIO DE 1999

DEDICATORIA:

A mis padres.

Porque gracias a su ejemplo y apoyo, que me acompañarán durante toda la vida, ahora es posible la culminación de esta importante meta.

A mi hermana, a Carlitos y a Lalito.

Que me motivan a seguir adelante.

RECONOCIMIENTOS:

Al DR. J. LEONARDO SOTO SUMUANO: Por su tiempo, su cuidadosa asesoría y su amistad.

Al DR. CARLOS ISLAS: Por sus consejos para la realización de esta tesis.

A LIZ Y MOY: Por brindarme su amistad durante este importante periodo de mi vida.

A MIS COMPAÑEROS, Sandra, Luis, Alberto, Fernando, Pablo, Javier y Adriana, por su amistad y ayuda.

A LOS BOLILLEROS: Roberto, Eduardo, Toño, Omar, Moy, Rufino, Manolo, Daniel, Sergio y Luis, por el recuerdo de los viejos tiempos y deseandoles la mejor de las suertes, donde quiera que se encuentren.

A SERGIO1, SERGIO2 Y RUBEN: Por sus animos para seguir hasta el final.

A LUPITA: Quien me ayudo por tanto tiempo.

AL PERSONAL DEL SEIS: Juan Jesús, Miguel, Angélica, Toño y Lupita, Gracias por todo.

A MONICA: Por conseguir mis libros.

A ALBERTO Y OSCAR: Por su apoyo en los laboratorios.

A MARTHA ELENA: Por su compañía, su apoyo y los animos que me dió para terminar esta Tesis.

RESUMEN

El trabajo de investigación realizado explica la ubicación del middleware dentro de la estructura de Tecnología de Información, el futuro de los negocios en la era de la información, es el manejo de los datos, de una manera confiable, transparente y portable de las aplicaciones.

Middleware es la base para lograr que la relación proveedores, industria y clientes sea productiva, logrando plena satisfacción de los clientes, habilidad de adaptarse a las necesidades y requerimientos de los mismos y respondiendo de manera inmediata.

El problema de comunicación entre bases de datos heterogéneas, diferentes tipos de redes, el potencial del uso de internet, nos lleva a la necesidad de recurrir a middleware para lograr que estas trabajen de manera conjunta. Sin embargo, el funcionamiento de middleware, sus raíces, sus características, no estaban claras, en esta tesis todos estos puntos quedan esclarecidos.

La investigación comienza con una explicación de los fundamentos Cliente/Servidor, pues esta arquitectura forma la base del middleware, para después empezar a esclarecer como trabaja middleware, que es el software entre un programa de aplicación distribuida y el entorno computacional en el que se encuentra, incluyendo el sistema operativo y las capacidades de red.

Cliente/Servidor, los desarrolladores de aplicaciones distribuidas e integradores de sistemas usan middleware para crear sistemas de información distribuida mas rápida y confiablemente.

Existen diferentes maneras en las que middleware puede ser definido y limitado, middleware debe ser de propósito general y soportar cliente/Servidor y aplicaciones distribuidas en una variedad de aplicaciones de diferentes dominios, tener interfaces que estén basadas en estándares o tener especificaciones disponibles de manera pública y tener implementaciones en al menos una plataforma de cada proveedor para lograr comunicación entre estos.

En esta investigación se encuentra la descripción de los diferentes tipos de middleware, beneficios, interfaces, background general, estándares y servicios middleware, esclareciendo estos puntos, se evaluó la ubicación del desarrollo de middleware en las empresas mexicanas y sus futuras aplicaciones.

INDICE.

Lista de Tablas

Lista de Figuras

INTRODUCCION.

En los últimos años la globalización que se lleva a cabo en el mundo de los negocios obliga a empresas a realizar cambios en su modelo administrativo, el cual esta siendo impactado por elementos externos (aperturas, alianzas estratégicas, cambios constantes en el mercado, y encuentra una solución a sus problemas en un esquema técnico (modelo tecnológico o informático). Los elementos externos piden mas productividad, piden distribución de la información, con el objetivo de eliminar distancias y tiempo de acceso, esto presupone la necesidad de un sistema administrativo nuevo, donde los sistemas (de información tanto desde el aspecto técnico como de planeación) son vitales, a partir de esta tendencia ha nacido la idea de Cliente-Servidor como el modelo tecnológico que apoya a las funciones administrativas y propone una solución para enfrentar esta transición.

La necesidad de un nuevo enfoque se plantea por competitividad, los modelos administrativos y tecnológicos de otros paises no son aplicables a nuestro país, de ahí la necesidad de estudiar como las empresas mexicanas medianas y grandes están ubicadas dentro de este modelo, como se enfocaria su migración y que posibilidades existen de éxito, el modelo tecnológico o informático impacta al modelo administrativo y este impacta al modelo productivo.

Enfrentando la creciente competencia y los rápidos cambios en el mercado actual, las organizaciones buscan reordenarse para sobrevivir y competir. La Tecnología de Información (TI), y en particular el paradigma Cliente/Servidor, son parte integral de este proceso de reordenamiento. Cliente/Servidor no es solo otro producto; este permite usar la tecnología para soportar la forma en que trabajan los negocios, por este motivo Cliente/Servidor se ha convertido en el principal enfoque de la Tecnología de Información de los 90's.

La realidad hoy es que gran numero de compañías están operando con una mezcla de sistemas viejos y nuevos, muchos de los cuales dan valor agregado al trabajo, pero no se pueden comunicar fácilmente unos con otros.

Ahora los negocios están usando la "Cuarta ola" de la computación- Cliente/Servidor para lograr que estos sistemas trabajen juntos y lograr que toda la gente que necesita información dentro de una compañía logre tenerla disponible.

Alrededor del mundo, los negocios están sufriendo una dramática transformación. La corta vida de los productos y la fragmentación de los negocios están disminuyendo las ganancias. La globalización de los negocios esta demandando rápidos flujos de información. Los clientes esperan constantes mejoras en la calidad, tiempos de respuesta y servicio. Los negocios se están apoyando en la tecnología para manejar su mas importante capital: Los datos corporativos.

La forma de clasificar el impacto de la tecnología en los negocios desde los 60's hasta la actualidad es a través de cuatro "olas".

La Primera Ola.

En los 60's el uso de mainframes soportaba los negocios que se caracterizaban por su centralización, estos sistemas proporcionaron beneficios, pero ahora, estos tienen que conectarse a sistemas de redes de computadoras personales.

La Segunda Ola.

Para los 70's, la innovación y las economías de escala hicieron de las mini-computadoras una opción financieramente atractiva, estas fueron adquiridas por compañías para tareas especializadas como planeación de la producción, control de procesos y otros trabajos "departamentales", esto creo islas de información inaccesibles por el resto de la compañía.

La Tercera Ola.

En los 80's la característica principal fue la demanda de computadoras poderosas para elevar la productividad personal, el resultado fue el uso de las PC's y pronto las compañías las empezaron a ligar por medio de Redes de Area Local (LAN's), por sus siglas en ingles, esto habilito a grupos de usuarios a comunicarse unos con otros y compartir recursos. Las LAN fueron el primer paso de Cliente/Servidor.

La Cuarta Ola.

En los 90's muchas compañías están operando en uno o mas de estas primeras fases evolutivas, pero estas necesitan integrarlas dentro de la estructura que soporta la forma en que se trabaja. Se necesita fácil acceso a la información a través de toda la compañía y existe la necesidad de ligar esta de manera electrónica.

Esta es la "Cuarta Ola": Ligar varios sistemas y equipos en una red (incluyendo todos los distintos estándares para operar y comunicar), un gran reto que es el concepto central de la computación Cliente/Servidor.

Existen algunas reglas generales para empezar a adentrarse en Cliente/Servidor. Renovar una aplicación existente, por instancia, puede ser una opción bastante atractiva por un lado pero tal vez no factible en Costo-Beneficio, por el otro. Usualmente una nueva aplicación basada en una reingenieria de procesos se puede esperar de mas alto beneficio que simplemente migrar una aplicación existente.

Migrando las aplicaciones: El hacer el menor cambio en las aplicaciones y re-explotarlas a través de una estructura Cliente/Servidor, puede llevar a las compañías a ganar enormes beneficios de nuevas capacidades disponibles pare ellos en sus estaciones de trabajo.

Los beneficios de los sistemas abiertos son tales como:

Portabilidad: Aplicaciones, datos y usuarios pueden ser movidos fácilmente de un sistema computacional a otro.

Interoperabilidad: Los usuarios pueden trabajar con aplicaciones y datos los cuales están distribuidos a través de computadoras las cuales están conectadas pero pueden tener diferentes arquitecturas.

Escalabilidad: El sistema puede ser expandido al tamaño que la compañía lo requiera.

Como resultado, la organización de Sistemas de Información tiene que hacer frente a tareas que anteriormente caían dentro del ámbito de los proveedores. La problemática que esto implica es:

- Soporte de la gestión empresarial: El departamento de Sistemas de Información debe estudiar el modelo Cliente/Servidor y la forma de implantarlo eficazmente. Ellos poseen generalmente, los conocimientos necesarios para apoyar la gestión de la empresa, formulando objetivos de negocio y proponiendo Tecnologías de Información que los soporten.

- Selección de estándares y tecnologías de información: Como los estándares de sistemas abiertos no están solidamente establecidos, la responsabilidad de su elección también se ha desplazado desde 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 infraestructuras Cliente/Servidor que incluyan:

  -Plataformas Operativas.
  -El entorno de desarrollo de aplicaciones.
  -La gestión del sistema distribuido.

- Definición de arquitecturas de aplicación: Establecer los principios de diseño que garanticen aplicaciones portables, interoperables y distribuidas.

- Desarrollo de aplicaciones corporativas: La responsabilidad de disponer de aplicaciones de ámbito general, para que las utilicen múltiples usuarios de diversos departamentos, seguirá siendo del departamento de Sistemas. Una infraestructura Cliente/Servidor adecuada permitirá la integración de estas aplicaciones con las originadas por los usuarios.

- Integración de aplicaciones: La integración de aplicaciones desarrolladas en la empresa con aplicaciones estandard es fundamental. Muchas de las aplicaciones estandard implican sus propias normas de instalación, parametrización y soporte de Middleware Cliente/Servidor.

Como parte da la problemática tenemos que desde el punto de vista de datos las compañías prefieren que estos se encuentren centralizados para facilitar su administración y mantenimiento, sin embargo, también es deseable que puedan ser accesados desde cualquier situación geográfica y de manera transparente (descentralizados), esta situación provoca un gran reto que necesita ser solucionado por la parte técnica que apoya los procesos del negocio.

Como un aspecto para mejorar lo actual y dar solución a estos problemas, se propone MIDDLEWARE, esta palabra implica software y hardware que es aplicado tanto a nivel de comunicaciones y redes, como para aplicaciones y compartición de bases de datos. Trata de ser una solución para poder comunicar cualquier equipo (AS-400, Lotus Notes, HP-9000, etc.), de forma transparente para los usuarios, proporcionando una base fácil de mantener y administrar desde el punto de vista sistemas, lo cual hasta ahora sigue siendo un tabú, middleware es un concepto muy nuevo y proporciona un tema de estudio interesante.

Se va a investigar este tema, ya que como se dijo anteriormente es un concepto bastante nuevo, todavía no hay nada escrito formalmente acerca de este, y crea confusión aun entre personas expertas de México que tienen que enfrentarse a esta tendencia tecnológica, un estudio amplio del mismo puede proporcionar orientación, aclarar el panorama y ubicar a personas de alta gerencia para formarse un criterio de planificación de sus telecomunicaciones en los negocios, elemento clave para destacar en el Siglo 21.

La aplicación mas importante de la tecnología del computo en los 90's sera la que ayude a las personas a trabajar mejor en conjunto (PC Magazine, 1995), dentro de este entorno, el área de comunicaciones es una de las mas grandes en México y posee una de las tasas de mayor crecimiento alrededor del mundo, (Gustavo Guerrero, Revista RED ejecutivos, marzo 1994). Ademas el potencial de la mejor maquina que pueda existir en el mundo no sirve para nada si esta no puede compartir recursos (discos duros, CD Roms, impresoras, escáneres) , comunicarse con otra, mejorar la calidad del trabajo. (PC Computing México, abril 1995).

Sin duda, nos encontramos en la era de la Información, comparable con la Revolución Industrial por su impacto en los negocios; si bien ahora este enfoque es a los procesos de negocios, la divulgación de la tecnología es tan alta y los productos pueden ser tan iguales que su única diferenciación hacia otro competidor puede ser solo el servicio, (Ing. Ricardo Rendón, 1994). Para proporcionar un buen servicio se debe contar con una infraestructura de comunicaciones dentro de todas las empresas, para conocer lo que es obvio para todos, sin que por esto deba entenderse como fácil, el estado actual de la empresa, ¿como están mis ventas?, ¿como se comporta el mercado?, ¿cual es el nivel de

satisfacción de mis clientes hacia mi?. Para lograr sobrevivir en la era de la información se debe obtener información confiable tanto de sensores internos de la empresa, como externos, (Peter Drucker, 1993) "La empresa que sobreviva y ademas sobresalga en el futuro no sera la mas grande sino la que mas información tenga y mejor la aproveche". Hasta aquí hemos hablado de los principios básicos de la administración.

¿Que tienen que ver las telecomunicaciones con el párrafo anterior?, en la actualidad la cantidad de proveedores de "soluciones", como IBM, Microsoft, Novell, etc., ofrecen en su publicidad soluciones, pero que dice un empresario, ¿Como me vas a vender una solución, si ni siquiera yo mismo se cual es mi problema? (Raúl García, Jefe de informática de laboratorios PISA en Guadalajara). Es abrumadora la cantidad de plataformas, productos integradores de redes, etc. que se encuentran en el mercado hoy día, así mismo, el monopolio mexicano de comunicaciones (TELMEX), en cuestión de tres años ha implementado una cantidad de servicios de telefonía, como son RDI, ATM, protocolos como X.25, etc. los cuales nos sirven para transmisión de voz, datos y video (este ultimo de manera prehistórica, comparado con los paises del primer mundo), debido al ancho de banda disponible. Pero bueno, estas cuestiones técnicas ya serán materia de estudio mas delante, a lo que me refiero es que muchas veces tenemos problemas para seleccionar nuestro equipo de redes, ¿lineas privadas o publicas?, ¿fibra óptica?...

La cultura de nuestro país se mueve mas cada día hacia las aplicaciones completas, el mercado comenzara a demandar el uso de redes para mejorar la administración, "La empresa que no maneje información y telecomunicaciones esta trazando su camino al fracaso", (Dr. David Alanís, Octubre 1995), para consentir a nuestros clientes, para ganar ventajas competitivas, tendencias de Internetworking, Groupware, comercio electrónico, popularización de Internet, facilidad del correo electrónico, multimedia, los cada día mas famosos agentes inteligentes, esto nos lleva a lo mismo, LA IMPORTANCIA DE LA INFRAESTRUCTURA DE REDES.

Terry Blevins, Jefe del departamento de Arquitectura y Estandares de AT&T en Ohio y Gregory Steele del Departamento de Interoperabilidad de AT&T (1995), proponen que Middleware es el nombre que corre entre un programa de aplicación y recursos del sistema de bajo nivel que una aplicación requiere para desempeñar una función, sostienen que Middleware puede ser diseñado para accesar información y otros recursos para una aplicación aislandola de los detalles de recuperación y transmisión de datos, entonces esto haría que una aplicación fuese fácil de correr y de reformatear datos que distintas aplicaciones necesiten aislandolas de inevitables cambios en el entorno computacional y haciendo estas mas estables cuando los cambios ocurran.

Actualmente muchas aplicaciones distribuidas contienen propiedades de Middleware para intercambio convencional de datos solamente entre elementos de la misma aplicación. Sin embargo, las propiedades de Middleware de diferentes proveedores es incompatible, una situación que genera problemas de integración, flexibilidad, escalabilidad, reutilización, confiabilidad y desempeño. Dennis DeBrosse (1995), afirma que Middleware llega a ser un

estandard que podrá volver este problema menos complejo, esto podrá hacer mas fácil la implementación de nuevas aplicaciones integrando el software y protegiendo al sistema de cambios.

John W. Verity (1995), da una visión del futuro de Middleware diciendo que los estándares futuros deben soportar integración de nuevas tecnologías tales como voz, video, audio e imágenes en las aplicaciones, sin provocar disturbios en los sistemas que ya existen en operación, Joanie Wexler (1995), dice que Middleware es una criatura nacida del caos, ya que las corporaciones tienen redes de todos los tamaños con ramificaciones de todas las marcas de proveedores, entonces Middleware viene a ser mas crucial ahora para poder interactuar con quien sea en donde sea y con el ambiente que sea de manera transparente. "Middleware puede traducir datos de diferentes plataformas de diferentes proveedores en formatos estándares para todos", Jim Duffy (1995).

La emergencia de sistemas Middleware para sistemas distribuidores vista como una tecnología en expansión y empowering, el World Wide Web (WWW) (Berners-Lee, 1994), pueden capitalizar esta proveyendo acceso a sistemas distribuidos. Por medio del WWW se integrarian servicios particularmente a OSF's DCE (Distributed Computing Environment) [HREF 1, 1993] y OMG's CORBA (Common Object Request Broker Architecture) [HREF 2, OMG, 1993]. WWW puede proveer los siguientes beneficios:

- Acceso a un gran rango de servicios.
- Acceso a servicios de Middleware sin la necesidad de instalar y administrar el software asociado.

La integración de WWW y middleware puede capacitarnos a recibir servicios de dos formas:

- Clientes de WWW pueden invocar servicios de middleware.
- Clientes de Middleware pueden solicitar servicios de WWW.

Existen personas que no están muy de acuerdo con Middleware y lo llaman Muddleware (Jonathan Adams, IBM UK, 1995) y de aquí nace una controversia algo fuerte que también trataremos a profundidad en el desarrollo de la tesis.

Pero ¿Que es Middleware?, este termino ha sido sujeto de abuso por parte de los diferentes proveedores del mismo, que han manipulado sus características resultando en una confusión en el Mercado. De aquí la referencia a Muddleware.

Middleware ha sido definido de diversas maneras, entre ellas, como una herramienta que habilitala interoperabilidad y su propósito es de conectar aplicaciones, las cuales se puede esperar que sean remotas una de otra. Esta puede hacerse con muchos métodos

diferentes y distintos grados de funcionalidad, lo que no es nuevo; el cambio esta en las demandas a este Middleware.

Las dos "olas" de la computación Cliente-Servidor.

Como el mundo de la tecnología de información se ha vuelto mas complejo y heterogéneo la necesidad de soluciones middleware para facilitar interoperabilidad de los sistemas y portabilidad de las aplicaciones se ha incrementado de manera importante. Estaremos envueltos en un mundo donde la simplicidad en los servicios de conexión deben de ser suficientes; Debe existir transparencia en las conexiones, facilidad de administración, portabilidad de funciones, desarrollo rápido y rápida explotación de funciones distribuidas, para proporcionar a las aplicaciones capacidades para soportar los negocios.

Algunas aplicaciones de Middleware soportan en cierta forma entornos de ejecución de aplicaciones distribuidas. Es deseable que las aplicaciones, datos y código estuvieran todas en un solo lugar, también esperamos que las aplicaciones construidas puedan correr en diversas plataformas, lo que implica el termino (re-arreageable component aplications). Tipicamente esta "Primera ola" de la computación Cliente/servidor usa middleware para soportar la distribución de funciones a través de Middleware API, esto implica que la aplicación por si misma es monolítica y que es distribuida por medio de servicios tales como, impresoras, servicios de datos, etc. Tal método tiene claras limitaciones futuras, pero también tiene algunos puntos a su favor. Por ejemplo, esto implica que la administración de sistemas distribuidos sea controlada de una forma razonable.

La Primera Ola de la computación Cliente/Servidor a estado mas enfocada a soportar la Reingenieria de Estructura de los negocios, es decir, en soportar a la organización de los 90's (Empowerment, aumentar la productividad personal, etc.).

La segunda Ola de la computación Cliente/servidor se enfoco a Reingenieria de Procesos (Downsizing, aplicaciones de misión crítica y el upsizing de compartir aplicaciones LAN's para proporcionar una nueva manera de trabajar a las compañías y cumplir con las demandas que les implica su negocio. Esta segunda Ola es la que maneja grandes demandas de Middleware como soporte para funciones distribuidas.

Los upsizers esperan que las aplicaciones y códigos sean interoperacionales y provean integración. Esta tendencia se ha llamado recientemente Objetos Distribuidos, donde, funciones y datos son distribuidos en un entorno de "objetos cooperativos".

Consecuentemente Middleware ofrece algunas "soluciones" que pueden ser de tres diferentes categorías:

Allow you to connect...

Esta categoría puede proveer al usuario con servicios de conexión, pero este debe de ser requerido (o pedido) por el usuario. Este tipo de Middleware esta caracterizado por ofrecer un significante reto a los programadores de aplicaciones, a menos que un poderoso lenguaje de 4GL sea usado los programas pueden ser muy lentos de desarrollar.

Connect you...

Esta clasificación de Middleware puede proporcionar servicios de conexión transparentes. Estas aplicaciones hacen llamadas usando el lenguaje nativo de Middleware API's y middleware satisfacen el requerimiento ya sea de manera local o remota. Este tipo de middleware requiere que los empleados sean entrenados para el entorno especifico de ejecución, requerimiento extra de habilidad puede ser un inhibidor para usar este tipo de Middleware.

Protect you...

Este es un tipo de middleware altamente funcional. Hace mas que solo proporcionar interoperabilidad, también aísla la aplicación del usuario del entorno (hardware y sistema operativo) en el cual se esta corriendo. Esto significa que puede proveer un alto grado de portabilidad y evitar dependencias del entorno. La portabilidad es limitada para plataformas en las cuales el middleware esta disponible, esto es sabido por todos, pero este tipo de middleware es muy simple de soportar y puede hacer que las aplicaciones sean fáciles de escribir resultando en gran productividad de las aplicaciones y de los programadores.

A partir de estas ideas se puede detectar la importancia de una guía clara de middleware. Contestar a las preguntas de ¿que es?, ¿para que sirve?, ¿como lo aplico?, lo que se presentara y es el tema central de esta tesis en la que ademas se sugeriran maneras de migración y de implementación de middleware.

A lo largo de este documento procederemos con el primer capítulo justificando el modelo Cliente/Servidor, costos y beneficios y sentando las bases de los términos que se usarán a lo largo del documento, durante el capítulo dos se describirán los elementos de la arquitectura Cliente/Servidor y las características de cada uno, así como las técinas de procesamiento cooperativo, en el tercer capítulo se describe middleware y sus características en los sistemas abiertos. En los capitulos cuatro y cinco se describen los problemas y beneficios middleware así como las alternativas disponibles, en el sexto y último capítulo se encontrarán los resultados de la investigación y la comparación entre estas alternativas.

CAPITULO I. JUSTIFICACIÓN DEL MODELO CLIENTE/SERVIDOR.

En este capítulo encontraremos las razones que impulsan a las compañías a evaluar el modelo Cliente/Servidor, los costos, beneficios y factores de éxito que estos generan, además de una rápida introducción a este modelo, con el fin de entender el concepto Middleware.

1.1 CLIENTE SERVIDOR

Definición de Computo Cliente/Servidor: La computación Cliente/Servidor se a visto impulsada por el desarrollo de las computadoras personales, las redes de área local (LANs), las redes de área amplia (WANs), la interoperabilidad de las comunicaciones, las interfaces gráficas de usuario y los sistemas de administración de base de datos, de esta manera logramos el crecimiento de las aplicaciones y de las redes, de una manera sencilla evitando problemas bastante complejos tanto para los programadores como para los administradores de red, logrando disminuir los tiempos de respuesta y aumentando la transparencia de los datos.

Existen varias razones que impulsan el crecimiento de las aplicaciones cliente/servidor:

1. La demanda de sistemas mas fáciles de usar, que contribuyan a una mayor productividad y calidad.
2. El precio/rendimiento de las estaciones de trabajo y los servidores.
3. La creciente necesidad de acceso a la información para tomar decisiones y de soportar los procesos mediante unas aplicaciones mas ajustadas a la estructura organizacional de la empresa, que permita realizar las operaciones de manera mas natural.
4. La utilización de nuevas tecnologías y herramientas de alta productividad mas aptas para la dinámica del mercado.

La organización de sistemas de información tiene que hacer frente a nuevas tareas que antes eran solo de la incumbencia para los proveedores tales como:

Soporte de gestión empresarial: El departamento de sistemas de información debe estudiar el modelo Cliente/servidor y la forma de implantarlo eficazmente, apoyando la gestión de la empresa.

Selección de estándares y tecnologías de información: Como los estándares de sistemas abiertos no están sólidamente establecidos, la responsabilidad de su selección también se ha desplazado desde 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 infraestructuras cliente/servidor que incluyan:

- Plataformas operativas.
- El entorno de desarrollo de aplicaciones.
- La gestión del sistema distribuido.
- Definición de arquitecturas de aplicación: Establecer los principios de diseño que garanticen aplicaciones portables, interoperables y distribuidas.

Desarrollo de aplicaciones corporativas: La responsabilidad de disponer de aplicaciones de ámbito general, para que las utilicen múltiples usuarios de diversos departamentos, seguirá siendo del departamento de sistemas de información, una infraestructura cliente/servidor adecuada permitirá la integración de estas aplicaciones con las originadas por los usuarios.

Integración de aplicaciones: La integración de aplicaciones desarrolladas en la empresa con aplicaciones estándar es fundamental, muchas de las aplicaciones estándar implican sus propias normas de instalación, parametrización y soporte de middleware cliente/servidor.

1.2 COSTOS Y BENEFICIOS DE CLIENTE/SERVIDOR.

1.2.1 COSTOS:

Educación.
Nuevas herramientas.
Nuevas metodologías.
Nuevas plataformas
Mayor complejidad en el entorno, la arquitectura y la gestión del sistema.

Salvo que se opte por una implementación muy a la ligera, descuidando aspectos claves, los costos iniciales de los sistemas distribuidos pueden ser substancialmente mas elevados de lo previsto, algunos de estos aspectos que no se deben olvidar 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.
- Arquitectura de aplicaciones y diseño para un entorno distribuido.
- Distribución de software/gestión de licencias.
- Nuevas guías de procedimientos operativos.
- Soporte técnico a los usuarios.

1.2.2 BENEFICIOS.

Mejores aplicaciones, escalabilidad y funcionalidad.
Usuarios satisfechos.
Racionalización de los costos informáticos.
Adaptación mas flexible a los procesos del negocio.

1.2.3 IMPLICACIONES DEL MODELO CLIENTE/SERVIDOR.

Este modelo implica 5 aspectos muy importantes para la empresa:

1) 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 unos procesos de negocio en cambio constante.

2) Nuevos roles de sistemas de información y de los usuarios.

El rápido desarrollo de aplicaciones exige que los usuarios participen activamente en el proceso de desarrollo. Los departamentos de los usuarios tienen responsabilidades en las funciones, procesos de negocio y facilidad de uso. La responsabilidad de sistemas de información se centra en proporcionar infraestructuras, aplicaciones integradas y definir estándares corporativos para las tecnologías.

3) 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 negocios.

4) Nuevas herramientas de desarrollo:

Las aplicaciones se desarrollan en base a "prototipos", construyendo nuevas funciones a partir de una combinación de componentes de desarrollo propio y aplicaciones estándar disponibles comercialmente.

5) Nuevo proceso de desarrollo:

Se necesita un nuevo proceso de desarrollo iterativo que acorte desarrollos basados en prototipos, en los que puedan participar los usuarios conjuntamente con los profesionales de sistemas de información.

1.3 APLICACIONES CLIENTE/SERVIDOR.

Una aplicación cliente/servidor se compone de varios procesos cliente y servidores que se pueden distribuir en una red. El middleware Cliente/Servidor conecta los procesos que componen la aplicación. Este termino ya muy extendido se utiliza para describir aquellas funciones que son necesarias para proporcionar transparencia de localización.

Es a través del middleware que estos procesos se conectan e interactúan constituyendo aplicaciones. El middleware agrupa una serie de funciones y servicios (directorios, seguridad, tiempos, etc.), 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: comunicaciones, directorios, integridad, etc.

Mientras que en el middleware define las plataformas y el nivel de transparencia de localización, las verdaderas ventajas de implantar Cliente/Servidor solo se pueden conseguir mediante un adecuado diseño del sistema y de la aplicación, y con una acertada elección de los elementos de desarrollo (herramientas, lenguajes). El objetivo es conseguir una total flexibilidad para, en caso necesario, distribuir las distintas partes de una aplicación entre varios procesadores sin que esta tenga que modificarse, para esto, una aplicación Cliente/Servidor debe disponer de los siguientes atributos:

Separación de funciones: Para poder distribuir partes de una aplicación entre los procesadores, su lógica debe dividirse en los siguientes módulos funcionales:

- Lógica de Presentación.
- Lógica de negocio.
- Lógica de datos.

Si se diseñan y se implantan de manera apropiada, los módulos de función pueden distribuirse y redistribuirse en distintos procesadores, y los flujos de trabajo pueden cambiarse en función de las necesidades del negocio. Además se pueden elegir herramientas especificas y/o tecnologías para implantar los distintos tipos de funciones. La elección del Middleware Cliente/Servidor determina a su vez, el nivel de portabilidad e interoperabilidad deseado.

Encapsulación de servicios: Con objeto de que el acceso a dichos módulos funcionales sea conocido y accesible por clientes y servidores, las funciones deben encapsularse, es decir, rodearse de una interface bien definida, accesible mediante mensajes o llamadas bien definidas.

Internamente, la función podrá cambiarse o mejorarse, incluso utilizando diferentes herramientas o lenguajes sin que los clientes tengan que ser notificados.

Un servidor por lo tanto, puede considerarse como un objeto, este objeto si se diseña correctamente puede ser compartido por muchos procesos de clientes distintos y diferentes aplicaciones (reusabilidad), los procesos servidores no están dedicados a procesadores físicos.

Portabilidad: Las diferentes "piezas" de la aplicación necesitan ser colocadas en diversos procesadores de la red, para que esto sea posible, es necesario que dichas piezas sean desarrolladas con atributos de portabilidad que permitan la ubicación en sistemas de diferentes arquitecturas y fabricantes con mínimos o nulos cambios. La perfecta solución cliente/servidor es independiente del hardware o de los sistemas operativos y debe permitir la combinación de diferentes plataformas.

Operación Sincrona/Asincrona: Múltiples clientes pueden acceder a múltiples servidores. Los clientes siempre inician la petición, los servidores esperan pasivamente las peticiones de los clientes. Un cliente puede iniciar una petición en modo sincrono o asincrono. El proceso sincrono significa que el cliente espera hasta que el servidor ha cumplido la petición o hasta que haya expirado un plazo determinado.

En el modo Asincrono, el proceso cliente continua en cuanto el servidor acusa recibo de la petición. Una vez terminado el proceso servidor, se notifica al cliente. El modo asincrono es útil para procesos largos, por ejemplo la impresión de un fichero. En algunos casos especiales el proceso servidor conlleva la notificación asincrona al proceso cliente. Ejemplos de este tipo son los servidores que soportan dispositivos de I/O, como impresoras o interfaces gráficas de usuario.

Uno de los retos mas importantes de las organizaciones 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 Middleware mas adecuados.
- Seleccionar las herramientas de desarrollo de aplicaciones apropiadas.
- Diseñar las aplicaciones para un entorno distribuido.

1.4 MODELOS DE DISTRIBUCIÓN CLIENTE/SERVIDOR

Dependiendo de las funciones que se distribuyan, distinguiremos los tres modelos de distribución siguientes:

- *Presentación Distribuida:* Que aísla la presentación del resto de la aplicación.
- *Datos Distribuidos:* Que separa el acceso a los datos del resto de la aplicación.
- *Función Distribuida:* Que permite asignar las funciones de la aplicación a distintos procesadores.

1.4.1 Presentación Distribuida:

En este modelo la parte de la aplicación que se ocupa de la presentación de los datos esta situada remotamente respecto de la lógica de negocio y los datos de la aplicación. La lógica de presentación se desarrolla generalmente con alguna de las herramientas de desarrollo Cliente/Servidor. La presentación distribuida en su forma mas simple puede consistir en una interfaz gráfica del usuario para acceder a programas de proceso de transacciones ya existentes, alternativamente, se pueden realizar, como parte de la Presentación Distribuida, todos los diálogos con el usuario, incluyendo la validación de datos y la iniciación de la lógica de negocio y acceso a datos.

1.4.2 Datos Distribuidos:

Cuando se distribuyen los datos su acceso y manipulación quedan separados del resto de la aplicación, como Archivos Distribuidos y Bases de Datos Distribuidas.

Archivos Distribuidos: Se utilizan para compartir recursos de la red, lo que es común a este tipo de servidores es la lógica de la aplicación, que siempre se ejecuta en el proceso cliente, mas exactamente consiste en una redirección física de las solicitudes de escritura y lectura, con muchos mensajes fluyendo por la red para encontrar los datos pedidos.

Bases de Datos Distribuidas: Permiten utilizar de forma mas eficiente el procesador que controla la base de datos. El proceso cliente pasa las peticiones SQL en forma de mensajes al servidor de la base de datos. Todas las operaciones de búsqueda se ejecutan en el servidor y únicamente se devuelve el resultado a la aplicación invocante. Un posible inconveniente es que el resultado de una operación de búsqueda de la Base de datos pueden ser múltiples filas de datos que en un entorno WAN, puede tener implicaciones negativas en el rendimiento.

1.4.3 Funciones Distribuidas:

Este modelo proporciona la máxima flexibilidad y permite a los departamentos de desarrollo un control total sobre donde 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 volver a redistribuir funciones y datos casi siempre son necesarias importantes modificaciones o volver a diseñar la aplicación.

Un proceso cliente invoca a un proceso servidor que reside en el mismo procesador o en otro procesador de la red. El servidor puede realizar funciones relativas al negocio y acceder a archivos o bases de datos, devolviendo el resultado al proceso cliente que lo ha invocado. El cliente recibe el resultado sin saber si procede del mismo procesador o de otro distinto. Un servidor de aplicaciones puede, a su vez, actuar como cliente, solicitando funciones de otros servidores de la red..

1.5 FACTORES DE ÉXITO.

Los elementos del éxito de una estrategia son:

Independencia del suministrador: Como siempre existirá un determinado nivel de dependencia del suministrador, el sistema debe construirse bajo el supuesto de que cada uno de los componentes podrá ser sustituido, si se necesita un producto diferente o nuevo, o si el producto y/o el suministrador deja de existir. El nivel de flexibilidad o apertura puede medirse por el tipo de componentes que pueden ser sustituidos y por las implicaciones asociadas a su sustitución.

- Hardware del sistema.
- Sistema operativo y software del sistema.
- Middleware Cliente/Servidor.
- Herramientas y lenguajes de desarrollo de interfaces del usuario, acceso a datos y lógica del negocio.
- Aplicaciones estándar

Elección de herramientas de desarrollo, lenguajes y Middleware Cliente/Servidor: Como muchos entornos de desarrollo contienen funciones Middleware Cliente/Servidor, la elección de sus componentes es básica. Para conseguir los objetivos mencionados de portabilidad e interoperabilidad, hay dos exigencias clave. La interoperabilidad depende del cumplimiento de estándares, mientras que la portabilidad puede conseguirse mediante software que se ajuste a estándares, o que sea de naturaleza propietaria portable.

Mientras que las interfaces de programación de aplicaciones (API's), son muy importantes para la portabilidad de aplicaciones, los formatos y los protocolos son básicos para la interoperabilidad.

Arquitectura de aplicaciones: La frase "Abierto es como se diseña, no lo que se compra" que ha acuñado el Gartner Group, indica que para la apertura es mas importante la arquitectura y diseño de la aplicación que la selección de un determinado producto. Un buen diseño de la aplicación puede reducir la dependencia. Los productos pueden utilizarse de formas diferentes y un diseño y una utilización inteligentes marcaran las diferencias.

1.6 DISEÑO DE APLICACIONES CLIENTE/SERVIDOR.

La total flexibilidad relativa a la distribución de funciones y datos en un entorno cliente/servidor, no se consigue fácilmente sin costos. Las aplicaciones tienen que diseñarse adecuadamente. Una aplicación debe dividirse en funciones independientes y estas trasladarse a procesos cliente/servidor, enfatizando la importancia de la separación de funciones.

Por lo tanto en cualquier aplicación bien diseñada, las funciones siguientes funciones deben desarrollarse como componentes individuales según sus características, es decir, funciones relativas a:

- Lógica de presentación.
- Lógica del negocio.
- Lógica de datos.
- Otro conjunto de funciones llamado servicios de aplicación.

1.6.1 LÓGICA DE PRESENTACIÓN, DE NEGOCIOS Y DE DATOS.

Lógica de presentación: Se refiere a la interacción entre la lógica de negocio y los gestores de recursos de presentación, y que se ocupan de la presentación de información en pantallas, impresoras, multimedia, etc. Separando la presentación de la lógica de negocio y de datos se consigue el máximo nivel de portabilidad y distribución.

Por razones practicas esta parte se ejecuta normalmente en la estación de trabajo del usuario, mientras que las funciones de la lógica de negocio y de datos pueden distribuirse en distintos procesadores.

