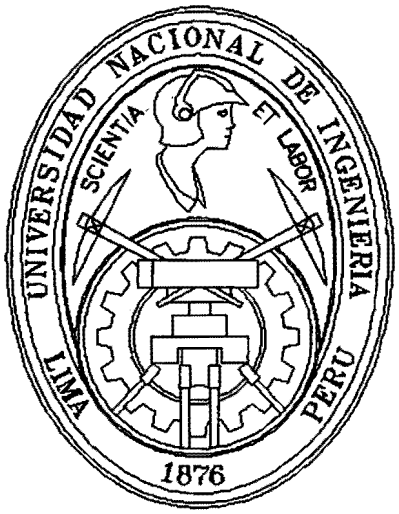
Universidad Nacional de Ingeniería
FACULTAD DE INGENIERIA INDUSTRIAL Y DE SISTEMAS



ARQUITECTURA DE DESARROLLO DE APLICACIONES INTRANET (WINDOWS DNA)

INFORME DE INGENIERIA

Para Optar el Título Profesional de:

INGENIERO DE SISTEMAS

EDUARDO MARTIN SALDARRIAGA RENGIFO

Lima - Perú
1998

A mis padres

INDICE GENERAL

Sumario

En la actualidad, las empresas requieren de sistemas de información, los cuales tengan un ciclo de desarrollo rápido y que se adapten con facilidad y en el menor tiempo posible a los cambios. Otro factor importante es la aplicación de los conceptos de *Internet* en los sistemas de información de las empresas. La tecnología Internet ha demostrado ser buena para compartir información entre sus millones de usuarios en el mundo. Se requiere que dicha tecnología se pueda utilizar en los sistemas de información de las empresas. Esto permite compartir información de manera rápida y sencilla entre los empleados de la empresa, desarrollo rápido de las aplicaciones, etc. Al tipo de aplicación o sistema que utiliza la tecnología de Internet al interior de las empresas, se le denomina *aplicación Intranet.*

Microsoft ha propuesto una arquitectura de desarrollo de sistemas de información, denominada "Arquitectura Windows DNA", basándose en componentes de software re utilizable, que acelera el proceso de desarrollo y modificación de los sistemas de información.

Conclusiones

La Arquitectura Windows DNA, utilizando la Tecnología de Componentes COM, define una arquitectura de desarrollo basada en componentes de software, que permiten el desarrollo rápido de aplicaciones, a un menor costo.

Al dividir una aplicación en tres capas lógicas (Interfaz Gráfica de Usuario, Reglas de Negocio y Fuentes de Información) se logra independencia y rapidez en el desarrollo de las diferentes partes de una aplicación.

1 Descriptores Temáticos

- Windows DNA
- Arquitectura de Desarrollo de Aplicaciones Intranet
- Acceso Universal de Datos
- Component Object Model
- COM
- Microsoft
- Aplicaciones Intranet
- Internet

2 Objetivo

El presente trabajo tiene como objetivo presentar la Arquitectura de Desarrollo de Aplicaciones Intranet de Microsoft, denominada Windows DNA. Esta arquitectura permite construir aplicaciones cliente/servidor tanto para Internet como para intranet.

Son objetivos de este trabajo:

- Explicar en forma clara y concisa las diferentes partes de esta Arquitectura.
- Presentar la estructura de aplicaciones intranet o Internet en un ambiente de red distribuido.
- Explicar el desarrollo rápido de aplicaciones mediante la reutilización de código existente.
- Presentar mediante una aplicación de ejemplo los conceptos y beneficios de la Arquitectura Windows DNA.

4 Introducción

Las computadoras han permitido a las empresas ser más eficientes en sus tareas diarias. Si pensamos cómo se mantenían actualizados los libros mayor y de diario en la contabilidad de una empresa hace 30 años, y cómo actualmente con la ayuda de las computadoras las empresas pueden tener al día su información, nos da una idea clara de los beneficios de las computadoras.

Los *programas de computadora*, más conocidos como **sistemas** o **aplicaciones**, permiten automatizar cualquier trabajo de oficina. Por ejemplo, existen Sistemas de Facturación que permiten automatizar todo el proceso de emisión, cobranza y registro contable de las facturas de una empresa. Hace treinta años ya se hacían sistemas de este tipo; sin embargo, estos eran difíciles de crear y mantener. A esto hay que agregar que el precio de las computadoras era muy elevado y requerían un enorme espacio de almacenamiento.

Hace unos veinte años con la aparición de la *computadora personal* (conocida como *PC*) y el acelerado avance tecnológico, el precio de las mismas empezó a decaer, permitiendo que las empresas pudieran utilizar en mayor escala programas de computadoras para diferentes necesidades. Por ejemplo, en la actualidad casi todas las empresas poseen algún sistema contable, de facturación o de planillas, los cuales hacen su trabajo más eficiente.

Luego, las computadoras personales se empezaron a unir, una con otra, en las denominadas *Redes de PC*, a un costo bajo. Esto permitió que los empleados de una empresa pudieran compartir la información de una manera más eficiente. Con las redes de computadoras, apareció el concepto de aplicaciones *cliente/servidor*, que permitía tener una parte de la aplicación en la PC, y tener las bases de datos en una computadora o servidor central. Esto hizo que las aplicaciones se hicieran más fáciles de crear y a un costo más bajo. Luego con la aparición del sistema operativo Windows, las PC (y las aplicaciones instaladas en él) se hicieron muy fáciles de utilizar.

Hace unos diez años apareció un gran fenómeno, denominado Internet, que ha cambiado la vida de todos. Internet es una gran red de computadoras unidas entre sí a escala mundial, que permite que todos estemos comunicados en lo que se ha denominado Aldea Global. Actualmente, podemos enviar mensajes a personas que están en otras partes del mundo en tan solo unos segundos. Asimismo, podemos buscar información sobre cualquier tema en los millones de computadoras conectadas a la red de Internet. Sin embargo, su uso, también, se ha extendido a otros ámbitos como el comercio, publicidad, etc. Por ejemplo, en la actualidad las empresas ofrecen sus servicios y productos a través de Internet. De esta manera, las empresas pueden llegar a millones de clientes, con costos muy bajos en términos de publicidad y ventas. El fenómeno de Internet ha introducido un nuevo tipo de sistemas o aplicaciones, denominadas *aplicaciones Internet* que son utilizadas por millones de personas.

La tecnología de Internet se ha introducido en los últimos cinco años en las empresas para crear un nuevo tipo de aplicaciones, denominadas, *aplicaciones intranet*. Este tipo de aplicación utiliza la tecnología de Internet internamente en la empresa. El uso común de este tipo de aplicación es para compartir información. Por ejemplo, un sistema intranet de Recursos Humanos permite que el empleado pueda ver en su PC toda la información de cursos ofrecidos por la empresa, o ver las últimas modificaciones a los manuales de procedimientos, etc. Se calcula que para el Año 2000, el 80 % de las aplicaciones en las empresas serán aplicaciones de tipo intranet

Para construir aplicaciones intranet, Microsoft propone utilizar la **Arquitectura de Desarrollo de Aplicaciones Intranet**, ó **Windows DNA** por sus siglas en inglés. Entre los beneficios de utilizar esta arquitectura está la facilidad para construir aplicaciones eficientes y fáciles de modificar.

El presente trabajo explica a detalle la Arquitectura definida por Microsoft para desarrollar aplicaciones intranet.

5 Arquitectura de Desarrollo de Aplicaciones Intranet

Las organizaciones requieren aplicaciones que puedan construirse fácilmente y que se adapten rápidamente a los cambios. De esta manera, las empresas están más preparadas para afrontar los retos de la modernidad.

Para satisfacer esta necesidad, Microsoft ha definido una arquitectura para construir aplicaciones modernas para ambientes intranet o Internet. Esta arquitectura es denominada **Windows DNA** (Distributed IntraNet Applications, por sus siglas en inglés), o **Arquitectura de Aplicaciones Distribuidas en Internet/intranet**.

Esta arquitectura propone construir aplicaciones identificando tres componentes fundamentales para cualquier sistema:

a) La **Aplicación misma**, la cual debe estar dividida de manera lógica en (1) **Interfaz Gráfica de Usuario**, (2) **Procesos de Negocio**, y (3) **Almacenamiento**. A este tipo de esquema se le denomina Tres Capas (Three Tier) o N Capas (N-tier).
b) la **Infraestructura de Red** sobre la cual se monta la aplicación, y
c) Las **Herramientas o Lenguajes de Programación** necesarios para construir la aplicación.

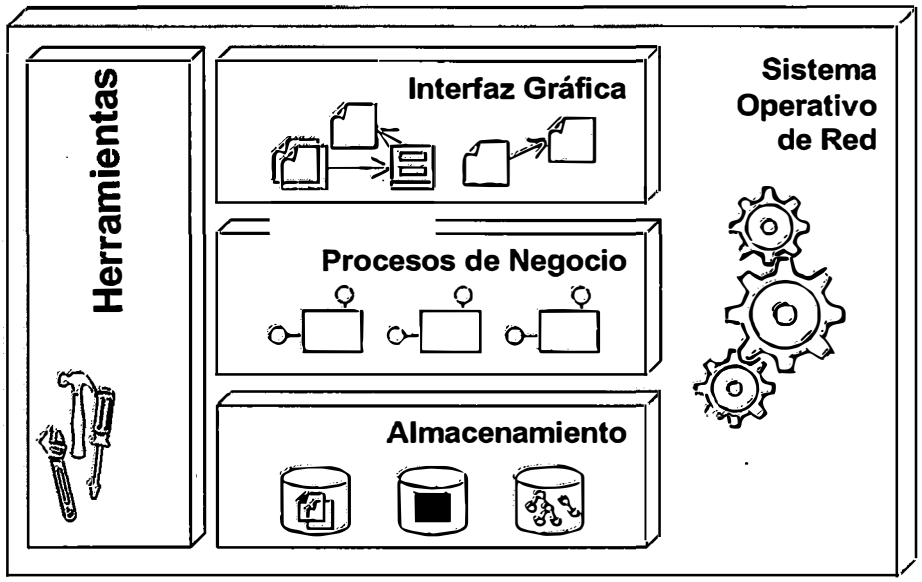


Figura 1: Arquitectura de Desarrollo de Aplicaciones Intranet de Microsoft (Windows DNA)

Como vemos en la Figura 1, la Arquitectura Windows DNA define una aplicación de tipo cliente/servidor de varias capas (interfaz gráfica, procesos de negocio, almacenamiento de información). Uno de los beneficios importantes de esta arquitectura es la posibilidad de crear **aplicaciones** cliente/servidor tanto para intranet como para Internet.