Lógica de negocio: Contiene las funciones de la aplicación y es la razón por la que existe esta. La lógica de negocio debe estructurarse de forma que correspondan a funciones (u objetos) que el usuario pueda invocar a través de la GUI.

Lógica de datos: Se refiere a la interacción entre la lógica de negocio y los gestores de recursos de datos; trata de la recuperación y actualización de los datos, soportando archivos planos, jerárquicos, relaciónales y bases de datos orientadas a objetos.

La lógica de datos (también llamada servicios de acceso a datos) esta ligada a la Interface de Programas de Aplicación (API), para archivos planos o jerárquicos y para las bases de datos relaciónales u Orientadas a Objetos.

Relación con la plataforma operativa: Las funciones de la lógica de presentación, de negocio y de datos, si se desarrollan con lenguajes o herramientas adecuados, son portables y pueden situarse en el procesador mas adecuado para su ejecución. Según se define en los Servicios de aplicación, en cada procesador también hay grupos de funciones. El Middleware aísla los elementos de la aplicación de los aspectos de distribución relativos al acceso de recursos locales y remotos (dispositivos, programas, datos).

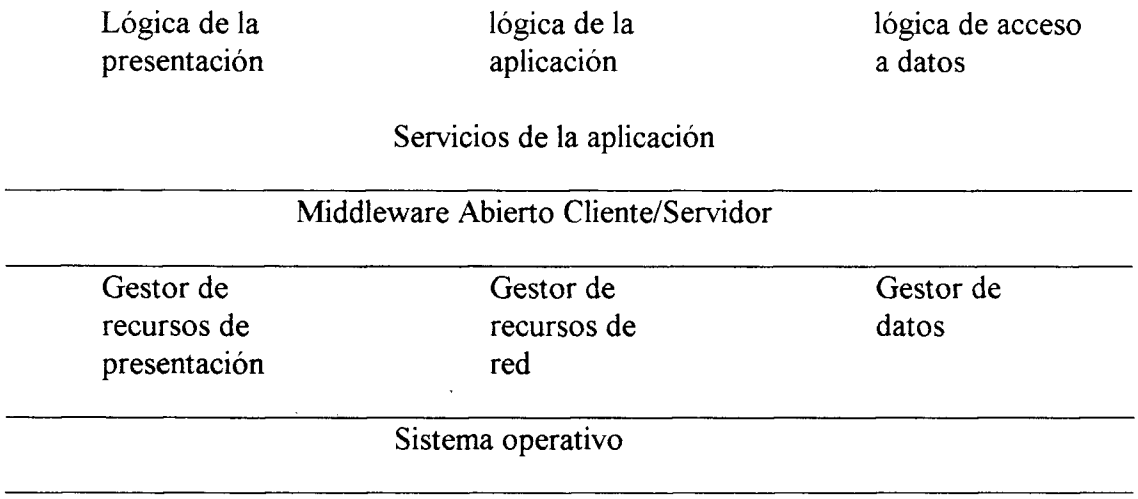


Figura 1.1. Partes del Entorno Distribuido.

1.6.2 GESTORES DE RECURSOS.

Para conseguir mas flexibilidad, los Gestores de Recursos se han desligado del Sistema Operativo y se han convertido en programas separados que funcionan por encima de el. Pueden ser suministrados por cualquier proveedor, y no exclusivamente por el del correspondiente Sistema Operativo.
Conceptos.

Los gestores de Recursos son programas de software que acceden a recursos y proporcionan una API. Un recurso puede ser dispositivo, pero también puede ser un archivo, una base de datos, un programa o un objeto, existen tres diferencias:

- Gestor de Recursos Local.
- Gestor de Recursos Remoto.
- Gestor de Recursos Distribuido.

Gestor de Recursos Local: Es el que solo puede operar dentro de los confines de un único sistema, proporcionando un servicio que relaciona el Sistema Operativo local. Ejemplos son la gestión de memoria y la distribución de tareas.

Gestor de Recursos Remoto: Puede operar en un sistema remoto, pero puede ser accedido como si fuera local. Se compone de dos partes:

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

Existen varios productos con esta capacidad tales como: OS/2 LAN Server, AS/400 Client Access y LANRES/VM.

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

- La familia DB/2 gestionando una base de datos distribuidos.
- La familia CICS gestionando un sistema de archivos distribuido y un entorno de aplicaciones distribuidas.

1.6.3 ACCESO A UN RECURSO.

El propósito de un gestor de recursos remotos o distribuidos es facilitar a los clientes acceso transparente a los recursos que controla.

El cliente no sabe donde esta situado el recurso y puede interactuar con el gestor de recursos en la forma mas adecuada (por ejemplo, SQL, para acceder a bases de datos relaciónales, OSF/Motif para pantallas). En un entorno distribuido todo puede ser considerado un recurso. No solo los tradicionales como impresoras y bases de datos, sino también los programas de aplicación que actúen como servidores de aplicaciones. La única diferencia es que estos no pueden acceder a ningún gestor de recursos que controle un recurso físico.

1.7 ARQUITECTURAS DE DOS Y TRES NIVELES:

El Middleware Cliente/Servidor y/o las herramientas de desarrollo de aplicaciones utilizadas, normalmente dictan el tipo de distribución y el nivel de flexibilidad, así como los procesadores soportados.

1.7.1 APLICACIÓN DE DOS NIVELES.

Los servicios de presentación, la lógica de la presentación y la lógica de aplicación y acceso de datos es, normalmente, un bloque monolítico que se ejecuta en la estación cliente, mientras la base de datos esta situada fisicamente en el servidor.

1.7.2 APLICACIÓN DE TRES NIVELES.

Las funciones de la aplicación están divididas en la lógica de presentación, lógica de la aplicación o de negocio y la lógica de datos, distribuidos entre múltiples aplicaciones de la red, como resultado, pueden existir servidores de aplicación o de transacciones que ejecuten funciones accesibles por muchos procesos cliente con distintas formas posibles de interacción con el usuario.

1.7.3 SISTEMAS ABIERTOS FRENTE A PROPIETARIOS.

La apertura de una plataforma, software o solución no se mide en términos absolutos. Es interpretada por algunos como el grado de cumplimiento con estándares de jure, mientras que otros consideran que un desarrollo propietario, que es portable entre plataformas hardware de distintos proveedores, puede ser una solución abierta. La apertura siempre tiene que juzgarse en relación con necesidades especificas, como por ejemplo:

- Independencia del software y del hardware.
- Portabilidad de la aplicación.
- Cumplimiento de estándares, de facto o de jure.
- Productos soportados por distintos proveedores.
- Productos propietarios que tengan aceptación en el mercado.

1.8 SELECCIÓN DE SOFTWARE PARA LA PLATAFORMA OPERATIVA.

1.8.1 APERTURA.

En un entorno de sistemas distribuidos, los sistemas operativos deben proporcionar los mecanismos que permitan a los procesos independientes intercambiar y compartir datos. Otras funciones importantes son la multitarea controlada por el propio sistema operativo, con capacidad de despachar los trabajos en base a sus propiedades.

1.8.2 SISTEMA OPERATIVO.

Debe ser posible soportar aplicaciones con datos distribuidos, función distribuida y presentación distribuida. Probablemente, la primera aplicación se pueda desarrollar perfectamente con datos distribuidos. La próxima puede requerir que la función o la presentación estén distribuidas. Se origina un grave problema si los productos de middleware o la herramienta de desarrollo elegidos no permiten esta posibilidad.

1.8.3 SOPORTE DE MULTIPLES MODELOS DE DISTRIBUCION CLIENTE/SERVIDOR.

Aunque no siempre es el caso, muy probablemente será necesario soportar interfaces gráficas de usuario, y hay otras tecnologías que están emergiendo cada vez con mas rapidez (ejemplos típicos en este momento son multimedia y la tecnología de Orientación a Objetos). El middleware es la plataforma que permite todo esto y debe hacer posible la incorporación de nuevas tecnologías tan pronto como sea posible, si ello puede beneficiar a los usuarios.

1.8.4 SOPORTE DE LAS TECNOLOGÍAS MAS IMPORTANTES.

Teniendo en cuenta que la mayoría de los entornos Cliente/Servidor serán heterogéneos, es decir, construidos a base de componentes de mas de un proveedor (tanto de hardware como de software), estos componentes deben ser capaces de operar entre ellos. En la practica, esto significa, que tendrán que ajustarse a los estándares. Los productos middleware deben usar servicios estándar de distribución, en lugar de proporcionar los suyos propios de forma propietaria. El Open Blueprint es una herramienta útil para seleccionar arquitecturas, formatos y protocolos de naturaleza abierta.

1.8.5 HERRAMIENTAS DE DESARROLLO DE APLICACIONES.

Como se ha explicado, el conjunto de aplicaciones no se compone únicamente de desarrollos propios, sino también de aplicaciones estándar disponibles comercialmente. Todos estos componentes deben estar integrados al máximo y aparecer ante el usuario como una aplicación única y homogénea que soporte a sus necesidades. La integración no solo debe estar a nivel de datos, sino preferiblemente a nivel de función.

1.9 INTEGRACIÓN DE DESARROLLOS PROPIOS Y APLICACIONES ESTÁNDAR.

La relación entre clientes y servidores no debe ser biunivoca, sino que un cliente debe tener acceso a múltiples servidores de forma concurrente, y un servidor debe poder ser accedido concurrentemente por múltiples clientes. Las implicaciones mas significativas de esta característica son:

-Los servidores deben residir en plataformas con sistema operativo multiusuario y multitarea.

-Los clientes deben tener acceso a los servidores tanto en modo sincrono como asincrono.

1.9.1 RELACIÓN CLIENTE/SERVIDOR.

Nunca se deben seleccionar productos de middleware ignorando las herramientas de desarrollo, no solamente por la importancia de que estas soporten la plataforma operativa, sino también porque muchas herramientas de desarrollo de aplicaciones incluyen componentes middleware.

1.9.2 ENTORNO DE DESARROLLO DE APLICACIONES.

Respecto a esta área, no existe una solución valida universal para todas las problematicas posibles, ya que las situaciones son muy diferentes en cada empresa; distinto Sistemas Operativos, distintos modelos de distribución cliente/servidor o empleo de diferentes tecnologías, para ayudar en la selección de las herramientas para un entorno determinado, es necesario entender la relación entre la Plataforma Operativa y el Entorno de Desarrollo de Aplicaciones.

1.9.3 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 razones siguientes:

- Una plataforma operativa para la que no existan herramientas de desarrollo, dificilmente puede ser útil.

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

1. 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 conectarse.
2. La interface entre los procesos cliente y servidor no es externa, en este caso, un proceso servidor no puede ser invocado por un proceso cliente construido con una herramienta diferente.
3. Los directorios y el sistema de seguridad se desarrollan de forma propietaria. En el mejor de los casos, esto ocasiona una complejidad añadida si se requiere la utilización de otros productos middleware. El caso mas desfavorable se produce cuando las funciones middleware de un producto limitan el tipo de sistemas y de servicios de comunicación soportados.

- Muchas herramientas de desarrollo soportan únicamente un subconjunto de los distintos modelos de Distribución Cliente/Servidor, o solamente las soluciones cliente/servidor de dos niveles.

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, en el que puedan desarrollarse nuevas aplicaciones añadiendo mas procesos cliente y servidor.

1.9.4 PROCESO ITERATIVO.

La nueva tendencia para el desarrollo de aplicaciones es utilizar un diseño iterativo, también conocido como proceso en espiral. Las herramientas Cliente/Servidor y Orientadas a Objetos son muy adecuadas para este tipo de procesos, en los que, una vez terminada la etapa de análisis, los programadores trabajan directamente con el usuario utilizando prototipos que se generan muy rápidamente.

1.9.5 CLIENTE/SERVIDOR Y ORIENTACIÓN A OBJETOS.

Una infraestructura Cliente/Servidor y el Entorno de Desarrollo de Aplicaciones deben ser capaces de incorporar las tecnologías orientadas a objetos, aunque no se usen inicialmente. Dentro de este mundo se pueden identificar varias disciplinas: Interfaces de Usuario (OOUI), Diseño OO, Programación OO y Bases de Datos OO.

1.9.6 Interfaces de Usuario Orientadas a Objetos (OOUI).

Una OOUI es una clase de espiral de GUI basada en el concepto objeto-accion (en lugar de accion-objeto) e incorpora las siguientes características generales:

- El usuario selecciona objetos mediante un mouse y lo arrastra hacia el objetivo.
- El usuario obtiene una respuesta inmediata a las acciones seleccionadas.

La interface no tiene estados o modos que alteren el significado de las acciones del usuario.

- Los objetos se presentan al usuario WYSIWYG.
- Los objetos y el significado de las acciones son consistentes, tanto dentro de una aplicación como entre varias.

1.9.7 Diseño Orientado a Objetos.

La construcción de objetos es una tarea compleja e iterativa. Utilizando las herramientas Orientadas a Objetos para construir al menos los clientes GUI, y quizás las herramientas mas tradicionales para construir servidores de negocio encapsulados, se facilitara la incorporación gradual de la tecnología Orientada a Objetos.

1.9.8 Programación Orientada a Objetos (OOP).

La unidad básica de organización en programas Orientados a Objetos es el objeto. Este se compone de los datos y del método, es decir, las acciones que se van a realizar sobre los datos. El método se invoca enviando un mensaje al objeto.

1.9.9 Servidores de Aplicación como Objetos.

Como los servidores de negocio encapsulados tienen interfaces claramente definidas, el impacto que pueda tener esta migración sobre los clientes de un servidor de negocios encapsulado será mínimo. La ventaja es que el uso de este enfoque hace posible la migración gradual hacia un entorno Orientado a Objetos; los componentes de la aplicación construidos son tecnologías procedural y de Orientación a Objetos pueden coexistir y trabajar conjuntamente en el mismo entorno del sistema.

1.10 COMPONENTES DE TIEMPO DE EJECUCIÓN (MIDDLEWARE).

Muchas herramientas de desarrollo proporcionan un componente para ejecución con funciones de Middleware Cliente/Servidor. Si ese es el caso, se deberían de tener presentes los siguientes aspectos:

- ¿Es el middleware propietario o se utilizan servicios de distribución que cumplen estándares de la industria?
- ¿Que modelos de distribución cliente/servidor están soportados?
- ¿Es externa la interfaz entre los componentes cliente y servidor?
- ¿Permite la herramienta aplicaciones en dos y/o tres niveles?
- ¿Cual es el nivel de integración soportado?: Integración en el puesto de trabajo, recursos compartidos o enlace directo entre aplicaciones.
- ¿Que tecnologías están soportadas? o ¿se trata de una herramienta abierta para acoger nuevas tecnologías (por ejemplo multimedia, orientación a objetos)?

1.10.1 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 la apertura y los desarrollos propietarios. Probablemente cuanto mas productiva sea la herramienta, mas cerrado y propietario será su carácter y, por tanto, mayor la dependencia del proveedor. El remedio no es recurrir a API's de bajo nivel para conseguir el máximo nivel de apertura. Algunos factores a tomar en cuenta son:

- Una plataforma de sistemas y aplicaciones claramente definida.
- Definición de estándares, formatos y protocolos soportados.
- Una arquitectura de aplicación que también contemple: Plataforma Operativa y Desarrollo de aplicaciones, delimitando el nivel permitido de características de tipo propietario.
- Principios de diseño de aplicaciones que definan como se diseñan y desarrollan las aplicaciones Cliente/Servidor.

1.10.2 CLIENTE/SERVIDOR:

Que tan cierto es que la arquitectura cliente/servidor será la forma de trabajar de los 90's, se menciona que la arquitectura cliente /servidor pude reducir el costo de mantenimiento de software, incrementar la portabilidad de las aplicaciones, mejorar el desempeño de los sistemas y las redes y llegar a eliminar la necesidad de usar minicomputadoras y mainframes, inmediatamente caemos en un debate acerca de cuales de estas promesas son realistas,

primero analicemos un poco de la evolución de los entornos computacionales y encontremos un lugar para el modelo cliente/servidor.

1.10.3 PROCESAMIENTO BASADO EN EL HOST.

Este es el entorno mas primitivo para soportar procesamiento de aplicaciones, no cuenta con capacidades para aplicaciones distribuidas. El procesamiento basado en el Hostson desempeñadas en un solo computador al que son atadas las conocidas como terminales tontas, este sistema se considera como no-distribuido.

1.10.4 PROCESAMIENTO MAESTRO-ESCLAVO.

Este fue el siguiente paso que se dio hacia los entornos distribuidos, como su nombre lo indica las computadoras esclavas son atadas a una computadora maestra que desempeñaba funciones relacionadas a procesamiento y a las aplicaciones, las computadoras esclavas eran capaces solo de desempeñar tareas locales tales como validación de campos en las pantallas, edición y procesamiento de las teclas de función.

1.10.5 PROCESAMIENTO CLIENTE/SERVIDOR.

Este modelo emergió como un sistema que se encontraba normalmente en las redes de área local (LAN), donde las PC's son atadas a un sistema que les permitía compartir recursos (Como archivos en discos duros o impresoras, en la tecnología LAN, los equipos que permitían tale capacidades eran llamados servers o servidores en español, aquí las PC's eran capaces de desempeñar sus aplicaciones y solo algunas funciones como las de impresión y archivos I/O eran distribuidas, como estos archivos eran leídos y actualizados completamente este tecnología exigía servidores exclusivamente dedicados a estas tareas como ocurrió en Novell NetWare y LAN Manager de Microsoft, el modelo cliente/servidor es una extensión natural de esta tecnología, el modelo cliente/servidor implica un procesamiento cooperativo requerimientos hechos por un cliente a un servidor el cual procesa el requerimiento y envía el resultado al cliente. Actualmente el procesamiento es iniciado por un cliente y parcialmente controlado por el servidor.

Arquitecturalmente, el procesamiento cliente servidor requiere:

- Comunicaciones robustas y confiables entre clientes y servidores.
- Interacciones cooperativas cliente/servidor que son iniciadas por un cliente.
- Aplicaciones distribuidas de procesamiento entre un cliente y su servidor.
- Exige a los servidores el control sobre que clientes tienen acceso a los datos y servicios.
- Exige a los servidores capacidades para resolver conflictos que puedan surgir entre los requerimientos de los clientes.

1.10.6 PROCESAMIENTO PEER-TO-PEER.

Esta arquitectura es diferente a la de peer-to-peer de las DOS LAN, ya que su principal característica es que todos los participantes en este sistema pueden tener las características de ser quienes solicitan requerimientos o también de proveer servicios. Es decir que puede actuar como cliente para otros servidores y como servidor para otros clientes incluyendose a sí mismo, teniendo como capacidad que sea transparente para el usuario el procesamiento de los datos incluso en una amplia variedad de plataformas de hardware y software. Una meta importante del procesamiento peer-to-peer es la de soportar bases de datos en red donde los Sistemas Manejadores de Bases de Datos (DBMS), trabajen también de manera transparente, esta es actualmente la meta para los diseñadores de sistemas distribuidos.

1.10.7 VENTAJAS DE LA COMPUTACIÓN CLIENTE/SERVIDOR.

Debido a las dificultades técnicas para soportar el procesamiento peer-to-peer, este se a vuelto una tarea difícil de lograr, el modelo cliente/servidor puede no ser un tipo de procesamiento peer-to-peer ideal, pero sí un razonable acercamiento al desarrollo de los sistemas distribuidos. Este modelo provee significantes beneficios y puede ser usado en las soluciones de los negocios de hoy.

Existen beneficios que pueden resultar de la adopción de una arquitectura cliente/servidor:

- Ayuda a las corporaciones a adoptar tecnologías de computación desktop mucho mejor, gracias al considerable poder que tienen las computadoras de hoy, antes disponibles solo en mainframes y a una fracción de su costo.

- Puede propiciar que los datos se ubiquen fisicamente cerca de la fuente donde son procesados y esto disminuye el trafico de la red (y por lo tanto el tiempo de respuesta), pueden ser reducidos, disminuyendo también el ancho de banda y los costos requeridos.

- Capacidades para manejar interfaces gráficas de usuarios (GUI), que facilitan la navegación y dan consistencia a las pantallas, como resultado de esto se puede esperar que las inversiones en entrenamiento y educación disminuyan así como también nuevos productos o desarrollos pueden ser fácilmente integrados y la resistencia de los usuarios finales puede ser minimizada.

- Nos lleva a la aceptación de los sistemas abiertos, de hecho clientes y servidores pueden estar corriendo en diferentes plataformas de software y hardware, evitando dependencia de proveedores.

También existen algunas desventajas del modelo cliente/servidor que debemos señalar:

- Si un servidor es cargado con demasiadas aplicaciones lógicas, dicho servidor se puede volver un cuello de botella.

- Las aplicaciones distribuidas especialmente aquellas diseñadas para procesamiento cooperativo, son mas complejas que las no-distribuidas, sin embargo, algunas de estas complejidades pueden tratar con diseño de sistemas modulares, esto consiste en desbaratar un gran problema en muchos problemas pequeños.

Se dicen muchas cosas acerca de que cliente/servidor puede volver a los usuarios no técnicos en desarrolladores profesionales de software y que además puede forzar a las minicomputadoras y mainframes a desaparecer, todo esto es altamente cuestionable, la mayoría de las publicaciones actuales enfatizan solo un lado del modelo cliente/servidor, el lado relacionado a la distribución de datos, sin embargo, esta arquitectura envuelve mucho mas que datos, debemos tomar en cuenta las redes y comunicaciones de datos, presentación distribuida y procesamiento de transacciones distribuidas, estándares y sistemas abiertos.

CAPITULO II. ELEMENTOS CLIENTE/SERVIDOR

A lo largo de este capítulo, se comprenderán los elementos Cliente/Servidor, tales como Interface de Usuario (GUI), servidores, las redes y comunicaciones y como deben de trabajar en conjunto para entender cuál es la distribución adecuada de los componentes de las aplicaciones en la arquitectura Cliente/Servidor, como deben ser distribuidos, accesados y administrados los datos, así como se explicará como debe ser la administración y distribución del software en un ambiente distribuido y la implementación de seguridad a datos y programas en el entorno Cliente/Servidor.

2.1 REQUERIMIENTOS GENERALES PARA UNA GUI ESTANDAR.

2.1.1 PORQUE UNA GUI ESTÁNDAR.

La amplia variedad de interfaces disponibles actualmente, puede confundir a usuarios y desarrolladores, cada nueva interface requiere re-entrenamiento y a veces exige que las aplicaciones sean modificadas, por lo tanto, se debe buscar que una GUI estándar con una sola interface de programas de aplicación (API) común, esto será de gran impacto para la productividad.

Portabilidad: Las aplicaciones y habilidades de los usuarios deben ser portables a través de varias plataformas de sistemas abiertos, una GUI estándar puede mantener una API estable para cada plataforma.

Standards Compliance: Este requerimiento es la clave para los sistemas abiertos y marca los lineamientos para el trabajo en grupo, refiriendose a la interoperabilidad, estas especificaciones están listadas en el Interclient Communications Conventions Manual (ICCCM).

Herramientas de desarrollo: Cualquier GUI que vaya a ser considerada como un estándar debe ser acompañada por herramientas de desarrollo (es decir que faciliten los nuevos desarrollos y que trabajen bajo diferentes plataformas.

Flexibilidad: Capacidad para acomodar nuevos displays y otros equipos de I/O.

Internacionalización: Aseguramiento de la portabilidad, en vistas al mercado global (lenguajes de otros países, símbolos especiales, unidades monetarias, horarios y fechas a nivel mundial).

Independencia de plataformas: Es decir que opere sin importar el sistema operativo ni los protocolos de red.

2.2 CARACTERÍSTICAS DE LA GUI.

En general las GUI trabajan con "ventanas", que se pueden traslapar unas con otras, conteniendo "objetos" llamados iconos con capacidades de minimización, eliminando casi por completo el tecleo de comandos que algunas veces pueden resultar complejos para los no-técnicos, la GUI debe ser capaz de aceptar y procesar eventos asincronos iniciados por el usuario o sistema en cualquier momento.

2.2.1 TIPOS DE EVENTOS.

Eventos de Mouse: Ocurre cuando se hace "click" en la pantalla.
Eventos de teclado: Ocurren al teclear.
Eventos de Menú: Cuando se despliega un menú en la pantalla.
Eventos de actualización de ventana: Ocurren cuando se traslapan las ventanas y estas tienen que ser redibujadas.
Eventos de resizing: Cuando se modifica el tamaño de una ventana.
Eventos de activación o desactivacion: Ocurren cuando hay cambios en la pantalla activa.
Eventos de inicialización/terminación: Cuando se destruye o se limpia una pantalla.

2.2.2 EVENTO DE DISTRIBUCIÓN:

API: aplication programing interface, rutinas especificas de librerías que desempeñan la función de crear ventanas y desplegar gráficos, existen varios modelos descritos a continuación.

Modelo evento de loop: Rutinas para checar eventos pendientes y regresar a ellos sino han sido completados.

Modelo evento callback: Rutinas que son llamadas desde teclado o mouse y responder a ellas.

Modelo híbrido: Combina las dos anteriores, un evento puede llamar otra rutina API, para manejar otros eventos.

2.2.3 CARACTERÍSTICAS DE SALIDA DE LA GUI.

Existen muchas características que distinguen a una GUI de otra.

Coordinación de espacios: Describe un sistema de coordenadas bidimensional, para situar pixeles en la pantalla para definir un punto de inicio de operaciones y definir la resolución de un espacio dibujado.

Algoritmos de dibujo: Describe la manera particular en como una GUI dibuja, centra y conecta líneas.

Efectos de color: Desgraciadamente no existe un código de color establecido en la GUI's, estos difieren de la plataforma e incluso algunos usuarios los personalizan.

Presentación de texto: Es diferente en un entorno gráfico de un sistema basado en caracteres, en una GUI el texto es tratado como un gráfico lo cual permite diferentes tipos de letra, tamaños, etc.

2.2.4 X WINDOWS SYSTEM.

Usaremos este sistema para ejemplificar una GUI.

En el entorno cliente/servidor la capa de presentación necesita soportar estas características criticas:

- Las aplicaciones corriendo en los servidores deben de ser capaces de accesar transparentemente la lógica de presentación que corre en los clientes.

- La lógica de presentación en un cliente desempeña servicios requeridos por la aplicación corriendo en el servidor (entonces el cliente actúa como un servidor de presentación).

Xwindows desarrollado por el MIT, IBM y DEC esta basado en una arquitectura cliente servidor, proporciona:

- Transparencia entre los protocolos de comunicación de la red entre una aplicación y su lógica de presentación.
- Alto desempeño e independencia de gráficos con equipos.
- Jerarquía para dar tamaño a las pantallas y traslapar las ventanas.
- El modelo X consta de un display, un servidor y programas clientes, el display X es el hardware que incluye una pantalla con capacidad de mapeo de bits, teclado, teclado e interfaces de mouse, el servidor X controla los programas y soporta la interface, clientes y servidores X pueden correr en el mismo o diferentes nodos de la red utilizan el X protocol para el intercambio de mensajes.
- X Window, utiliza una ventana raíz de la cual se despliegan otras pantallas.
- El sistema X se maneja por eventos, es decir el cliente hace un requerimiento y espera hasta que el servidor envía respuesta y así sucesivamente.
- La administración de X reside en el cliente y desde ahí controla ventanas e iconos.
- Para comunicar un cliente X con su servidor X requiere el X protocol que son una serie de rutinas en C, llamadas Xlib que ha sido enriquecida con la creación de toolkit.

entonces tenemos que:

1. En el sistema X, los clientes inician las interacciones con el servidor, Xserver.
2. Un cliente requiere de un servidor X para desempeñar funciones de entrada y salida.
3. Un servidor X soporta a múltiples clientes.
4. El cliente y el servidor X pueden residir en la misma maquina.

2.3 SERVIDORES.

2.3.1 FUNCIONES.

Los cambios en la tecnología han llevado a la especialización de los servidores, como un ejemplo los Database servers, aunque esta especialización se ha extendido a funciones de comunicación, emulación de terminales, fax, administración de librerías y correo electrónico. Las funciones que desempeñan los servidores son determinadas por los tipos de requerimientos de los clientes, a continuación se describen algunas funciones hechas por los usuarios a los servidores:

Compartición de archivos, (File sharing): En un entorno de trabajo en grupo se deben de compartir archivos y eso hace posible la existencia de los file servers, para controlar que nadie entre a un archivo cuando un cliente lo este utilizando y para transferencia de archivos entre clientes.

Printer sharing: Print servers, compartir periféricos.

Databas Access: acceso concurrente a bases de datos de manera transparente (DBMS).

Comunicación serviles: Comunicativos Servers donde se hacen los requerimientos de comunicación y conexiones a procesadores remotos.

Facsímile services: Fax server.

La especialización de los servidores debe ser evaluada dependiendo de la carga de trabajo que se tenga, estos pueden ser especializados para satisfacer requerimientos de propósito general tales como:

Soporte multi-usuario: Soportar sistemas de multitasking.

Escalabilidad: Prever la posibilidad de crecimiento de la red y que el servidor soporte a este.

Performance: Desempeño en el tiempo de respuesta, que tanto importa.

Storage: Capacidad del servidor para soportar almacenamiento a gran escala como las herramientas CASE y los DBMS.

Multimedia: Capacidades de manejar imágenes, hipertexto y almacenamiento óptico tales como DASD y WORM (Write-Once-Read-Many).

Networking: Capacidades de red, si un sistema es diseñado con las características de red en mente la arquitectura de software y hardware pueden ser óptimamente integradas con las interfaces y protocolos de red.

Aunque en la actualidad existen varias arquitectura cliente servidor que trabajan con mainframes, la tecnología de las microcomputadoras, ha proporcionado mayores economías de escala.

2.3.2 ARQUITECTURA HARDWARE PARA SERVIDORES.

El grado de especialización de los servidores se puede lograr por sus capacidades de hardware (tales como tecnología RISC y Symetric MultiProcesing SMP).

2.3.3 CONSIDERACIONES DE LOS SISTEMAS.

Entre los factores que afectan el desempeño de los servidores se encuentran la arquitectura del CPU, tamaño e implementación de los métodos para procesar instrucciones, la habilidad de los compiladores para optimizar el desempeño, la tecnología de los chips, y el sistema operativo. Desde el punto de vista de las aplicaciones tomemos en cuenta que la mayoría se desarrollan en C mas que en lenguaje ensamblador.

Para evaluar el desempeño de un equipo nos podemos concentrar en los sig. puntos:

Acortar las instrucciones de la ruta de lenguaje.
Mejorar el porcentaje de instrucciones por ciclo.
Aumentando la velocidad del ciclo del reloj.

2.3.4 RISC VS CICS.

RISC, Reduced Instructions Set Computer, las instrucciones son decodificadas directamente por el hardware, incrementando la velocidad de procesamiento. La tecnología RISC actual divide al procesador en tres partes, The branch processor, que decodifica las instrucciones y las asigna a otras dos unidades, el procesador de punto flotante y el procesador integer, cada una de estas parte puede desempeñar varias instrucciones simultáneamente.

2.3.5 MULTIPROCESAMIENTO SIMETRICO.

Este tipo de procesamiento se ha vuelto indispensable para soportar aplicaciones complejas. Es mas barato tener la facilidad de aumentar el performance de la red cuando tenemos capacidad de instalar mas procesadores, esto nos lleva a otro tipo de especialización de servidores, los multi-processor-based servers. Existen dos tipos genéricos:

Loosely-Coupled o Multiprocesador de memoria distribuida, que incorpora un gran numero de procesadores autosuficientes, cada uno con sus propios sistemas operativos y recursos, estos se comunican entre si por canales de I/O a través de redes, teniendo comunicación a nivel interprocesadores, son relativamente lentos.

Tightly-Coupled o Multiprocesador de memoria compartida, consiste de un pequeño numero de procesadores (relativamente), que comparten una memoria común, sistema operativo común y I/O canales comunes, los procesadores seleccionan tareas a ejecutar de una tarea común y se interconecta por un sistema de bus de alta velocidad, son muy eficientes.

De estos son los últimos proveen verdadera escalabilidad a bajo costo.

2.3.6 ARQUITECTURAS ESCALABLES DE HARDWARE.

Asegurarse que los servidores tengan capacidad para aumentar elementos de procesamiento y buses de I/O para soportar elementos periféricos y ejecutar las aplicaciones de manera transparente al usuario.

2.3.7 ARQUITECTURAS ESCALABLES DE SOFTWARE.

Los sistemas operativos diseñados para soportar SMP (procesamiento SIMETRICO), son mas complejos.

Estos sistemas operativos deben de contar con estrategias de diseño para evitar actualizaciones simultáneas a estructuras y códigos de datos en la memoria común compartida, El balanceo dinámico de cargas es un requerimiento critico para el Software SMP, distribuir las cargas a todos los PE's disponibles (procesing elements), deben facilitar la computación en paralelo, donde múltiples aplicaciones pueden correr simultáneamente, existen varias técnicas para desarrollar aplicaciones en paralelo capaces de llevar a cabo un máximo de performance en un SMP.

2.4 REDES Y COMUNICACIONES.

2.4.1 SISTEMAS DE COMUNICACIÓN.

Un sistema de comunicación es un mecanismo que permite a los recursos distribuidos de una red intercambiar control de la información y datos, estos deben de ser transparentes para el usuario final.

2.4.2 COMUNICACIÓN Y DISTRIBUCIÓN.

La arquitectura cliente/servidor es un caso especial de procesamiento cooperativo y este a su vez es un caso especial de procesamiento distribuido, pero antes de seguir, ¿Porque tanto interés en los sistemas distribuidos?

- Los avances tecnológicos en la microelectronica han cambiado el porcentaje precio/desempeño resultando en bajos costos.
- Los costos de interconexión y comunicaciones se han reducido dramáticamente en los últimos años.
- Los usuarios demandan capacidades mas económicas, rápidas, sofisticadas y confiables.

2.4.3 FUNCIONES DE UN SISTEMA DE COMUNICACIÓN.

Naming and addressing: Un sistema de comunicaciones maneja nombres para objetos tales como procesos, puertos, mailboxes, sistemas y sesiones entre usuarios (mantenimiento de mapas y directorios), una dirección destino es mapeada dentro de una ruta o un objeto tal como un gateway.

Segmenting: Si los mensajes o archivos a transmitir de los usuarios son muy largos, el sistema de comunicaron lo debe fragmentar para la transmisión y volverlo a ensamblar antes de entregarlo, razones para la segmentación:

- Mensajes muy largos incrementan los retrasos en el acceso a otros usuarios.
- Los mensajes cortos mejoran la eficiencia y reducen los porcentajes de errores en la transmisión.
- Los buffers internos, usados por el sistema de comunicación, pueden limitar el tamaño del mensaje transmitido.
- Cuando varios datos siguen una ruta particular de transmisión, cada componente de la red puede tener diferentes tamaños de mensaje.
- En algunas redes la fragmentación de mensajes permite el uso de links de datos en paralelo, soportando transmisión en paralelo y evitando retardos.

Blocking: Es un protocolo usado para combinar mensajes de diferentes usuarios dentro de un solo bloque de transmisión.

Flow Control: Generalmente son atados a la red mas usuarios que los recursos que comparten bajo el concepto de que no todos van a utilizar los recursos al mismo tiempo sin embargo, existen horas pico de trafico en los que la red se congestiona, el control de flujo evita este efecto.

Sinchronization: Sincronizar entre las partes transmisora y receptora en ajuste de velocidades, señal de inicio, byte y bloques, las entidades deben mantener información por la vía del protocolo de comunicación.(APPC por ejemplo).

Priority: Identificar que usuarios tienen preferencia sobre los otros.
Control de error: Para proveer confiabilidad, el control de error debe incluir funciones tales como detección de errores, corrección y recuperación.

2.5 CAPAS, PROTOCOLOS E INTERFACES.

Consideremos un modelo típico de comunicaciones persona-a-persona, que consta de tres capas:

Capa Cognitiva: Se refiere a que los dos puntos deben de saber lo que están hablando, debe existir entendimiento entre estos.

Capa de lenguaje: Que las palabras tengan los mismos conceptos e ideas, hablar el mismo lenguaje, sin esto cualquier comunicación es imposible.

Capa de transmisión física: Proveer los medios para la comunicación (papel, cableado, etc.).

Todos las arquitecturas de red deben compartir los mismos objetivos de alto nivel, estos son:

1. Conectividad que permita conectar hardware y software dentro de un sistema de red uniforme.
2. Modularidad que permita construir componentes de propósito general dentro de diversos sistemas de red.
3. Confiabilidad, para soportar comunicaciones libres de error.
4. Fácil de implementar, logrando que las capacidades de comunicación sean transparentes para el usuario.

2.5.1 Capas con que cuenta una arquitectura de sistemas distribuidos:

Capa de aplicaciones: La capa mas alta de la arquitectura, administra los procesos de aplicación, distribución de datos, comunicación entre procesos y descomposición de las funciones de aplicación.

Capa de sistema operativo distribuido: Esta capa soporta los nombre, direcciones, recursos locales compartidos, protección, sincronización, intercomunicación y recuperación, también es responsable de crear el Single System Image.

Capa kernel y de administración local: Soporta el sistema operativo distribuido dentro de los nodos individuales, procesos de comunicación local, memoria, acceso de I/O y multitarea, comunicación peer layer.