Utilizando esta arquitectura las empresas pueden construir aplicaciones escalables, que se pueden montar sobre ambientes de redes pequeñas o en grandes redes con muchos servidores. Asimismo, las aplicaciones pueden acceder a diferentes fuentes de información independiente de su ubicación física y del tipo de fuente de información (estructurada o no estructurada). Por otro lado las aplicaciones clientes pueden ser de múltiples plataformas (Windows, UNIX, etc.).

Finalmente, el beneficio más importante que se logra con esta arquitectura, es construir aplicaciones cuyos componentes se pueden re utilizar, posteriormente, en otras aplicaciones, logrando de esta manera el gran sueño de poder crear soluciones a partir de "pedazos" de otros sistemas previamente creados.

6 Interfaz Gráfica de Usuario en la Arquitectura Windows DNA

La Arquitectura Windows DNA permite crear aplicaciones modernas donde la interfaz gráfica de usuario puede ser de arquitecturas diferentes. La aplicación cliente puede tener interfaces Windows, o Browsers de cualquier tipo (Windows, UNIX, etc.).

La clave para esta independencia en el tipo de interfaz gráfica de la aplicación cliente está en centrar todas las reglas del negocio de la aplicación en componentes encapsulados de código que sean utilizados desde la interfaz del cliente.

Los tipos de interfaz gráfica que permite esta arquitectura son: Windows (denominadas Win32) y Browsers de cualquier tipo (UNIX, Windows, etc.).

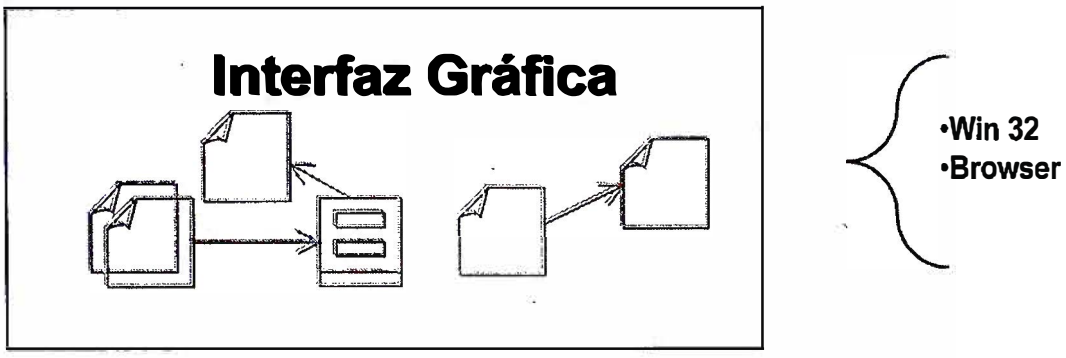


Figura 2: Interfaz Gráfica de Usuario. *Puede ser tipo Windows o Browsers de*

cualquier plataforma.

6.1 Interfaz Gráfica de Usuario tipo Browser

La interfaz gráfica más popular en la actualidad es el Browser (más conocido como Explorador o Navegador). Esto se debe a que en los últimos años se ha vivido la gran revolución de la Internet que ha cambiado la vida de todos con la gran ventaja de poder comunicarnos con otras partes del mundo de manera casi automática, y de poder buscar y encontrar rápidamente información de cualquier tipo. Precisamente, las aplicaciones denominadas intranet, donde se utilizan los mismos estándares del Internet dentro de las organizaciones, se han hecho cada vez más populares. Así, se calcula que para el año 2000 el 85% de las aplicaciones tendrán interfaces de tipo intranet, donde el Browser será el cliente universal de cualquier aplicación.

La Arquitectura Windows DNA permite crear aplicaciones donde el Browser puede ser de cualquier tipo, incluyendo aquellos de plataforma Windows (Internet Explorer, Netscape, etc.), como los de plataforma UNIX.

La interfaz gráfica tipo Browser responde a la necesidad de contar con un cliente universal para todas las aplicaciones. El beneficio fundamental de esto es que sólo basta tener instalado un Browser en la máquina del cliente para poder acceder a cualquier aplicación, sin necesidad de tener que instalar una aplicación cliente en cada computadora para poder acceder a la aplicación. Es importante mencionar que este tipo de aplicaciones sigue siendo del tipo cliente/servidor con un beneficio adicional: el cliente es un cliente universal: el Browser.

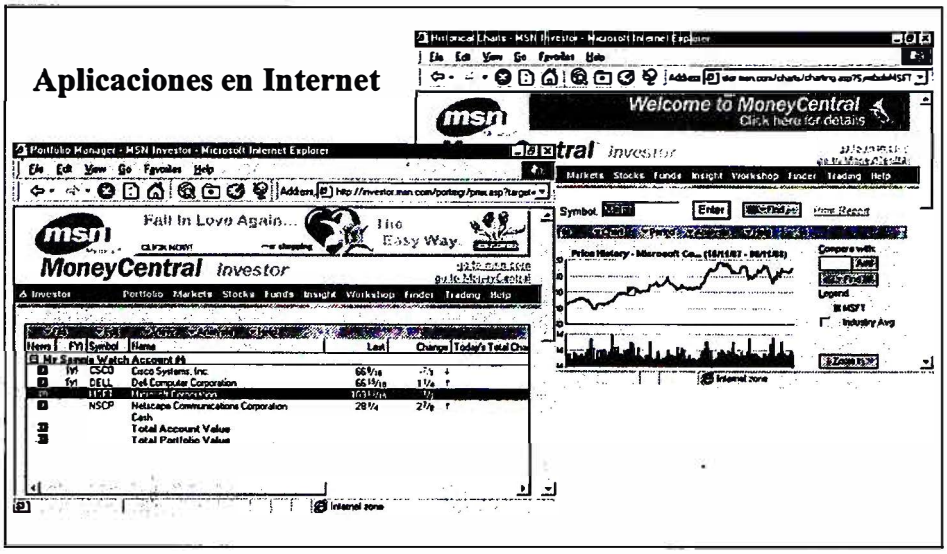


Figura 3: Interfaz Gráfica de Usuario tipo Browser. *Microsoft Investor es una aplicación que está en Internet. Permite analizar las acciones en la bolsa de New York. Es una Aplicación cliente/servidor en Internet. El usuario sólo necesita tener un Browser instalado.*

Las aplicaciones de este tipo tienen la característica que utilizan el paradigma introducido por los browsers, como son páginas web, navegación entre páginas (página hacia adelante, página hacia atrás), hipervínculos (posibilidad de dar clic en algún texto o imagen e ir directamente a la página relacionada), etc.

6.2 Interfaz Gráfica de Usuario tipo Windows (Win32)

Las aplicaciones más populares en los últimos 10 años, por los beneficios que ofrece, son las denominadas **aplicaciones cliente/servidor** en plataforma Windows. Este tipo de aplicaciones requieren tener instalado en la computadora del usuario, una aplicación denominada "**aplicación cliente**", la cual hace uso de

los servicios que le proporciona el servidor de datos o el servidor de aplicaciones.

La arquitectura Windows DNA permite seguir creando este tipo de aplicaciones, pero con el beneficio adicional que los procesos de negocio residen en uno o varios servidores de aplicaciones centralizados, los que a su vez se conectan con los servidores de datos. A este tipo de esquema se le denomina Tres Capas (3 Tier) o N Capas (n Tier). El beneficio de una arquitectura de este tipo es que los cambios a los sistemas, que por lo general ocurren en las reglas de negocio, se hacen en la capa central, manteniendo las aplicaciones cliente la misma interfaz (que es lo que menos cambia en los sistemas), sin necesidad de tener que reinstalar las aplicaciones cliente.

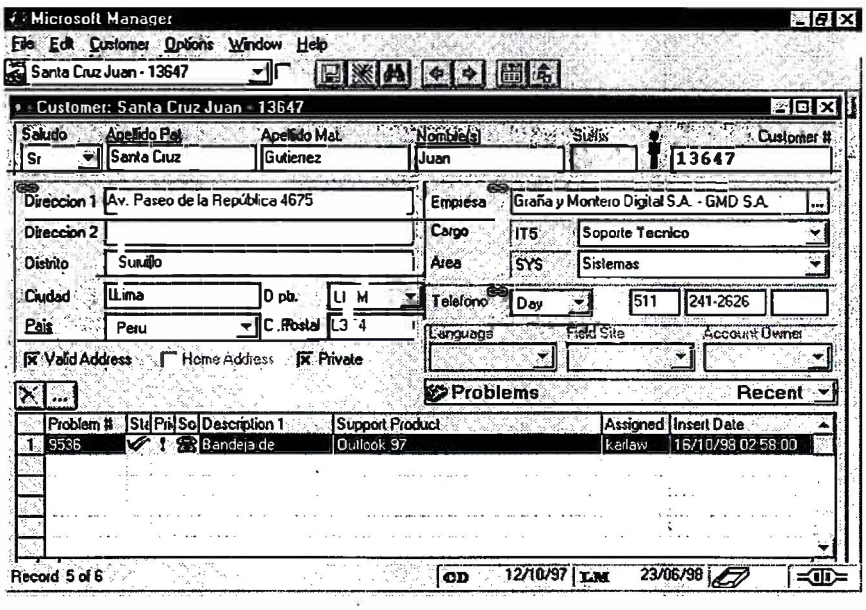


Figura 4: Interfaz Gráfica de Usuario basada en Formularios de Windows. *Las aplicaciones cliente/servidor requieren una aplicación cliente instalada en la computadora del usuario.*

La Figura 4 muestra una aplicación cliente/servidor de seguimiento de clientes. La aplicación cliente está instalada en la computadora del usuario. Este tipo de interfaces gráficas de usuario se basan en formularios (Forms) de Windows y se les denomina Win32. Los formularios hacen uso de la plataforma Windows para crear interfaces amigables y fáciles de usar.