Capa de los sistemas de comunicación: Soporta las comunicaciones requeridas por las tres capas anteriores.

La arquitectura de capas provee los siguientes beneficios:

Independencia de las capas: Cada capa depende solo de la capa debajo de ella, pero no de la implementación de las capas mas bajas.

Flexibilidad: Un cambio en una capa no debe de afectar las demás capas (como avances en tecnología, etc.).

Simplifica la implementación y el mantenimiento: Descomponiendo en pequeñas secciones la arquitectura y el diseño modular.

Estandarización: Encapsulación de la capa de funcionalidad, servicios e interfaces dentro de una arquitectura definida permite establecer estándares.

2.6 PROCESAMIENTO COOPERATIVO Y ADMINISTRACIÓN DE DATOS EN UN ENTORNO CLIENTE/SERVIDOR.

Las principales preguntas a las que debe responder un diseñador de un entorno cliente servidor son:

¿ Cual es la distribución adecuada de los componentes de aplicaciones en la arquitectura C/S? ¿Donde deben residir cada uno de estos componentes (lógica de presentación, lógica del negocio y procesamiento lógico de la Base de Datos?

¿ Como deben de ser distribuidos, accesados y administrados los datos?

¿ Como pueden los componentes de interacciones ser implementados?

¿ Cuales son los roles de y los requerimientos para la administración de transacciones distribuidas?

¿ Como puede la administración y distribución del software ser implementada en un entorno distribuido?

¿ Como debe ser implementada la seguridad a datos y programas en un entorno cliente/servidor?

A continuación se responderá a estas preguntas.

2.6.1 CLIENTE/SERVIDOR Y PROCESAMIENTO COOPERATIVO.

En C/S el procesamiento cooperativo es dividido entre un sistema cliente y un sistema servidor (no necesariamente iguales pero si de manera equilibrada), trabajando en conjunto para desempeñar una función.

Recordemos, que los mayores beneficios de una arquitectura C/S es el poder y el precio/desempeño de las microcomputadoras, elevar la productividad, facil-de-usar y facil-de-aprender son ganancias asociadas a este entorno y al desarrollo de GUI, íntimamente relacionadas en C/S, estos beneficios se llevan a cabo a través de la especialización de componentes de hardware en la arquitectura C/S.

El termino Cliente/servidor tiene dos significados, uno en relación a procesamiento cooperativo entre sus componentes y otro en su relación entre el hardware del cliente y el servidor.

El modelo cliente/servidor ofrece una vista del entorno desde la perspectiva de la estación cliente, un usuario final del entorno C/S puede accesar varios recursos que estén distribuidos entre múltiples servidores, localizados en los nodos de la red y conectados vía comunicaciones de red. Un usuario de C/S debería de contar con un sistema SSI Single System Image, de manera que estos accesos fueran transparentes para el usuario (Sin importar la capa física ni los Sistemas operativos), esta participación se logra mediante el procesamiento cooperativo y este es el fundamento de C/S.

2.6.2 ESTRUCTURA DEL PROCESAMIENTO COOPERATIVO.

C/S utiliza procesamiento cooperativo distribuido para:

-Componentes de procesamiento cooperativo distribuido entre clientes (normalmente presentación y algunas partes de la lógica del negocio) y servidores (partes de la lógica del negocio, lógica de bases de datos y DBMS).

- Soporte de interacciones entre clientes y servidores en un ambiente cooperativo.

Sin embargo existen varios posibles estilos de procesamiento cooperativo entre aplicaciones distribuidas, estos pueden ser definidos como:

- Presentación distribuida (DP).
- Presentación remota (RP).
- Lógica del negocio distribuida (DBL).
- Administración distribuida de datos (DDM).
- Administración remota de datos (RDM).

2.7 TÉCNICAS DE PROCESAMIENTO COOPERATIVO.

Para que una arquitectura de este tipo funcione necesariamente se necesitan técnicas de comunicación, teóricamente existen tres tipos básicos de técnicas de comunicación para sistemas cooperativos que la arquitectura cliente servidor puede usar:

Pipes
Remote Procedure Calls
Client/server SQL interactions

Pipes: o tubos que funcionarían como los tubos para el agua, entran por un lado y salen por el otro, los detalles del transporte de la información permanecen ocultos para el usuario y esta técnica impone mínimos requisitos de formateo y protocolos para usarlos.

Remote Procedure Call (RPC): Es un mecanismo por el cual un proceso puede ejecutar otro proceso o subrutina que reside en un diferente y usualmente remoto, sistema posiblemente corriendo en un sistema operativo diferente.

Client/server SQL interaction: Para el uso de bases de datos relaciónales donde se cuenta con un servidos de bases de datos, este tipo de técnica impone grandes restricciones de formateo y protocolos.

2.7.1 SINGLE SYSTEM IMAGE (SSI).

Muchos sistemas diferentes pueden participar en la ejecución de un requerimiento de algún usuario pero esto es transparente para él, sistemas multiclientes/multiservidores.

2.7.2 PRESENTACIÓN DISTRIBUIDA.

El primer fragmento de aplicación que interactúa con el usuario final es la lógica de presentación y esta a su vez interactúa con la lógica del negocio, el porque de que esta porción se instale en los clientes son:

- Pantallas de alta resolución, que prácticamente permiten "pintar" el monitor.
- Facilidad de los entornos basados en ventanas y el apoyo de mouse, track ball y plumas electrónicas.
- Las estaciones de trabajo tienen capacidades de manejar gráficos en 2-D y 3-D.

2.7.3 PROCESAMIENTO DISTRIBUIDO.

Mientras la lógica de presentación esta al frente del usuario final, es realmente la lógica del negocio y de la base de datos la que representa la esencia de la aplicación, pero donde situarla?, bueno existen 3 soluciones posibles:

1. Instalar la lógica de procesamiento de aplicaciones en el cliente.
2. Instalar la lógica de procesamiento de aplicaciones en el servidor.
3. Instalar la lógica de procesamiento de aplicaciones fragmentada, entre cliente y servidor.

Aunque puede parecer lógico cada solución tiene sus ventajas y desventajas.

2.7.4 INSTALANDO LA LÓGICA DE PROCESAMIENTO EN EL CLIENTE.

Ventajas.

- Disminución de trafico en la red.
- Price/performance.
- Terminal de I/O, entonces lo mas lógico es que los datos o procesamiento lógico se encuentren lo mas cerca de la fuente como sea posible.
- No hay necesidad de sincronización entre los fragmentos de procesamiento lógico de las aplicaciones, lo cual es necesario cuando estos están dispersos entre cliente y servidor.

Desventajas.

- Probablemente exista la necesidad de mantener copias de la misma lógica de negocios en cada cliente, lo cual dificulta la administración.
- Los beneficios de un modulo compartido no es posible.
- No se aprovecha el poder de los servidores como debe de ser, y tal vez el cliente sea sobrecargado lo que nos llevaría a aumentar el poder de los clientes.

2.7.5 INSTALANDO LA LÓGICA DE PROCESAMIENTO EN EL SERVIDOR.

Ventajas.

- Elimina código de redundancia, simplifica el mantenimiento del sistema, aprovechamiento del poder del servidor.
- Reduce trafico en la red.
- No hay necesidad de sincronización.

Desventajas.

- Sobrecarga del servidor (puede llevar a upgrade).
- Se puede afectar el tiempo de respuesta al aumentar el numero de interacciones y aumenta el trafico de red.
- El poder de los clientes se puede desperdiciar, reduciendo el retorno de la inversión.

2.7.6 INSTALANDO LA LÓGICA DE PROCESAMIENTO DE MANERA FRAGMENTADA.

Este método combina a los dos anteriores y aunque puede parecer la mejor opción, es también el mas complicado, hay que buscar el balance de la fragmentación, existe sincronía (las interacciones entre un cliente y servidor son sincronas).

2.7.7 CLIENTES.

ROLES Y FUNCIONES DE LOS CLIENTES.

Las funciones mas significantes desempeñadas por un cliente en un entorno cliente/servidor son las funciones de presentación y algunas de la lógica del negocio, la lógica de presentación es la capa que interactúa con los usuarios finales y con la lógica del negocio. existen varias formas para convertir las presentaciones basadas en caracteres a presentaciones que acepten iconos sin necesidad de cambiar las aplicaciones.

2.7.8 ADMINISTRACIÓN DE LA PRESENTACIÓN.

La administración de la presentación debe estar basada en estándares que lleven a los sistemas a interactuar con los usuarios finales en un estilo consistente que pueda ser portable a través de una amplia variedad de plataformas de hardware y sea capaz de interoperar con otras aplicaciones de sistemas abiertos.

Ha continuación se analizaran los 5 estilos de procesamiento cooperativo que se pueden implementar en un entorno cliente/servidor.

2.8 FUNCIONES DE LA LÓGICA DE PRESENTACIÓN.

En general, las funciones de presentación se desempeñan por el cliente, provocando la especialización del cliente en estas funciones, desde el punto de vista del procesamiento cooperativo, la manera en que las funciones de presentación son separadas del resto de las aplicaciones determina dos estilos de presentación cooperativa: Presentación distribuida y presentación remota.

2.8.1 Presentación distribuida.

Es cuando la parte de la presentación de la aplicación esta dividida entre dos o mas nodos de la red, es decir dos dispositivos (Cliente y servidor), consiste de componentes front-end y back-end, los componentes front-end manejan la parte fisica de la interface del usuario (despliegue de pantallas, GUI's, ventanas, colores, fuentes, teclado y mouse, por lo tanto se localiza en la interface del usuario que puede ser una PC, WS o terminal, reside en el nodo cliente.

El componente back-end se encuentra en un nodo diferente al de front-end, aunque cumplen con algo en común, las funciones de presentación, este se encuentra en el servidor, cooperando con el cliente.

2.8.2 Presentación Remota.

Es cuando la parte de presentación de una aplicación es instalado en un nodo mientras el resto de la aplicación se encuentra en otro, este tipo de presentación puede ser soportado por medio de remote procedure calls (RPC).

2.9 FUNCIONES DE LA LÓGICA DE APLICACIÓN.

El tener la lógica del negocio en un solo lugar puede parecer simple pero esto crea los sig. problemas, una solo localización puede crear cuellos de botella especialmente en un entorno de alto volumen de transacciones, además puede disminuir la confiabilidad del sistema al crear un solo punto de falla, se desperdician los beneficios de la computación distribuida y probablemente se exija upgrade del hardware donde reside la lógica, por estas razones se deben de distribuir en varios nodos.

2.9.1 FUNCIONES DISTRIBUIDAS.

Estas son complejas ya que requieren un gran numero de interacciones I/O, la lógica del negocio relacionada a las funciones de presentación usualmente residen en el nodo del usuario final o cliente, el procesamiento de negocios relacionado a la base de datos es normalmente instalado en el nodo que contiene la base de datos, de este modo se reducen un poco el intercambio de interacciones (incluyendo implementaciones de RPC y de comunicaciones programa a programa, aunque debemos de tener en cuenta que los usuarios también quieren mantener bases de datos en sus PC's (hojas de calculo, procesadores de palabras), para lo que es aconsejable tomar en cuenta la posibilidad de existencia de diferentes tipos de procesamiento y diferentes DBMS.

2.9.2 PROCESAMIENTO DE TRANSACCIONES.

Una transacción puede ser definida como una secuencia de acciones predefinidas, desempeñadas por una aplicación, para cumplir funciones de negocio.

La información afectada por la s transacciones es almacenada en un sistema que puede ser accesado y cambiado en tiempo real por la ejecución de estas transacciones, esta forma de trabajar se refiere a OLTP On Line Transaction Processing System,

2.10 FUNCIONES DE LA LÓGICA DE ADMINISTRACIÓN DE DATOS.

Permiten a los usuarios y aplicaciones a manejar almacenamiento y recuperación de datos de la empresa, la confiabilidad de esta información es vital para la planeación de estrategias y de adquisición de ventajas competitivas.

2.10.1 Arquitectura de la administración de datos y datos distribuidos.

Existen dos tipos de datos, los archivos (almacenamiento de registros y bloques de registros y las bases de datos (entidades operacionales administrativas).

Las funciones de la lógica de administración de datos están diseñadas para:

-Almacenar, recuperar, y actualizar grandes volúmenes de datos.
-Mantener la integridad y seguridad de datos almacenados en la base de datos.
-Proporcionar una interface consistente al usuario con una aplicación.
-Soportar múltiples usuarios y proporcionar un tiempo de respuesta razonable.

La arquitectura de un entorno distribuido cooperativo esta caracterizada por:

-Los datos deben ser distribuidos a través de varios sistemas (nodos).
-El procesamiento debe ser distribuido a través de varios nodos y administrado por un distributed transaction manager (control de la red, coordinación de procesos, sincronización y calendarización).
-El acceso a los datos distribuidos debe ser multicapas para soportar interfaces del usuario, controladores de I/O, diccionario de datos, directorio, catalogo.
-Utilización de DBMS y RDBMS.

2.10.2 ADMINISTRACIÓN REMOTA DE DATOS.

Consiste de dos componentes: Procesamiento lógico de los datos (parte del código que manipula datos dentro de la aplicación) y procesamiento de bases de datos (procesamiento lógico de la base de datos).

Arquitecturalmente hablando, la administración remota de datos es implementada vía un modelo de procesamiento front-end/back-end, donde el sistema front-end contiene la lógica del negocio y el sistema back-end contiene la base de datos y corre el DBMS.

2.10.3 ADMINISTRACIÓN DISTRIBUIDA DE DATOS.

Trata con las bases de datos distribuidas a través de múltiples nodos, esta distribución nos permite situar los datos cerca de su fuente, proporcionando alta disponibilidad de los datos por la existencia de múltiples copias de datos críticos en diferentes locaciones y elimina el potencial "punto de falla único". Es caracterizado por:

-Los datos y el DBMS son distribuidos a través de múltiples nodos, incluyendo el nodo con la aplicación lógica.
-Las funciones de administración de datos son distribuidas entre un sistema front-end (lógica de procesamiento de datos o DML) y otro back-end (base de datos o DBMS).

La principal ventaja de la distribución de datos es la habilidad para accesar datos localizados en muchos lugares de manera transparente, una base de datos distribuida propiamente instalada permite a cada nodo ser configurado de tal manera que este pueda manejar el monto de datos que residen en el nodo, la complejidad de las aplicaciones y el numero de usuarios.

2.10.4 Métodos de distribución de datos.

Ofrece los siguientes beneficios:

- Situar los datos cerca de su fuente.
- Alta disponibilidad de datos.
- Acceso a datos de manera mas eficiente.
- Balanceo de cargas de acceso a datos.
- Facilita el crecimiento de aplicaciones y usuarios.

Existen varios métodos para accesar datos secuencial, jerárquica, networking y relacional.

2.10.5 CLIENTE SERVIDOR ES UN NUEVO PARADIGMA.

Existen muchas cuestiones derivadas del cambio de paradigma que supone el modelo Cliente/Servidor. La mayoría de ellas, si no todas, son cruciales para el éxito de la implantación Cliente/Servidor, pero normalmente no se tratan de la forma adecuada si el departamento de Sistemas de Información no participa activamente en los proyectos Cliente/Servidor. En general, a los departamentos de usuarios y a la línea de negocio no les preocupa estos aspectos.

Si se quiere evitar la proliferación de distintos sistemas y plataformas de aplicación y la incompatibilidad entre los diversos desarrollos, el departamento informático debe asumir la responsabilidad corporativa en la selección de:

- Plataformas del sistema (por ejemplo, procesadores para los servidores y estaciones de trabajo para los usuarios, así como sistemas operativos soportados).

- Estándares, formatos y protocolos aplicables al hardware y software de sistemas y aplicaciones.

- Software para los componentes de Middleware de las Plataformas Operativas y herramientas de desarrollo de aplicaciones.

- Software de aplicación estándar disponible en el mercado.

No existen soluciones estándar Cliente/Servidor, aunque pueden existir bloques que se pueden reutilizar. Ni siquiera se puede asegurar, de forma general, que un modelo de distribución, una herramienta o una 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 y software y conocimientos.

El levantamiento de una infraestructura técnica que se aparte de las necesidades del negocio esta condenada al fracaso. Los factores que determinan la solución Cliente/Servidor son:

- Las necesidades del negocio y como las tecnologías informáticas pueden soportarlas para conseguir los objetivos comerciales.
- El marco de tiempo disponible para la nueva puesta a punto.
- El estudio económico.
- 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 aplicaciones ya 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, preferiblemente sin tener que modificarlas.

Para México, se prevé un gran crecimiento del Mercado Middleware, el aumento de los ambientes informáticos distribuidos arroja perspectivas muy optimistas para el mercado de productos middleware. Según la firma investigadora Ovum, este segmento tendrá un crecimiento de un 200 por ciento de aquí a finales de siglo hasta alcanzar un volumen de tres mil millones de dólares.

Si hasta hace poco el sector dominante era el de productos de conectividad de bases de datos, a partir de ahora el motor de este mercado será el middleware orientado a objetos y a ambientes distribuidos.

Al aumentar la demanda de sistemas de información flexibles e interfuncionales, es mayor la necesidad de productos que ofrezcan una interface sin fisuras entre los componentes de esos sistemas, es decir, se incrementa la necesidad de middleware.

Si tomamos este punto de partida, queda claro que el middleware se presenta como la clave para el proceso distribuido, según la consultora Ovum, el termino se ha visto rodeado en los últimos tiempos de una gran confusión. Por eso, es fundamental dejar claro que entendemos por middleware.

Ovum define middleware como un software de conectividad, de utilización inmediata, que soporta un proceso distribuido en run-time y es utilizado por los desarrolladores de aplicaciones para crear software distribuido.

2.10.6 ¿DONDE INTERVIENE?

Un análisis del mercado de este ambiente permite dividir el middleware en cinco categorías:

1) Productos de conectividad de bases de datos.
2) Middleware orientado a mensajes.
3) Middleware de ambiente de proceso distribuido.
4) Brokers de petición de objetos.
5) Monitores de proceso de transacciones distribuidas.

2.11 ESTANDARES Y SISTEMAS ABIERTOS.

Debemos enfatizar la diferencia entre una arquitectura y un producto, una arquitectura es una colección de definiciones, reglas y términos que son usados como guías para la construcción de un producto y un producto es una implementación especifica de un producto.

2.11.1 SISTEMAS ABIERTOS:

Vistos desde un punto de vista de TI, una empresa tiene diferentes tipos de software y hardware de diferentes proveedores que muchas veces no son compatibles entre sí, la megatendencia de hoy es de unir sistemas entre empresas y clientes, así los clientes necesitan extender sus negocios ligándose con sistemas de proveedores y distribuidores así como ellos a sus clientes unificándose en un todo coherente, para interconectarse estos sistemas deben tener arquitecturas abiertas para adoptar interfaces basadas en estándares.

Tomemos en cuenta que la capacidad de un sistema abierto no recae en el sistemas operativo, un sistema abierto debe de tener las siguientes características:

- Cumplir con estándares de la industria para programación, comunicaciones, redes, administración de sistemas, presentación, servicios del sistema e interfaces.
- Portabilidad de las aplicaciones a través de los sistemas.
- Escalabilidad de las aplicaciones.
- Interoperabilidad de los sistemas.

2.11.2 SISTEMAS PROPIETARIOS.

Existen organizaciones que juegan papeles claves en la arena de los sistemas abiertos, algunos de ellos son:

POSIX: Portable Operating System Interface for computer environment, apoyado por la IEEE, para proveer consistencia entre entornos operativos diferentes.

X/Open: Una organización no lucrativa fundada en 1984 para solucionar problemas causados por software e incompatibilidad de los sistemas , juega un papel importante en la definición de un entorno común de aplicaciones (CAE por sus siglas en ingles), el cual se contiene en una guía llamada XPG3 (para la tercera liberación de esta guía).

International Organization for Standardization (ISO): Tiene como meta desarrollar, acelerar y promover varios estándares y los productos que implementan estos a través de su modelo de referencia Open Systems Interconnection (OSI).

Corporation for Open Systems (COS): Concentra sus esfuerzos en el área de OSI, ISDN, estándares de prueba y certificaciones.

Object Management Group (OMG): Es una organización internacional envuelta en desarrollo de sistemas y software orientados a objetos.

Open Software Foundation (OSF) y Unix International (UI): Son los dos mas grandes proveedores de tecnología para los entornos de sistemas abiertos.

2.11.3 BENEFICIARIOS DE LOS SISTEMAS ABIERTOS.

Como podemos definir un ideal de los sistemas abiertos:

- La computación ocurre transparentemente a través de la red.
- Aplicaciones, recursos, funcionalidad y poder pueden ser compartidos a través del entorno.
- Los usuarios pueden ser provistos de grandes posibilidades de portabilidad, interoperabilidad y escalabilidad de las aplicaciones.

Los beneficiados son:

Proveedores de hardware y software: Disminuyendo el costo de desarrollo de sus productos, así como el tiempo de desarrollo de los mismos, disminuye el gasto de tiempo y esfuerzo en la creación de nuevos productos propietarios, los sistemas abiertos no eliminan la competencia sino que solo la transforma, para ver quienes agregan valor a sus productos, logrando especialización en productos y plataformas de hardware, calidad, servicios y optimización de soluciones propietarias ya existentes.

Usuarios finales: Asegurar la funcionalidad de sus sistemas sin importar el proveedor, la operabilidad y conectividad de sus redes puede ser simplificada.

2.11.4 MÉTODO PARA DISTRIBUCIÓN.

Hoy los requerimientos de cliente/servidor incluyen la libertad para almacenar datos y correr aplicaciones en una variedad de plataformas interconectadas, las implementaciones de cliente/servidor deben estar disponibles en un entorno abierto, flexible y que soporte multi-proveedores donde varias tareas puedan ser corridas en paralelo, proporcionando un alto desempeño y mejor utilización de los recursos, una arquitectura cliente/servidor debe ser construida así y representa un caso especial de la computación distribuida- un procesamiento distribuido cooperativo.

El siguiente paso critico es el de los modelos de entorno distribuidos.

2.11.5 MODELOS DISTRIBUIDOS.

El desarrollo de sistemas distribuidos ha sido afectado por dos fuerzas opuestas.

La primera que afecta directamente a los usuarios finales en relación a las aplicaciones, el desarrollo de aplicaciones en estaciones de trabajo y PC's proporciona significantes ganancias precio/desempeño, la otra es que los usuarios finales demandan autonomía local y funcionalidad adicional (como una GUI flexible y consistente) que incremente la productividad del usuario.

La segunda fuerza, tiene sus raíces en la necesidad del usuario final de accesar los datos corporativos, afectando la disponibilidad, desempeño e integridad.

Estos requerimientos no son posibles de satisfacer en sistemas pequeños, estas razones complican los modelos distribuidos.

2.11.6 ENTORNO MULTILAZO (Multitiered environment).

Las organizaciones mas grandes en los EUA han empezado a moverse a una arquitectura conocida como "tres lazos" (three-tiered), instalando en el primer lazo al mainframe, en el segundo lazo al servidor y en el tercer lazo a las estaciones de trabajo (ver figura).

Por supuesto esta arquitectura puede ser expandida horizontalmente adicionando mainframes al primer lazo, para administrar la red en este tipo de arquitectura necesitamos un enlace de comunicación entre los tres "lazos" o capas haciéndola mas difícil de construir y administrar, sin embargo el resultado el desempeño se ve aumentado.

El modelo multitiered parece ser razonablemente flexible de implementar, sin embargo, existen unas preguntas que debemos contestar como son:

¿ En que lazo o lazos deben estar situados los datos?
¿ En que lazo o lazos debe ser situada la lógica de aplicaciones?
¿ En que lazo o lazos debe estar situada la interface del usuario?

Estas preguntas reflejan las propiedades intrínsecas de la arquitectura cliente/servidor.

Procesamiento de la lógica de aplicación: Es una parte del código de aplicación que facilita la interacción del usuario con un equipo tal como PC o estación de trabajo y desempeña tareas como formateo de pantalla, lectura y escritura de información en la pantalla, etc. se refiere básicamente a las interfaces gráficas del usuario o GUI's.

Procesamiento de la lógica del negocio: Parte del código de aplicación que usa las entradas de los datos para desempeñar tareas de negocios (3GL, 4GL y objetos).

Procesamiento de la lógica de la base de datos: Parte del código de aplicación que manipula datos dentro de las aplicaciones, usando Data Base Management System (DBMS) en SQL (Lenguaje estructurado de búsqueda) y lenguaje de manipulación de datos (DML) usualmente desarrollados en 3 y 4GL.

Procesamiento de base de datos: DBMS pero es importante tomarlo en cuenta desde el punto de vista arquitectural.

Combinando todos estos recursos en un entorno distribuido podemos alcanzar grandes costo/beneficio.

Aclarando este punto regresemos a nuestra pregunta clave ¿Que recursos deben ser distribuidos y cuales son consecuencia de tal distribución?

La arquitectura cliente/servidor emplea procesamiento corporativo distribuido para:

- Aplicaciones en los clientes típicamente la presentación y algunas partes de la lógica del negocios son instaladas en los clientes y algunas partes de la lógica del negocio, la lógica de la base de datos y DBMS en los servidores.
- Organiza las interacciones entre clientes y servidores de manera cooperativa, tratando de equilibrar la capacidades del sistema.

Un diseñador debe tratar de encontrar el equilibrio, para esto se pueden hacer algunas recomendaciones:

- En general la lógica de presentación es puesta en los clientes y estos puestos en el tercer lazo, así como un poco de la lógica del negocio, por lo menos tareas tales como edición de pantalla y aquellas piezas de código que sean de un cliente en particular.

- Si el procesamiento lógico de la base de datos esta incrustada dentro de la lógica del negocio y si los cliente mantienen una baja interacción entre los datos, entonces también puede ser puesta dentro de los clientes.

- Tomando en cuenta que se trabaja en grupo y que todos comparten una base de datos la lógica de la base de datos y los DBMS deben ser instalados en el servidor.

Para sistemas mas complejos la situación de los componentes puede ser decidida por:

- La cantidad de datos relevantes que puede usar alguna aplicación.
- El numero de usuarios activos corriendo aplicaciones alrededor de estos datos.
- El numero de interacciones entre varios componentes de la aplicación.
-Las características técnicas de las plataformas seleccionadas para clientes y servidores.

CAPITULO III. MIDDLEWARE

En este capítulo se explicará el concepto middleware, la importancia y necesidad de funciones de este tipo y donde se encuentra dentro de las partes del entorno distribuido estudiando este concepto en su estructura y arquitectura teórica, se describirán las asociaciones middleware que existen y algunos productos middleware que existen en el mercado.

3.1 DEFINICIONES DE MIDDLEWARE.

Normalmente la definición es imprecisa para los usuarios, esto significa que existen varios significados en operación.

La definición mas simplificada que se puede encontrar es: Middleware es visto como un facilitador para que la gente trabaje en equipo en un ambiente de oficina.

La industria se inclina por definir Middleware como: Una colección de herramientas que están disponibles a través de la red para permitir a los clientes obtener o entregar tres funciones clave, como son:

- Compartir información en tiempo real.
- Colaborar en la administración, edición y cambio de la información.
- Diseminar información a la gente apropiada (un entorno controlado).

Uno de los valores mas notables del middleware, en la actualidad, donde se han notado las capacidades middleware es en su amplia demanda para llevar a cabo Reingenieria de Procesos.

Middleware se ha implementado notablemente para extenderse organizacional y geográficamente proporcionando ventajas competitivas.

Hoy todas las piezas de Middleware están ahí. Pero la integración del Middleware todavía presenta muchos problemas.

3.1.1 ¿QUE ES MIDDLEWARE?

Una capa de software que se sitúa debajo de la capa de aplicaciones y por encima de el sistema operativo de las computadoras. Middleware interconecta a los dos proporcionando estándares para servicios y funciones.

Esto significa que el software de aplicación puede ser escrito sin importar el sistema operativo en el cual esta corriendo, esto ahorra tiempo para los desarrolladores de aplicaciones y proporciona gran flexibilidad para distribuir la misma aplicación a través de diferentes sistemas.

El Middleware consiste de herramientas de software que enriquecen la productividad y permiten a los protocolos manifestarse como librerías de software, necesarias para introducir la descentralización de la infraestructura de TI, la utilización de Middleware es útil para comunicar equipos de diferentes proveedores y distintos sistemas con un enfoque a sistemas abiertos o cliente/servidor.

Los negocios de hoy enfrentan grandes retos: la competencia es global, las industrias y los productos que se desarrollan para satisfacer necesidades deben volverse mas flexibles para responder y localizar nuevos requerimientos, en cuestión de meses México tendrá un cambio importante en sus telecomunicaciones donde varias compañías ofrecerán distintos servicios, las compañías deben de estar en estrecho contacto con sus clientes y proveedores, además existe una continua y enorme presión para trabajar rápida y eficazmente con pocos recursos y alta calidad en los resultados.

Mas y mas la supervivencia y el éxito en estas condiciones demandan el uso de estrategias de tecnología de información para asegurar una ventaja competitiva.

Una de las mas marcadas tendencias en la tecnología de información de hoy es la emergencia de trabajar con redes como un gran sistema distribuido. Como las PC's lo fueron para los 80's, trabajar con multiplataformas corporativas de redes lo es para los 90's. Un numero de factores contribuyen a esta tendencia, los cuales son:

- Las oportunidades para mejorar el precio/desempeño del hardware.
- El vasto porcentaje de usuarios de información no-tecnicos y la necesidad de accesar datos rápida y fácilmente.
- La necesidad de datos en sistemas heterogéneos para ser distribuida ampliamente a través de la empresa.
- Los ineludibles requerimientos de un mercado global y distribuido.
- Decrementar los ciclos de desarrollo de aplicaciones haciéndolos rápidos y flexibles.
- La aceleración de un movimiento hacia los negocios descentralizados y autónomos.

El mercado de redes ha ido evolucionando y progresando en desarrollar y entregar soluciones interoperables a protocolos de bajo nivel. Ahora es muy factible enlazar redes de diferentes tipos y equipo de diferentes proveedores.

Desafortunadamente, existen problemas de interconectividad al nivel de las capas de servicios de red y de aplicaciones. Algunos de los grandes problemas que han retardado la tendencia de ir hacia los sistemas distribuidos son:

- La complejidad de desarrollar aplicaciones distribuidas combinadas con la falta de conocimientos y habilidades para escribir estas aplicaciones, especialmente sobre múltiples o diferentes plataformas.

- La dificultad en soportar y mantener tales aplicaciones.

- Estandarizar los diferentes tipos de datos y formatos de mensaje entre sistemas heterogéneos.

- La falta de conocimientos y habilidades estimados para cada sistema operativo, protocolos de red y plataformas de hardware.

La dificultad en transformar grandes montos de aplicaciones de mision-critica de mainframes dentro de aplicaciones distribuidas Cliente/Servidor.

3.2 IMPORTANCIA DE MIDDLEWARE.

Para lograr este tipo de adaptabilidad y respuesta, los administradores de Sistemas de Información necesitan tener la libertad de compra de software que vaya de acuerdo a la infraestructura tecnológica de la organización.

La infraestructura de TI típicamente consiste de protocolos de comunicación heterogéneos, sistemas operativos, herramientas de desarrollo, bases de datos y múltiples plataformas de ejecución.

Para llevar a cabo este tipo de libertad de interorperabilidad es necesaria una capa de software la cual no requiera a los desarrolladores de aplicaciones corporativas aprender teoría de comunicaciones. acceso a bases de datos, formato de datos, estructura de mensajes y media docena de entornos operativos diferentes. En efecto, se necesita una nueva capa "middle" aislando a los desarrolladores de software de tratar con intrincados detalles de los protocolos de comunicación, diferencias entre aplicaciones, sistemas operativos y plataformas de hardware. La existencia de tales capas de software es obligatoria para llevar a cabo la independencia de software y de hardware, el cual es el mayor requerimiento de los sistemas abiertos, esta capa de software es llamada Middleware.

3.2.1 MIDDLEWARE.

3.2.1.1 NECESIDAD DE FUNCIONES MIDDLEWARE.

Así como los Servicios de aplicación aíslan a estas del entorno operativo y de los detalles de la configuración, el Middleware separa a las aplicaciones de los aspectos de distribución. Se trata de un software que proporciona un conjunto de servicios para conseguir transparencia de ubicación de los recursos distribuidos. Con ello se libera al programador de las complejidades de las comunicaciones, de construir y mantener un directorio, así como de desarrollar funciones de seguridad para controlar el acceso a las funciones servidoras.

A través de los servicios de Middleware, los recursos de la red pueden ser sucedidos como si fueran locales. Las aplicaciones solicitan servicios a través de un API; el Middleware redirecciona la solicitud a la ubicación física del gestor del recurso, que puede estar en el mismo procesador o en otro distinto de la red.

A fin de conseguir transparencia local/remota, los productos Middleware disponen de los siguientes servicios:

1. Directorios y nominación.
2. Comunicaciones.
3. Seguridad.

4. Gestor de transacciones.

Normalmente, no existe un único producto que proporcione todas estas funciones. Se consiguen mediante la combinación de diferentes componentes de software que se complementan entre si. Cada vez con mayor frecuencia hay programas de distintos suministradores que participan en una solución. Muchos de ellos tienen un desarrollo exclusivo o propietario de estas funciones. Por razones históricas, se produce un solapamiento y duplicación que dificulta el mantenimiento del sistema. Generalmente se espera que este nivel de funcionalidad lo proporcionen los proveedores del sistema, aunque a veces la carencia de determinadas funciones obliga a los departamentos informaticos a buscar soluciones tácticas.

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

Mantener varios directorios con diferentes formatos, implica costos elevados en la gestión del sistema distribuido, y puede resultar muy difícil de operar, especialmente en un entorno de múltiples proveedores.

Comunicaciones. Estos servicios son necesarios entre la parte cliente y la servidora del Gestor de Recursos Remoto o Distribuido. Los tres modelos siguientes están aceptados por la industria:

- Conversacional.
- Llamada a procedimiento remoto (RPC).
- Cola de Mensajes.

Como estos modelos tienen atributos distintos, deberán utilizarse en base a las necesidades especificas del sistema y de la aplicación.

Los productos Middleware seleccionados para la Plataforma Operativa que proporcionan solo un subconjunto de los tres modelos, no deber ser mutuamente exclusivos. Además, la flexibilidad y la portabilidad aumentan cuando estos productos disponen de servicios de comunicación independientes de la red, en lugar de utilizar directamente el protocolo de transporte. 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.

Tener mas de un sistema de seguridad en molesto tanto para el usuario, que tiene que conectarse a varios sistemas con distintos identificadores y contraseñas, como para las personas responsables de mantener los directorios de seguridad.

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

Es probable que en un entorno distribuido haya que sincronizar los recursos gestionados por los distintos gestores. Esto solo puede conseguirse teniendo un único gestor de transacciones, y no con cada gestor incorporando sus propias funciones. Uno de los objetivos clave de Open Blueprint es superar estos problemas potenciales seleccionando API's, formatos y protocolos para estas funciones relacionadas con Middleware. Cuando los productos Middleware utilicen con el tiempo estos estándares, podrán ejecutarse e interoperar conjuntamente en la misma o en distintas plataformas, de una forma coherente. Un criterio básico de selección de productos Middleware es el grado en que utilizan, o tienen previsto utilizar, los estándares que se describen en el Open Blueprint. Lo mas importante es verificar hasta que punto se utilizan los Servicios de Sistemas Distribuidos según se definen en Open Blue print, en lugar de los equivalente funcionales proporcionados por los productos de forma propietaria.

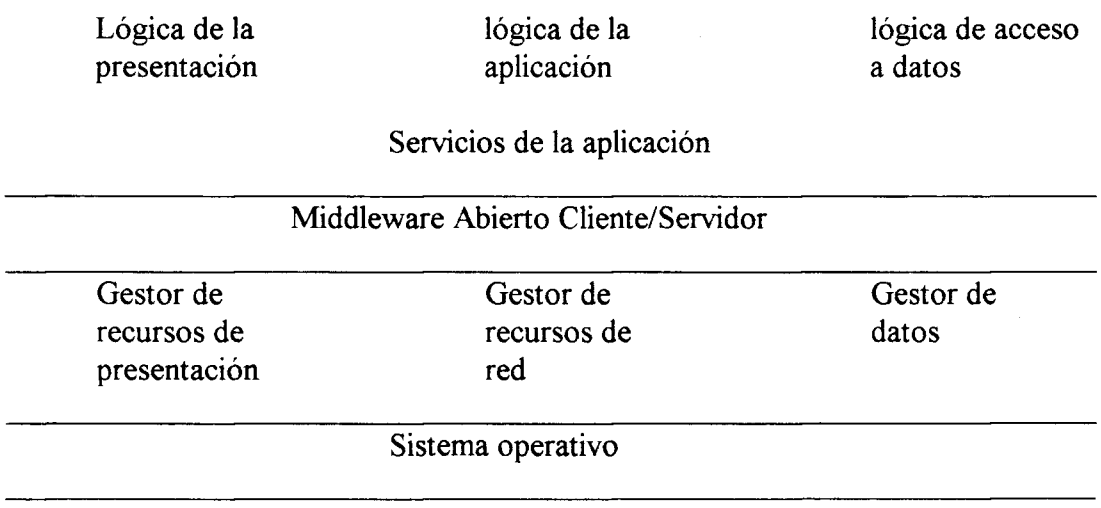


Figura 3.1. Partes del Entorno Distribuido.