7 Las Reglas del Negocio en la Arquitectura Windows DNA

La parte más importante, y a la vez más cambiante, de un sistema son las Reglas o Procesos del Negocio. Por ejemplo, una regla del negocio es la forma de calcular el sueldo variable de un vendedor basado en criterios específicos. Podemos por ejemplo, pagarle basándose en un porcentaje de ventas, y luego de algún tiempo cambiar el criterio de pago, por un porcentaje basándose en los nuevos clientes que el vendedor ha captado. Así vemos que mientras las interesas de usuario y las fuentes de información pueden permanecer durante mucho tiempo sin ser modificadas, las reglas del negocio si cambian periódicamente.

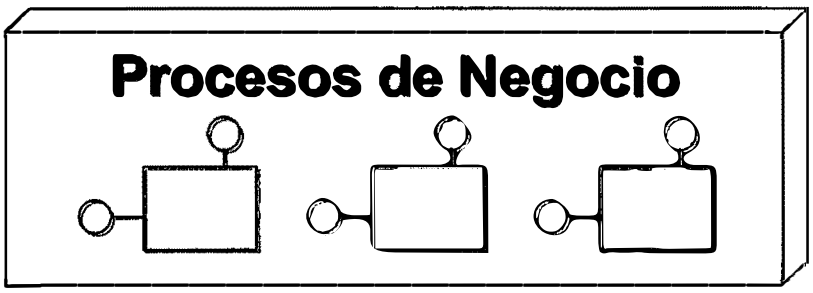


Figura 5: Reglas y Procesos del Negocio. *Las reglas de negocio son los elementos más importantes y a la vez más cambiantes de un sistema.*

La arquitectura Windows DNA hace uso de la **Tecnología de Componentes** de Microsoft (Component Object Model o COM) como estándar para crear las reglas del negocio. Esta Tecnología propone crear "pedazos de código" que proporcionan

alguna funcionalidad o servicio, encapsulada en un archivo binario. Un archivo binario es un archivo de tipo DLL (Dynamic Link Library).

Dentro de este archivo o **componente** estará las reglas del negocio del sistema, agrupadas bajo el concepto de clases. A su vez, las aplicaciones cliente, invocarán los servicios encapsulados en el componente, sin necesidad de conocer cómo se implementaron o programaron estos. El componente es una caja negra (archivo DLL) invocada desde la aplicación cliente.

La Tecnología de Componentes descompone una aplicación en componentes de Reglas de Negocio, dependiendo de las características y agrupamiento de estas reglas. Así, una aplicación de Facturación constaría de los componentes de Reglas de Negocio de Clientes, Productos, Facturas, etc.

7.1 La Tecnología de Componentes en la Arquitectura Windows DNA

La Tecnología de Componentes o **Component Object Model (COM)** la podemos definir como un estándar que permite crear código que contiene un conjunto de servicios que ofrecen una funcionalidad específica (por ejemplo, el Cálculo del descuento de Quinta Categoría) en un archivo binario (archivo DLL). Luego este servicio pueda ser utilizado desde otros programas.

La Tecnología de Componentes de Microsoft tiene tal aceptación en el mercado de software, que por ejemplo, sistemas tan completos como Baan, PeopleSoft y SAP han anunciado el soporte a componentes COM.

Utilizar la Tecnología de Componentes significa un cambio en la forma como se han estado programado las aplicaciones. De enfoques basados en colocar las reglas de negocio en el mismo código del cliente, o de colocarlas en procedimientos sin ninguna agrupación lógica; a esquemas donde se hace el análisis y el diseño de la aplicación con alguna metodología orientada a objetos. El análisis orientado a objetos proporciona los **objetos de negocio**, incluyendo sus propiedades y servicios (métodos).

Luego, en el diseño de la aplicación se agrupan los objetos de negocio en **componentes de código.** Estos componentes pueden ser re utilizados en otras aplicaciones que necesiten los mismos servicios.

Con la Tecnología de Componentes COM se logra el gran sueño de muchos años de lograr crear nuevas aplicaciones basándose en **código re utilizable** que se haya creado anteriormente.

Otra de las características importantes de la Tecnología de Componentes, es que los componentes pueden ser creados en cualquier lenguaje de programación que permita crear Componentes COM, y luego estos Componentes son utilizados desde otros lenguajes de programación, sin importar en que lenguaje fue creado el Componente. Así, por ejemplo, se puede crear un componente COM en Visual Basic, el cual contenga las reglas del negocio de una aplicación. Luego, desde una interfaz gráfica programada en Visual C++ (o Visual FoxPro,

o incluso una página web), hacer uso de los servicios que ofrece ese componente COM en particular.

Es precisamente lo anterior, lo que lo diferencia de un esquema de programación basado en objetos, donde el re uso de código sólo es posible utilizando el mismo lenguaje de programación en que fue creado el objeto. Así en un esquema de programación por objetos, si creamos el objeto en Visual C++, sólo podremos re utilizarlo mediante el concepto de herencia en otro programa en el mismo lenguaje Visual C++; más aún debemos contar con el código fuente del objeto.

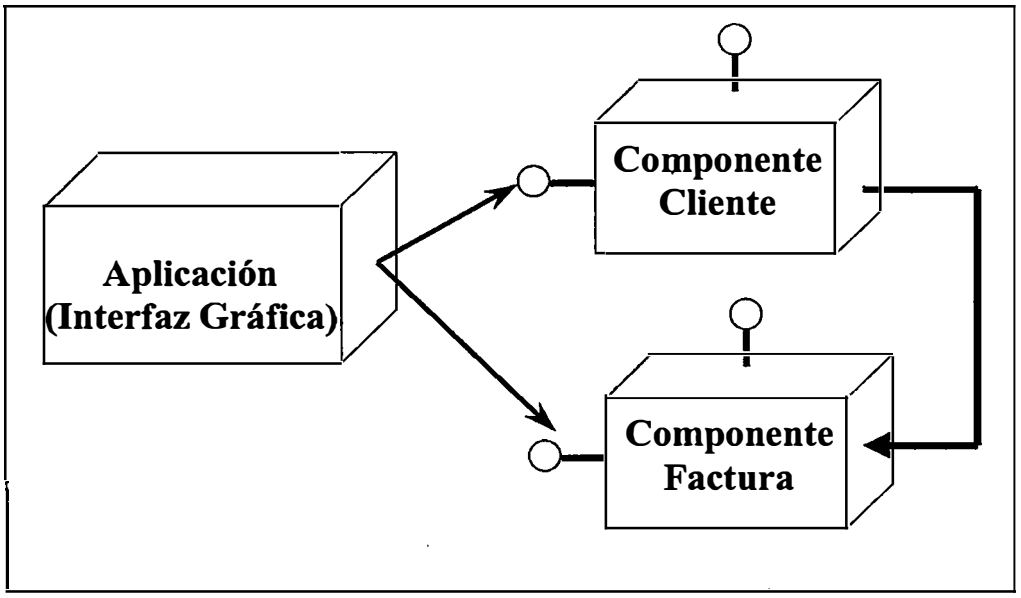


Figura 6: Tecnología de Componentes. *Las aplicaciones hacen uso de los componentes simplemente invocado sus servicios (métodos). Un componentè puede hacer uso de los servicios de otros componente.*

La Figura 6 muestra una aplicación cliente que hace uso de la funcionalidad ofrecida por los componentes que contienen las reglas de negocio de los Clientes y las Facturas de un sistema. Estos componentes pudieron haberse creado anteriormente para otros sistemas; sin embargo, son utilizados desde esta nueva aplicación. Un componte, a su vez, puede hacer uso de los servicios de otro componente. En la Figura 6, el componente Cliente hace uso de la funcionalidad del componente Factura.

7.2 Características fundamentales de la Tecnología de Componentes (COM)

a) Estándar binario, una vez creado el componente como archivo binario, puede ser utilizado desde cualquier lenguaje de programación, sin necesidad de tener a disposición el código fuente del componente.

b) Se implementa a través de cualquier lenguaje que permita crear un componentes COM: Visual Basic, Visual FoxPro, Visual C++, Visual J++, Power Builder, Delphi, etc.

c) Permite el re uso de código a través de la simple invocación a los servicios que pueda ofrecer el componente.

d) Ampliamente adoptado por la industria de software. Las versiones actuales de Windows NT y Windows 98, así como los lenguajes de programación de Microsoft (conocido como Visual Studio) están enteramente creados mediante componentes COM. Microsoft Office 2000, la próxima versión de Office, cuenta con más de 600 componentes COM a los que incluso los programadores pueden acceder para hacer uso de las características de Office desde sus aplicaciones.

e) Un Componente COM esta totalmente encapsulado, una vez creado el componente (archivo binario), no hay forma de saber como están programados los servicios que ofrece; simplemente se invocan.

f) La única forma de acceder a un componente es a través de una **"interfaz"**, o clase primaria que todo Componente COM posee. Esta interfaz, a su vez, llama a todos los servicios que el Componente puede ofrecer, logrando de esta manera un mayor grado de encapsulamiento.

g) Los componentes COM pueden residir en diferentes ubicaciones físicas y aún así pueden ser invocados desde otros programas o desde otros componentes, sin necesidad de conocer previamente su ubicación física.

h) Un Componente COM puede contener varias clases dentro de sí, donde cada clase representa un agrupamiento lógico de servicios y propiedades, generalmente, asociado a un Objeto de Negocio (es decir, que existe en la realidad) identificado en el proceso de análisis de la aplicación.

7.3 Diferencia entre un Componente COM y un Objeto

Es muy importante mencionar las diferencias entre un Componente COM y un Objeto para poder entender claramente la Tecnología de Componentes COM. Un Componente COM y un Objeto, en la forma como se le conoce en la Teoría de Objetos, son diferentes.

La primera gran diferencia entre un Componente COM y un Objeto, es que la única forma de poder utilizar un Componente COM es a través de una clase

denominada "interfaz", la cual proporciona todo el mecanismo de comunicación con los servicios que el componente puede ofrecer.

Todos los Componentes COM contienen una interfaz estándar denominada **IUnknown**. Esta "interfaz" la contienen todos los Componentes COM y es compilada junto con las otras clases o interfaces que pueda contener el Componente COM.

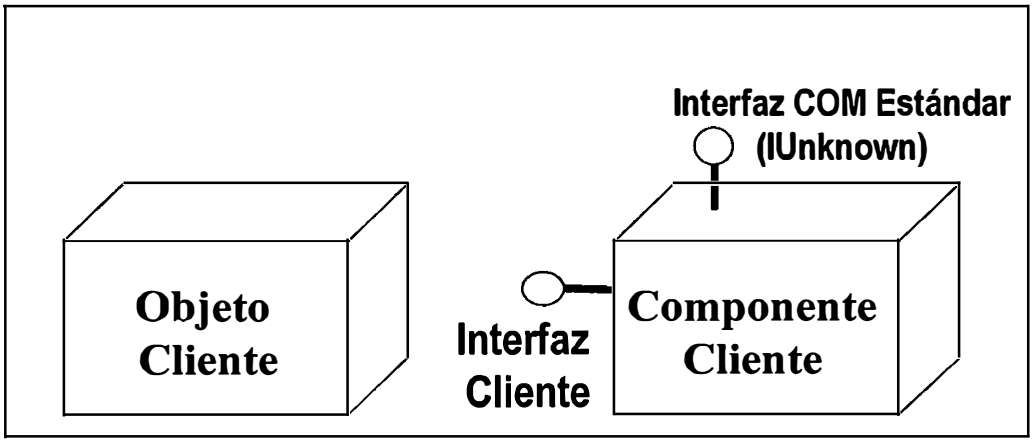


***Figura 7: Objeto y Componente COM**. Un Componente COM y un Objeto no son lo mismo. La única manera de utilizar con un Componente COM es a través de sus "interfaces".*

En la Figura 7 vemos como un Componente COM para un sistema de Clientes contiene dos interfaces: Una es la interfaz estándar de todo Componente COM conocida como IUnknown, y la otra interfaz, que para este ejemplo se le ha llamado "Cliente", la cual contiene realmente la funcionalidad que queremos modelar de los Clientes de nuestro sistema; es decir, los servicios (métodos) y propiedades. Este Componente Cliente podrá ser invocado desde otros

programas, de manera transparente, sin importar cómo se programó; y más aún, sin importar en qué lenguaje de programación se construyó.

Los lenguajes como Visual Basic, Visual J++ y Visual C++ contienen Asistentes para la creación automática de un Componente COM, de tal manera que el programador sólo se preocupa de crear la funcionalidad misma que ofrecerá el componente.

Otra diferencia es que un Componente COM puede contener varias **clases** dentro de sí, donde cada clase representa un agrupamiento lógico de servicios y propiedades, generalmente asociado a un Objeto de Negocio encontrado en el proceso de análisis. Un Objeto de Negocio es aquella entidad de la vida real que se quiere representar mediante un programa. Así por ejemplo, siguiendo con nuestro ejemplo anterior, el Componente COM para nuestro sistema de Clientes pudo haber contenido también la funcionalidad de los Productos y de las Facturas. Si este hubiera sido el caso, nuestro componente COM lo hubiéramos llamado Componente Facturación, el cual hubiera contenido, en el más simple de los casos, cuatro interfaces: la estándar (IUnknown), Cliente, Producto y Factura. A su vez, cada una de las tres últimas interfaces, hubieran contenido los servicios (métodos) y propiedades de los Clientes, Productos y Facturas de la vida real.

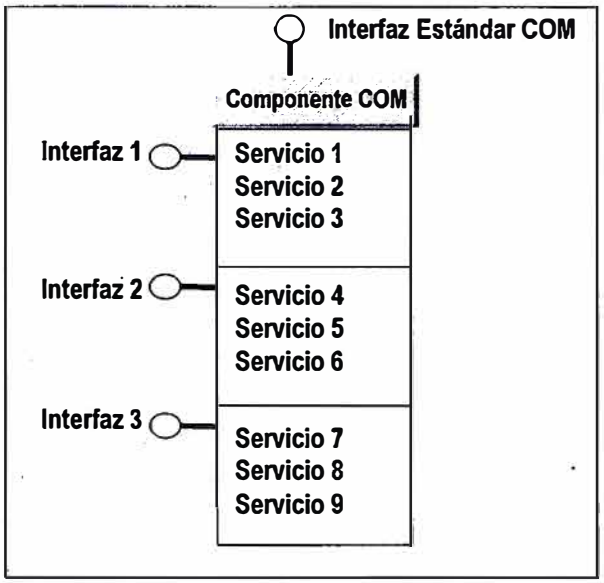


Figura 8: Componente COM con varias interfaces. *Un Componente COM puede contener varias interfaces que representen varios Objetos de Negocio encapsulados en un sólo Componente o Caja Negra.*

7.4 Administración de Componentes COM en un ambiente distribuido

Uno de los principales beneficios de crear Componentes COM es que los cambios que tengamos que hacer en las Reglas del Negocio se harán únicamente en él(los) Componente(s) que se necesita cambiar y no en toda la aplicación.

Las aplicaciones clásicas se han estado creando de tal manera que el código que representaba las Reglas del Negocio se mezclaba con el código de la interfaz gráfica de usuario. Esto provocaba que ante cambios en alguna Regla del Negocio, se tuviera que recompilar toda la aplicación y más aún reinstalar la aplicación en cada computadora donde se estuviera utilizando. Por el contrario,

con un enfoque basado en Componentes COM, los cambios se hacen en el Componente necesario, sin tocar el código de la interfaz gráfica de usuario. Más aún con un enfoque basado en Componentes COM centralizados en uno o varios servidores, con las interfaces de usuario accediendo a estos servidores, los cambios sólo se hacen en el servidor que contiene el Componente COM involucrado en el cambio. Con esto se logra que los cambios en las Reglas del Negocio que es lo más frecuente en las organizaciones, sólo signifiquen actualizar los Componentes COM en el servidor(es) de aplicaciones, sin representar un cambio en las aplicaciones clientes, representando esto un gran ahorro en los costos de mantenimiento de los sistemas.

Para administrar los Componentes COM que residen en uno o varios servidores de aplicaciones y que son utilizados por muchas aplicaciones clientes que acceden en forma simultánea, existe el Microsoft Transaction Server (MTS). El MTS permite tener en uno o varios servidores centrales, los diferentes Componentes COM de una aplicación y administrar el uso de estos Componentes COM por las diferentes aplicaciones cliente que lo requieran. Uno de los principales beneficios de crear Componentes COM para el Microsoft Transaction Server es que los Componentes se programan pensando en que serán usados en un ambiente monousuario, siendo el MTS quien se encarga de administrarlos en un ambiente multiusuario.

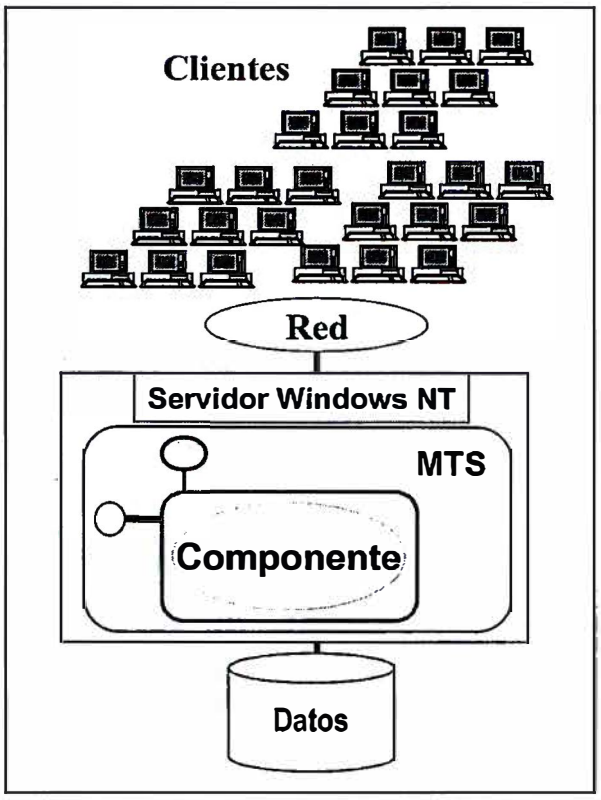


Figura 9: Microsoft Transaction Server (MTS). *Permite centralizar los Componentes COM en uno o varios servidores centrales. Los Componentes contienen las Reglas del Negocio. Las aplicaciones clientes utilizan los servicios de los Componentes COM a través de la red.*

Otro de los beneficios de crear un Componente COM que será administrado por MTS es que si el Componente involucra una actualización en una base de datos, en caso de ocurrir algún error y la transacción no ha terminado, todo vuelve al estado inicial de la transacción en la(s) base(s) de datos involucrada(s). Microsoft Transaction Server permite crear servicios (métodos) transaccionales,

donde la característica es: Se completa toda la transacción que se está procesando (Commit), o no se procesa nada (Rollback). Esto permite crear aplicaciones muy seguras y robustas, siendo el MTS y no el programador, quien administra todo el esquema transaccional.

En la Figura 9 podemos ver cuál es la arquitectura de una aplicación cuando se utiliza Microsoft Transaction Server (MTS). Siguiendo con el ejemplo anterior, el Componente COM de Facturación es administrado por MTS, el cual a su vez, reside en un servidor Windows NT. El Componente COM de Facturación contiene todas las Reglas de Negocio de Productos, Clientes y Facturas. La aplicación cliente instalada en cada una de las computadoras de los usuarios del Sistema de Facturación utiliza los servicios ofrecidos por el Componente COM de Facturación. A su vez este último es quien hace todo el trabajo de interacción con la base de datos utilizada en el sistema.

Precisamente, la Figura 9 ayuda a explicar lo que es un esquema de tres capas (Three Tier) lógico y n capas (N-tier) de implementación física. Las tres capas lógicas son la interfaz gráfica de usuario, las Reglas de Negocio y las fuentes de información (base de datos). Cuando este esquema lo llevamos a un diseño físico (implementación) tenemos: la aplicación cliente, el o los componentes conteniendo las reglas de negocio en uno o varios servidores de aplicaciones y las fuentes de información, también, en uno o varios servidores de datos. A esto se le denomina esquema físico o de implementación de N Capas o N-tier.