3.3 ESTRUCTURA O ARQUITECTURA TEORICA.

3.3.1 Oferta de Middleware Modelos propuestos.

Una serie de desarrollos tecnológicos y sociológicos, nos llevan a redefinir las corporaciones y los procesos organizacionales para operar en un mundo abierto y heterogéneo, entre ellos están:

- El surgimiento de las computadoras personales.
- La revolución Cliente/Servidor.
- La creciente importancia del UNIX.
- El énfasis en los sistemas abiertos.
- La llegada de la tecnología orientada a objetos. (un nuevo paradigma de la TI, el paradigma procedural debe coexistir con una migración discreta a la tecnología orientada a objetos.
- La necesidad de colaboración (Groupware).
- El acceso sencillo a la supercarretera de información.

3.3.2 Requerimientos para la Tecnología de Información.

3.3.2.1 Acceso Global Transparente.

Los usuarios necesitan accesar datos y aplicaciones donde quiera que estas residan. Este acceso debe ser transparente para que los usuarios no encuentren barreras o limites fisicos (Single System Image).

3.3.2.2 Interoperabilidad.

Los productos de todos los proveedores deben ser capaces de trabajar juntos, es decir, el uso de interfaces, protocolos y formatos estándar.

3.3.2.3 Flexibilidad de las plataformas.

Los usuarios deben tener acceso a todas las funciones sin ser restringidos por un solo hardware o plataforma de software.

3.3.2.4 Desarrollo de aplicaciones rápido y eficiente.

Los desarrolladores necesitan herramientas para crear aplicaciones distribuidas cliente/servidor. Estas herramientas deben ocultar la del entorno de la red. Permitiendo que los desarrolladores puedan:

- Diseñar aplicaciones que permitan la distribución de funciones.

- Construir aplicaciones que se puedan ejecutar en múltiples plataformas.
- Probar y mantener aplicaciones en un entorno distribuido.

Los desarrolladores necesitan herramientas que permitan crear componentes de software reusable y ser capaces de responder a requerimientos de cambio y creación de prototipos rápido.

3.3.2.5 Manejabilidad.

Los administradores de sistemas requieren herramientas y automatización para simplificar las complejidades de el entorno, la manejabilidad debe ser construida dentro del entorno, además, la flexibilidad y usabilidad de las herramientas de administración deben ser mejoradas continuamente.

3.3.2.6 Protección de la Inversión.

Las inversiones en tecnología son bastante fuertes, ninguna compañía puede ignorar estas inversiones y empezar de nuevo.

3.3.3 Oferta de Middleware.

3.3.3.1 ¿que es Message Oriented Middleware?

Message Oriented Middleware es una capa que proporciona software la cual soporta múltiples protocolos de comunicación, múltiples lenguajes de programación y múltiple ejecución de plataformas (de hardware y de software). Esta reside entre las aplicaciones del negocio y la infraestructura de red de múltiples protocolos de comunicación o entre las aplicaciones por si mismas, dependiendo de la implementación.

Message Oriented Middleware se refiere al proceso de distribuir y controlar datos a través del intercambio de registros conocidos como mensajes. Es decir, comunicación process-to-process en un entorno distribuido proporcionando modelos para la transmisión de mensajes que no proporcionan otros entornos tales como DCE.

La característica clave de un Modelo de Mensajes es que este natural y únicamente, soporta ambos tipos de comunicación (Sincrona y Asincrona) y permite el manejo de eventos, mas que el procesamiento procedural. Además, los productos basados en el Modelo de Mensajes proporciona portabilidad, escalabilidad y otras importantes características.

La comunicación asincrona permite enviar y recibir procesos para ser activados en diferentes momentos. Frecuentemente, las implementaciones de comunicación asincrona implican el uso de queues donde los mensajes pueden ser almacenados para procesamiento subsecuente por parte del sistema de mensajes.

La comunicación asincrona es siempre sin bloqueos: después de enviar un requerimiento, o un mensaje el mensajero es capaz de continuar un trabajo mientras el receptor esta formulando una respuesta.

El mensajero no tiene que dejar de trabajar y esperar, se debe de resaltar que la comunicación asincrona sin bloqueo puede también ser implementada sin queues permitiendo simplemente el envío de procesos. Generalmente la respuesta subsecuente puede ser recibida vía una notificación automática usando funciones de callback o por polling de mensajes hasta que exista una respuesta.

La función de callback proporciona un mecanismo para activar una función automáticamente después de que la ejecución de una función especifica ha sido completada. Por ejemplo, el programa mensajero puede especificar una función que puede recibir el control inmediatamente después de que un mensaje enviado ha sido completado.

La comunicación sincrona es un modelo de comunicación process-to-process. Esto indica que ambos procesos (mensajero y receptor) son activados al mismo tiempo y también mantienen una sesión activa entre ellos. Fundamentalmente, la comunicación sincrona es un modelo de bloqueo (el proceso de envío es bloqueado), llamado Syncronous lockout, hasta que el receptor completa el proceso del requerimiento y devuelve los resultados. El control va con la función de proceso activa.

Para usar modelos de comunicación MOM's asincrono o sincrono, la información puede ser transmitida sobre diferentes medios de comunicación, como líneas telefónicas asincronas dial-up o líneas sincronas de datos de alta velocidad.

Para usar MOM's es necesario separar las aplicaciones distribuidas o componentes de la misma aplicación distribuida en mensajes. Los mensajes portan requerimientos y aplicaciones de datos. MOM ha sido usado exitosamente por muchos corporaciones del Fortune 500 implementando mision-critica, high transaction, aplicaciones distribuidas y enterprise-scale en entornos heterogéneos.

La anchura de las capacidades de la comunicación asincrona tales como soportar message quening mesaage passing, combinadas con capacidades de callback y manejo de eventos, hace de message oriented middleware algo muy atractivo para los desarrolladores bajo entornos cliente/servidor. Para proporcionar estas capacidades asincronas, MOM explota totalmente el ineherente paralelismo de la red. En los últimos dos años muchas corporaciones han optado por soluciones bajo MOM (Message Oriented Middleware).

La flexibilidad de las aplicaciones y sistemas de software es uno de los mas importantes aspectos que enfrentan muchos Gerentes de Sistemas que hacen decisiones tecnológicas hoy. El hecho de que MOM sea la única tecnología de middleware que soporta múltiples tipos de comunicación lo convierte en el mas robusto, funcional y flexible middleware que se encuentre hoy, cada persona que tome decisiones de tecnología necesita

considerar esta tecnología como un componente importante de su infraestructura de TI, como una solución táctica para desarrollar y soportar aplicaciones distribuidas.

Un buen ejemplo de esto es la industria del cuidado de la salud, un gran proveedor encontró que debía comunicarse con un amplio rango de partes envueltas para ayudar a disminuir los costos. La compañía no puede gastar el tiempo y el dinero en constantes evaluaciones de tecnologías emergentes para computación distribuida. Hay una inmediata necesidad de enlazar múltiples sistemas heterogéneos que puedan soportar millones de demandas cada mes. para solucionar este problema la compañía cambio su infraestructura de TI hacia message oriented middleware.

3.4 ASOCIACIONES MIDDLEWARE.

En la búsqueda de resolver este problema, se han formado diferentes grupos que tratan de definir estándares, o proponer soluciones (recordemos que una solución se puede convertir en un estándar), estos grupos son:

3.4.1 MOMA (Message Oriented Middleware Asociation).

Este grupo es radicalmente orientado a los objetos, manejar los problemas de interconexión por medio de MOM (Message Oriented Middleware).

MOMA es una asociación mundial de usuarios, proveedores y consultores dedicados a enriquecer la interoperabilidad de la computación distribuida Cliente/Servidor vía message oriented middleware. La asociación a tomado un rol de liderazgo, educando a sus miembros y usuarios de la industria acerca de los beneficios de message oriented middleware.

Entre sus miembros se encuentran:

Applied Communications, Inc.
AT&T
Computer Associates International, Inc.
Digital Equipment Corporation.
IBM UK Labs, Ltd.
Motorola, Inc.
Novell, Inc.
SunSoft, Inc.
Yankee Group.

3.4.2 X/Open:

X/Open es una organización no lucrativa fundada en 1984 para resolver problemas causados por la incompatibilidad de software y sistemas. X/Open juega un rol muy importante en definir un entorno común de aplicaciones (CAE, Common Application Environment).

3.4.3 Object Management Group (OMG).

Es una organización internacional envuelta en el desarrollo de sistemas y software en la estructura de la tecnología orientada a objetos. Esta organización dispone de mas de cien compañías involucradas en el desarrollo de sistemas y software así como usuarios de software. El objetivo del OMG es desarrollar un marco común para aplicaciones orientadas a objetos basado en normas de la industria. La adopción de este marco hará posible

desarrollar un entorno de aplicaciones heterogéneo que acoja las principales marcas de hardware y los sistemas operativos mas difundidos.

OMG a desarrollado CORBA (Common Object Request Broker Architecture), una especificación de mensajes basada en objetos.

3.4.4 Open Software Foundation (OSF) y UNIX International (UI).

Que son los dos mas grandes proveedores para el entorno de sistemas abiertos. Es una organización de investigación y desarrollo sin fines lucrativos cuyo objetivo es proporcionar una solución de software que permita que las computadoras de distintos proveedores trabajen conjuntamente en un verdadero entorno de sistemas abiertos. OSF utiliza un proceso abierto para seleccionar e implementar tecnología (DCE), fue fundada en 1988.

Algunos de sus miembros son:

Hewlett-Packard
IBM
Digital Equipment Corporation
Microsoft Corporation
Locus
Apollo
Siemens
Transarc.

3.5 PRINCIPIOS DE MIDDLEWARE.

Middleware se define como una colección de servicios de computación distribuida que, habilitan a los clientes y servidores para comunicarse e interorperar unos con otros, de manera confiable, correcta y flexible.

3.5.1 CLASES DE MIDDLEWARE.

Es muy difícil una clara y distinguida categorización del middleware, esto es, que existen varias clasificaciones diferentes, las cuales pueden ser usadas para dividir diferentes tecnologías middleware dentro de grupos separados y estas clasificaciones se overlapan una con otra.

3.5.1.1 BASES DE DATOS.

La primera área en la cual middleware fue ampliamente usado fue en la administración de las bases de datos. Dentro de la administración de bases de datos, la primera función de middleware fue la de proporcionar algún grado de abstracción de la administración local de datos a las aplicaciones cliente con dos propósitos.

1. Gateways para otros productos DBMS- gateway-oriented middleware es usado para permitir a una aplicación que normalmente corre en sistema propietario a accesar datos contenidos en una base de datos distinta, por ejemplo, por ejemplo una base de datos Sybase accesando datos de ORACLE.

2. Acceso a Multiple-DBMS- Esta clase de base de datos middleware proporciona acceso estandarizado para múltiples productos DBMS.

La siguiente figura ilustra los dos tipos de middleware de base de datos.

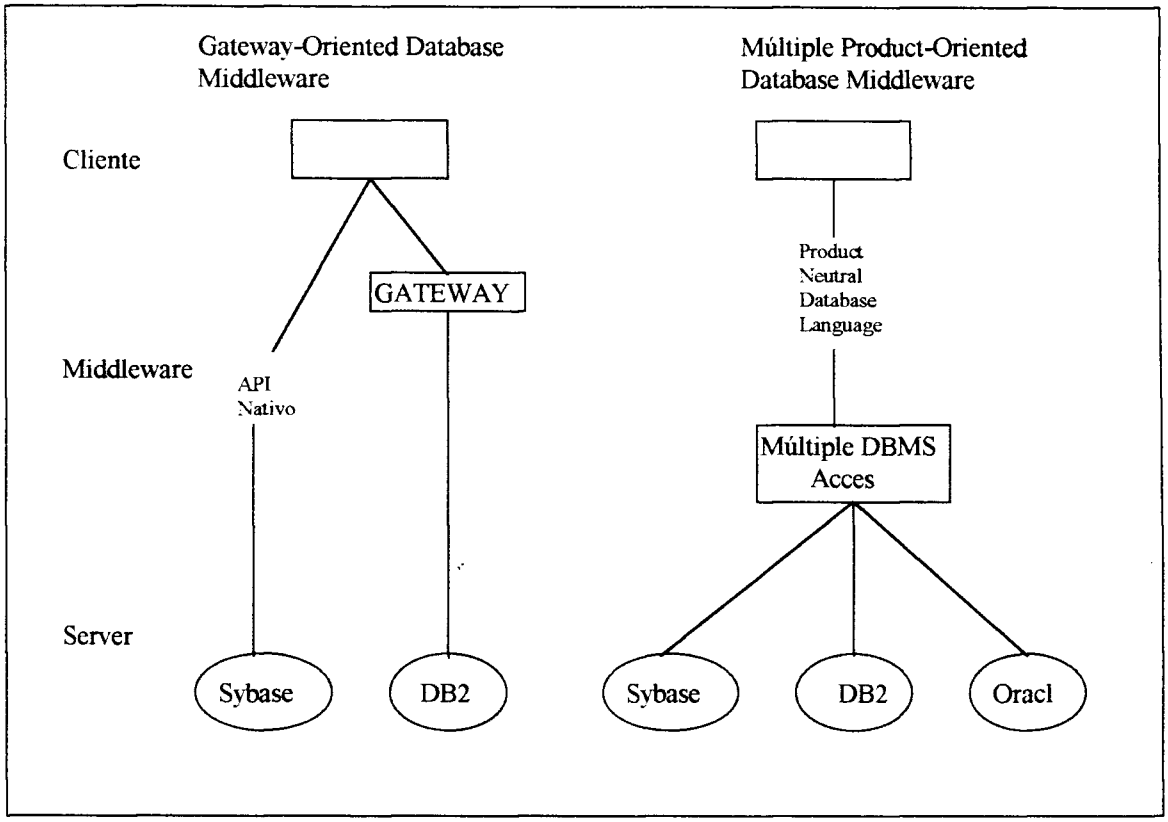


Figura 3.2.- Tipos de Middleware de Base de Datos.

3.5.1.2 PEER-to-PEER.

Otro tipo de middleware puede se nombrado peer-to-peer middleware. el cual mas que envolver aplicaciones cliente y servidores de bases de datos se usa para:

- Los clientes se comunican entre ellos mismos (aplicaciones corriendo entre clientes).
- Servidores comunicándose entre ellos (procedimientos corriendo en un servidor de bases de datos requieren datos de otro servidor).
- Clientes o servidores comunicándose con algún servicio middleware (requerimientos del directorio de información y aplicaciones de correo electrónico).

3.5.1.3 OBJECT-ORIENTED.

Otra categorización es la de middleware orientado a los objetos y middleware no orientado a los objetos. Los detalles de esta categorización ya han sido tratados con anterioridad.

3.6 CLASIFICACIONES DE MIDDLEWARE.

La figura 3.2 ilustra un recurso de como el middleware puede ser categorizado, en las publicaciones de middleware es común hacer distinciones entre diferentes clases de middleware, pero rara vez se continua usando la misma taxonomía, a pesar de eso las líneas de demarcación tienden a ser diferentes de un estudio a otro tales como bases de datos contra no-bases de datos (peer-to-peer), o RPC contra Messaging contra Orientación a objetos, el siguiente diagrama como cualquier producto middleware puede ser categorizado en más de una forma, en la siguiente sección se distinguen varios usos entre diferentes tipos de middleware.

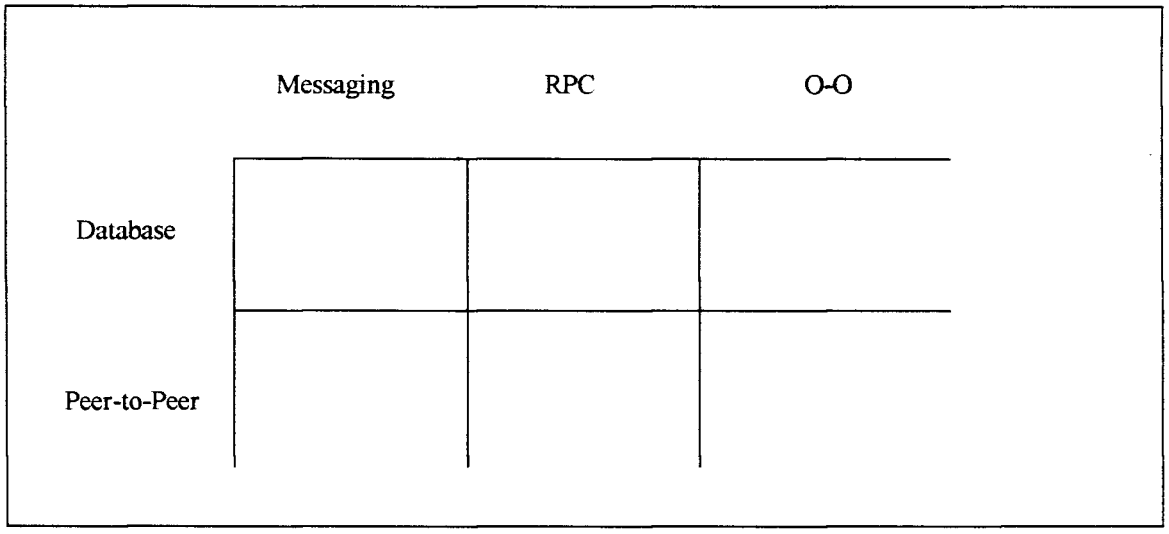


Figura 3.3.- Una taxonomía de clases de middleware.

3.6.1 DATABASE MIDDLEWARE

3.6.1.1 RAICES.

Dos grandes tendencias de la tecnología de los 80's- el desarrollo de las PC's y el uso de minicomputadoras basadas en computación departamental, trajeron consigo varios problemas con respecto a la organización de los datos. Específicamente, las aplicaciones corrían en una plataforma que necesitaba accesar a datos corriendo en otra plataforma, usualmente con hardware y sistemas operativos diferentes.

Se esperó que eventualmente el advenimiento de la tecnología de bases de datos distribuida podría proporcionar la solución a los problemas de acceso, pero existen un numero de razones que por las cuales esta no ha sido la solución idónea tales como tecnología inmadura, problemas de desempeño y complejidad en la administración para

tratar de evitar estos problemas los productos actuales proporcionan algún grado de acceso heterogéneo a datos.

Conforme los 80's fueron avanzando, un gran numero de tendencias externas, especialmente aquellas relacionadas a las fusiones y adquisiciones de las corporaciones, llevaron a la necesidad de servicios middleware para el acceso a las bases de datos, esto causo la expansión del uso del middleware y el crecimiento en el mercado de estos productos.

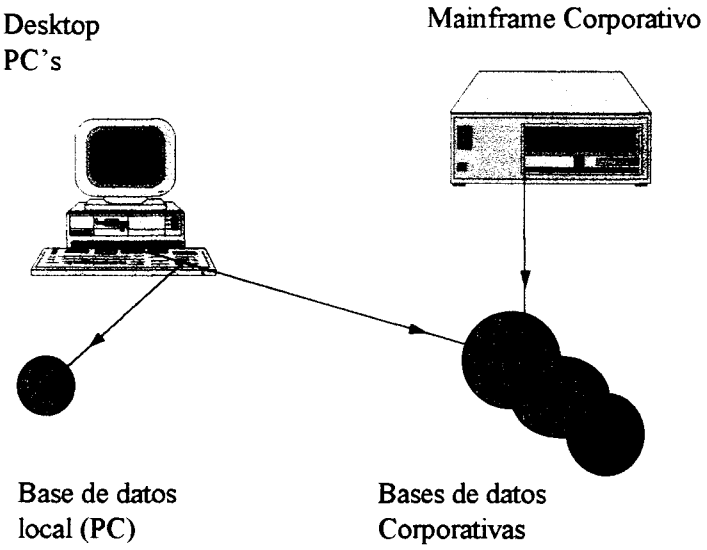


Figura 3.4.- Acceso a datos entre distintas plataformas.

El acceso a bases de datos heterogéneas se volvió una necesidad, las facilidades de compartirlas varían de un producto a otro, el esfuerzo por estandarizar llevo al desarrollo de SQL y Open Database Connectivity (ODBC).

Sin embargo, el acceso a datos por si solo no es suficiente para muchas aplicaciones, existe la necesidad de muchos servicios adicionales que se necesitan una de estas áreas de servicio es la replicación de las bases de datos, la duplicación controlada de algunas porciones de información a través de múltiples plataformas fisicas. Irónicamente, una de las principales razones de la computación distribuida es la de eliminar tanta duplicación de datos como sea posible por medio de Distributed Database Management System (DDBMS), sin embargo la duplicación resulta muy necesaria en algunos casos de tal modo que existen productos middleware con un alto poder de replicación, la replicación middleware empezó a aparecer a mediados de los 90's, es importante resaltar que una variedad de diferentes aplicaciones están disponibles y la efectividad de cada uno puede variar de acuerdo a factores tales como el uso destinado, por ejemplo, algunos modelos de replicación son mas apropiados para Sistemas de Soporte a las Decisiones (DSS) y para Sistemas de Información

para Ejecutivos (EIS), por sus capacidades de almacenar "Data warehouses", o bases de datos informacionales (en lugar de transaccionales u operacionales.

Las funciones necesarias para el soporte de replicación deben ser localizadas y administradas por software middleware.

3.6.1.2 TRANSPARENCIA DE LOCALIZACIÓN.

Otra clase de middleware orientado a las bases de datos es el que es usado para proveer un grado de transparencia, los orígenes de la transparencia están en los DDBMS, y su principal función es la de tener un middleware que proporcione un directorio de datos que pueda ser usado para búsquedas dinámicas y determinación de localización de la información.

3.6.1.3 TENDENCIAS FUTURAS.

El área del middleware esta desarrollándose rápidamente por lo tanto es importante tener en cuenta que los detalles técnicos se pueden volver obsoletos muy pronto, pero existen algunos puntos muy importantes en los cuales es imprescindible poner atención a su evolución.

Middleware de bases de datos: Replicación y Data warehousing continúan siendo las áreas de crecimiento más grandes y es de esperarse que al final de los 90's esta área pueda lograr un crecimiento maduro, para soportar estos métodos de computación distribuida es necesario que productos de middleware más robustos se encuentren en el mercado.

Tecnología RPC: La convergencia de las tecnologías MOM y RPC pueden afectar esta área ampliamente, además de Distributed Computing Environment (DCE), que podria desarrollarse a tener más allá de pequeñas aplicaciones departamentales.

Message-Oriented Middleware: La tecnología MOM puede volverse el componente de interoperabilidad predominante en el entorno (oponiéndose al mundo de aplicaciones procedurales en que vivimos).

Object-Oriented Middleware: El principal desarrollador de esta tecnología es CORBA quien proporciona estándares orientados a objetos para lograr sistemas abiertos a gran escala.

Todos estos son discutidos mas a detalle adelante en este documento.

Como se había mencionado con anterioridad, el software que proporciona capacidades de transparencia de funciones distribuidas así como estándares basados en interfaces de programas de aplicación (API's) y protocolos de red es llamado Middleware, este software reside en la mitad entre un programa de aplicación y el sistema operativo y las capacidades de red de un sistema. Middleware soporta interfaces basadas en estándares

(incluyendo API's y protocolos de red), que enriquecen las capacidades de distribución de las aplicaciones, interoperabilidad entre aplicaciones y portabilidad de aplicaciones entre sistemas.

Los estándares middleware especifican interfaces para operaciones distribuidas entre distintos sistemas (protocolos de comunicación), interacción con los usuarios, acceso de datos, administración de sistemas y de red y programación de sistemas. Los estándares middleware proporcionan especificaciones disponibles al publico en general para una interface y habilitan el desarrollo de múltiples implementaciones de capacidades middleware que están disponibles en múltiples sistemas.

Los desarrolladores de aplicaciones y los integradores de sistemas pueden usar servicios de middleware como pequeños bloques para construir aplicaciones del tipo enterprise-wide que usen recursos de computación distribuida de manera efectiva. Estos sistemas proporcionan a los usuarios la información que ellos necesitan, donde, cuando y en el formato que ellos necesitan.

3.7 Message Oriented Middleware (MOM) y Database-Centric Distributed Computing.

Mientras la computación distribuida soporta las soluciones departamentales, Message Oriented Middleware (MOM) proporciona la flexibilidad y escalabilidad necesaria para soluciones de grandes empresas con grandes volúmenes de datos.

3.7.1 Computación Distribuida con base de datos centralizada (2-Lazos).

En la computación distribuida con bases de datos centralizada, existen normalmente tres componentes lógicos: Servicios de presentación, lógica de negocio y acceso a las bases de datos. Estas se ilustran en la siguiente figura:

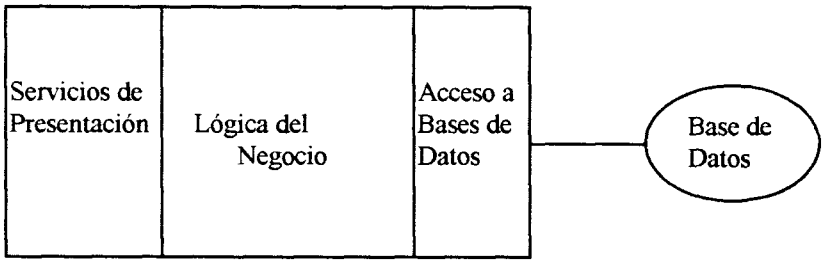


Figura 3.5. Database-centric Computing.

Los servicios de presentación (pantallas de I/O y aceptación de entradas de los usuarios). La lógica del negocio, normalmente SQL o Stored Procedure Calls, encapsulan practicas de negocio tales como transacciones financieras. El acceso lógico a la base de datos consiste de software de Bases de Datos (como DB2 u Oracle), que controlan el acceso fisico de los datos. La expresión "Database-centric" es usada para describir aplicaciones basadas en esta arquitectura porque la lógica de negocio incluye diferencias directas referencias directas a objetos de la base de datos tales como tablas y columnas.

Muchas veces las aplicaciones de bases de datos centralizadas corren en entornos de mainframe que son monolíticos con terminales atadas directamente, pero un numero cada vez mayor de aplicaciones han empleado configuraciones fisicas de 2-Lazos que incluye clientes y servidores. En este caso un protocolo de bases de datos cliente/servidor tal como Oracle SQL Net's o Sybase Open Server son usados para comunicaciones entre clientes y servidores.

Esto es ilustrado en la siguiente figura:

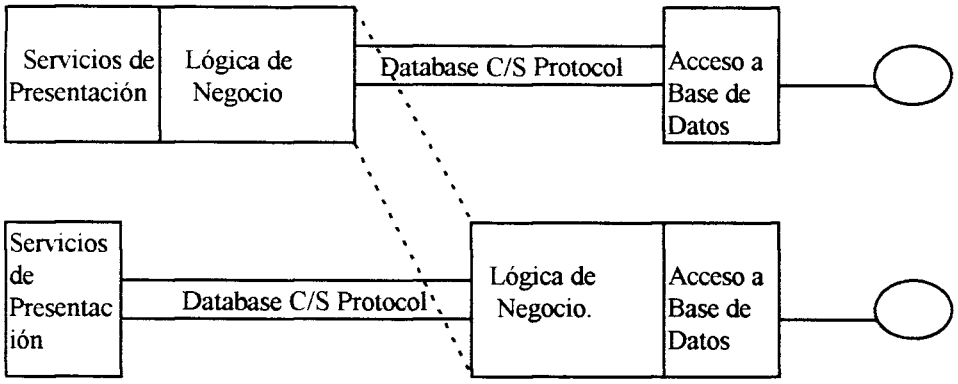


Figura 3.6. Comunicaciones entre Cliente y Servidor

Como se ilustra, la lógica de negocio puede ser localizada en el cliente o en el servidor, pero invariablemente es localizada en uno o en otro. Mover la lógica de aplicación al cliente reduce el monto de procesamiento requerido en el servidor, mejorando la escalabilidad del sistema. Para búsquedas adhoc y aplicaciones simples la lógica de negocio es usualmente localizada en el cliente.

A causa de una limpia división entre componentes lógicos de aplicación y la relativamente simple computación distribuida de 2-Lazos, este tipo de arquitectura de bases de datos es ampliamente usada.

Por muchas razones, las empresas están cambiando su computación distribuida de 2-Lazos a lógica y funciones distribuidas de 3-Lazos. En este caso, las aplicaciones distribuidas están comprendidas de componentes de lógica de negocio C/S que son conectadas por medio de middleware.

Un atributo clave que distingue a la arquitectura distribuida de 2-Lazos con base de datos centralizada es que la lógica de negocio y/o las funciones de negocio están inherentemente distribuidas y encapsuladas dentro de mensajes que son intercambiados usando message-queuing o message-passing.

La arquitectura de computación distribuida utilizando middleware orientado a los mensajes es ilustrado en la siguiente figura.

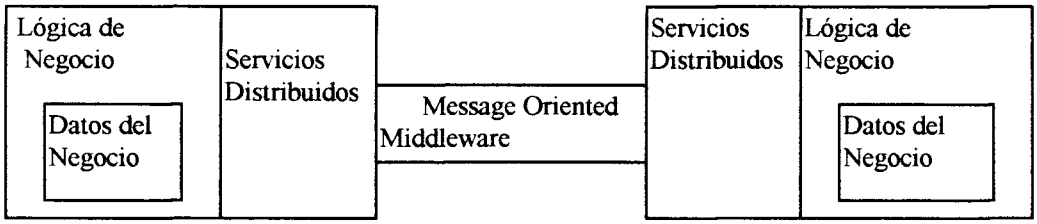


Figura 3.7. Message Oriented Middleware Distributed Computing Arquitecture

Message oriented middleware proporciona una colección de servicios distribuidos que almacenan diferentes procedimientos de llamadas. Muchas implementaciones middleware ofrecen un amplio orden de servicios distribuidos tales como seguridad, alta disponibilidad, servicios de transacciones, soporte y administración de plataformas heterogéneas.

La manera en que las aplicaciones distribuidas interactuan en una computación distribuida basada en middleware es que las aplicaciones del servidor ofrece servicios tales como "add customer", y las aplicaciones del cliente normalmente las invocan por medio del envío de mensajes para requerir el servicio. La manera en que los nuevos servicios son introducidos al sistema y la manera en que los clientes las invocan es una función de el middleware. En algunas arquitecturas, la configuración de archivos es modificada o las utilidades introducen nuevos servicios. En arquitecturas mas sofisticadas cuando una aplicación comienza, este avisa sus servicios. Esta ultima permite a los entornos distribuidos ser mas dinámicos por permitir servicios nuevos o modificar servicios existentes a ser introducidos, en cualquier momento sin la necesidad de hacer cambios.

Para invocar un servicio, un cliente crea un mensaje conteniendo datos del negocio y lo envía usando message queuing o message-passing dependiendo de la implementación. El middleware envía el mensaje al servidor que procesa el mensaje y responde con otro mensaje de sus propios datos de negocio, un ejemplo de tal intercambio es ilustrado en la siguiente figura.

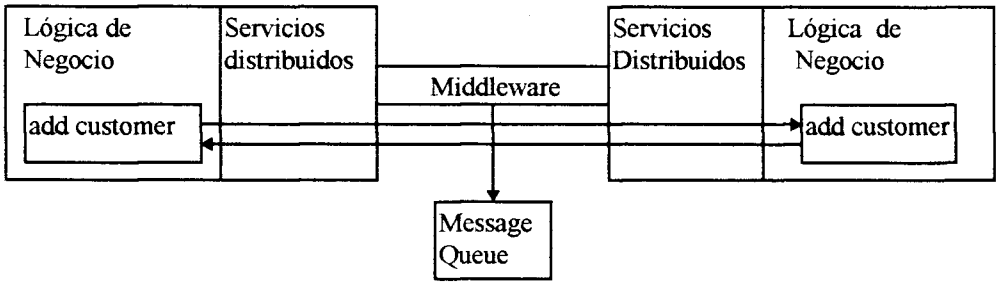


Figura 3.8. Client/Server Message-Passing.

En este ejemplo se debe notar que la única distinción entre el cliente y el servidor es quien inicia el intercambio de mensajes, de hecho, middleware habilita no solo C/S sino que también computación distribuida peer-to-peer. Las aplicaciones de bases de datos que son distintas y usan middleware son inherentemente distribuidas porque la lógica del negocio es siempre dividida. Con una solución MOM, las aplicaciones del servidor pueden beneficiarse ampliamente por el uso de cualquier extensión de SQL que se encuentre en el mercado y/o procedimientos de bases de datos no portables mientras que todas las aplicaciones cliente permanecen independientes de la base de datos. Esto nos capacita para reemplazar un servidor de bases de datos simplemente por tener la nueva aplicación de los servidores en una nueva base de datos emulando los mismos componentes de negocio.

Los distintos protocolos de bases de datos cliente servidor pueden intercambiar muchos mensajes en la red para ejecutar una sola orden de SQL, si utilizamos message oriented middleware solo un par de mensajes son utilizados para cada interacción. Esto reduce significativamente el trafico de la red y mejora el desempeño del sistema. Además, a causa de que el middleware maneja todo el trafico de mensajes, este puede proporcionar valor-agregado a los servicios, incluyendo seguridad, identificación de servicios y administración de red.

Usando una técnica llamada "middleware encapsulation", el acceso a distintos sistemas y data warehouses es extendido vía aplicaciones middleware para servidores, convirtiendo este servidor en un servidor disponible para clientes middleware, protegiendo inversión en sistemas que ya existen y extendiendo la capacidad de recuperación y acceso a datos. Desde el punto de vista del cliente que el middleware simula un solo sistema con un solo acceso para el.

NOTA: Message oriented middleware (MOM), es una arquitectura flexible que proporciona soluciones distribuidas, si se cuenta con este tipo de middleware entre clientes y servidores, los clientes pueden enfocarse en la lógica de negocio y datos y de esta manera elegir el mas efectivo sitio de lazos lógico y fisicos de acuerdo a sus necesidades. En implementaciones middleware sofisticadas, las aplicaciones distribuidas pueden beneficiarse de una gran gama de servicios distribuidos como son seguridad y reconfiguraciones que agreguen valor para resolver problemas de aplicación. Por estas razones se prevé un gran crecimiento de MOM como la base de la siguiente generación cliente/servidor. Ha continuación se agregan dos gráficas que pueden ilustrar un poco esta tendencia.

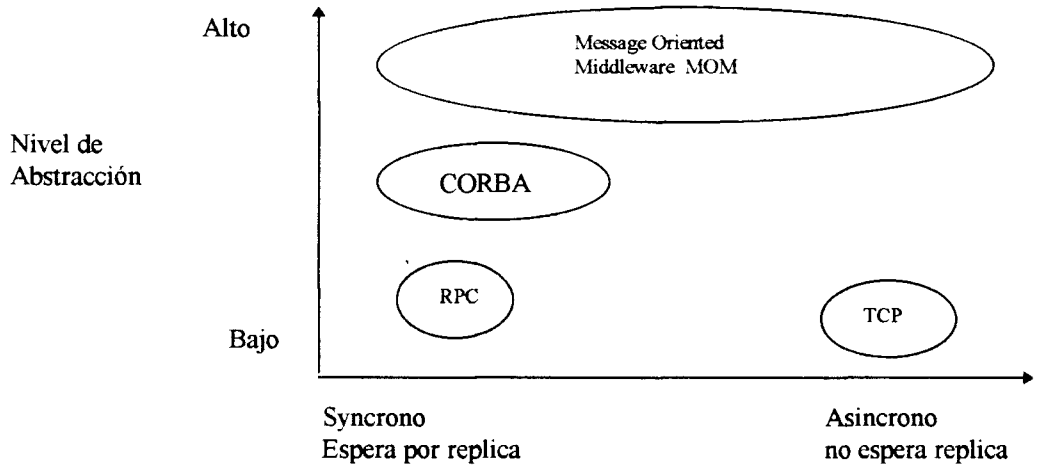


Figura 3.9. Flexibilidad de Comunicaciones.

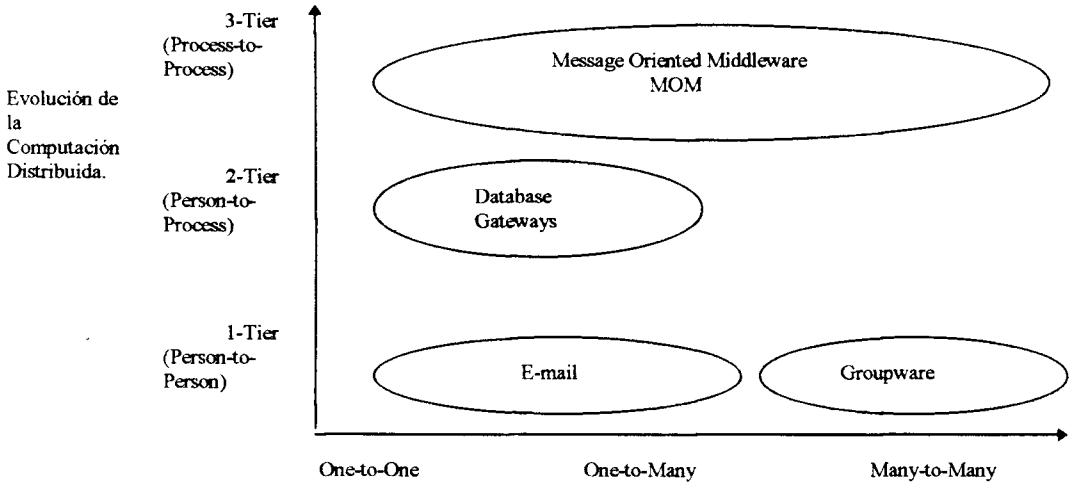


Figura 3.10. Conectividad Distribuida.