La Arquitectura Windows DNA permite crear aplicaciones distribuidas a través de la red de la organización. Los beneficios de un esquema de este tipo son, entre otros, la posibilidad de crear aplicaciones utilizando componentes previamente creados para otras aplicaciones similares, o incluso mejorar los componentes existentes; el poco esfuerzo de mantenimiento de la aplicación en la medida que los mayores cambios están, en la mayoría de los casos, en las reglas de negocio que residen en servidores centralizados de aplicaciones. Otro de los beneficios es la posibilidad que la misma aplicación que fue creada para unos pocos usuarios, sin necesidad de hacer cambios, pueda servir para cientos y hasta miles de usuarios, sólo repotenciando el hardware de los servidores de aplicaciones. La aplicación cliente y los Componentes COM no necesitan ningún cambio. Con esto se logra aplicaciones robustas y altamente escalables.

8 Fuentes de Información en la Arquitectura Windows DNA

La información crítica de los negocios ya no está solamente en base de datos relacionales; por el contrario, reside ahora, en un sinnúmero de otros formatos, llámese archivos de voz, gráficos, videos, imágenes, textos, sistemas de mensajería, etc.; a los cuales las empresas desean acceder para poder analizarlos, y que puedan participar en sus procesos de análisis de información y de toma de decisiones.

Precisamente, la Arquitectura Windows DNA permite construir aplicaciones donde la fuente de información puede ser estructurada o no estructurada. Podemos almacenar la información de las aplicaciones en una base de datos relacional (RDBMS), como en un archivo texto, o incluso en un sistema de mensajería sin una estructura relacional de información.

Para lograr lo anterior, la Arquitectura Windows DNA hace uso de una tecnología denominada Acceso Universal a Datos o **Universal Data Access** (UDA) de Microsoft. Esta tecnología tiene el principio fundamental que la información reside en diferentes formatos y ubicaciones físicas dentro de la organización y que lo que realmente se requiere es un esquema de Acceso Unico a cualquier fuente de información, más que un esquema de base de datos universal.

La tecnología de Acceso Universal a Datos define una forma estándar de acceder a los datos, utilizando las mismas sentencias de programación cualquiera sea el tipo y formato de la información. El beneficio principal que obtienen los programadores es que pueden crear aplicaciones que puedan conectarse a diferentes fuentes de información y sólo deben aprender un sólo estándar de conexión y no uno para cada tipo de formato de información que quieran acceder.

8.1 Tecnología de Acceso Universal a Datos (Universal Data Acccess)

La tecnología de Acceso Universal a Datos consta de tres elementos: OLE DB, ActiveX Data Object (ADO) y Open Database Connectivity (ODBC).

OLE DB es una interfaz o driver de conexión a bajo nivel para diferentes fuentes de información, desde fuentes estructuradas a no estructuradas. Así, existen drivers de OLE DB para Microsoft SQL Server, como drivers para archivos texto, información gráfica, sistemas de mensajería, etc. OLE DB esta basado en la tecnología COM, por lo que crear un driver OLE DB es simple y está a la mano de cualquier proveedor de información. Por ejemplo, ya existen drivers OLE DB para Oracle, DB/2, etc. El programador no requiere utilizar directamente el driver de OLE DB; sólo necesita conocer con las sentencias de programación de ActiveX Data Object (ADO).

Open Database Connectivity (ODBC). Es el acceso más conocido a las bases de datos relacionales. Adicionalmente, OLE DB for ODBC utiliza los drivers de ODBC existentes para acceder a las bases de datos relacionales.

ActiveX Data Object (ADO), es una interfaz de alto nivel. ADO es un modelo de programación (sentencia de programación) que evita que el desarrollador tenga que manipular directamente los drivers de OLE DB u ODBC. Todos los lenguajes de programación de Microsoft: Visual Basic, Visual C++, Visual FoxPro, e incluso Visual J++ (el compilador de Java de Microsoft) hacen uso de ADO para acceder a las diferentes fuentes de información, tanto relacionales como no relacionales. Con esto, los desarrolladores pueden acceder prácticamente a cualquier tipo de información y sólo necesitan aprender un único modelo de datos (ADO), ganando tiempo en el desarrollo de aplicaciones cliente/servidor tanto para intranet como para Internet.

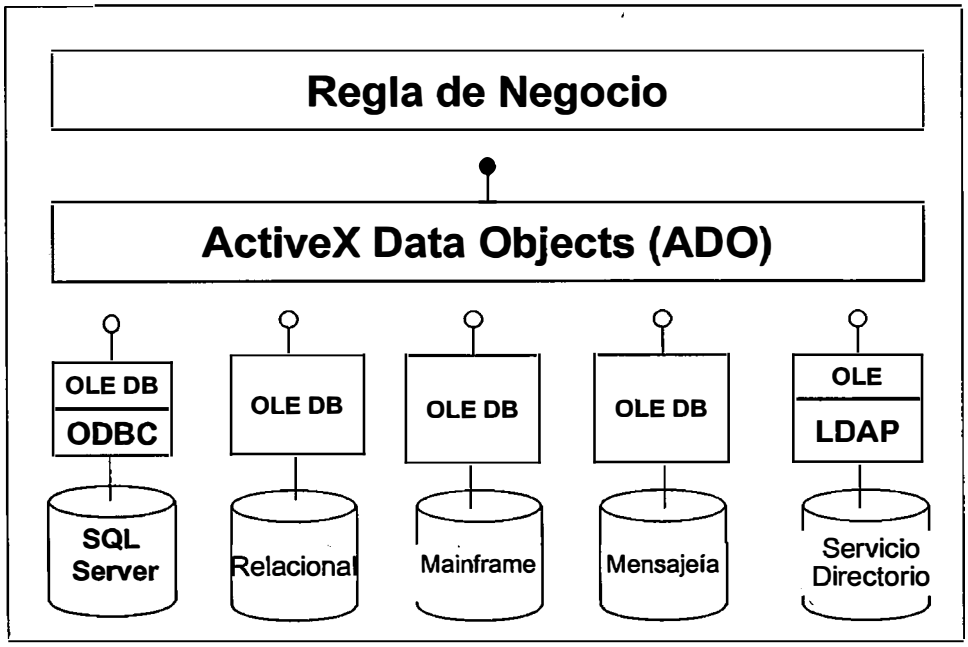


Figura 10: Acceso Universal a Datos (Universal Data Access). *Esta Tecnología permite acceder a diferentes formatos de información (base de datos relacionales, sistemas de mensajería, voz, texto, etc.) utilizando un único modelo de programación denominado ActiveX Data Object (ADO)*

La Figura 10 muestra la Tecnología de Acceso Universal a Datos. Desde el Componente de Reglas de Negocio se accede a fuentes de información de diferentes formatos utilizando un único modelo de programación ActiveX Data Object (ADO), y un único driver de conexión denominado OLE DB. Como se aprecia en el gráfico para acceder a una fuente de información determinada, sólo se requiere el driver de conexión adecuado, las sentencias de programación son las mismas. Esto permite que los programadores sólo requieran aprender un único modelo de acceso, sin importar la fuente de información a conectarse.

Siguiendo con nuestro ejemplo anterior, el Componente COM de Facturación contenía tres interfaces, una de ellas era la que contenía toda la funcionalidad de Objeto de Negocio Producto. Desde uno de los servicios (métodos) de la interfaz Producto podemos acceder a la información de la tabla relacional Productos, a través de una sentencia de programación que utilice el modelo de datos ADO (ActiveX Data Object). A su vez, en el servidor de aplicaciones donde está el Componente COM de Facturación, está instalado el driver de OLE DB para el servidor de datos relacional Microsoft SQL Server (o cualquier otra fuente de información) donde reside físicamente la tabla. El servidor SQL Server retorna los datos al servicio que lo invocó. Este último, a su vez, hace el proceso de las reglas de negocio que requiere y puede retornar alguna información a la aplicación cliente o directamente actualizar la tabla Productos o cualquier otra tabla involucrada en la regla de negocio.

9 Diseño de una aplicación utilizando la arquitectura Windows DNA

Anteriores se ha mencionado cada uno de los elementos o capas que conforman una aplicación: Interfaz Gráfica de Usuario, Reglas de Negocio y Fuentes de Información. La arquitectura Windows DNA define una serie de principios que se deben seguir para la construcción de cada una de estas capas. Lo que vamos a explicar, ahora, es cómo unir estas capas para crear aplicaciones distribuidas tanto para ambientes de cliente/servidor en intranet o Internet, como para aplicaciones cliente/servidor tradicionales.

9.1 Radiografía de una Aplicación Windows DNA

La parte central de cualquier sistema son las reglas del negocio que representan la forma cómo funcionan los procesos de negocios dentro de las organizaciones. Para modelar estas reglas del negocio, la Arquitectura Windows DNA propone crear Componentes COM. Estos Componentes tienen la ventaja que pueden ser creados en cualquier lenguaje de programación y luego ser utilizados desde la aplicación que requiera los servicios que el componente ofrezca; aún cuando esta aplicación se esté desarrollando en un lenguaje de programación diferente al que se utilizó para crear el Componente.

Más aún, los Componentes COM pueden ser luego instalados en uno o varios servidores de aplicaciones desde donde las aplicaciones cliente hacen uso de los

servicios que ofrecen estos Componentes. Para administrar estos Componentes en los servidores, existe el Microsoft Transaction Server, el cual es un administrador de componentes, permitiendo crear aplicaciones transaccionales y altamente escalables con el mínimo de esfuerzo.

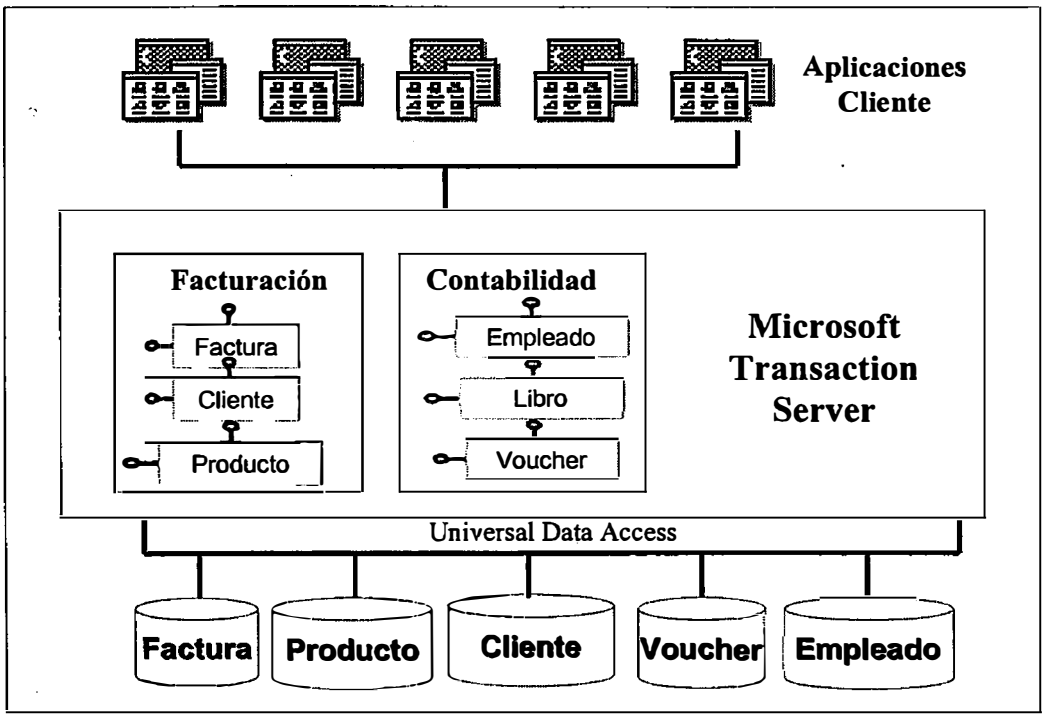


Figura 11: Radiografía de una Aplicación Windows DNA. *Las aplicaciones cliente utilizan los servicios de los Componentes de Regla de Negocio. Son los componentes quienes manipulan las fuentes de información.*

La Figura 11 nos una radiografía de una aplicación Windows DNA. Las aplicaciones cliente (interfaz gráfica de usuario) hacen uso de los servicios ofrecidos por los diferentes componentes de Reglas de Negocio (por ejemplo, Factura, Empleado, etc.). Los Componentes son administrados por el Microsoft Transaction Server (MTS). A su vez, los componentes son quienes manipulan

las fuentes de información a través de la tecnología de Acceso Universal a Datos. Por ejemplo, la tabla Factura podría estar almacenada en un servidor SQL Server, mientras que la tabla Producto podría estar almacena en una base de datos Access.

9.2 Distribuyendo una Aplicación Windows DNA

Una vez creada la aplicación, podemos decidir la distribución de los Componentes COM en varios servidores de aplicaciones; así como la separación de las fuentes de información en varios servidores de datos. Esto permite que una aplicación escale o crezca según la necesidad del negocio, y de la carga de trabajo que recibirán tantos los Componentes como las fuentes de información. Para escalar sólo necesitamos agregar nuevos servidores, no se necesita en absoluto cambiar la aplicación cliente, y menos los Componentes COM. Más aún, como se explicó anteriormente, los Componentes COM fueron creados pensando en que iban a ser usados en un ambiente monousuario, siendo el Microsoft Transaction Server quien administra todo el esquema transaccional y multiusuario.

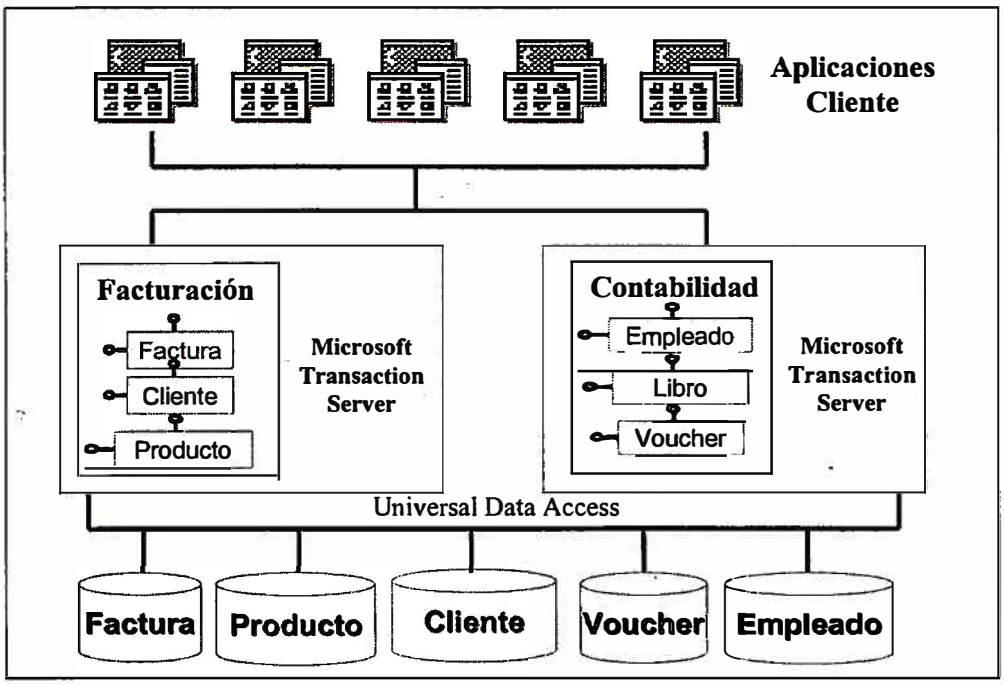


Figura 12: Aplicación Windows DNA Distribuida. *Los Componentes y las fuentes de información pueden estar instalados en varios servidores. Las aplicaciones cliente no necesitan ser modificadas.*

La Figura 12 muestra cómo se puede distribuir una Aplicaciones Windows DNA en varios servidores de aplicaciones y de datos. Las aplicaciones clientes (interfaz gráfica de usuario) pueden acceder a diferentes componentes, sin importar su ubicación física dentro de la red de la organización. A su vez, los Componentes COM, a través de la Tecnología de Acceso Universal a Datos, pueden acceder a fuentes de información de diferentes formatos (relacional, gráfica, texto, etc.) que pueden estar, también, en diferentes ubicaciones físicas.

9.3 Centralización de Reglas de Negocio en Componentes COM

Las aplicaciones desarrolladas con la Arquitectura Windows DNA permiten diferentes tipos de interfaces gráficas de usuario, tanto en Windows, como Browsers de diferentes plataformas (Windows, UNIX, etc.). Esto se logra gracias a que las Reglas de Negocio están centralizadas en uno o varios servidores de aplicaciones, y accedidas desde las aplicaciones cliente.

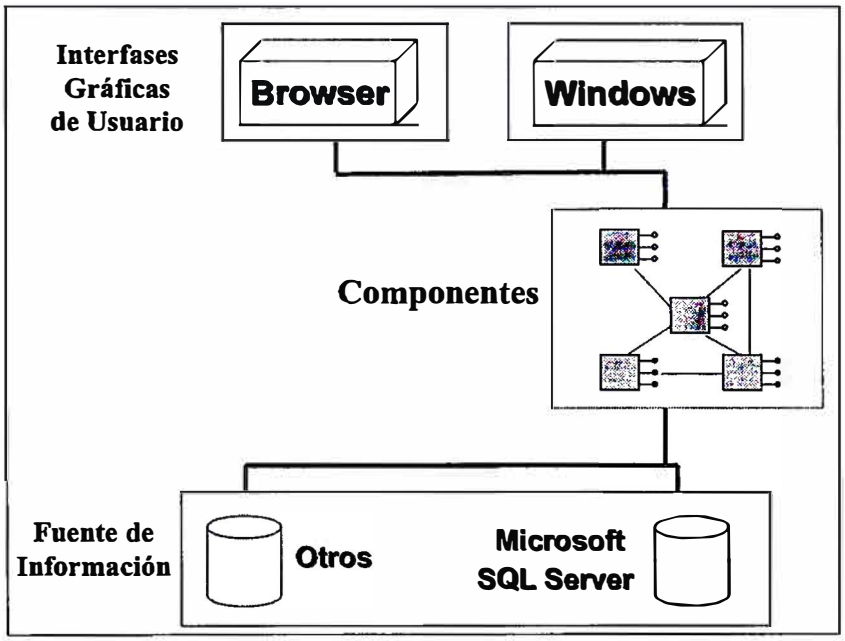


Figura 13: Tipos de Interfaces Gráficas de Usuario de una Aplicación Windows DNA.

La Figura 13 muestra como una aplicación creada utilizando là Arquitectura Windows DNA puede tener varios tipos de interfaces gráficas de usuario. Se puede crear una aplicación que va a tener una interfaz gráfica tipo Windows (denominadas Win32), y la misma aplicación, con la misma funcionalidad, tener

una interfaz tipo Browser, de diferentes plataformas (Windows, UNIX, etc.). Este último tipo de aplicación sería para un ambiente de intranet o Internet.

La clave para poder crear aplicaciones que tengan diferentes tipos de interfaces gráficas de usuario, está en crear las Reglas del Negocio en Componentes COM centralizadas en uno o varios servidores de aplicaciones. Las aplicaciones cliente pueden hacer uso de los servicios que ofrecen los Componentes COM.

Uno de los beneficios de la Arquitectura Windows DNA, es la posibilidad de crear aplicaciones cliente/servidor para ambientes corporativos basados en Windows y que sean fácilmente adaptados para ofrecer servicios a través de Internet o de una intranet. Sólo basta crear la interfaz gráfica en ambiente Browser y hacer uso de los mismos Componentes COM que utiliza la aplicación corporativa.

9.4 Topología de una aplicación intranet o Internet

El componente principal de una aplicación en ambiente de Internet o intranet es el Servidor Web, el cual proporciona los servicios necesarios para enviar las páginas web a las computadoras conectadas al Servidor Web. Como hemos mencionado anteriormente, las aplicaciones Windows DNA pueden ser llevadas rápidamente a un ambiente Internet o intranet. Si todas las Reglas del Negocio han sido programadas creando Componentes COM, las páginas web que conforman la interfaz gráfica de usuario pueden hacer uso de estos Componentes.