3.7.1.1 Message Oriented Middleware (MOM) & Remote Procedure Calls (RPC).

Con la llegada de la computación de escritorio y la proliferación de sistemas de administración de bases de datos relaciónales (RDMS), el paisaje de las grandes empresas computacionales cambio drásticamente. Como la computación se había movido dentro de sistemas centrales, un gran monto de información almacenada existía en los Mainframes, pero el acceso a estos datos se ha extendido a través de la empresa, desde luego, es deseable que esta sea integrada dentro de un todo sin remiendos. No solamente podemos trabajar en conjunto mas efectivamente en un entorno como estos, pero la administración de los sistemas, reusabilidad del código y una reducción en los costos de largo plazo están facilitados por una empresa integrada.

Una creciente tecnología por conquistar la diversidad de las redes es el Middleware. Middleware es ampliamente referido como software que provee la ilusión de homogeneidad en la red para una aplicación mientras que al mismo tiempo opera sobre topologias heterogéneas de redes. Hoy existe un enorme interés en middleware y es justo decir que dos de los mas populares son los modelos RPC y MOM. Esto es debido a que la mayoría de los proveedores se han alineado con RPC y soportan su método de comunicaciones, pero existe una pregunta que debemos tomar en cuenta: ¿Es RPC el mecanismo ideal para las comunicaciones inter-empresariales?, aquí tiene lugar una controversia la cual vamos a discutir.

El numero de compañías implementando middleware esta creciendo rápidamente. Existe la tecnología llamada Remote Procedure Calls (RPCs), que ofrece soluciones practicas y robustas a varios tipos de problemas de la computación distribuida. De acuerdo a algunas fuentes, mas de 4 millones de maquinas soportan RPCs en este momento.

La otra tecnología que tenemos es la de message oriented middleware (MOM), y esta tecnología es mas reciente y se esta incrementando rápidamente porque aparenta ser mas simple, mas confiable y de mejor desempeño. Otros tipos de middleware también han proliferado, tales como middleware para bases de datos, middleware para procesamiento de transacciones, middleware para comunicaciones y tipos similares de productos. Muchas de estas soluciones tienen una clara posición en el mercado, sin embargo, existe gran confusión- ¿Cuando uno reemplaza al otro?, ¿Cuales aplicaciones son mejor manejadas por cada cual?, ¿Estos son sustituibles o complementarios?, es claro que ambas de estas tecnologías tienen un rol en la computación distribuida y pueden coexistir dentro del entorno distribuido.

3.7.1.2 Definición de RPC.

Las Remote Procedure Call (RPC), toma un concepto de programación familiar, llamando a una subrutina y aplicándola a un entorno de red. Las interacciones dentro de este modelo son program-to-program, estas comunicaciones son sincronas y de bloqueo. Esto se muestra en la siguiente figura.

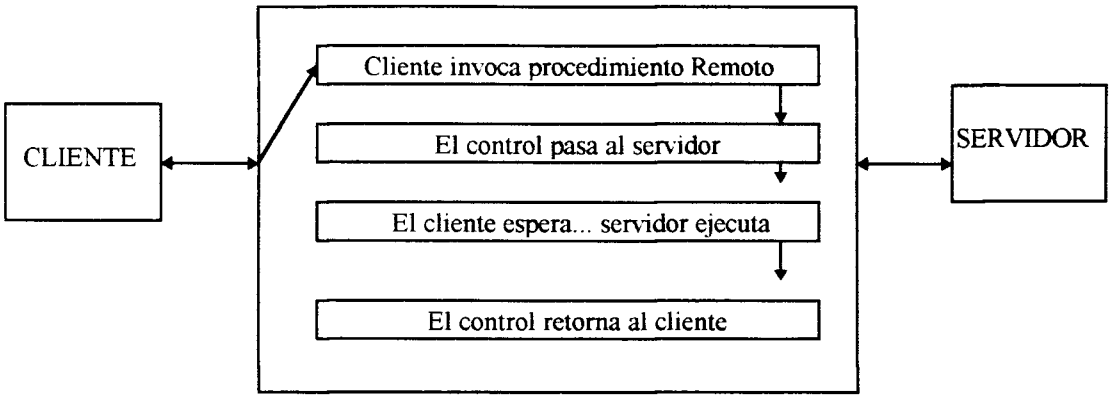


Figura 3.11. Remote Procedure Calls.

Usando RPC, un desarrollador puede llamar una subroutina que no solo puede estar en un nodo remoto, sino que también corre en un sistema operativo diferente. Los proveedores de RPC, han ido consolidando una pareja de mecanismos: el OSF/DCE RPC y el SUN ONC RPC. Un RPC puede ser usado para simples programas de Requerimiento/Respuesta tales como actualizaciones o búsquedas en bases de datos. En construir aplicaciones cliente servidor RPCs juega un papel muy importante. El RPC es generalmente provisto por los vendedores como módulos de procesamiento para clientes y como librerías de código para adicionar capacidades entre aplicaciones sobre los nodos de la red. Cuando las aplicaciones y los procesadores son capaces de conectarse y correr recursos de RPC, los usuarios son capaces de tomar una completa ventaja de procesamiento distribuido a través de la empresa.

3.7.1.2.1 Beneficios de RPC.

RPC lleva a cabo el objetivo de aislar a los programadores de los detalles de las capas bajas de la red. El programador no ve nada mas que la semántica del lenguaje de llamadas-retorno del lenguaje con el que ya esta familiarizado. El programador no requiere un gran entrenamiento en los complicados detalles de comunicaciones.

Quizás mas significativamente, diferentes esquemas de distribución pueden ser tratados rápidamente sin mayor impacto en la programación cuando el RPC es usado. La unidad básica de la perspectiva de un programa apoyado en RPC es el nombre de la función. Tal aplicación no tiene el conocimiento directo del protocolo nativo o de la semántica de la red. Cuando una nueva distribución de carga de trabajo va a ser implementado, la función a ser remotamente ejecutada es portada a una nueva locación en otro server. Por lo tanto nuevas aplicaciones pueden ser desarrolladas como proyectos piloto relativamente cortos.

Los sistemas RPC tienden a ser un buen trabajo para normalizar los datos. Este aísla las aplicaciones de las complejidades de la red y de la arquitectura de los CPU. Como estas

tecnologías inevitablemente cambian, la lógica de las aplicaciones de la empresa permanecen sin afectarse.

3.7.1.3 Message Oriented Middleware.

Message oriented middleware proporciona un entorno de procesamiento completo que permite a los desarrolladores y usuarios conectar datos y código de un sistema a otro o de interconectar procesos usando interfaces consistentes definidas. Message-based middleware. este proporciona interfaces de programación de alto nivel que resume la capa de comunicaciones de datos y los componentes de acceso de cada parte de un sistema distribuido, proporcionando un enlace en cada sistema operativo. Messaging es el proceso de distribuir y controlar datos por medio de el uso de mensajes.

Messaging permite la abstracción de los procesos de las capas bajas en verbos comprendidos por el API de la arquitectura de mensajes. Una vez que esta arquitectura es puesta en un lugar, las conexiones son resumidas dentro de un pequeño numero de funciones expresadas como parte del API, el cual es distribuido a través del sistema. Los principios de message oriented middleware son como se muestran en la siguiente figura:

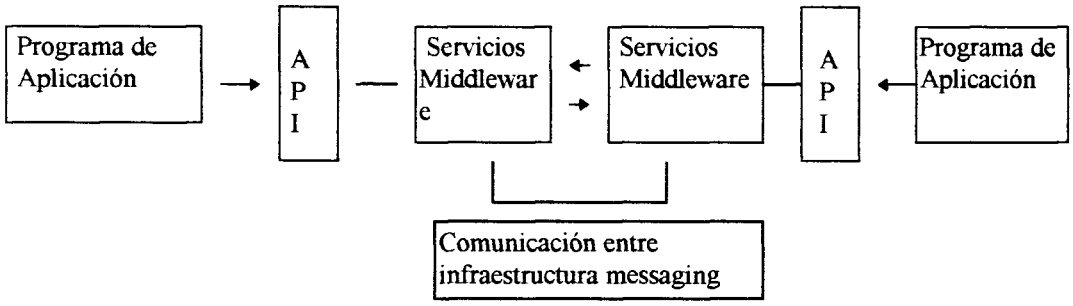


Figura 3.12. Message Oriented Middleware.

Messaging aísla a los clientes de los procesos back-end y libera a la aplicación de estos procesos, esta liberación es la mas importante distinción entre message oriented middleware y RPC-based middleware. Esta característica permite a los desarrolladores rehusar patrones de acceso a datos capaz de cubrir necesidades de negocio, liberándolos de los métodos de acceso impuestos por la estructura de datos y conexiones existentes. El mensaje es transmitido como una entidad independiente no limitada por el emisor o el receptor.

3.7.1.4 Comparación de MOM y RPC.

Cuando los RPCs son usados, las subroutinas se ligan con cada aplicación y se comunican sobre una sesion entre las dos aplicaciones.

Cuando se utiliza MOM, las comunicaciones se comunican usando mensajes vía API con el API invocando los servicios de una capa de middleware que no requiere de una sesión, los RPCs requieren una respuesta después de que una llamada a RPC es ejecutada, message oriented middleware usa la naturaleza asincrona de la comunicación para habilitar un proceso o aplicación a enviar un mensaje a el middleware mientras trabaja en otra cosa. Los RPCs pueden tener este comportamiento, pero generalmente solo usando técnicas mas complejas de programación.

MOM y RPC se soslayan en algunas partes de la computación distribuida, RPC utiliza comunicación sincrona, los programadores han trabajado bien con RPC considerándolo una extensión de la programación normal, RPC usa estándares para la representación de datos fundamentados en DCE.

MOM, por otro lado, es mas apropiado para soportar modelos peer-to-peer. El modelo de messaging es mas apropiado para aplicaciones complejas multi-lazos porque aparte de comunicaciones sincronas, también soporta comunicaciones asincronas. El desempeño de muchas aplicaciones distribuidas, tales como integración de aplicaciones que accesen muchas diferentes fuentes de datos para satisfacer la búsqueda de un usuario, puede ser ampliamente enriquecida por messaging asincrono. MOM es mas apropiado para soportar objetos distribuidos.

En adición, la siguiente generación de aplicaciones distribuidas puede necesitar soportar modelos mas complejos de distribución. Hay fuertes indicadores que los modelos de requerimientos/respuesta ofrecidos por RPC pueden no ser suficientes en aplicaciones y procesos a gran escala. Hay muchas aplicaciones en las empresas que pueden requerir un método mas flexible para computación distribuida. Muchas aplicaciones necesitan soportar comunicaciones sincronas y asincronas.

3.7.1.5 Comparación ORB y RPC.

Los mecanismos de invocación de ORB (Object Request Broker), y RPC (Remote Procedure Call), son similares pero tienen importantes diferencias. Con RPC se llama a una función especifica (el dato es separado), en contraste con ORB, donde se llama un método dentro de un objeto especifico. Diferentes clases de objetos pueden responder al mismo método. Cada objeto administra sus datos.

La siguiente tabla sumariza las propiedades y atributos de MOM y RPC, en general. Dependiendo de la compañía que ofrece un producto middleware estas pueden variar un poco y tienen que ser consideradas individualmente.

Propiedades/Atributos	MOM	RPC
Asincrono sin bloqueo	si	no/limitado
Sincrono bloqueo	si	si
Sincrono diferido	si	si
Escalabilidad amplia	si	limitada
Soporte multi-protocolo	si	limitado
Soporte a sistemas multioperativos	si	si
Soporte DCE	no	si
Prioridad de mensajes	si	no
Opciones de entrega	si	no
Soporte a arquitecturas 3-Tier	si	no sin dificultad
Disponible en el mercado	si	si
Fácil de aprender	si	si
Capacitación para programadores	si	no
Programación API	si	no

Tabla 3.1. Propiedades y Atributos de MOM y RPC.

Las propiedades básicas de las alternativas middleware disponibles son:

	MOM	RPC	DB Gateway	Propietario
Bloque Sincrono	si	si	si	si
No-Bloque Asincrono	si	no	no	si
Escalabilidad Enterprise	si	no	no	no
Soporte multiprotocolo	si	no	si	no
Soporte Multi.operativo	si	si	si	si
Prioridad de Mensajes	si	no	no	si

Tabla 3.2. Propiedades Básicas de las Alternativas Middleware.

3.8 PRODUCTOS MIDDLEWARE.

Existen 5 principales categorías de productos middleware, estas son:

Remote Procedure Calls, (RPC).
Message Oriented Middleware (MOM).
Productos de Conectividad de Bases de Datos.
Distributed Computing Environment, (DCE).
Distributed Transaction Procesing Monitor (DTPM).

La elección de cada una de estas categorías y como vamos a implementar nuestro middleware, corresponde a una evaluación independiente, para elegir cada uno de estos o varios depende de nuestra configuración en particular de los sistemas instalados y nuestro propósito de conectividad.

Existe un middleware en el mercado que es muy importante, este middleware desarrollado por IBM se presenta a continuación.

3.8.1 Messaging & Queuing Series (MQSeries).

MQSeries ayuda a resolver problemas como son:

- Ambientes mixtos.
- Programación Compleja.
- Opciones limitadas de diseño.
- Dificultad para coordinar datos.

3.8.2 MQSeries e Internet.

Este producto ofrece varios beneficios a los usuarios de MQ:

- Acceso a clientes y proveedores, comercio electrónico.
- Clientes de bajo costo, (GUI atractiva).
- Red de área global barata (Comunicación entre compañías).

3.8.2.1 Beneficios de MQSeries a WWW.

- Procesamiento Asincrono (Fuera de línea).
- Garantiza la entrega.
- Soporte de mensajes transaccionales.
- Ambiente de programación conveniente.
- Aísla la red interna de internet.

- Puente para redes no-IP.

3.8.3 Utilizando MQSeries en Internet.

3.8.3.1 Aplicación MQ- a través de internet.

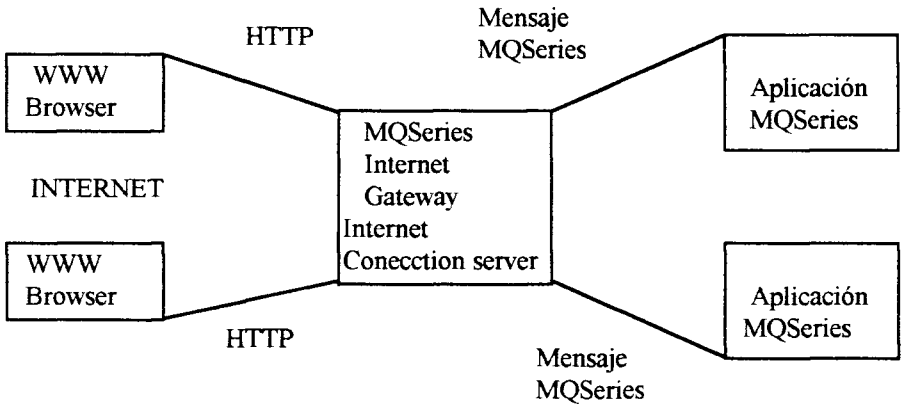
- Posible ya que MQSeries soporta TCP/IP.
- MQ garantiza la entrega.
- Seguridad MQ para proteger datos sensitivos.
- Adecuado para comunicación dentro de la misma empresa y entre empresas.

3.8.3.2 Aplicación MQ - Web Browser.

- Es necesario usar Internet Gateway.
- No se necesita aplicación MQ o código en el Browser.

3.8.3.3 Acceso desde el Web a Aplicaciones MQSeries.

Figura 3.13. MQSeries y el WEB.



A continuación se describen algunos productos middleware que se encuentran en el mercado:

Comunicación RPC.

Interfaces	Protocolos	Organización
DCE RPC	DCE RPC	X/Open

Messaging & Queuing.

Interfaces	Protocolos	Organización
MQI		IBM

Object Management.

Interfaces	**Protocolos**	**Organización.**
CORBA		OMG
Common Object Services		OMG
Lifecycle Services		OMG
Externalization Services		OMG
Identity		OMG
	IIOP, DCE-CIOP	OMG

Seguridad.

Interfaces	**Protocolos**	**Organización**
DCE SEC_RGY	DCE Registry	OSF
GSSAPI		IETF/OSF
DCE SEC_ACL	DCE ACLs	OSF
	Kerberos	MIT/OSF

Transaction Manager.

Interfaces	**Protocolos**	**Organización**
Transactional C	Encina	Transarc
TX		X/Open
XA		X/Open
	OSI TP	ISO/IEC
CICS API		IBM
IMS TM API		IBM
	SNA Sync Point	IBM
Transaction Service		OMG

Interface del Usuario.

Interfaces	**Protocolos**	**Organización**
OS/2 PM		IBM
Motif		OSF
Windows		Microsoft
X Window Xlib		X/Open
X Window Xt intrinsics	X Window System	X/Open

Bases de Datos Relacionales.

Interface	**Protocolos**	**Organización**
SQL (1992E)		ISO/IEC
SQL CLI		X/Open
	DRDA	IBM
ODBC		Microsoft

Bases de Datos Jerárquicas.

Interfaces	**Protocolos**	**Organización**
DL/I		IBM

Storage Management

Interfaces	**Protocolos**	**Organización**
Backup Services		X/Open

Monitores de Transacciones

Interfaces	**Protocolos**	**Organización**
CICS API		IBM
IMS TM API		IBM

NOTA: Esta lista es solo una pequeña guía de los productos middleware, existen en el mercado mas de 400 productos middleware entre las 5 categorías.

CAPITULO IV. PROBLEMAS Y BENEFICIOS DE MIDDLEWARE.

Hoy los sistemas de información son una herramienta clave para ayudar a las empresas a responder a las condiciones de mercado cambiantes, moviendo la información electrónicamente, estos sistemas proporcionan a los trabajadores acceso a la información que necesitan cuando la necesitan. Esto se refleja en una importante ventaja competitiva.

Por muchos años diferentes departamentos en la empresas resolvieron sus problemas de negocios con la compra de plataformas de hardware, sistemas operativos y aplicaciones de diferentes proveedores. Hoy, los usuarios en varios departamentos dentro de una empresa necesitan compartir información no solo entre sus departamentos sino también entre clientes, socios y proveedores externos.

Los sistemas de información intra-empresariales se están volviendo de misión crítica, y algunas veces el cuello de botella, para accesar información. Mientras estos sistemas son efectivos también se vuelven complejos, con muchos componentes y muchas interdependencias, volviéndose costosos y difíciles de mantener.

El middleware advierte las diversas necesidades del escritorio, workgroup y entornos de los mainframes y de manera particular, la conectividad e interoperabilidad entre estos entornos. De esta manera, middleware ayuda a hacer que esta información sea accesible a todos los usuarios. Para soportar una amplia variedad de estándares para procesamiento distribuido, middleware ayuda a separar la información de "pequeñas islas separadas" de computación para comunicarlas efectivamente.

Los sistemas de información se pueden encontrar en múltiples locaciones y organizaciones. Los diseñadores de sistemas deben construir cuidadosamente estos sistemas a partir de un gran número de componentes, los sistemas deben estar en muchos departamentos y en compañías separadas. Cada uno de estos departamentos o compañías pueden tener una base instalada de sistemas de información para satisfacer sus necesidades locales. Una estrategia para crear un sistema de información inter-empresarial debe contar con el hecho de que la información local puede venir de una variedad de manufactureros y una variedad de sistemas operativos.

Como la tecnología de información avanza, existe la necesidad de tomar ventaja de la diversidad de tecnología mientras se ocultan las diferencias no-esenciales. En otras palabras, las compañías necesitan adoptar una amplia variedad de interfaces estándares que permitan a los diversos sistemas a interoperar y aislar aplicaciones de las diferencias existentes. Los servicios de middleware proporcionan esta capacidad.

4.1 MIDDLEWARE COMPRENDE UNA VARIEDAD DE SERVICIOS.

El middleware puede ser visto como una variedad de servicios, como se muestra en la siguiente figura. De esta manera se accesan servicios de aplicaciones por medio de un servicio middleware conocidos como API's. El programa de aplicación hace un requerimiento a través de un servicio de API. El servicio middleware procesa el requerimiento y regresa un resultado.

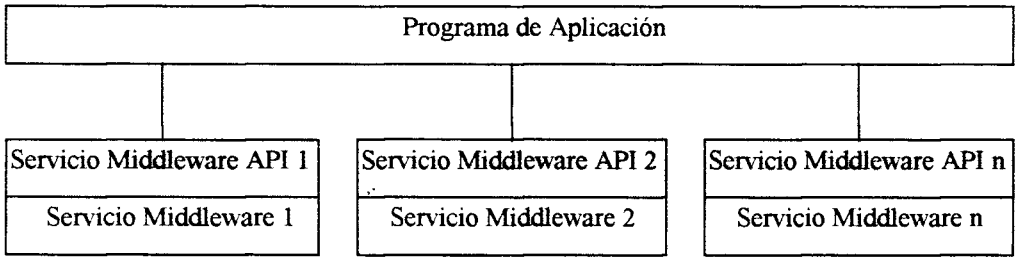


Figura 4.1.- Middleware es un conjunto de servicios accesados a través de sus API's.

Un ejemplo de un servicio de middleware es RPC (Remote Procedure Call), el servicio de RPC comunica con un servidor remoto que contiene la implementación de el procedimiento. El server remoto ejecuta el procedimiento y el servicio RPC regresa los resultados al programa que inicio el requerimiento. El desarrollador de la aplicación no necesita saber donde esta el servidor remoto o si el procedimiento fue ejecutado local o remotamente.

Algunos servicios de middleware son relativamente independientes, pero muchos están integrados con otros servicios middleware. Algunos servicios hacen uso de otros servicios para proporcionar parte de sus funciones (por ejemplo, los servicios de messaging usan el servicio de directorio de datos) y otros proporcionan estilos similares de interface (por ejemplo, los servicios distribuidos integrados tienen API's similares).

La siguiente figura representa otra manera de mostrar los servicios middleware, esta figura es equivalente a la figura anterior, pero ahora se enfatiza la integración de los servicios middleware.

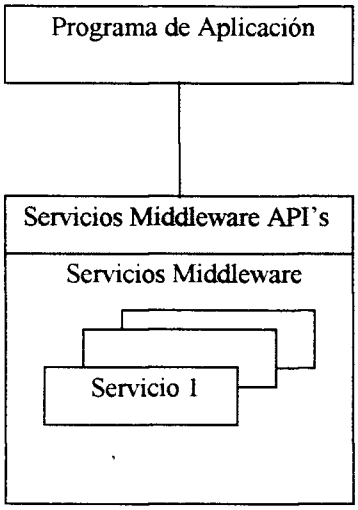


Figura 4.2.- Servicios Middleware Integrados.

Los servicios middleware hacen que sea fácil para las aplicaciones trabajar juntas porque los servicios API ocultan las diferencias de los varios recursos no-esenciales de los sistemas de información. Las aplicaciones pueden fácilmente accesar múltiples recursos a través de la red y usar estos recursos para intercambiar información con otras aplicaciones. De esta manera, los servicios middleware ayudan a integrar diversos servicios de hardware, sistemas operativos, redes, datos, procesamiento y usuarios.

Una de las claves para integrar una diversa variedad de recursos es el uso de estándares. La mayoría de los servicios middleware y sus interfaces están basadas en una variedad de fuentes, este tema lo veremos a fondo un poco más adelante.

Muchos servicios middleware son servicios de alto-nivel; esto es, una sola llamada al servicio puede desencadenar complejas situaciones de procesamiento. Por ejemplo, un solo requerimiento de un servicio de acceso a datos puede recuperar muchos renglones de información de una o más base de datos relacional de algún lugar remoto usando SQL. Esto significa que las aplicaciones pueden usar estos servicios para completar una sola operación, visto por el lado de los programadores, usando servicios middleware, ellos pueden lograr esto con unas pocas líneas de código. Pocas líneas de código reducen la necesidad de crear aplicaciones complejas y de involucrar a los programadores de temas complejos de comunicación.

4.2 DESARROLLANDO MIDDLEWARE PARA APLICACIONES DISTRIBUIDAS.

Mientras existen varios beneficios para las funciones distribuidas de un sistema de información a través de redes inter-empresariales, los esfuerzos de programación para hacer esto posible han sido muy fuertes. Para lograr que un sistema sea verdaderamente distribuido, es decir, que tenga la habilidad de mover las funciones de procesamiento de la información de una aplicación a través de la red sin costosas modificaciones al programa. Hoy los servicios de middleware asumen la carga, haciendo posible la operación a través de redes de área local y de área amplia, además, la mayoría de los servicios middleware son transparentemente distribuidos, es decir, que los programadores y usuarios los accesan a través de sus API's sin importar donde o como los servicios son desempeñados en los nodos de la red.

Uno de los beneficios de usar servicios middleware es el de proporcionar a la aplicación protocolos y transporte independientes para comunicarse entre los componentes de una aplicación distribuida. Esto ayuda a llevar a cabo la portabilidad de la aplicación más fácilmente.

La siguiente figura muestra la estructura básica de una aplicación distribuida tradicional, donde una aplicación se comunica con otra aplicación (quizás idéntica).

La aplicación debe incluir la lógica de aplicación y dos tipos de networking code:

Application protocol support code: Este código hace que las aplicaciones sean capaces de formatear e interpretar los mensajes necesarios para comunicar los datos de las aplicaciones y de procesar estos mensajes en la secuencia correcta. Por ejemplo, este código debe implementar el protocolo X o PEX para transmitir imágenes gráficas.

Transport interface code: Este código es la interface entre Application protocol support code y las funciones de transporte de la red el código de interface de transporte hace las llamadas apropiadas a la red para enviar y recibir los mensajes que envía el protocolo de la aplicación sobre un específico transporte de red.

Por ejemplo, una aplicación de acceso remoto de datos necesita un protocolo de aplicación para hacer transferencias de código y para transformar los datos en el formato necesario para transmisión.

La aplicación debe de ser enviada sobre el transporte especificado, el cuál tiene su propio protocolo para manejar la transmisión.

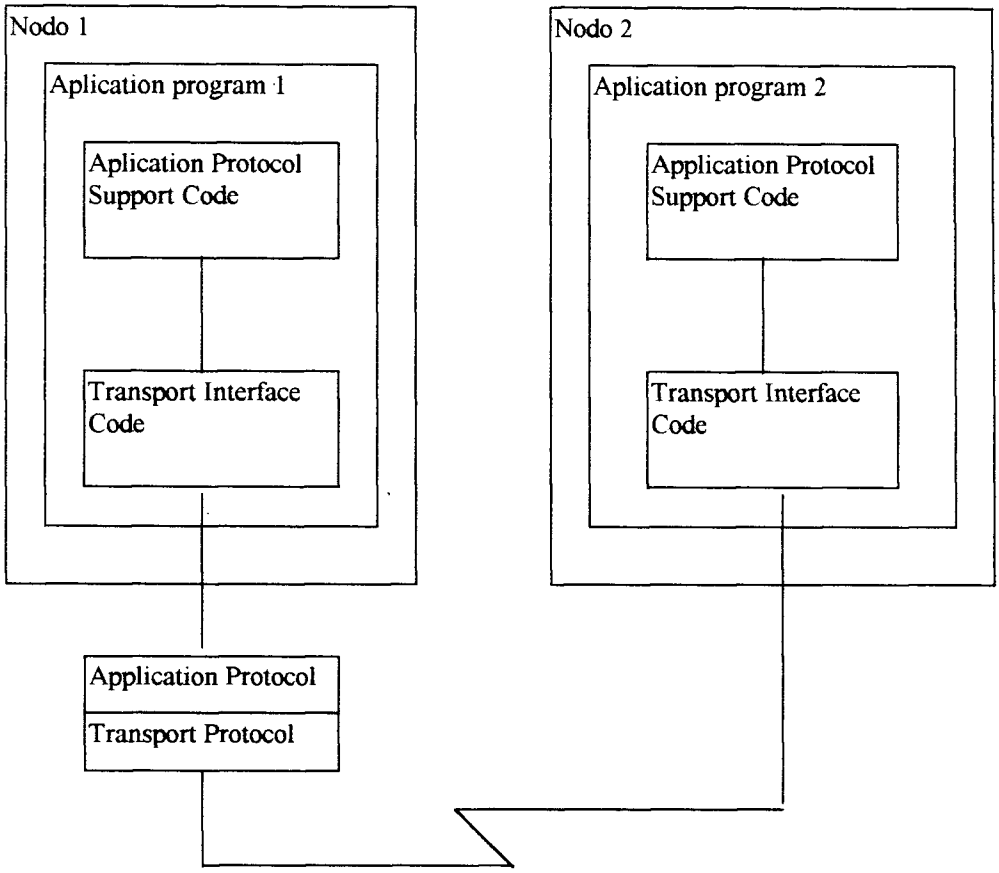


Figura 4.3.- Estructura de una Aplicación Distribuida Tradicional.

Una desventaja de esta técnica de programación es que es costosa de modificar para correr en otro transporte. Para adicionar transporte de otro transporte, los programadores deben de reescribir ambos la Application protocol support code y el transport interface code.

Dependiendo de el diseño de la aplicación, la lógica de la aplicación, application support code y la interface de transporte pueden extenderse a través de componentes inseparables para ser divididos dentro de módulos con interfaces bien definidas.

La siguiente figura muestra una forma más efectiva de diseñar una aplicación que pueda soportar múltiples transportes de red.

La aplicación divide el código de interface de transporte en dos partes:

Código de interface independiente del transporte: Este es un solo grupo de código que proporciona una interface común para múltiples transportes de red.

Código de interface dependiente del transporte: Este es código especializado para cada mecanismo de transporte soportado.

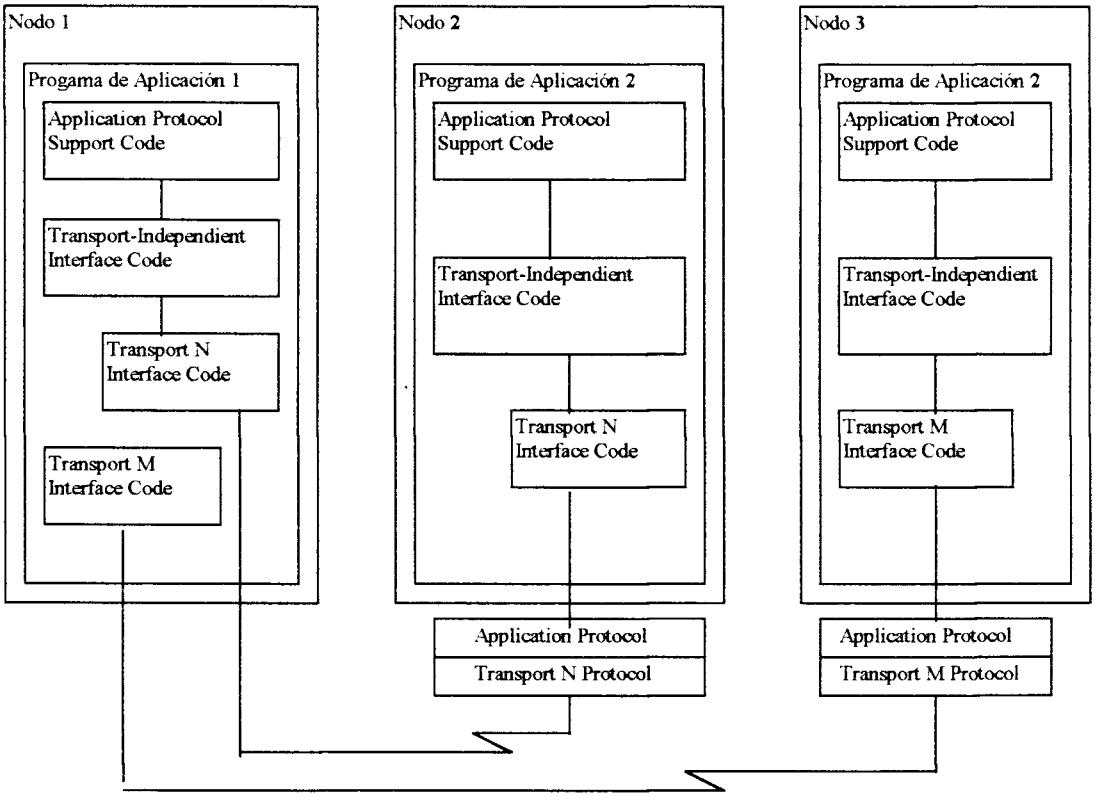


Figura 4.4.- Programación de Aplicaciones Distribuidas Transporte Independiente de Red.

El protocolo de comunicación soporta interfaces de código para el transporte independiente. Para soportar un nuevo transporte, los programadores deben escribir nuevo código de interfaces dependientes de transporte para ese transporte y adicionar un pequeño monto de código al transporte independiente para llamar al nuevo código de transporte. Esto no requiere cambios para el código la aplicación de soporte de protocolo.

La técnica descrita en la figura anterior representa una mejora, pero está aún requiere el diseño y soporte de una aplicación para soportar el protocolo y código de interface para el transporte. Para evitar escribir ambos códigos los desarrolladores de las aplicaciones pueden utilizar servicios middleware, el cual entonces desempeña las operaciones distribuidas, como se muestra en la siguiente figura.

Por ejemplo, los programadores pueden usar servicios para compartir archivos o servicios para accesar servicios para accesar datos remotos para la aplicación. Los programadores deben también usar los servicios de impresión distribuida para imprimir los resultados en una impresora remota. Los programadores requieren los servicios de una manera estándar a través de API's de cada servicio. Los servicios mencionados manejan los detalles de las interfaces particulares de la red.

A causa de que los servicios middleware son transparentemente distribuidos, la localización de los servicios requiere una decisión de configuración, no una decisión de programación, esto enriquece la flexibilidad del sistema.

Los servicios middleware trabajan sobre una gran número de redes de área local y de redes de área amplia, estos incluyen trabajar con estándares públicos tales como OSI y TCP/IP, estándares de facto definidos por la industria como LAN Manager y AppleTalk y redes propietarias tales como DECnet y SNA.

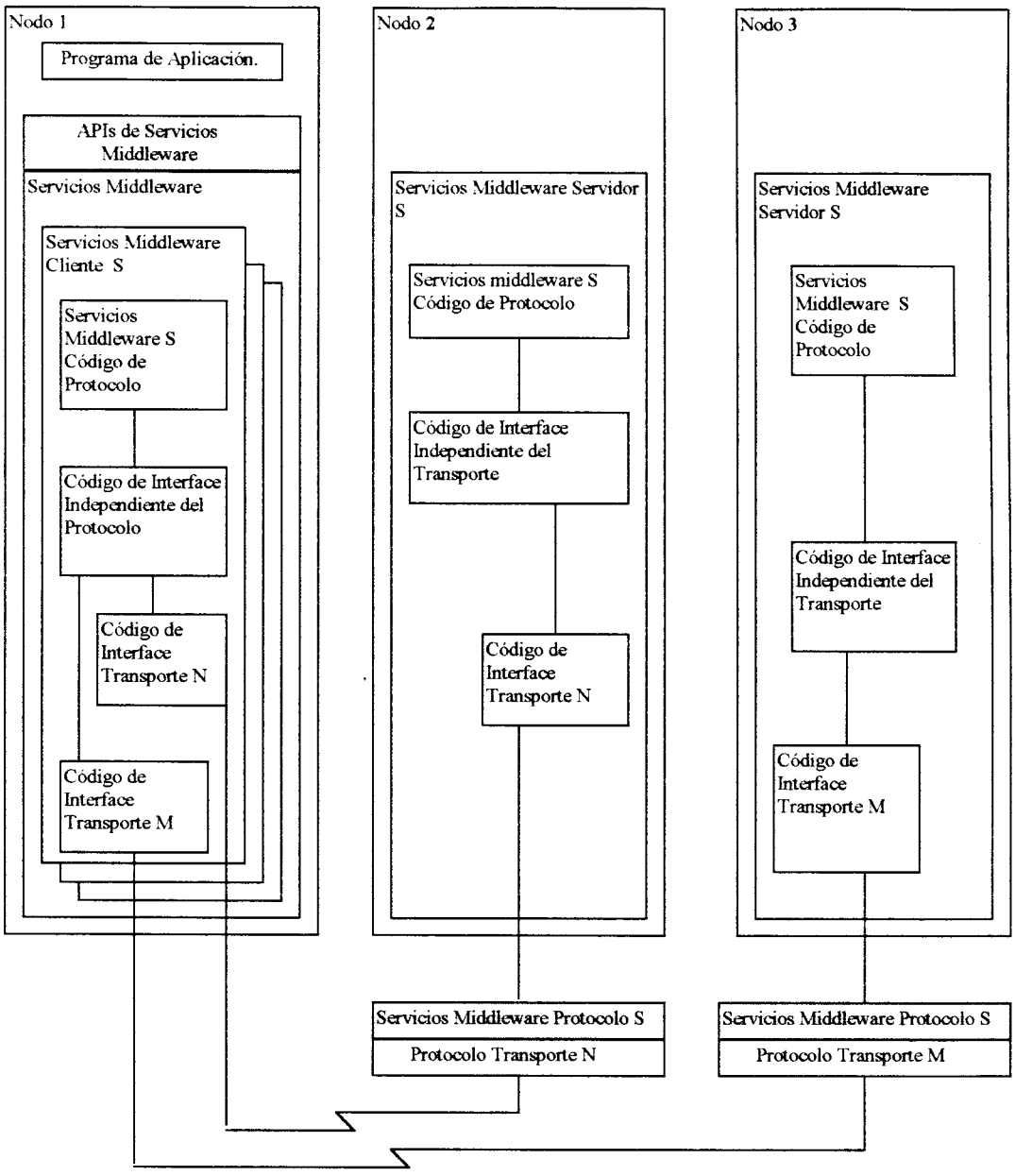


Figura 4.5.- Programación Distribuida basada en los Servicios.