Microsoft ha creado un tipo especial de página Web denominada **Active Server Pages** (ASP) o Página Activa de Servidor cuya particularidad es tener código Visual Basic Script o JScript junto con el código HTML. El código Script se ejecuta en el servidor antes de enviar el código HTML al usuario. Al ser estándar el código HTML, el Browser del usuario que se haya conectado al servidor web puede ser de diferentes plataformas: Windows, UNIX, etc.

El código Script permite hacer uso de los servicios ofrecidos por los Componentes COM, con lo cual se puede construir aplicaciones cliente/servidor para ambientes multiplataforma, tanto en intranet como en Internet.

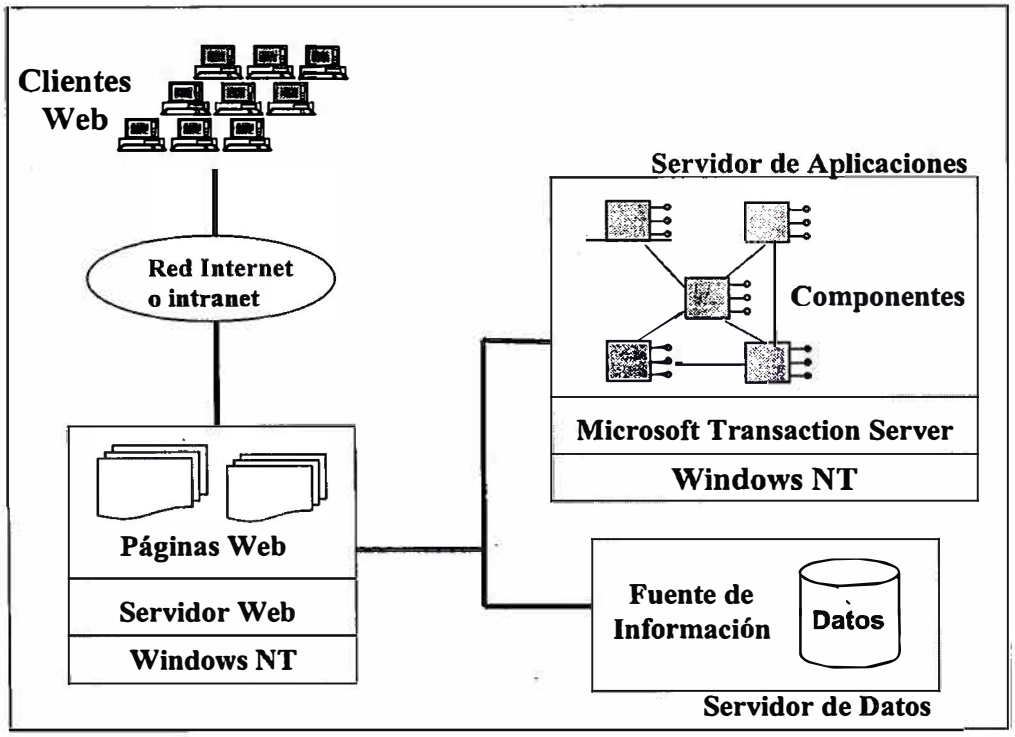


Figura 14: Topología de una Aplicación intranet o Internet

La Figura 14, muestra la topología de una aplicación para ambiente de intranet o Internet. Los usuarios se conectan utilizando un Browser al servidor Web, el cual a su vez está conectado a los Servidores de Aplicaciones y de Datos. Desde las páginas web, se hace uso de los servicios que ofrecen los Componentes, los cuales están administrados por el Microsoft Transaction Server. Los Componentes contienen las Reglas del Negocio de la Aplicación, y consultan y/o actualizan la información.

10 Beneficios de la Arquitectura Windows DNA

Las organizaciones requieren aplicaciones que puedan construirse fácilmente y que se adapten rápidamente a los cambios. De esta manera, las empresas están más preparadas para afrontar los retos de la modernidad y de la competencia.

Precisamente, para satisfacer esta necesidad, Microsoft ha definido una arquitectura para desarrollar aplicaciones distribuidas para ambientes de intranet e Internet, denominada Windows DNA. Utilizando la arquitectura Windows DNA, las empresas pueden construir aplicaciones escalables que puedan crecer con la organización; accediendo a fuentes de información de diferentes formatos; re utilizar componentes que contengan las Reglas del Negocio; utilizando interfaces gráficas de usuario de diferentes plataformas (Windows, UNIX, etc.).

La Arquitectura Windows DNA proporciona los siguientes beneficios

Flexibilidad de la Aplicación: Las aplicaciones Windows DNA se dividen en tres elementos independientes entre sí: Interfaz gráfica de usuario, Reglas de Negocio y Fuente de Información. Esto permite la modularidad de la aplicación y adaptación rápida a los cambios. Al momento de implantar la solución, los componentes de reglas de negocio y las fuentes de información pueden estar en diferentes servidores lo que facilita la escalabilidad de la aplicación.

Independencia en el tipo de Interfaz Gráfica de Usuario: La arquitectura Windows DNA permite crear aplicaciones con interfaz gráfica de usuario que utilicen Browsers de diferentes plataformas (Windows, UNIX, etc.). Asimismo, se puede crear interfaces gráficas que utilicen la plataforma Windows al máximo.

Acceso Universal a Datos: Microsoft ha desarrollado una tecnología que permite acceder a diferentes fuentes de información: base de datos relacional, voz, gráficos, sistemas de mensajería, etc. El acceso a los datos se logra a través del modelo de programación denominado ActiveX Data Object (ADO).

Re utilización de componentes: Las Reglas de Negocio de una organización se pueden modelar y encapsular en Componentes de código utilizando la Tecnología de Componentes de Microsoft (Component Object Model, COM). Estos componentes se pueden re utilizar en otros sistemas, permitiendo el desarrollo rápido de aplicaciones.

Elección del Lenguaje de Programación: Los Componentes COM pueden ser creados con diferentes lenguajes de programación. El Componente COM es un archivo binario (archivo DLL), por lo que puede ser utilizado desde otros lenguajes de programación, aún cuando sea diferente al lenguaje de programación en que se creó el componente. Esto permita que los desarrolladores se concentren en las Reglas del Negocio, más que en los aspectos técnicos.

Escalabilidad: Los Componentes de Reglas de Negocio se ejecutan dentro del Microsoft Transaction Server, lo que permite que la misma aplicación sea utilizada por unos cuantos usuarios, como por cientos, y hasta miles de usuarios concurrentes. Microsoft Transaction Server administra los componentes COM, proporcionando el manejo transaccional y multiusuario de la aplicación en forma automática. Uno de los beneficios de utilizar Microsoft Transaction Server (MTS) es que los programadores crean las aplicaciones pensando que serán utilizadas en un ambiente monousuario, el MTS proporciona el manejo multiusuario.

11 Desventajas de la Arquitectura Windows DNA

- La Arquitectura Windows DNA se aplica solamente en aquellas empresas donde los servidores de red son Servidores Microsoft Windows NT. Es indispensable, por la tecnología que se requiere en el Servidor Web o Servidor Intranet, que dicho servidor sea Microsoft Windows NT.
- La combinación de tecnologías y productos que emplea la Arquitectura Windows DNA, requiere de analistas de sistemas altamente entrenados en todos estos productos y tecnologías para su adecuada y eficiente utilización.

12 Conclusiones y Recomendaciones

12.1 Conclusiones

- La Arquitectura Windows DNA define una arquitectura de desarrollo basada en componentes de código, que permiten el desarrollo rápido de aplicaciones.
- La Arquitectura Windows DNA se aprovecha mejor en una red donde los servidores son Windows NT.
- Al dividir una aplicación en tres capas lógicas (Interfaz Gráfica de Usuario, Reglas de Negocio y Fuentes de Información) se logra independencia y rapidez en el desarrollo de las diferentes partes de una aplicación.
- La tecnología de Componentes COM, permite crear software que puede ser utilizado en otros sistemas, reduciendo el tiempo y costo de desarrollo. Sin embargo, dicha tecnología requiere un proceso formal de aprendizaje para llegar a dominarlo y utilizarlo adecuadamente.
- Para aplicar adecuadamente la Arquitectura Windows DNA, los analistas de sistemas requieren tener conocimientos muy actualizados en tecnología de información.

12.2 Recomendaciones

- Se recomienda evaluar la arquitectura Windows DNA para el desarrollo de aplicaciones modernas en una organización, siempre y cuando los servidores de red de la empresa sean Windows NT.
- Para aplicar esta arquitectura, los analistas de una empresa deben primero capacitarse en tecnologías y productos tales como Windows NT, Internet, Component Object Model (COM), Sistemas Distribuidos, Microsoft Transaction Server, etc. Esta capacitación es necesaria, por que la Arquitectura Windows DNA junta todas estas tecnologías y productos en un solo esquema conceptual.
- Es importante antes de empezar a crear aplicaciones con la Arquitectura Windows DNA, evaluar como otras empresas del país han utilizado esta arquitectura. Dicha evaluación permitirá tener parámetros de comparación y referencia de cómo la Arquitectura Windows DNA se aplica a la realidad de las empresas peruanas.

13 Bibliografía

Microsoft Developer Network (MSDN), Diversos Artículos, Microsoft Press, 1998

Introducing Intranets, Gordon Benett, QUE, 1996

Microsoft Transaction Server 2.0, Roger Jennings, SAMS Publishing, 1997

Software Project Survival Guide, Steve McConnel, Microsoft Press, 1998

Microsoft TechNet, Diversos Artículos, Microsoft Press, 1998

14 Anexos

14.1 Anexo 1: Ejemplo de Aplicación

Como parte del trabajo se presentará un ejemplo de aplicación de la Arquitectura Windows, el cual se mostrará al momento de la sustentación del presente trabajo.

Descripción del Ejemplo de Aplicación:

Desarrollar una aplicación de **Mesa de Ayuda** (*Help Desk*), en la cual los *Usuarios* puedan ingresar los problemas que tengan, tanto de hardware como de software. Luego, estos problemas son asignados en forma automática, a *Ingenieros de Soporte Técnico* quienes ayudan a los usuarios a resolver sus problemas. La interfaz gráfica de usuario que se empleará en este ejemplo de aplicación será de tipo Browser.