4.3 DESARROLLANDO MIDDLEWARE PARA APLICACIONES PORTABLES.

Las **plataformas** computacionales (pares de hardware y sistemas operativos), vienen en muchas variedades. Derivado de las diferencias entre estas plataformas, el software creado para correr en una plataforma usualmente no corre en otra sin costosas y tardías modificaciones. Mientras una empresa puede evitar este problema estandarizando todo el software a través de una sola plataforma a través de toda la empresa esto muchas veces no es practico o monetariamente efectivo. Entonces se requiere **portabilidad** en las aplicaciones; esto es, la habilidad para mover aplicaciones entre plataformas sin una costosa reingeniería de la aplicación.

Con técnicas tradicionales de programación no portables, como se muestra en la siguiente figura, los programadores escriben aplicaciones que dependen de un sistema operativo específico y una arquitectura de máquinas.

Portar tal aplicación a otra plataforma es, usualmente, bastante costoso, ya que un gran porcentaje del código de la aplicación necesita modificaciones o rediseño.

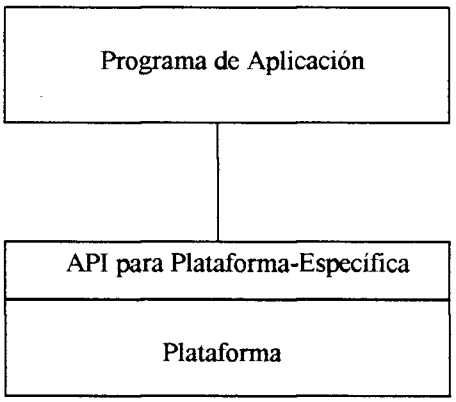


Figura 4.6.- Programación Tradicional no Portable

La siguiente figura muestra como los programadores pueden mejorar la portabilidad de una aplicación aislando las partes dependientes de la plataforma de la aplicación a una pequeña porción de código (Código de interface para plataforma x). Para soportar una nueva aplicación los programadores necesitan reimplementar solamente estos módulos.

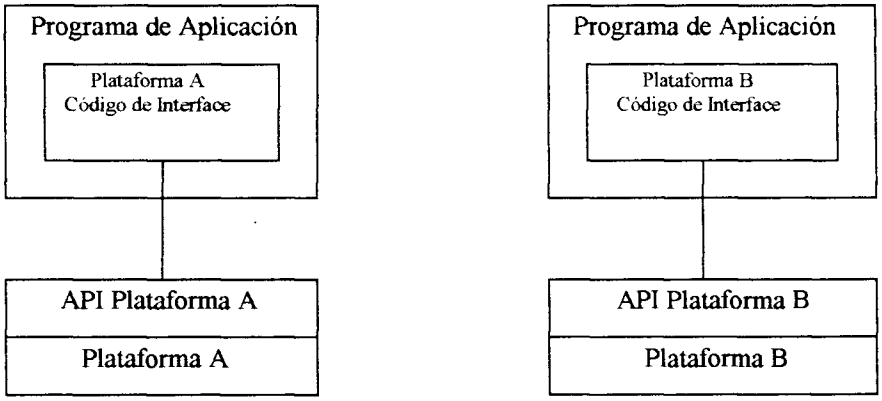


Figura 4.7.- Programación Portable Basada en las Aplicaciones.

La siguiente figura muestra como los programadores pueden mejorar la portabilidad, codificando el programa de aplicación para usar las API's portables de los servicios de middleware. De esta manera, los programadores pueden escribir aplicaciones portables que dependen de una variedad de servicios disponibles en distintas plataformas y tienen pocas o ninguna dependencia de la plataforma.

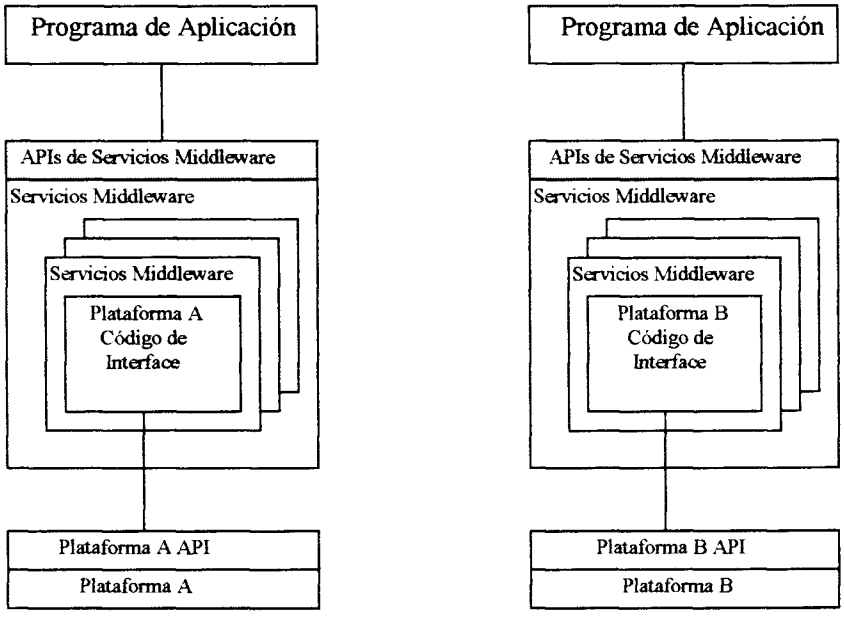


Figura 4.8.- Programación Portable Basada en el Servicio.

Los servicios middleware proporcionan portabilidad a la aplicación poniendo una "máscara" a las diferencias entre las plataformas. Los programadores pueden pensar en el todo de los servicios como un sistema virtual que es independiente del hardware, sistema operativo y especificaciones de la red.

El programa de aplicación no necesita complicadas API's para una plataforma específica, en lugar de eso, el programa de aplicación llama las API's de los servicios middleware y los servicios middleware llaman a las API's de la plataforma específica. La colección de API's proporcionada por la plataforma base está referida como la interface del sistema (SI).

Middleware ayuda a llevar a cabo portabilidad en las aplicaciones de varias maneras:

- *Servicios basados en estándares y API's:* Siempre que sea posible, los servicios middleware y API's están basadas en estándares, esto permite a múltiples proveedores a implementar capacidades middleware en sus plataformas, haciendo el middleware disponible en múltiples entornos computacionales.

- *Lenguaje middleware en estándares binding:* Muchos servicios middleware (estándares y especificaciones), describen ambas, las funciones que están disponibles a través del API y de la actual sintaxis de llamada de las funciones de uno o más lenguajes de programación. La descripción de las funciones de llamada y los parámetros de la sintaxis para un lenguaje específico es llamada "binding", la función de los mismos en diferentes lenguajes pueden ser diferentes, basadas en las capacidades de los distintos lenguajes. Cuando las implementaciones middleware conforman estándares binding, la portabilidad es enriquerida porque los proveedores pueden implementar middleware en distintas plataformas.

- *Servicios middleware de alto nivel:* Los servicios middeware eliminan significantes montos de código de aplicación, porque los servicios proporcionados son de alto nivel y proporcionan muchas de las funciones necesarias. Menos código significa aportar menos esfuerzo.

4.4 DISTRIBUTABILIDAD Y PORTABILIDAD PROPORCIONADA POR LOS SERVICIOS MIDDLEWARE.

La siguiente figura combina distributabilidad y portabilidad en un solo diagrama, cuando las aplicaciones usan middleware, la interoperabilidad de la aplicación es también enriquecida. El middleware no solo implementa estándares API's que enriquecen la portabilidad, sino que también implementa estándares middleware de los protocolos de comunicación.La interoperabilidad es enriquecida porque múltiples proveedores pueden implementar servicios middleware que generen o respondan a los mismos protocolos de comunicación.

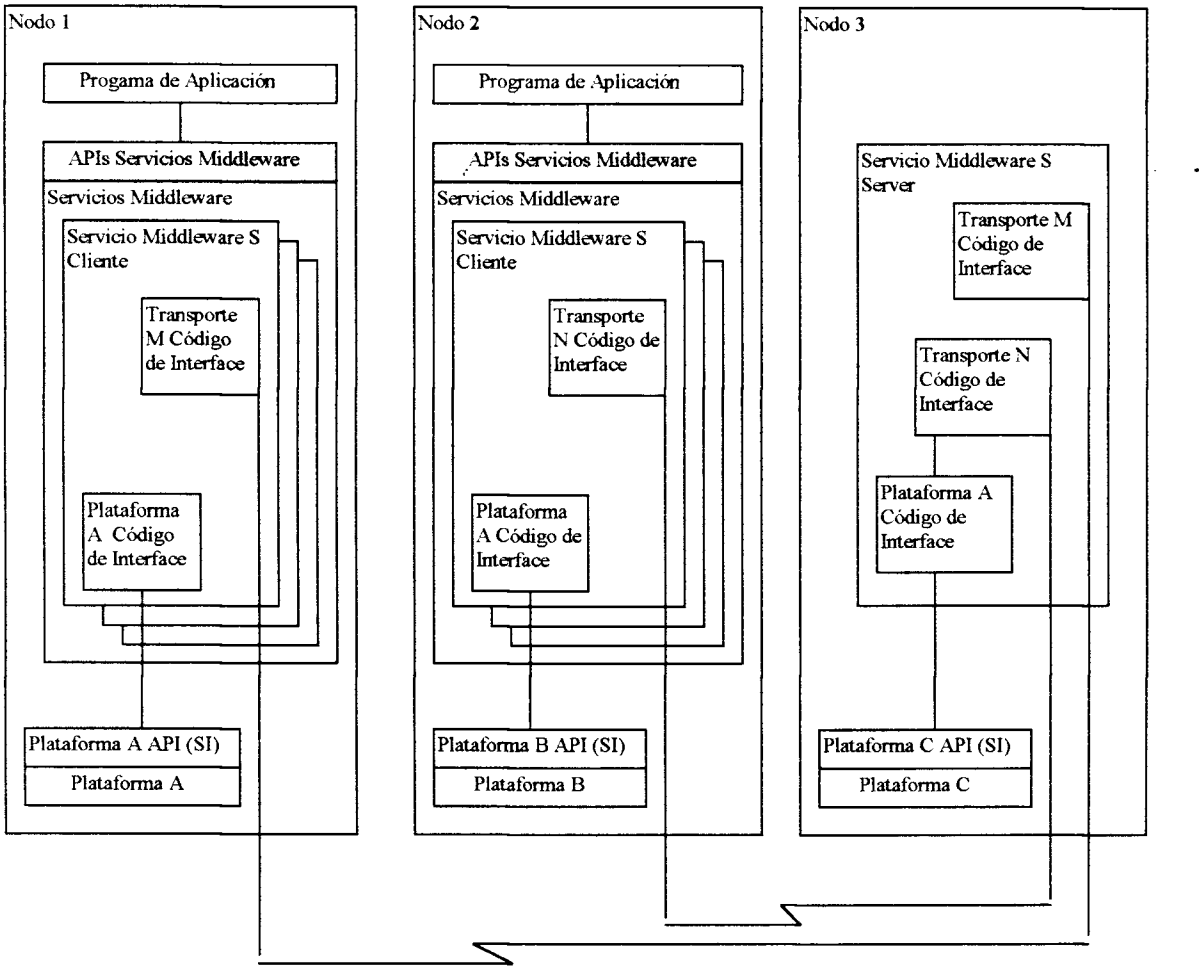


Figura 4.9.-Programación portable basada en servicios distribuidos.

3.4.1 CONTEXTO DE LOS SERVICIOS MIDDLEWARE.

Los servicios middleware proporcionan alto nivel de transparencia de las funciones distribuidas que los programadores pueden accesar a través de API's. Estas API's aíslan a los programas de aplicación de las complejidades de las especificaciones de transporte de la red y de las interfaces específicas de la plataforma, debido a que estos servicios se encuentran enmedio (middle), entre los programas de aplicación (que se encuentran arriba), y las múltiples redes y plataformas (que se encuentran abajo), estos servicios son llamados **servicios middleware**.

El contexto de los servicios middleware se describe en la siguiente figura, la cuál simplifica la figura anterior con más detalles ocultos de la red, la figura siguiente representa un modelo middleware de alto nivel y de cómo construir aplicaciones usando middleware.

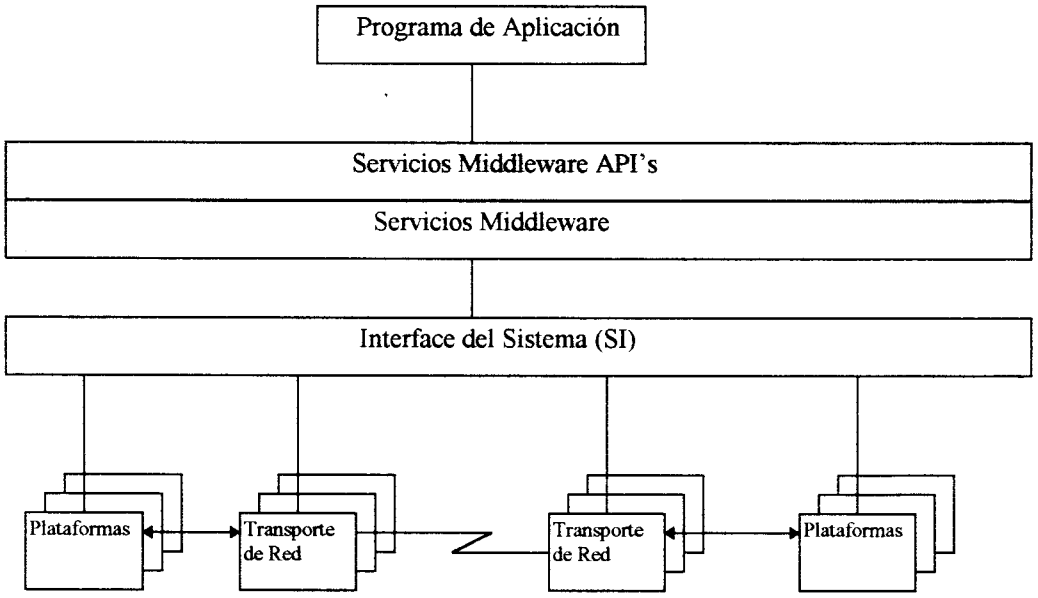


Figura 4.10.-Modelo de Servicios Middleware.

CAPITULO V. INTERFACES MIDDLEWARE.

El middleware y otros sistemas computacionales pueden ser definidos por sus interfaces. Las interfaces middleware separan las partes dependientes de la aplicación de las partes independientes. Por ejemplo, un programa de aplicación que usa un servicio middleware es dependiente de la interface de la aplicación y es independiente de los detalles de como el middleware es implementado.

Existen seis tipos de interfaces . Estas interfaces son interfaces funcionales que middleware usa en la entrega normal de los servicios. Estas interfaces son:

- *Application Programming Interface:* Los programas de interface de las aplicaciones (**API**), especifica como los programas de aplicación interactúan con los servicios middleware. Todos los servicios middleware tienen un API, aunque en pocos casos el API es **nativo** (plataforma específica).

- *System Programming Interface:* Las interfaces de los sistemas (**SPI**) especifican como un programador de un sistema puede extender o modificar el comportamiento de un servicio middleware. No todos los servicios middleware ofrecen un SPI.

- *Communications Interface:* Las interfaces de comunicaciones (**CI**) (o aplicaciones al nivel de protocolos de comunicación) especifican como los componentes middleware, corriendo en diferentes sistemas en una red, intercambian datos e información de control, ambos entre ellos mismos y con otros componentes. Casi todos los servicios middleware tienen un CI. Sin embargo, en pocos casos, el CI es **privado** (no documentado).

- *User Interface:* La interface gráfica (**UI**) especifica como los usuarios interactúan con el sistema de información. Todos los servicios de presentación tienen una UI.

- *Data Interface:* La interface de datos (**DI**) especifica la sintaxis y la semántica para soportar los formatos de los datos. Pocos servicios ofrecen DI, pero la mayoría de los servicios definen el acceso a datos a través de APIs solamente, manteniendo su DI privada.
- *Management Interface:* Interface de administración (**MI**) es la administración equivalente para cada uno de los tipos primarios de interfaces.

La siguiente figura muestra como se representan estas cinco interfaces primarias (API, SPI, CI, UI y DI). Esta también muestra como estas interfaces están relacionadas a las aplicaciones y a los varios tipos de sistemas.

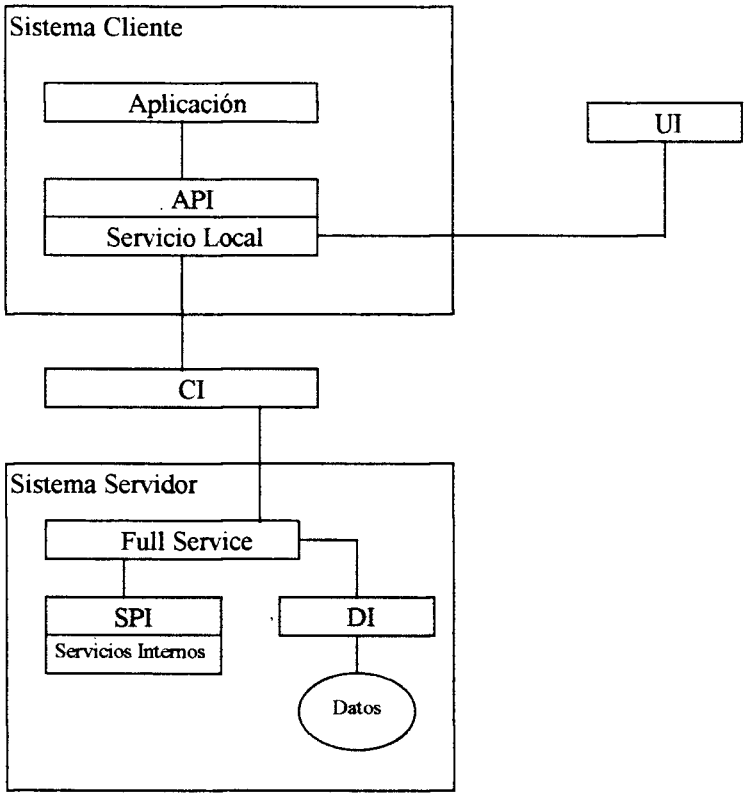


Figura 5.1.- Interfaces Middleware.

5.1 APPLICATION PROGRAMMING INTERFACE.

La interface de programas de aplicación (API) es el mecanismo por el cuál los programas de aplicación interactúan con un servicio middleware. Para invocar las funciones de un servicio, los desarrolladores de aplicaciones usan el servicio API.

Cada especificación API describe la sintaxis abstracta y la semántica del servicio, incluyendo definiciones de parámetros y reglas de uso. Lenguajes específicos de programación para "atar" (Binding) proporcionan la sintaxis concreta para los APIs.

La especificación de un API, aunque es crucial, no especifica por sí sola bastante información para los desarrolladores de aplicaciones para invocar un servicio.

Los desarrolladores requieren otros dos tipos de información, los cuales incluyen las siguientes especificaciones:

- *Especificaciones del lenguaje de programación*: Estas describen la sintaxis y la semántica de operaciones comunes. Estas especificaciones son aumentadas por aplicaciones guías para ayudar a los programadores a escribir código portable y fácilmente distribuible.

- *Especificaciones del entorno del sistema*: Incluye todo lo que los desarrolladores necesitan conocer para usar la capa del sistema correctamente cuando construyen aplicaciones. Quizás la información mas importante incluida en las especificaciones del entorno del sistema sea la lista de servicios disponibles a los desarrolladores para traer información del sistema.

Estas especificaciones son conocidas en la industria del software como "Perfiles de Servicios API".

5.1.1 Interface de Programación del Sistema.

Una interface de programación del sistema (SPI) es el mecanismo por el cuál los programadores de sistemas pueden extender o modificar un servicio middleware o una estructura. Una SPI es idéntica en forma a un API, pero esta es una interface de mas bajo nivel, por lo tanto, no es usual que los programadores usen SPI. Los programadores de aplicaciones usan los servicios middleware para implementar la lógica de las aplicaciones, pero no modifican el comportamiento de los servicios.

Pocos servicios middleware ofrecen una SPI. Por ejemplo, los servicios de messaging y la estructura de monitor de transacciones ofrece SPIs. Estos SPIs permiten a los desarrolladores adicionar soporte para transportes de red adicionales (messaging service)

y usuarios adicionales (transaction processing monitor framework) más allá de aquellos soportados por los servicios de implementación.

5.1.2 Interface de Comunicaciones.

La interface de comunicaciones (CI) es el mecanismo por el cuál varios componentes middleware, corriendo en diferentes sistemas en una red, pueden intercambiar datos e información de control. Los implementadores de servicios usan aplicaciones al nivel de protocolos de comunicación para implementar interfaces de comunicaciones. Algunos servicios soportan protocolos alternativos para permitirles interoperar con un diverso conjunto de componentes distribuidos.

Los desarrolladores de aplicaciones raramente necesitan saber acerca de los CIs porque los servicios middleware están en un nivel alto y transparentemente distribuidos. Sin embargo, los CI son de interés para los integradores de sistemas porque ellos necesitan saber que protocolos usa un servicio middleware para permitir al servicio interoperar con otros componentes de sistemas distribuidos middleware o no-middleware.

La CI describe a los protocolos de la aplicación los servicios que pueden usar para cada tipo de transporte de red (por ejemplo, TCP/IP, OSI, DECnet, PC o LANs). Llamadas a Procedimiento Remoto (RPC) implícitamente define el protocolo de comunicación para algunos servicios basados en la descripción de interfaces de los servicios de red. El uso de RPC permite al servicio desempeñar funciones distribuidas.

La CI también incluye Gateways para otros servicios middleware o no-middleware que están disponibles en la red (por ejemplo, para comunicarse con un sistema IBM).

Con excepción de pocos servicios de solamente uso local, tales como multihreading service, todos los servicios middleware tienen una CI, aunque algunos son privados.

5.1.3 Interface del Usuario.

La interface del usuario (UI) es el mecanismo por el cual los usuarios interactúan con el sistema de información. Una interface del usuario puede incluir entrada, salida o ambas. La UI debe manejar una variedad de tipos de dispositivos y un amplio rango de roles del usuario, lenguajes y culturas.

Los servicios de presentación middleware manejan interfaces del usuario. Estos servicios normalmente definen un mecanismo básico de interacción, el cual algunas veces también incluye un estilo, juntos, estos son comúnmente conocidos como el "look and feel" de la UI. Los desarrolladores de aplicaciones usan los mecanismos de servicios de presentación básicos y agregan sus propios estandares UI.

Los dispositivos de UI incluyen computadoras personales (corriendo Windows), estaciones de trabajo que usan Motif y el sistema X Windows, terminales de despliegue de vídeo e impresoras, otros incluyen reconocimiento de voz y equipo text-to-speech, escaners de imágenes y tableros gráficos.

Los servicios de presentación proporcionan el mecanismo de interface gráfica que los desarrolladores de aplicaciones pueden usar para crear las aplicaciones de UI.

5.1.4 Interface de Datos.

Una interface de datos (DI) es una estructura de información que permite a otra rutina intercambiar (leer y/o escribir) datos con un servicio sin ir a través de un mecanismo de llamada o messaging. La DI incluye almacenamiento de formatos y cualquier sintaxis de datos que los programadores o usuarios puedan requerir.

El middleware promueve que el uso de definiciones estandares de datos y especificaciones para el intercambio de formatos para lograr interoperabilidad en las aplicaciones. Los mecanismos DI incluyen file passing (usando SGML), shared database access (usando esquemas SQL) o common memory (usando el X Window System paste buffer).

Pocos servicios middleware ofrecen una DI. Por ejemplo los servicios de impresión especifican un número de formatos de datos, incluyendo PostScript que son entendidos por los servicios de impresión. Para muchos servicios middleware el DI es privado. A pesar de comunicarse con el servicio a través de grandes estructuras de datos, los programadores usan los servicios API para pasar pequeños ítems de datos como argumentos para operaciones específicas. Este uso del servicio API proporciona alguna flexibilidad.

5.1.5 Interface de Administración.

La interface de administración (MI) permite a los administradores del sistema y a los programadores instalar, monitorear y controlar información. La MI para un componente es actualmente un conjunto de interfaces y consiste de la administración equivalente para cada uno de los tipos de interfaces primarias (API, SPI, CI, UI y DI). En otras palabras, la MI puede incluir un API de administración, una CI de administración, una SPI de administración, una UI de administración y/o una DI de administración.

La siguiente figura muestra como se organiza la interface de administración con las interfaces primarias.

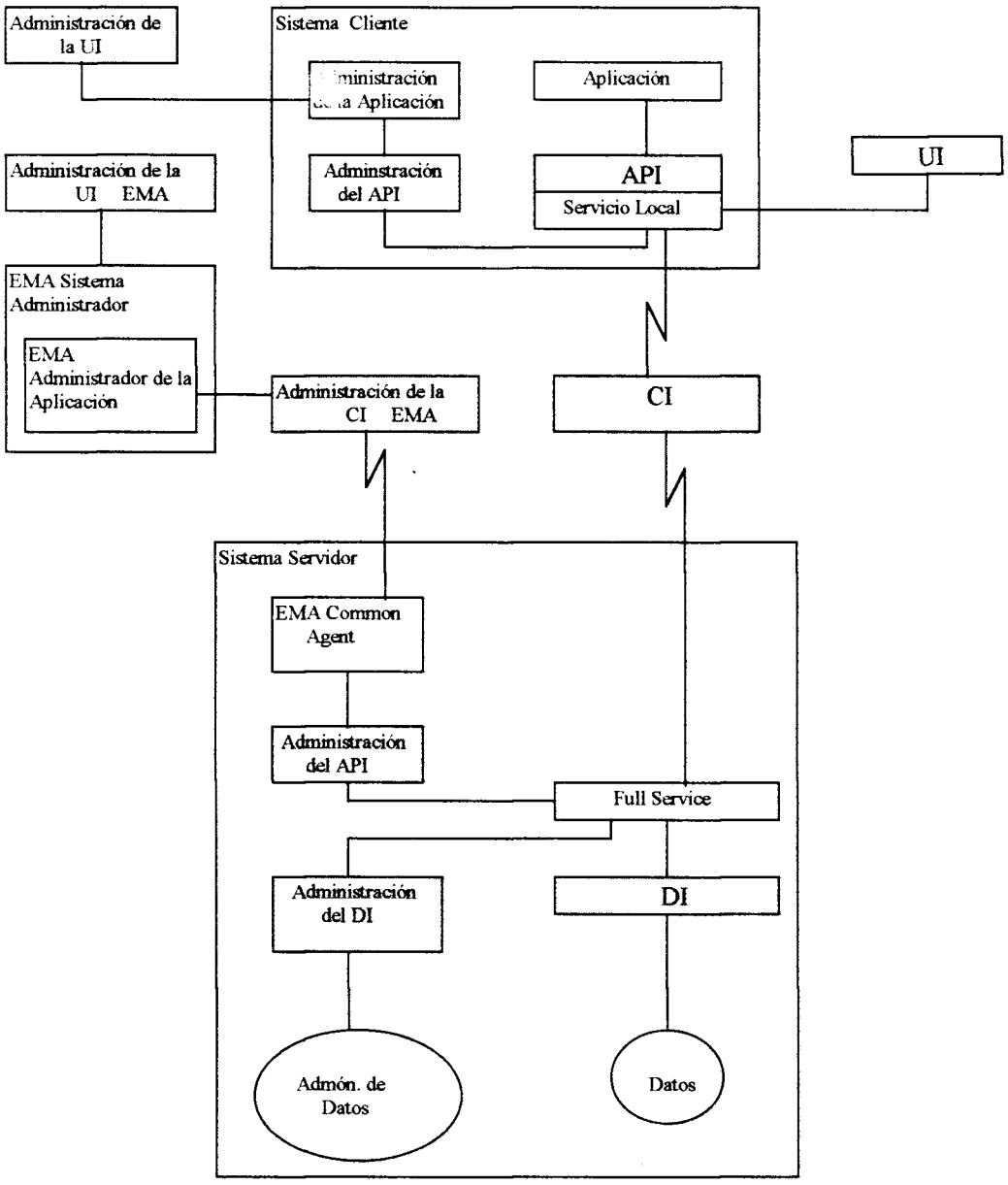


Figura 5.2.- Interfaces de Administración Middleware.

NOTA: El acronimo EMA aparece cuando la interface o componente de administración es de acuerdo a Enterprise Management Architecture.

5.2 ATRIBUTOS MIDDLEWARE.

El objetivo básico de middleware es el que la empresa sea capaz de crear, operar y envolver sistemas de información empresariales (enterprise-wide) que conozca las necesidades de los usuarios, sea flexible y efectivamente costeable. Para lograr este objetivo, una empresa debe establecer una **infraestructura** común - una que ofrezca un conjunto uniforme de servicios de red y que mantenga estas propiedades claves de manera uniforme (tales como apego a ciertos estandares) a través de todos los nodos en la red. De manera que en la red exista una sola manera de hacer las cosas en todos los nodos y todos los componentes puedan trabajar juntos fácilmente.

Los atributos middleware son:

1. Usabilidad (Usability).
2. Distributabilidad (Distributability).
3. Integración (Integration).
4. Apego a estandares (Conformance to standards).
5. Extensibilidad (Extensibility).
6. Internacionalización (Internationalization).
7. Administrabilidad (Manageability).
8. Performance.
9. Portabilidad (Portability).
10. Confiabilidad (Reliability).
11. Escalabilidad (Scalability).
12. Seguridad (Security).

5.2.1 Usabilidad Controla los Otros Atributos.

El propósito fundamental de un Sistema de Información es el de mejorar la efectividad de la gente (individuos y equipos), de acuerdo con esto usabilidad es el atributo primario que controla a los restantes.

En una empresa, uno de los grandes retos es compartir información a través de gente que esta geográficamente dispersa y que necesita trabajar en conjunto, en este punto la distributabilidad e integración es el siguiente atributo más importante. Estos atributos son esenciales para que una plataforma de Tecnología de Información (TI), sea eficiente, efectiva y proporcione soluciones flexibles.

5.2.2 Usabilidad (Usability).

Usability es el grado en el cual un sistema o sus componentes ayudan a los usuarios a realizar sus trabajos eficiente y efectivamente. Usabilidad tiene muchos contextos diferentes

como son usuarios y sistemas. Usabilidad significa diferentes cosas para un administrador de sistemas, para un desarrollador y para un usuario.

Los siguientes ejemplos muestran como otros atributos proporcionan usabilidad:

- Distributabilidad- Los usuarios finales pueden trabajar más eficientemente cuando no tienen que preocuparse de donde están los datos, en que nodo debe correr la aplicación o cómo intercambiar información con otro usuario.

- Integración- Los integradores de sistemas pueden construir un nuevo servicio fácilmente cuando este está basado en un formato común, tal como un directorio de servicios.

- Extensibilidad- Los usuarios finales pueden usar los sistemas más fácilmente cuando el sistema anticipa las necesidades de los usuarios.

- Performance- Los administradores de sistemas pueden administrar un sistema más fácilmente cuando un sistema se desempeña bien sin la necesidad de hacer constantes ajustes.

- Confiabilidad- Los programadores pueden desarrollar programas más fácilmente cuando los servicios de la red (capas bajas), son confiables.

5.2.3 Distributabilidad (Distributability).

Distributability proporciona a los clientes y componentes de un servicio, herramienta o aplicación para que sea ejecutada a través de múltiples componentes de hardware de la red. Distributabilidad se extiende de ninguno (cuando todos los componentes corren en una computadora) a distributabilidad completa (cuando cualquier componente puede correr en cualquier computadora con código completamente automático y migración de datos).

Los objetivos del middleware de distributabilidad incluye soporte para las siguientes capacidades:

- Distribución transparente de servicios para facilidad de uso: El cliente de un servicio no necesita saber si un servicio está distribuido o donde está localizado el servidor. Es posible diseñar, contruir, debug, probar, ejecutar y administrar aplicaciones y componentes de tal manera que la localización de los componentes no sea importante. La distribución no debe ser transparente solo para los programas de aplicación, pero el entorno de desarrollo de la aplicación por sí mismo debe ser transparentemente distribuido.

- Múltiples modelos de distribución, con énfasis en el modelo Cliente/Servidor.

- Distribución de datos, procesamiento y funciones de presentación. En la siguiente figura se muestran como pueden ser las aplicaciones de distribución y los servicios dentro de las diferentes categorías de funciones de software, resultando en diferentes estilos de distribución. Los desarrolladores pueden usar alguno o todos estos estilos sequencial o simultáneamente.

- Operación inmediata o por queued (para permitir al cliente y al servidor ser distribuidos en el tiempo y el espacio.

- Distribución geográfica (LANs, WANs).

- Distribución en todas las fases del ciclo de vida del sistema.

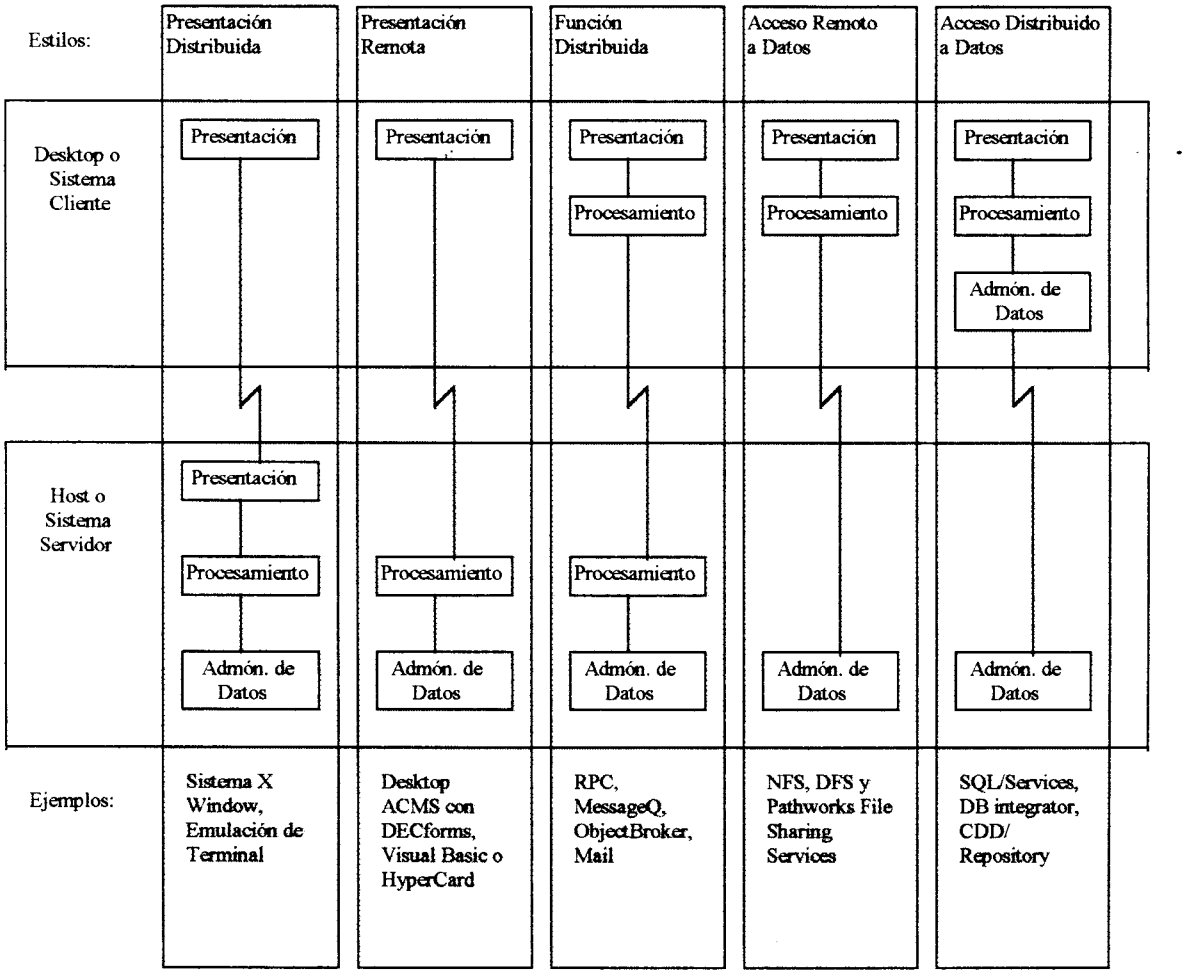


Figura 5.3.- Estilos de Procesamiento Distribuido.

5.2.4 Integración (Integration).

Integración es la habilidad de las aplicaciones para trabajar juntas, desempeñando tareas de forma consistente para el usuario. **Interoperabilidad** y **Uniformidad** son dos aspectos claves de integración.

Interoperabilidad es el grado en el cual un conjunto de componentes invoca e intercambia información efectivamente, tales como traer datos dentro y fuera de un componente o convertir datos como sea necesario. Uniformidad es el grado en el cual un conjunto de componentes son constantes con respecto a un conjunto de atributos.

Interoperabilidad es una condición necesaria, pero no suficiente para lograr la integración. Por ejemplo, dos sistemas pueden ser completamente interoperables, a través de un gateway, con cada uno proporcionando un conjunto completamente diferente de interfaces (interface del usuario, API, interface del sistema e interface de comunicación) y tener un bajo grado de integración. Por esta razón, la uniformidad es el aspecto clave de la integración.

Un alto grado de uniformidad, particularmente de uniformidad de interfaces, reduce la complejidad del sistema y hace el resultado del sistema más fácil de entender, crear, usar, modificar y administrar. La uniformidad de interfaces también ayuda a acomodar la diversidad encontrada en diferentes sistemas.

5.2.4.1 Trabajando con otros sistemas.

Mientras es verdad que un sistema construido con servicios middleware seguido de buenas prácticas de ingeniería de software puede ser altamente integrado, es raro encontrar un sistema de información donde todos sus componentes estén construidos en una base común de tecnología. Por el contrario, los sistemas de información normalmente contienen un gran porcentaje de aplicaciones y otros componentes desarrollados en bases tecnológicas múltiples y usando diferentes interfaces. Los sistemas necesitan interoperar con importantes aplicaciones (incluyendo sistemas existentes, Legacy Systems) a través de la empresa que no usan las mismas interfaces. No es necesario modificar el software existente (legacy software) para lograr interoperabilidad.

Para lograr la integración de los sistemas con legacy software, varias técnica pueden ser usadas:

- Gateways- Componentes que conectan una interface de comunicación con otra haciendo conversiones entre los sistemas de información (por ejemplo, un mail gateway que conecta IBM/PROFS y el producto de digital MAILworks 400).

- Convertidores- Converters, Componentes que traducen de una representación a otra (por ejemplo, un convertidor DDIF Digital Document Interchange Format a SGML Standard

Generic Markup Language); los Gateways usualmente tienen convertidores incrustados dentro de ellos.

- Encapsulación- El ocultamiento de una o más implementaciones específicas para una interface común (por ejemplo, un mensaje de servicio abstracto que puede usar uno o más servicios de comunicación para hacer la transferencia del mensaje).

Gateways, convertidores y encapsulación usualmente permiten interoperación para solamente un subconjunto de funciones que es común entre las diferentes interfaces, representaciones y servicios.

5.2.4.2 Ajuste a estandares (Conformance to Standards).

La mayoría de los servicios middleware implementan y/o usan capas por encima de los estandares de jure y de facto e interoperan usando estos estandares. Estos estandares forman un conjunto de objetivos que proporcionan a una empresa a conocer la demanda de un sistema abierto.

Existen muchos estandares ya establecidos o desarrollados. Donde existen estandares API los servicios middleware usan estos estandares, donde no existen, es necesario desarrollar interfaces estandarizadas para nuevas tecnologías.

Los ajustes del middleware a estandares incluye soporte para los siguientes perfiles globales:

- Base system profile: Es el fundamento de una estrategia de sistemas abiertos para una empresa proporcionando partes de non-networking de la interface del sistema (SI) que soporta componentes middleware de alto-nivel. Un ejemplo de un perfil de sistema base es Spec 1170 de X/Open, el cual es una especificación de los APIs para el sistema UNIX.

- Networking profile: Este puede incluir International Organization for Standardization (ISO) Open Systems Interconnection (OSI) standards, estandares de facto (como TCP/IP) y NetWare.

- Distributed application environment profile: El OSF Distributed Computing Environment (DCE) es el único perfil existente en la industria que cubre el entorno de aplicaciones distribuidas.

- Language standards profile: Un perfil de lenguaje estándar para una empresa debe contener lenguajes estandares de la ISO y American National Standards Institute (ANSI).

5.2.5 Extensibilidad (Extensibility).

Extensibilidad es la facilidad con la cual un sistema puede ser adaptado para conocer nuevos requerimientos. Extensibilidad incluye la habilidad para adicionar o cambiar una función o dato (tipo de dato, formato de archivo, esquema de la base de datos o modelos de información) *sin* los siguientes efectos:

Requerir cambios a las funciones, datos e interfaces existentes.

Introducir efectos colaterales inesperados (tales como degradación del performance, confiabilidad, mantenimiento o portabilidad).

Existen tres tipos de extensibilidad:

- Customization

Customization capacita a los clientes o usuarios finales para cambiar las características de presentación del sistema (la interface del usuario), tales como cambiar el color de la pantalla.

- Configuration

Configuration capacita a los instaladores o integradores de sistemas a cambiar el sistema y sus componentes, tales como hacer cambios en un lugar específico a los parámetros de la base de datos para conocer especificaciones del mercado, requerimientos de la plataforma o estandarización de los componentes para tener la misma interface del usuario y decrementar costos de capacitación y hacer la resolución de problemas más fácil.

- Evolution

Evolución capacita a los arquitectos y desarrolladores de los componentes para cambiar los componentes internos, tales como cambiar la base de datos.

5.2.6 Internacionalización (Internationalization).

Internacionalización es el grado en el cual un sistema de información es apropiado para usuarios y datos de diferentes culturas. (Cultura es más específica que lenguaje. Por ejemplo, el chino se habla en la República de China, en Singapur, en Hong Kong y en Taiwan, pero con diferentes culturas que separan las necesidades de procesamiento e interface del usuario).

Existen dos dimensiones clave de internacionalización:

- Monto de reingeniería requerida.

- Número de culturas soportadas.

Una aplicación internacional no debe requerir cambios en su fuente para soportar datos multilingües. Una aplicación totalmente internacional no debe requerir reingeniería para soportar datos multilingües (datos que tienen una mezcla de lenguajes).

5.2.7 Administrabilidad (Manageability).

Administrabilidad es el grado en el cual los administradores de sistemas pueden configurar, monitorear, diagnosticar, mantener y controlar los recursos de un entorno computacional de manera económica. La administración de sistemas distribuidos es un círculo de proceso de retroalimentación. Los administradores de sistemas monitorean el status de los recursos distribuidos y proporcionan controles de entrada para estos recursos, causando que el sistema distribuido opere de acuerdo a las políticas y metas de la compañía. Administrabilidad es un factor crítico de costo para la operación de sistemas heterogéneos, enterprise-wide e información distribuida.

Las metas para el middleware de administrabilidad incluye soporte para las siguientes capacidades para asegurar que los servicios middleware, aplicaciones y plataformas en las cuales trabajan sean constante y remotamente administrables.

- Administración de todos los recursos en el entorno, incluyendo:
    - Componentes del sistema (procesos timesharing y batch, estaciones de trabajo, Pcs, periféricos y storage media).
    - Componentes de red (ambos físico y lógico).
    - Software (sistemas operativos, software servidor, herramientas y aplicaciones).
    - Bases de Datos.
- Administración del ciclo de vida entero de estos recursos, incluyendo diseño, planeación, desarrollo, instalación, operación, mantenimiento y evolución.
- Soporte para todas las funciones de administración aplicable a estos recursos, tales como administración de la configuración, administración de fallas, desempeño, administración, contabilidad y seguridad.

5.2.8 Desempeño (Performance).

Desempeño es la medida de los recursos requeridos para desempeñar una operación. La métrica de medida del desempeño más común es el tiempo transcurrido (elapsed time) para desempeñar una operación. (El tiempo transcurrido desde el final de la entrada del usuario hasta el despliegue de los datos de salida es llamado **tiempo de respuesta**). El

desempeño puede también incluir medidas de el uso de otros recursos tales como el procesador, comunicaciones, entrada/salida y memoria.

Eficiencia es el porcentaje de recursos usados para el número de operaciones desempeñadas. **Utillización** es el relativo uso de un recurso. **Throughput** es la medida de el número de operaciones completadas en una unidad de tiempo. Un buen desempeño depende de una buena arquitectura y buenas implementaciones.

5.2.9 Portabilidad (Portability).

Portabilidad es la facilidad con la cual los desarrolladores pueden mover software de una plataforma a otra. Un alto grado de portabilidad es crucial para el éxito de un entorno de computación heterogéneo. Sin este, el software puede no estar disponible en ciertas plataformas o el costo para portar el software puede ser alto.

Para lograr portabilidad de las aplicaciones a través de las plataformas, el middleware tiene que se como sigue:

- Soportar portabilidad de las aplicaciones basadas en los perfiles, donde el software depende únicamente en lenguajes, servicios API y protocolos definidos por un perfil (esto es, un estándar aprobado). El software es portable a través de todas las plataformas que soportan dicho perfil.

- Proporcionar interfaces portables para servicios middleware a través de las plataformas.

- Soportar múltiples dimensiones de portabilidad, incluyendo APIs, programas, datos, usuarios, documentación y herramientas de desarrollo.

- Asegurar que los servicios middleware apropiados, protocolos, herramientas de desarrollo y otros componentes sean soportados cuando se necesiten.

5.2.10 Confiabilidad (Reliability).

Confiabilidad es el grado en el cual un sistema o parte de un sistema, produce salidas correctas en diferentes procesos (sin producir efectos no deseados).

La confiabilidad es medida de el tiempo entre fallas (MTBF), donde falla es definida como indisponibilidad de un sistema o parte de el. Disponibilidad es el porcentage de tiempo que el sistema esta disponible para producir resultados esperados. La disponibilidad es medida por MTBF/(MTBF+MTTR), donde MTTR es el tiempo para reparar o recuperar la falla.

La necesidad de construir aplicaciones confiables va más allá del dominio del procesamiento transaccional tradicional. Conforme las técnica de alta-confiabilidad se vuelven más amplias y con mejor relación cost-efectivas, más aplicaciones dominantes pueden confiar en ellas.

5.2.11 Escalabilidad (Escalability).

Escalabilidad es el grado en el cuál los desarrolladores pueden aplicar una solución a diferentes problemas. De una manera ideal, una solución debe trabajar bien a través de un gran rango de complejidad. En la practica, sin embargo, hay soluciones simples para problemas de baja complejidad. El problema viene cuando un problema de baja complejidad se vuelve con el tiempo en un problema de alta complejidad.

En un sistema de información del tipo enterprise-wide, algunas de las complejas medidas clave son el número de nodos del sistema, usuarios y otros recursos de la red. Esta complejidad requiere escalabilidad de la red, el cual es el grado en el que los componentes del sistema de información (tales como servicios, herramientas y aplicaciones) usan la infraestructura computacional efectiva y eficazmente para manejar problemas pequeños y grandes.

5.2.12 Seguridad (Security).

Seguridad es la protección de la información de modificaciones no autorizadas y la protección de los recursos de usos no autorizados.

Las metas para los servicios de seguridad incluyen la implementación de autentificación, control de acceso y auditar funciones para asegurar que lo siguiente ocurra:

- El servicio puede confiablemente identificar el usuario o sistema en los cuales una operación esta siendo ejecutada.
- El contexto de seguridad de el usuario o sistema que hace un requerimiento esta correctamente limitado para ejecutar el software.
- La información creada por las aplicaciones y servicios para un usuario o sistema permanece adecuadamente protegida de otros usuarios o sistemas.
- El software ejecutado esta limitado para usar solamente aquellos objetos y operaciones para los cuales el usuario o sistema que hace el requerimiento esta autorizado.
- Las aplicaciones que administran sus propios objetos tienen la habilidad de ejecutar acceso a estos objetos de una manera consistente.
- Las aplicaciones pueden soportar eventos de seguridad relevante.

5.2.13 Estandares y Perfiles (Standards and Profiles).

Los estandares han jugado un importante rol en los sistemas de información desde los tempranos esfuerzos por estandarizar los lenguajes de aplicación. El primer esfuerzo de estandarización trato de enfocar a los participantes en aplicaciones portables de software que pudieran correr en una variedad de sistemas. Hoy, la estandarización de la tecnología de información se enfoca no solamente en la portabilidad, sino también en la interoperabilidad y la usabilidad. En respuesta a requerimientos cambiantes, los estandares de la tecnología de información ahora cubren interfaces de sistemas operativos, computación distribuida, interfaces del usuario y mucho más.

5.3. ESTANDARES DEFINIDOS.

Los estandares son un punto de referencia y no una solución completa para una tecnología de información, así como un producto no es una solución completa para cada aspecto de un conjunto especifico de problemas de negocios. Los usuarios de la TI son fuertemente soportados por estandares, especialmente estandares que facilitan el desarrollo de sistemas abiertos y proporcionan control de la tecnología a los usuarios.

El middleware incorpora estandares de la industria como son:

- Estandares De Jure

  Los servicios middleware y sus interfaces están basados algunas veces en estandares de jure. Los estandares de jure son aquellos que han sido creados por organizaciones desarrolladoras de estandare formalmente reconocidas tales como la International Organization for Standardization (ISO), el American National Standards Institute (ANSI) , y el Institute of Electrical and Electronics Engineers (IEEE). Los estandares de la IEEE incluyen la Portable Operating System Interface (POSIX) 1003.n standards. Estos estandares son desarrollados bajo las reglas de un consenso en un foro abierto.

El uso de estandares de jure proporcionan servicios middleware para expander el rango de plataformas y redes comúnmente encontradas en sistemas de información del tipo entreprise-wide. Los estandares de jure son especialmente importantes para los cliente de la TI por las siguientes razones:

- La mayoría de los proveedores que ofrecen productos de tecnología está apegados a los estandares de jure.

- Los estandares de jure enriquecen la interoperabilidad y la portabilidad.

- Estandares De Facto

  Algunos servicios middleware también incorporan estandares de facto populares, especialmente aquellos usados para conectar PC´s a hosts IBM. El termino estándares de facto se aplica a un producto o sistema que es ampliamente usado y que algunos proveedores tienden a emular, copiar o usar (MS-DOS es un ejemplo de un estándar de facto).

Muchos servicios middleware también incluyen interfaces para integrar datos y aplicaciones que no cumplen con los estandares de la industria, tales como mail Gateways. Estas interfaces son popularmente usadas para proteger inversiones en tecnología existentes.

- Especificaciones

Las especificaciones describen características de hardware y software, pero no son estandares. Sin embargo, Las especificaciones de la tecnología de información están eventualmente aprobadas como estandares. Ejemplos de estas especificaciones son la especificación del sistema X Window, la especificación OSF/Motif y la *X/Open Portability Guide*.

5.4 TIPOS DE GRUPOS DE ESTANDARIZACION.

Existen varios tipos de grupos que producen estandares, así como consorcios y grupos de usuarios creando especificaciones para estandarización. Estos grupos incluyen lo siguiente:

- Organizaciones desarrollando estandares

1. Standards developing organizations (SDOs), son organizaciones acreditadas de voluntarios que buscan desarrollar estandares formales o de jure reconocidos. Las SDOs llevan a cabo la acreditación de estandares por consenso, el cual asegura que todas las partes interesadas puedan ayudar a desarrollar un estándar, las SDOs pueden ser organizaciones internacionales, regionales o nacionales como:

2. SDOs Internacionales: Tienen el más alto nivel de estandarización. Sus recomendaciones son reconocidas mundialmente. Las SDOs internacionales incluyen organizaciones tales como la International Organization for Standardization (ISO) y la International Electrotechnical Commission (IEC).

3. SDOs Regionales: Las más prominentes SDOs fueron fundadas en Europa. El rol de las SDOs Europeas a aumentado debido al establecimiento del mercado común europeo de comercio, este incluye el Comité Européen de Normalisation (CEN) que es el equivalente europeo de ISO.

4. SDOs Nacionales: Estas organizaciones administran los esfuerzos de estandares nacionales, la mayoría de las organizaciones nacionales son miembros de la ISO. Cada SDO nacional refleja el punto de vista de estandarización de cada nación. Las SDOs nacionales incluyen organizaciones tales como el American National Standards Institute (ANSI) y la British Standards Institution (BSI).

5. Organizaciones de estandares acreditadas por ANSI: Estas organizaciones desarrollan estandares para sus propias areas de interés. Estas organizaciones incluyen comites tales como (ASC Xn) y la IEEE.

- Regional workshops

Están involucrados con el desarrollo de estandares en un área especifica. Por ejemplo, la European Workshop on Open Systems (EWOS) esta involucrada con el desarrollo de estandares OSI para el desarrollo de perfiles estandares internacionales para el dibujo de ISO.

- Consorcios, asociaciones y grupos de usuarios.

Un consorcio es una asociación de compañías, vendedores o usuarios unidos con un fin común. Los consorcios no desarrollan estandares, pero ellos pueden indirectamente influenciar los procesos de estandarización. Algunos ejemplos de consorcios son:

- Open Software Foundation (**OSF**)
- **X/Open**

Otros grupos o consorcios influencian uno o más de los servicios middleware incluyendo el Internet Engineering Task Force (**IETF**), Multivendor Integration Arquitecture (**MIA**) Consortium, Object Management Group (**OMG**) y MIT **X Consortium**.

5.4.1 Perfiles Estándar.

Los perfiles son una serie de estandares que son usados juntos para soportar una función particular. Los perfiles describen un entorno de computación en particular. Este entorno puede ser limitado para una aplicación especifica o general tal como definir todos los estandares necesarios para una compañía.

La organización Internacional de Estandares (OSI) desarrollo el concepto de perfiles de su trabajo con varios cientos de estandares OSI, de este gran grupo, los usuarios necesitaban seleccionar una serie de estandares para la exitosa interconexión de sistemas y para necesidades funcionales especificas. Como resultado, muchas compañías y gobiernos han desarrollado su propio conjunto de estandares como un subconjunto del gran conjunto de estandares ISO. Este subconjunto es la lista de estandares aceptables para dicha compañía o gobierno. Estos subconjuntos de estandares son llamados perfiles.

5.4.2 Quién Desarrolla Perfiles.

Además de OSI, muchos grupos formales desarrollan perfiles, tales como:

- X/Open Company, Limited.

X/Open, una compañía internacional no lucrativa, construye perfiles para desarrollar especificaciones basadas en estandares internacionales formales y estandares de facto de la industria. Inicialmente, X/Open desarrollo un perfil llamado Common Applications Environment (CAE) que estaba basado en muchos estandares que describían interfaces de programación y protocolos de red. De el CAE, X/Open desarrollo la X/Open Portability Guide (XPG) Base Profile, la cual describe un mínimo conjunto de estandares para un sistema. X/Open está ahora definiendo conjuntos de componentes de el XPG que describe un perfil de una estación de trabajo, perfil de la base de datos y muchos otros. Muchas compañías y gobiernos usan los perfiles de X/Open como una guía para apegarse a los estandares.

- European Commission

La Comisión Europea (EC) usa la construcción de perfiles para tratar de tener productos apegados a los estandares. Un producto clave del trabajo de la EC es el European Procurement Handbook for Open Systems (EPHOS). Este libro proporciona información sobre los estandares y especificaciones sean de acuerdo a los estandares OSI desarrollados por la ISO.

La primera fase de EPHOS cubre las siguientes areas técnicas y de negocios:

- File transfer, Acces, and Management (FTAM) para controlar transferencias de información entre sistemas potencialmente distantes.

- Servicios de Red públicos y privados (X.400) para enviar información entre sistemas.

- Servicios de red de área amplia (WAN) con X.25 para la unión electrónica de sistemas remotos.

- Network Management Forum

Es un consorcio que incluye a la mayoría de las compañías de telecomunicaciones del mundo y a sus proveedores, formando un grupo llamado Service Provider Integrated Requirements for Information Technology (SPIRIT). El propósito de SPIRIT es el de desarrollar especificaciones de compra para plataformas de computación abiertas estandarizadas.

5.4.3 Quién Usa los Perfiles.

Los perfiles sirven a múltiples propósitos y funciones para varios usuarios dependiendo de sus necesidades. Un perfil típico de usuarios puede ser el siguiente:

- Clientes de Tecnología de Información

Las compañías usan perfiles cuando se preparan a comprar tecnología de información y también construyen perfiles para necesidades especificas de la compañía. Por ejemplo, una compañía puede necesitar un perfil que solo comprenda la computación cliente/servidor. Si no existen perfiles que comprendan esta área completamente, un consultor de TI puede usar componentes (productos estándar) de múltiples perfiles para construir el perfil necesitado.

- Consultores en Tecnología de Información

Los consultores en TI evalúan perfiles contra un conjunto de requerimientos de la compañía y también construyen perfiles específicos de la compañía, la gente que construye perfiles es llamada "profile writers".

- Desarrolladores de Software

Los desarrolladores de software hacen uso de los perfiles, especialmente de los perfiles más reconocidos de la industria como son X/Open, OSF y POSIX. Cuando los desarrolladores de software crean productos que cumplan con los mejores perfiles de sistemas abiertos, los productos se vuelven más fáciles de comercializar.

5.4.4 Beneficios de los Perfiles.

Existen varios posibles usos de los perfiles, como son:

- Para definir estandares de un entorno particular

- Para seleccionar productos abiertos basados en estandares o en APIs estandares

Los perfiles son especialmente usados por compañías enfocadas a los sistemas abiertos, sistemas downsizing o aplicaciones cliente/servidor. Los consultores de TI pueden usar los perfiles cuando evaluan las necesidades de una compañía para aplicar soluciones en TI.

- Para definir las relaciones entre estandares, particularmente en términos de interoperabilidad y portabilidad

Por ejemplo, los perfiles tienen coherencia, lo cual significa que los estandares usados dentro de un perfil pueden conectarse apropiadamente y trabajar juntos.

- Para identificar las partes que los perfiles de los usuarios deben comprender a través del uso de opciones

Las opciones son productos que no están basados en estandares, pero que completan la solución de TI de una compañía. Las opciones pueden tener una estrategia de migración a un producto estándar en un futuro cercano.

5.4.5 Tipos de Perfiles.

Existen diferentes tipos de perfiles, estos se pueden clasificar en diferentes categorías:

Open System Environment (OSE) profiles

Perfiles Middleware
Application Environment Profiles (AEP)

Estas aplicaciones cubren un amplio rango de entornos. Usualmente, el perfil OSE cubre la más amplia gama de perfiles, el AEP describe un entorno limitado y los perfiles middleware están entre ellos dos. La siguiente figura muestra los tipos de perfiles:

	Entorno General OSE	Perfiles Middleware	Application Environment Profiles (AEP)
Estandares Formales	POSIX 1003.n	FTAM	SQL-92 (ISO/IEC 9075)
Estandares Gubernamentales	NIST APP EC Informatics Architecture	FIPS 151-2	Standard SGMLGeneralized Mark-up Language
Estandares de la Industria	X/Open Spec 1170	DCE	X/Open Desktop

Figura 5.4.- Tipos de Perfiles.

5.4.6 Perfiles OSE (OSE Profiles).

Un perfil OSE es un perfil basado en múltiples componentes y referencias a otros perfiles. La lista de componentes forma el criterio de selección básico del cual una persona que construye perfiles extrae componentes para construir un perfil especifico para satisfacer un requerimiento funcional.

Por ejemplo, el National Institute of Standards and Technology (NIST) Application Portability Profile (APP) consiste de una lista de componentes que incluyen estandares formales, de facto y otros perfiles. Una compañía puede desarrollar un subconjunto de estos componentes para describir su entorno y usar este subconjunto como una base para conseguir productos de Tecnología de Información.

5.4.7 Perfiles Middleware (Middleware Profiles).

Los perfiles middleware consisten de componentes y otros perfiles que sirven a una función o a un entorno particular, estos perfiles están más limitados en alcance que un perfil OSE. Son un subconjunto de perfiles OSE. Por ejemplo, el entorno computacional de una empresa puede apegarse a un perfil OSE formado de varios perfiles, incluyendo muchos perfiles middleware, tales como Distributed Computing Environment (DCE).

5.4.8 Perfiles AEP (AEP Profiles).

El perfil AEP comprende un dominio específico de aplicaciones. Estas aplicaciones pueden ser relacionadas a la industria, tales como un entorno bancario, o pueden ser una aplicación genérica, tal como acceso a la información a través de SQL.

CAPITULO VI. UNA COMPARACION DE ALTERNATIVAS MIDDLEWARE.

En este capítulo se encontraran las conclusiones de esta investigación, comparando las diferentes alternativas middleware de acuerdo a las características de cada una.

6.1 Alternativas Middleware.

Existen varias formas diferentes de middleware en el mercado, incluyendo:

Basic Communication Protocols API's. (Por ejemplo CPI-C, TCP sockets).
Remote Procedure Calls (RPCs).
Database Middleware (usualmente SQL-focused).
Message Oriented Middleware.
Homegrown, Middleware desarrollado en casa.

Muchas corporaciones han invertido millones de dólares en desarrollar sus propias soluciones middleware, algunas veces consistentes de una combinación de APIs genéricas de comunicaciones, las cuales proporcionan solo comunicación sincrona.

Otras soluciones middleware necesitan algo de hardware, sistemas operativos e independencia de protocolos de comunicación. Un ejemplo es escribir código para un protocolo de comunicaciones especifico API que es implementado a través de múltiples plataformas de hardware y de software, tales como TCP sockets o conversaciones APPC. Algunos de estos mecanismos tienen una implementación común de API a través de múltiples plataformas (tales como WinSockets e Interfaces de programación para comunicaciones), para ocultar diferencias de comunicación entre protocolos.

Desafortunadamente, existen algunas interfaces que son muy complejas y requieren una gran inversión en entrenamiento de los protocolos de comunicación. Además, la amplia variedad de sistemas operativos, requiere código especializado y limita la portabilidad.

6.1.1 Remote Procedure Calls.

The Remote Procedure Calls (RPC), toma un concepto familiar de programación, llamando una subrutina y aplicándola a un entorno de red. Las interacciones dentro de este modelo son program-to-program y con comunicación sincrona y de bloqueo.

Usando RPC, un desarrollador puede llamar una subrutina que no solamente esta en un nodo remoto, sino que además corre en un diferente sistema operativo. Los proveedores de RPC se han consolidado en un par de mecanismos: El OSF/DCE RPC, y el SUN ONC

RPC. Un RPC puede ser usado por programas de requerimientos/respuestas tales como actualización y query de las bases de datos.

El RPC no es apropiado para todos los tipos de aplicaciones distribuidas. Los RPC son inherentemente sincronos en los procesos y habilitar a los mismos para soportar procesamiento asincrono es complejo y requiere la utilización de complejos sistemas operativos primitivos, esto crea problemas de portabilidad, soportabilidad e interorperabilidad, por ejemplo, un programa utilizando enlaces Solaris no puede ser fácilmente portable a otro sistema operativo utilizando enlaces DCE.

Otras desventajas adicionales de RPC, son que ambos lados, cliente y servidor deben ser enlazados con RPC, y el RPC nativo no reconoce semánticas transaccionales, la experiencia practica con RPCs ha demostrado que puede ser efectivo solo en pequeñas aplicaciones y principalmente en redes de área local (LAN's). En la gran mayoría de las aplicaciones basadas en RPC, hay un numero limitado de clientes (usualmente < 100), y con un mínimo numero de servidores, por lo tanto los RPCs no son escalables para aplicaciones enterprise-wide y soluciones de misión critica.

6.1.2 ¿ Porque Orientación a los mensajes en lugar de RPCs?

MOM soporta comunicación sincrona y asincrona y no esta asociada con ningún conjunto particular de protocolos. MOM es ideal para soportar objetos distribuidos y modelos de aplicaciones peer-to-peer, así como el modelo cliente servidor. Algunos productos de mensajes también reconocen semántica transaccional.

Además, la siguiente generación de aplicaciones distribuidas puede tener la necesidad de soportar modelos mas complejos de distribución. Hay fuertes indicadores de que el modelo requerimiento/respuesta ofrecido por RPC puede simplemente no ser suficiente. Hay muchas aplicaciones en las corporaciones que requieren métodos mas flexibles de computación distribuida.

Muchas aplicaciones necesitan no solamente soportar comunicaciones sincronas y asincronas, sino que también requieren la capacidad de soportar broadcasting/multicasting, notificación de eventos, almacenamiento, completa independencia de protocolos (no solo TCP/IP), y muchas otras capacidades.

La conclusión es que los RPCs no pueden ser usados por simples y relativamente pequeñas aplicaciones cliente/servidor, y que no soportan aplicaciones mas complejas las cuales están basadas en objetos distribuidos o modelos peer-to-peer.

El middleware de base de datos, permite diferentes aplicaciones cliente para accesar datos que están distribuidos a través de la red en múltiples plataformas y almacenados en bases de datos heterogéneas. Para la mayoría, los productos middleware de bases de datos proporcionan su propia API (de manera propietaria), con soporte para ODBC. Otros

productos middleware de bases de datos soportan solamente comunicación sincrona entre cliente y servidor, Fundamentalmente, tales comunicaciones están basadas en un lenguaje de SQL donde un proceso cliente envía un requerimiento de datos al servidor de la base de datos y el servidor de base de datos responde enviando la respuesta de regreso mientras el cliente espera.

6.1.3 ¿Porque Orientación a los mensajes en preferencia a una base de datos Middleware?

En adición para proveer una consistente interface para accesar multiples bases de datos, las soluciones middleware de bases de datos provee soporte para las sesiones de comunicación entre el Front-end y el Back-end y entre múltiples back-ends. En este contexto, el acceso a la base de datos debe soportar muchos sistemas operativos y protocolos de comunicación, soportando una amplia heterogeneidad.

El middleware de base de datos no es suficiente todavía para soportar procesamiento mas complejo en ambos lados (cliente y servidor). Para la mayoría de los productos, el middleware de base de datos esta basado en comunicación sincrona entre cliente y servidor. Esta comunicación esta basada en lenguaje SQL que no entrega la flexibilidad que un programa regular puede proveer. Un modelo SQL implica que no hay lógica del lado del servidor y que toda la lógica del negocio reside en el lado cliente. Esto crea mayores problemas en el área de seguridad de los datos, escalabilidad e interorperabilidad, así como en el desempeño. Además, la ausencia de la lógica del negocio en el lado servidor no proporciona algunas facilidades para filtrar y recuperar datos y como resultado los datos pueden ser transmitidos sobre la red innecesariamente.

Existe una excepción en este tema de que no existe lógica del negocio del lado del servidor. Los procedimientos de almacenamiento de la base de datos pueden ser usados para permitir la existencia de la lógica del negocio en el servidor, pero desafortunadamente, todos los procedimientos de almacenamiento de este tipo implican una solución propietaria, la cual es única a una implementación especifica de DBMS. No son portables a través de diferentes DBMS's y son muy dificiles de administrar. Aunque los procedimientos almacenados pueden ser compilados estos pueden ser un potencial cuello de botella en el desempeño.

Mientras los lenguajes de procedimientos de almacenamiento como Sybase's Transact-SQL pueden se vistos como una extensión de los dialectos tradicionales de SQL, estos todavía se quedan cortos con la riqueza y funcionalidad provista por lenguajes de 3GL y 4GL, estos procedimientos de almacenaje requieren la presencia de una base de datos relacional la cual obliga a conseguir licencias adicionales.

El punto principal es que el middleware de bases de datos es apropiado para soportar aplicaciones de bajo volumen donde los datos almacenados en bases de datos remotas y heterogéneas necesitan ser recuperados. Esto no es apropiado para aplicaciones de misión critica o de alto volumen.

Consideremos el ejemplo de una de las mas grandes compañías para el cuidado de la salud. La compañía proporciona soluciones de sistemas de información para mas de 700 hospitales y 10,000 médicos, proporcionando contabilidad de pacientes y administración de los servicios que se proporcionan a los clientes. Una subsidiaria esta construyendo una nueva infraestructura para el intercambio de la información entre clientes y compañías de seguro de la salud, empleados, agencias de gobierno e instituciones financieras.

La infraestructura de la red debe proveer la manera de combinar una inmensa heterogeneidad entre las empresas y las organizaciones, esta compañía sugirió el uso de procesamiento basado en los mensajes (message-based processing).

Esta compañía necesita cubrir las siguientes necesidades:

- Conectar cientos de usuarios in una red de aplicaciones distribuidas peer-to-peer.
- Proporcionar comunicaciones en tiempo real.
- Soportar trafico que soporte mensajes asincronos y manejo de eventos (event-drive).
- Soportar un amplio numero de plataformas.
- Minimizar el soporte y el tiempo de desarrollo.

A causa de la diversidad de aplicaciones envueltas, el intercambio de la información resultaría mejor con Messaging connectionless messages de forma asincrona.

El numero de de plataformas de hardware envueltas son como siguen: CICS/IMS, VMS, Stratus VOS, OS/2, UNIX, Windows NT, Windows, AS/400, cualquier conexión debe requerir adicional ancho de banda y una compleja topología de red.

La arquitectura cliente servidor basada en SQL, puede ser difícil de construir, si utilizamos una planeación con middleware orientado a los mensajes, podemos aumentar la portabilidad y el tiempo de respuesta de la red.

La necesidad de una capa de middleware ha sido reconocida por muchas de las compañías que forman parte del Fortune 500, algunas de las cuales han desarrollado sus propias soluciones middleware, generalmente, estas son soluciones híbridas las cuales se derivan de un protocolo de comunicaciones API o de una solución universal de acceso de datos o ambas.

Esto llevo a las compañías a enfrentar el problema de elevados costos de mantenimiento, como mas sistemas operativos y protocolos de comunicación entraban en el mercado, las soluciones hechas en casa necesitaban ser actualizadas para soportar estas. Además, como mas aplicaciones se volvían distribuidas, nuevas necesidades de funcionalidad necesitaban ser implementadas, las corporaciones no solo tenían altos costos de mantenimiento sino que además necesitaban aumentar las habilidades del personal interno y además de contratar servicios de soporte externo, lo que hacia de este problema una verdadera pesadilla.

6.1.4 MOM asegura flexibilidad y funcionalidad.

Para soportar diferentes modelos para la distribución y la comunicación, el middleware orientado a los mensajes proporciona alta funcionalidad y flexibilidad arquitectural en diseñar sus aplicaciones para soportar múltiples modelos de comunicación, estos modelos son:

- Modelo process-to-process.
- Message quening Model.

Ambos métodos están basados en mensajes que son enviados entre entidades. Cuando se usa el modelo process-to-process al menos dos procesos son necesitados para mantener una sesión activa entre ellos. En contraste cuando se usa message quening model, el proceso de comunicación entre cada uno de ellos es es queues, sin que exista una sesión entre ellos.

6.1.5 MOM ofrece flexibilidad y rápido desarrollo para las grandes compañías.

Una gran compañía del mercado electrónico ha sido uno de los grandes ejemplos de aplicaciones MOM.

El reto tecnológico que enfrento esta compañía fue enorme, las herramientas deberían ser tan flexibles como fuera posible para acomodar el gran numero de necesidades de aplicaciones de los desarrolladores, el grupo debía crear las herramientas de tal manera que los recursos de desarrollo de las aplicaciones pudieran ser aplicados al desarrollo de aplicaciones que soportaran el negocio, preferentemente que aprender complicadas comunicaciones de bajo-nivel y otros servicios para la computación distribuida. El grupo opto por construir sus herramientas usando tecnología de mensajes por su neutralidad de lenguaje, portabilidad, escalabilidad, capacidades sincronas y asincronas, transparencia de localización y su desempeño, todas criticas para soportar la siguiente generación, programación orientada a objetos y aplicaciones multimedia.

Flexibilidad fue la característica clave que llevo a la compañía a elegir MOM, específicamente las capacidades para soportar un amplio rango de herramientas cambiantes y de adaptación a tecnologías emergentes. La lista de herramientas desarrolladas por el grupo es muy impresionante. Los beneficios que el grupo ha obtenido de el uso de MOM son significantes.

El desempeño del sistema se ha mejorado bastante y el tiempo de desarrollo de aplicaciones dentro de la corporación ha sido reducido substancialmente. Con el uso de estas nuevas herramientas la compañía ha desarrollado un gran catalogo de aplicaciones en una fracción del tiempo normalmente requerido. La característica común de una compleja aplicación es la escalabilidad, confiabilidad y desempeño. Actualmente esta aplicación esta sirviendo mas de 1,000 usuarios utilizando múltiples sistemas operativos (Windows 3.1,

OS/2 y MVS) y protocolos de comunicación (IPX/SPX, TCP/IP y SNA LU6.2) con un tiempo de respuesta de segundos.

Mientras el process-to-process message passing típicamente requiere que ambos procesos estén entregando mensajes al mismo tiempo, este modelo es flexible en mensajes sincronos y asincronos y permite múltiples requerimientos procesados concurrentemente.

Message queuing ofrece un modelo diferente para aplicaciones de control y datos distribuidos. Una aplicación empuja los mensajes en el message queue y toma mensajes de el message queue, sin importar que los queues sean locales o remotos.

Por lo tanto, este modelo permite la comunicación asincrona, independiente en el tiempo entre los procesos. Message queuing es un mecanismo flexible, el cual tiene la capacidad de diferir la entrega de un mensaje y procesa y asegura que el mensaje será entregado al destino solo una vez (asegurando la entrega de los mensajes) y proporcionando recuperabilidad. Un programa distribuido basado en message queuing no tiene que esperar a su compañero para estar disponible y entregar una respuesta a su requerimiento. Este tipo de transmisión asincrona e independencia en el tiempo hace el uso de la red y de sus recursos eficientes. Cuando se usa messaging process-to-process, una falla causada por el servidor o el cliente crea una situación donde la aplicación entera no puede continuar procesando. En contraste, el queuing de mensajes crea una situación en la cual una falla por una de las partes no causa una falla en la aplicación entera.

Una analogía puede ser hecha entre un teléfono y una maquina contestadora, donde la persona que llama puede dejar un mensaje usando la contestadora, de manera que la persona a la que se destina el mensaje puede recuperar este mensaje después, si lo desea.

Message queuing es una buena técnica para aplicaciones que pueden tolerar el tiempo latente incurrido por múltiples componentes de comunicación con cada otro no directamente pero vía message queues. Este es un modelo efectivo para enlazar sistemas independientes.

6.1.5.1 MOM proporciona Flujo de Información Automático a una empresa manufacturera de metal.

Un buen ejemplo es la industria manufacturera del metal con sistemas independientes que manejan orden de captura y horarios de trabajo, control de ventas de piso y despacho de mercancía de las bodegas. La información fluye entre estos sistemas en forma de papel y cintas magnéticas, este flujo manual puede causar errores (como entrada de datos con errores).

Esta compañía eligió el flujo de información de manera automática entre estos sistemas en el intento de reducir problemas y costos.

Técnicamente el diseño necesitaba satisfacer los siguientes requerimientos:

Las aplicaciones existentes necesitaban ser retenidas. El usar un modelo simple de entrada/salida, eligiendo un modelo de message queuing.

La información necesitaba ser movida con seguridad y confiablilidad entre los sistemas, sin una carga adicional para el diseño de las aplicaciones.

Cada sistema operaba con diferentes prioridades y con diferentes escalas en el tiempo. La información necesitaba ser movida al tiempo de ser relevante para las aplicaciones es independiente de las actividades y prioridades del receptor. El diseño de message queuing permitió al emisor y al receptor operar en diferentes porcentajes de tiempo sin tener un diseño directamente orientado a las conexiones.

Soportar ambos modelos, process-to-process y message queuing, resulta en alta funcionalidad, la cual es la clave para la funcionalidad y el desempeño, el modelo asincrono event-drive messaging alivia el problema de bloqueo y es un excelente modelo para enlazar cliente y servidor donde es necesaria una respuesta inmediata.

6.1.6 MOM proporciona mas que una comunicación básica de APIs.

La fuerza fundamental de MOM es su habilidad para proporcionar un simplificado, simétrico y consistente API de comunicaciones a través de múltiples protocolos y sistemas operativos heterogéneos. Algunos de los actuales productos de message oriented middleware ofrecen mucho mas:

MOM, adiciona una arquitectura de red para soportar confiabilidad, comunicaciones de alto rendimiento para aplicaciones distribuidas de gran escala en entornos heterogéneos. MOM proporciona comunicaciones asincronas sin bloqueo y sincronas con bloqueo, transparencia de localización, administración eficiente de la sesión, prioridad de mensajes, consistencia en el control del flujo de mensajes, administración de time-out, rutas alternas de mensaje, manejo simplificado de errores y recuperación.

En algunos casos, MOM también proporciona balanceo de cargas y administración dinámica de los recursos.

6.1.7 Flexibilidad es la clave.

Mucha gente cree que las crecientes aplicaciones cliente/servidor van a ocurrir en el área de las funciones distribuidas. Message oriented middleware o simplemente messaging, puede ser usado para implementar muchos de los estilos de la computación cliente/servidor. Mientras otras tecnologías alternativas tales como RPCs o SQL pueden ser consideradas para reemplazar a MOM en la implementación de acceso remoto a datos o presentación de aplicaciones distribuidas, estas tecnologías no deben ser consideradas para el desarrollo de

aplicaciones mas complejas, de las cuales la mayoría, están basadas en múltiples funciones de aplicación y en recursos distribuidos a través de la red.

Message oriented middleware da a cualquier aplicación transparencia para accesar funciones, recursos y servicios de negocios localizados a través de la red. Esto es análogo a la manera en que la base de datos proporciona acceso a los datos almacenados en los discos. Cuando usamos sistemas de administración de bases de datos, los desarrolladores no se involucran en las intrincadas particularidades de bajo nivel de los métodos de acceso y de la organización interna de los datos. Cuando usamos MOM los desarrolladores no tienen que involucrarse acerca de los detalles de sistemas operativos, plataformas de hardware, diferencias entre las aplicaciones, diferencias entre las bases de datos, protocolos de comunicación o donde están localizados los recursos de la red.

Cuando dos o mas procesos están comunicándose vía mensajes, los cuales pueden contener datos, instrucciones de control o ambos, ellos crean un servicio de negocios independiente de su localización. En otras palabras, usando MOM, los limites entre dos o mas plataformas son removidas a una red virtual unificada, similarmente, el desarrollador de un sistema distribuido tiene acceso transparente a los servicios y recursos de la red vía mecanismo de mensajes.

Message oriented middleware proporciona la infraestructura de red necesaria que permite la comunicación independiente de los procesos. Para un desarrollo hecho en casa o un proveedor de software, la complejidad de los temas de comunicación se convierte en el alma de la simplicidad. Este proporciona una simple y consistente colección de interfaces que cubren ese rango de complejidad. Algunas veces el método basado en mensajes pretende ser una alternativa competitiva para otros mecanismos, especialmente RPC. Sin embargo, MOM no solo puede replicar la función de un RPC si es necesario, también puede aumentar la función de un sistema distribuido basado un mecanismo menos comprensivo de RPC.

MOM proporciona la flexibilidad para soportar los requerimientos de un amplio rango de necesidades de aplicaciones, otras formas de middleware tienden a ser mas convenientes para aplicaciones especificas.

Por ejemplo, los RPCs son convenientes para LANs relativamente pequeñas basadas en aplicaciones cliente/servidor, mientras que los gateways de bases de datos son convenientes para aplicaciones de soporte a la decisión.

6.1.8 Tecnologías emergentes y Message Oriented Middleware.

Message Oriented Middleware es consistente con la mayoría de la computación distribuida y arquitecturas orientadas a objetos en el mercado.

OSF's Entorno de computación distribuida.
OMG's CORBA nuevas tecnologías orientadas a objetos.

MOM es una tecnología complementaria que aumenta y extiende estas tecnologías.

Message Oriented Middleware puede aumentar y complementar DCE en las áreas en las cuales esta falta, específicamente messaging y message queuing. En teoría, para utilizar messaging, DCE puede utilizar redes heterogéneas existentes sin requerir la presencia de TCP/IP en cada nodo de red dentro de un entorno corporativo. En adición, si se usa messaging, el modelo de comunicaciones DCE puede ser enriquecido para soportar sistemas que manejen event-drive- una característica ausente dentro de la estructura DCE, además si se usa MOM, los clientes nunca están bloqueados mientras su requerimiento es procesado.

El resultado es lo mejor de ambos mundos, los usuarios finales y los desarrolladores y planeadores corporativos de TI pueden obtener todos los beneficios de estas dos importantes tecnologías.

Los modelos basados en los mensajes se pueden volver mas esenciales para incrementar las tecnologías orientadas a objetos. Primero, los objetos usan y pueden continuar usando el modelo MOM y comunicarse vía mensajes. Para hacer esto, los objetos distribuidos pueden no ser confinados a solo comunicación sincrona como lo exige el modelo RPC. En la actualidad Object Management Group (OMG) no provee bastante flexibilidad para que los objetos se puedan comunicar con otros. Esto es especialmente verdad donde UNIX y TCP/IP no están involucrados. Es verdad que la comunicación asincrona puede expandir la funcionalidad de los OMG's para alcanzar mas aplicaciones y plataformas. La tecnología de objetos puede beneficiarse substancialmente por la habilidad para ofrecer recursos dinámicos, el cual puede permitir a cualquier objeto en la red a encontrar a otro objeto dinámicamente a pesar de la capa baja de protocolo de red. Además, los objetos pueden residir en el nodo donde diferentes protocolos de comunicación están presentes.

Mientras DCE es una importante tecnología basada en los servidores y es buena para aplicaciones cliente servidor, es menos apropiada para aplicaciones distribuidas complejas a nivel empresa.

En resumen, MOM puede enriquecer la funcionalidad de DCE y OMG para proveer una plataforma de software para computación distribuida, mientras mantiene los principios arquitecturales de DCE y CORBA.

6.1.9 La interoperabilidad entre las aplicaciones es obligatoria.

La flexibilidad máxima en la infraestructura para computación distribuida es critica. Mientras que algunas compañías se mueven agresivamente a eliminar sus mainframes, existen numerosas compañías donde los mainframes son componentes claves para sus estrategias cliente/servidor. Estas compañías pueden terminar migrando el código a plataformas mas baratas en un largo plazo o pueden optar por mantenerse con sus

mainframes indefinidamente. Estas compañías requieren un mecanismo de desarrollo de aplicaciones que sea por si mismo portable e interoperable a través de múltiples plataformas, sistemas operativos y protocolos de comunicación. MOM esta capacitado para proporcionar estos mecanismos de desarrollo.

La interoperabilidad en el nivel de aplicaciones es clave. Últimamente, las compañías esperan que cualquier aplicación por objetos sea capaz de localizar a cualquier otra aplicación en la red e intercambiar mensajes y datos de control a pesar de la localización, protocolos de comunicación, tipos de datos, estructuras de los mensajes, campos y sistemas operativos de las plataformas fuente y objetivo donde estos objetos residen. MOM proporciona una fuerte capacidad para lograr a cabo este tipo de interoperabilidad en el mundo real.

Mientras la interoperabilidad es la clave, mas y mas compañías intentan llevar a cabo verdadera interoperabilidad, la existencia de una sola red lógica es deseable. Ello implica que una sola red es importante no solo para desarrollar aplicaciones distribuidas abiertas, sino también para la red y la administración del sistema.

MOM es capaz de proporcionar la capa de software requerida para unificar cada procesador, base de datos y aplicaciones, logrando esta interoperabilidad entre componentes heterogéneos de la red.

6.1.10 Messaging es una solución optima para aplicaciones empresariales.

Messaging proporciona mas facilidad de uso, flexibilidad, escalabilidad y mejor desempeño que otros mecanismos para aplicaciones distribuidas lógicas, permitiendo a cualquier numero de procesos independientes o componentes de software u objetos comunicarse con cualquier otro a través de cualquier red vía mensajes. MOM transforma la red en un solo sistema donde los nodos pueden ser vistos como una colección de procesos capaces de comunicarse con cualquier otro.

MOM ofrece un numero de atributos técnicos que la convierten en una solución extremadamente atractiva para desarrollar cualquier complejo sistema distribuido, algunos de estos atributos son:

- Comunicación sincrona y/o asincrona puede ser usada y mezclada dependiendo de los requerimientos de las aplicaciones.
- MOM puede soportar cualquier numero de protocolos de red concurrentemente y puede rutear el trafico de las aplicaciones entre diferentes protocolos si es necesario.
- Los mensajes son independientes del lenguaje y no procedurales.
- Messaging soporta varios modelos de aplicaciones distribuidas tales como cliente/servidor, peer-to-peer y objetos distribuidos, así como flujos de aplicaciones complejas.

Grandes proveedores de software están ya definiendo una variedad de software e interfaces que satisfagan estos requerimientos. Estas soluciones múltiples tienden a tener múltiples interfaces, dentro de la misma arquitectura y estos sistemas pueden tener diferentes propósitos. Sin la simplicidad y flexibilidad de MOM, el desarrollo y administración de aplicaciones distribuidas puede ser una tarea extremadamente difícil.

CONCLUSION Y RESULTADOS DEL EXPERIMENTO.

Ubicación del Middleware en su Relación con el Modelo OSI y El Open Blueprint.

A partir de las clases de middleware que existen podemos ubicar a las mismas dentro de la capa 5 a la capa 7 del modelo OSI, que son, la capa de sesión, la capa de presentación y la capa de aplicación.

En la siguiente figura se muestra esta relación.

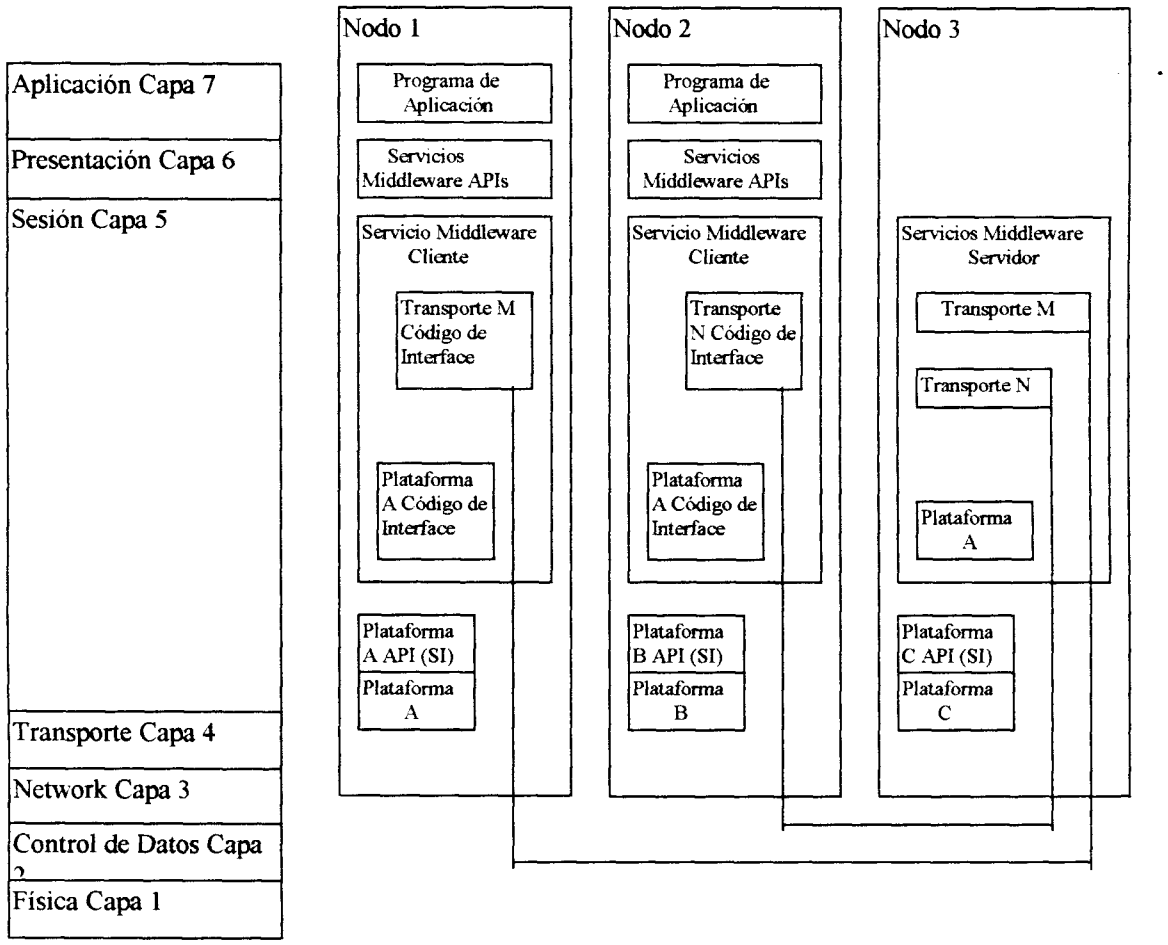


Figura C.1.- Relación Middleware con el Modelo OSI y el Open Blueprint.

El middleware consiste de un gran número de servicios, estos servicios se agrupan dentro de seis categorías:

Servicios de Presentación:

Estos servicios permiten a una aplicación interactuar con los usuarios. Algunos servicios operan a través de diferentes dispositivos de interface, un ejemplo de los servicios de presentación sería el Sistema de Servicios X Window.

Servicios de Comunicación:

Estos servicios permiten a una aplicación comunicarse con otras aplicaciones, ambas locales o remotas. Un ejemplo de un servicio de comunicación es "electronic messaging".

Servicios de Control:

Estos servicios permiten a una aplicación controlar programas de ejecución en entornos tanto locales como distribuidos, estos incluyen servicios para invocar funciones provistas por programas de aplicación, un ejemplo de los servicios de control es el servicio "object brokering".

Servicios de Información:

Estos servicios permiten a una aplicación definir, almacenar, accesar y manipular datos, estos proporcionan información para las herramientas y aplicaciones. Estos servicios soportan mecanismos de acceso múltiple para la información distribuida, un ejemplo de los servicios de información es el directorio de servicios.

Servicios de Computación:

Estos servicios permiten a la aplicación desempeñar complejas operaciones sobre los datos, estos servicios proporcionan a las aplicaciones valores en el tiempo para manipular datos y sincronizar actividades, un ejemplo de este servicio es el "Servicio de Tiempo Distribuido".

Servicios de Administración:

Estos servicios permiten a los administradores de los sistemas para administrar recursos fácil y consistentemente de nodos remotos. Los servicios de administración soportan el mismo esquema para administrar recursos de aplicación, esto permite a los administradores de sistemas administrar cualquier componente de cualquier lugar en la red. Un ejemplo de servicios de administración es el "management agent service".

La siguiente figura muestra las categorías de los servicios, los servicios en cada categoría y su relación con las plataformas.

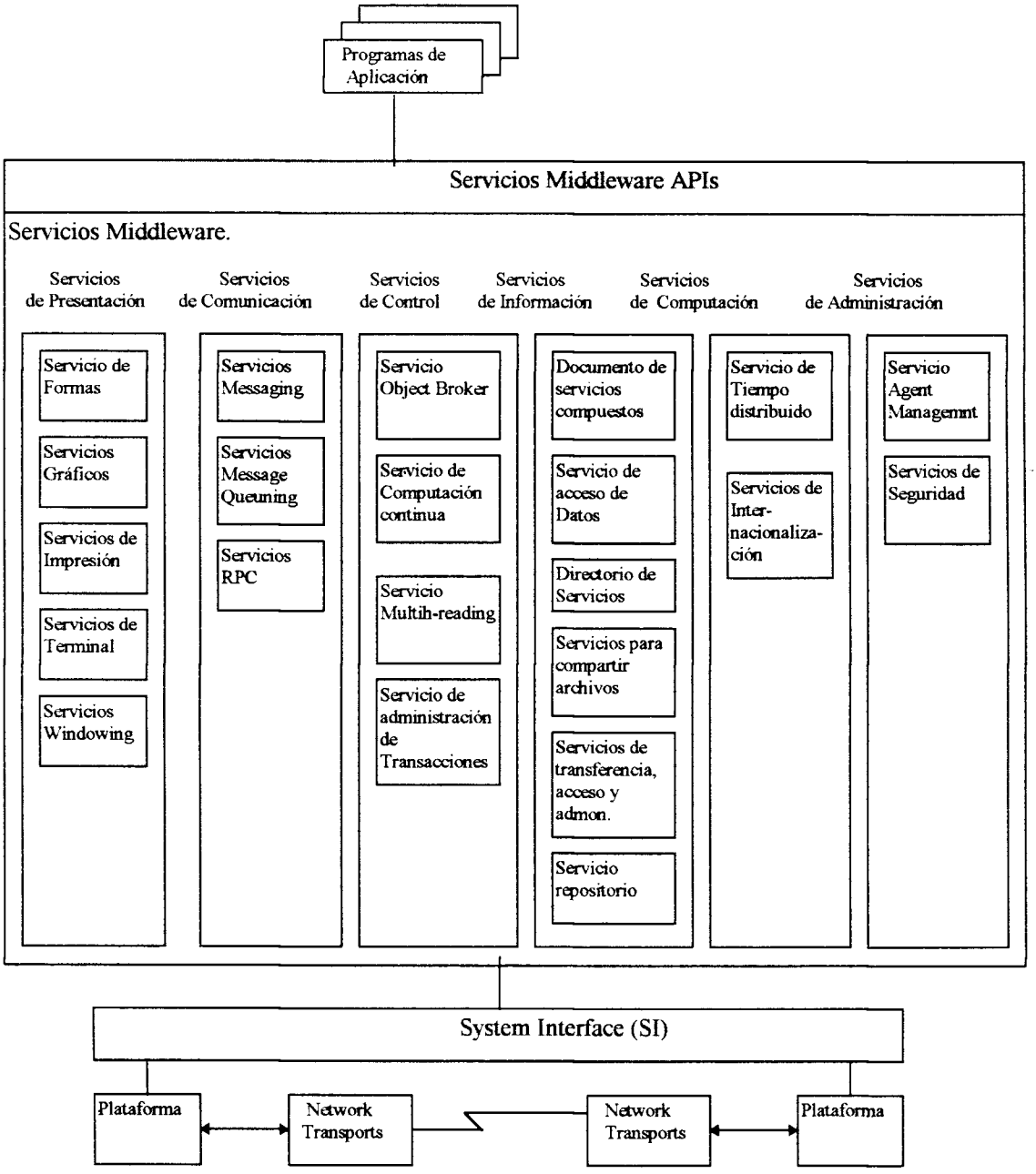


Figura C.2.- Categorías de los servicios, servicios por categorías y su relación con las plataformas.

De acuerdo a esta categorización encontramos que dentro del Open Blueprint propuesto por IBM consta de tres partes, Servicios de Red, Servicios de sistemas distribuidos y servicios de aplicaciones y aplicaciones habilitadoras, podemos ubicar los distintos tipos de middleware.

Ha continuación se presenta el Open Blueprint:

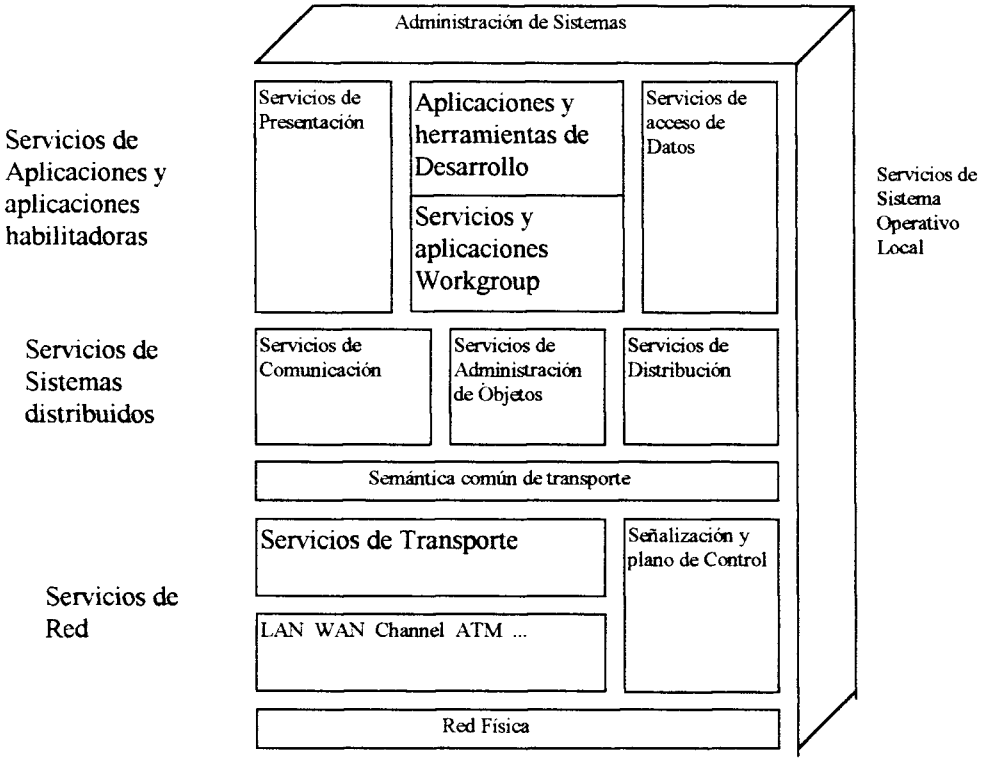


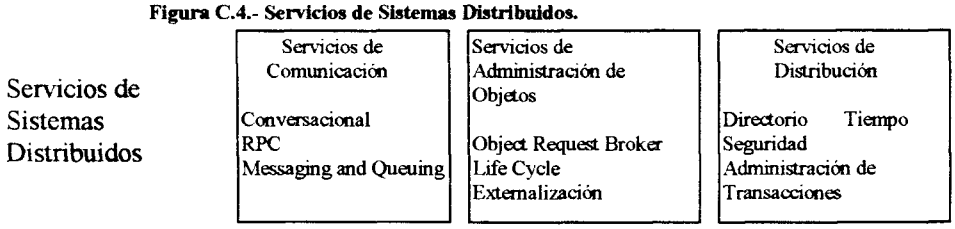
Figura C.3.- Open Blueprint.

El middleware queda completamente fuera del área de los servicios de red pero dentro de las áreas de servicios de sistemas distribuidos y de los servicios de aplicaciones.

SERVICIOS DE SISTEMAS DISTRIBUIDOS.

Dentro de esta parte del modelo tenemos tres partes que la forman, las cuales son: Servicios de Comunicación, Servicios de Administración de Objetos y Servicios Distribuidos, donde localizamos middleware como sigue:

Figura C.4.- Servicios de Sistemas Distribuidos.



SERVICIOS DE APLICACIONES:

Esta parte relacionada con las aplicaciones y las herramientas de desarrollo tiene tres partes, en las cuales encontramos situado el middleware, estas son: Servicios de Presentación, Servicios de Aplicación/Workgroup, Servicios de Acceso de Datos.

Estas partes contienen middleware como sigue:

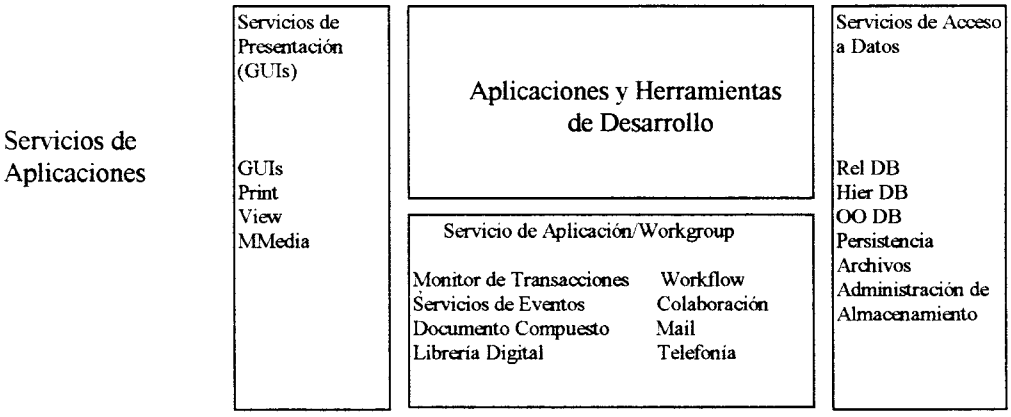


Figura C.5.- Servicios de las Aplicaciones.

Dentro de este modelo encontramos middleware aplicado dependiendo de sus capacidades pueden existir productos middleware que abarquen una sola, dos o las tres partes de una sola área del modelo Open Blueprint o incluso middleware que soporte varias partes del modelo.

Se diseño y se aplico una encuesta a diferentes compañías, con el objetivo de saber cuales de ellas conocen, o aplican middleware como una solución a sus problemas de conectividad.

Las compañías encuestadas son como sigue:

Pequeñas:	17.25%
Medianas:	51.72%
Grandes:	31.03%

Los giros de estas compañias son:

Gobierno:	10%
Privado:	90%

Este 90% se encuentra conformado por:

Sector educativo:	7.40%
Sector Turístico:	3.70%
Industria de la transformación:	48.16%
Sector de servicios:	40.74%

Esta encuesta se aplico a personas del departamento de Informática, de manera de lograr datos 100% confiables. Los resultados observados fueron:

El 65.52% de las personas conocen el concepto de sistemas abiertos y el 34.48% lo desconocen o no lo comprenden del todo.

1.- El porcentaje de existencia de sistemas operativos distintos en las empresas es como sigue:

El 79.31% de ellos utilizan mas de dos sistemas operativos distintos.
El 20.69% de ellos utilizan solo uno y de estos solo el 5.25% es porque se encuentra estandarizado.

De estos sistemas operativos el porcentaje de preferencia entre ellos es:

DOS	32.8%	UNIX	12.5%
OS-400	1.56%	Windows NT	26.56%
Netware	14.06%	HP 3000	0.99%
Windows	4.68%	Windows 95	7.81%

Resultando necesidades de interconexión entre estos.

2.- Por lo tanto existen distintas bases de datos que se encuentran en casos especiales como son, misma base de datos trabajando sobre diferentes plataformas, diferentes bases de datos trabajando sobre una misma plataforma y diferentes bases de datos trabajando sobre diferentes plataformas, en las compañías encuestadas se encontró que en el 3.44% estas se encuentran estandarizadas, el 20.68% de estas compañías no tienen esta situación y el 72.42% de estas compañías tienen una situación como las descritas anteriormente, lo que resalta la importancia de el conocimiento del middleware.

De este 72.42% la situación es como sigue:

A) Misma base de datos trabajando sobre diferentes plataformas: 43.33%

B) Diferentes bases de datos trabajando sobre una misma plataforma: 33.33%

C) Diferentes bases de datos trabajando sobre diferentes plataformas: 23.34%

3.- En el 72.42% de estas compañías existe una necesidad de compartir bases de datos heterogéneas para facilitar el trabajo y de esta manera cumplir con la misión-visión de la empresa, el 27.48% restante no necesita compartirlas, por diferentes razones como es la función de su departamento, la seguridad de los datos o estas ya están estandarizadas.

4.- La comunicación entre estas bases de datos distintas se realiza de la siguiente manera:

Uso de Gateways: 23.33%

Uso de DBMS (Database Management System): 40%
Se encuentran estandarizados y no hay necesidad de usar ninguno de los anteriores: 30%

Tienen necesidad de comunicarlas, pero todavía no lo hacen: 6.67%

Podemos observar la importancia del conocimiento del middleware otra vez, pues como lo vimos anteriormente el uso de DBMS's y de gateways es una parte fundamental de middleware, así como las empresas que se encuentran estandarizadas, obviamente aplicando middleware, sin saberlo y sin ubicarlo claramente dentro del entorno informático de la empresa y con la capacidad de ayudar y orientar a las empresas que tienen que comunicar sus bases de datos heterogéneas y todavía no lo hacen.

5.- En estas empresas el 41.37% tienen aplicaciones Cliente/Servidor, y del 58.63% restante el 35% de ellas están pensando en implementarlas.

6.-En el 44.82% de estas compañías existe más de una red dentro las mismas, existiendo de la siguiente manera:

Novell Netware:	45.71%	UNIX:	17.14%
Lantastic:	17.14%	Windows NT:	20.01%

7.- De este 44.82%, el 37.93% tiene una necesidad de comunicación entre estas redes distintas, en el 62.07% no existe necesidad de comunicación, pero de este 62.07% el 61.12%, tiene que comunicarlas en un futuro próximo.

Los tipos de acceso a las redes del total de las compañías encuestadas son:

RDI:	41.38%	Satelital:	10.34%
Modem:	48.28%	Mas de dos tipos de acceso:	36.37%

8.- Se encontró que en el 82.75% de estas compañías la comunicación es transparente, es decir, la información es recuperada, consultada y modificada sin que se tengan retrasos considerables en los tiempos de respuesta y sin importar donde esta ubicada físicamente, en el 10.34% se tienen problemas y en el 6.91% restante esta situación no aplica.

9.- De las personas encuestadas, el 93.1%, no conoce absolutamente nada sobre middleware, el 6.9% restante de estas personas lo conocen, aunque ninguna pudo dar una definición concreta de middleware, ni de sus características ni de su ubicación dentro de la Tecnología de Información.

10.- Una característica del Front-end de middleware es la Interface Gráfica de Usuario (GUI) que se utilice como estandar y facilite la nevegación en el sistema y la transparencia de localización de las aplicaciones a los usuarios finales, se encontró que en el 72.41 de estas compañías se utiliza alguna GUI, 27.59% de ellas no lo utilizan, pero de este porcentaje que no las utilizan el 12.5% esta migrando actualmente o evaluando una para su próxima instalación.

11.- Una característica muy importante del middleware es la de los servicios de comunicación que se utlizan, esta pregunta es muy importante, pues por ser muy técnica refleja claramente el nivel de conocimiento a middleware, solo el 50% de las personas encuestadas contestaron y se encontró que estos se utilizan como sigue:

Remote Procedure Calls (RPC). 40%
Message Queuing Service (Comunicación entre sistemas heterogéneos por medio de un intermediario llamado queue). 13.32%
Messaging Services (Este servicio es una implementación de la ITU, basada en el estandar X.400. 13.32%
Electronic Data Interchange (EDI). 33.36%

12.- Estas empresas calificaron la portabilidad de las aplicaciones de las empresas en la escala de 5 a 10, siendo 5 la peor y 10 la mejor como se describe a continuación:

Calificación	Porcentaje
5	27.59%
6	3.44%
7	3.45%
8	27.59%
9	20.69%
10	17.24%

Este grado de satisfacción resalta la importancia que existe en México del conocimiento del middleware, sus capacidades, su ubicación dentro de la infraestructura informática y de su importante misión: Facilitar la portabilidad de las aplicaciones y la transparencia de la ubicación de las bases de datos.

BIBLIOGRAFIA

A. Beitz and M. Bearman (1994), An ODP Trading Service for DCE. in Proceedings of the Firts International Workshop on Services in Distributed and Networked Environments (SDNE), páginas 42-49. June 27-28, 1994.

Alan R. Simon/Tom Wheeler. "Open Client/Server Computing and Middleware", AP Professional. 1995.

Berson, Alex y Jay Ranade. "Client/Server arquitecture", McGraw-Hill Series on Computer Communications. 1992.

Colonna-Romano, John and Patricia Srite. "The Middleware Source Book", Digital Press. 1995.

C. J. Biggs and W. Brookes, "Enhancing Interoperability of DCE Applications: A Type Management Approach", (SDNE), páginas 50-57. June 27-28, 1994.

Halsall, Fred, "Data communications, computer networks and open systems, Addison-Wesley, Third edition.

ISO Trader (1994), Rec. X./ ODP Trading Function ISO/IEC 13235: 1994, July, 1994.

Juan Carlos Usandizaga, Presidente IBM España, "Fundamentos Cliente/Servidor", Centro Europeo de Soluciones Cliente/Servidor de IBM, Documentos Computer World, segunda Edicion, Mayo 1995.

Krantz, Steve Project Leader, IBM Boca Raton Client/Server Migration Project. "Real World Client/Server", Maximum Press. 1995.

Leo Laverdure/Patricia Srite/John Colonna-Romano. "NAS Arquitecture Reference Manual, Middleware for Building Distributed Applications", Digital Press. 1993.

Lucio Stanca, "The Four Wave: A Bussines Guide to Client/Server Computing", IBM World Trade Europe/Middle East/Africa Corporation.

OMG (1993), The Common Object Request Broker: Architecture and Specification, Revisión 1.2, December 1993, Object Management Group, Cambridge U.S.A.

OSF (1993), OSF DCE Application Development Guide, Revisión 1.0, 1993. Open Software Foundation, Cambridge U.S.A.

P. van Eijik and A. Belinfante (1990), "The Term Processor Kimwitu, Manual and Cookbook." University of Twente Technical Report INF-90-45, 1990.

Time Berners-Lee, and Arthurs Secret (1994), "The World-Wibe-Web", Communications of the ACM, 37(8): 76-82, August 1994.

Terry Blevins, Dennis J. DeBrosse, Gregory Steele, CIO's of AT&T Technology, Dayton, Ohio. pp 14-19, 1995.

Vaskevitch, David, Microsoft Corporation. "Client/Server Strategies" 2nd Edition, IDG Books Worldwide. 1995.

Verity, John W. "Cyber-networks need a lot of spackle- Brokering babel: programs that give networks one voice", Bussines Week, Jun 26, 1995, pp 92-93.

Wexler, Joanie, "SAP readies client/server software suite", Computer World, Jun. 1995.

Referencias hipertexto (forma parte de la Bibliografía utilizada.):

HREF1: http//www.osf.org:8001/dce/index.html, DCE's Home Page.
HREF2: http//www.omg.org, OMG's Home Page.
HREF3: http//www.ansa.co.uk/phase3-activities/wwwCorba.html, ANSA Web project.
HREF4:http//www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/hastings/hastings.ht ml, Digital's CORBA Web Project.
HREF5: Message Oriented Middleware Association (MOMA).
http://www.sbexpos.com.