Objetos de Negocio de la Aplicación de Ejemplo:

Los Objetos de Negocio encontrados durante el proceso de análisis son:

- Usuario
- Ingeniero
- Incidente

Componentes COM de la Aplicación de Ejemplo:

Para la aplicación de ejemplo se ha creado un sólo componente COM, el cual contiene la siguiente descripción:

- **Componente:** Mesa de Ayuda
- **Nombre de Archivo:** COMUsuario.DLL
- **Lenguaje de Programación:** Microsoft Visual Basic 6.0
- **Clases (Interfaces):**
  - Usuario
  - Ingeniero
  - Incidente
  - IncidenteDetalle

Base de datos del sistema de Mesa de Ayuda:

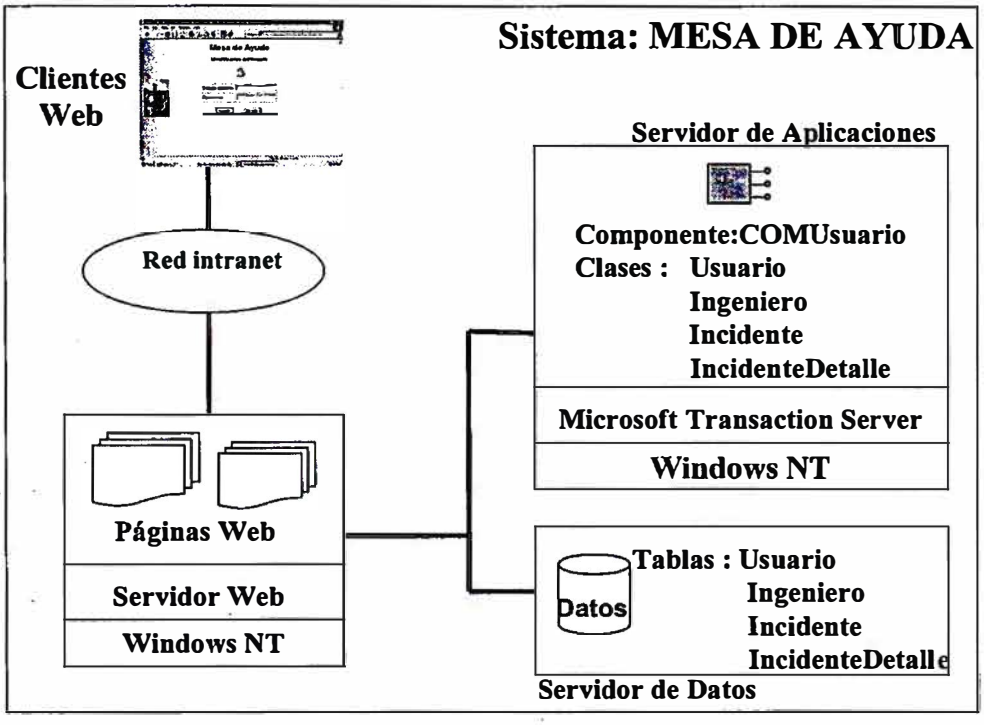
La base de datos utilizada para el ejemplo de aplicación de Mesa de Ayuda es SQL Server 7.0. La información de la base de datos es la siguiente:

- **Base de Datos:** Microsoft SQL Server versión 7.0
- **Nombre de Base de Datos:** MesaAyuda

- **Tablas creadas:**
  - Usuario
  - Incidente
  - Ingeniero
  - IncidenteDetalle
  - CargoUsuario
  - TipoIncidente

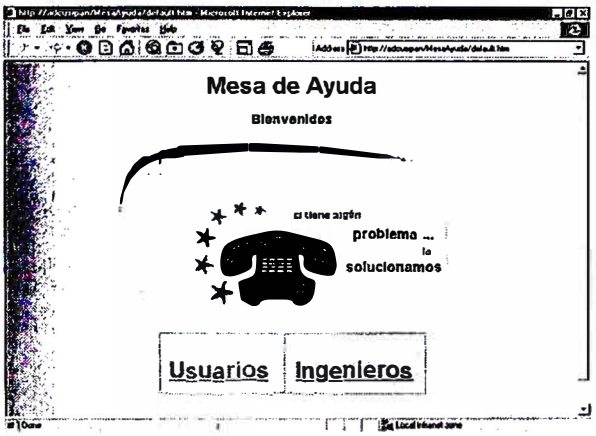
Topología de la Aplicación de Mesa de Ayuda:

El siguiente gráfico muestra los diferentes elementos del ejemplo de aplicación Mesa de Ayuda. El componente COM, COMUsuario, es instalado en un servidor Windows NT. Desde dicho componente se accede a la base de datos SQL Server 7.0 donde se almacena la información de Usuario, Ingeniero e Incidente. Desde las páginas web que residen en el servidor web se accede al componente COMUsuario. Las páginas web fueron creadas con Visual Interdev 6.0.

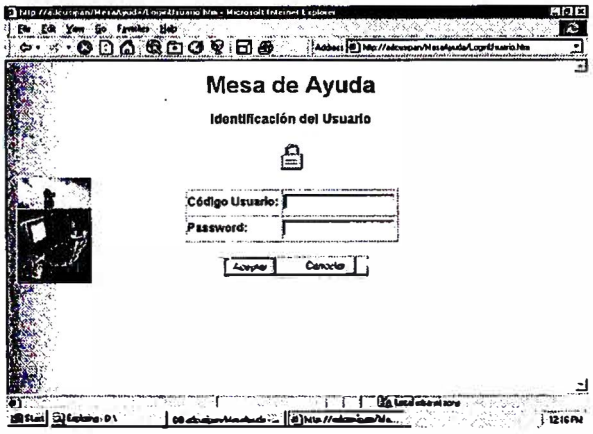


Pantallas de la Aplicación de Ejemplo:

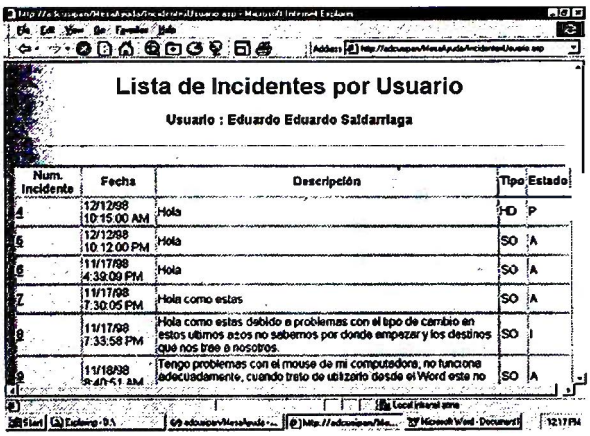
- **Pantalla de Ingreso al Sistema de Mesa de Ayuda.** Mediante esta pantalla el Usuario o el Ingeniero ingresan al sistema de Mesa de Ayuda (Help Desk).



- **Pantalla de Ingreso de los Usuarios**. En esta pantalla los Usuarios del Sistema de Mesa de Ayuda ingresan la clave y password respectivos para poder ingresar al sistema.



- **Lista de Incidentes por Usuario.** Esta pantalla muestra los Incidentes ingresados por cada usuario. Cada Usuario puede utilizar esta pantalla para hacer seguimiento a sus consultas y conocer al Ingeniero asignado a cada Incidente.



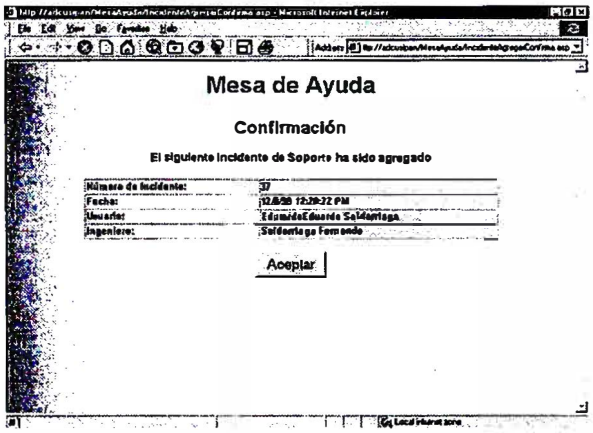
Lista de Incidentes por Usuario

Usuario : Eduardo Eduardo Saldarriaga

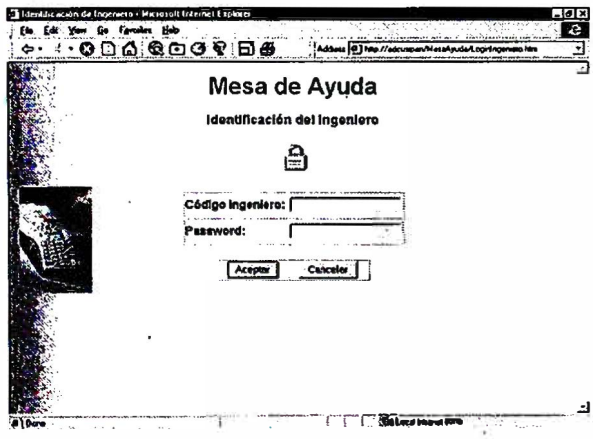
Num. Incidente	Fecha	Descripción	Tipo	Estado
4	12/12/98 10:15:00 AM	Hola	HD	P
5	12/12/98 10:12:00 PM	Hola	SO	A
6	11/17/98 4:39:09 PM	Hola	SO	A
7	11/17/98 7:30:05 PM	Hola como estas	SO	A
8	11/17/98 7:33:58 PM	Hola como estas debido a problemas con el tipo de cambio en estos ultimos años no sabemos por donde empezar y los destinos que nos trae a nosotros.	SO	I
9	11/18/98 [illegible] AM	Tengo problemas con el mouse de mi computadora, no funciona adecuadamente, cuando trato de utilizarlo desde el Word este no	SO	A

- **Pantalla de Ingreso de Incidentes.** Esta pantalla permite a los Usuarios ingresar nuevos Incidentes de Soporte Técnico. Los Incidentes pueden ser de software o hardware y son asignados automáticamente a un Ingeniero.

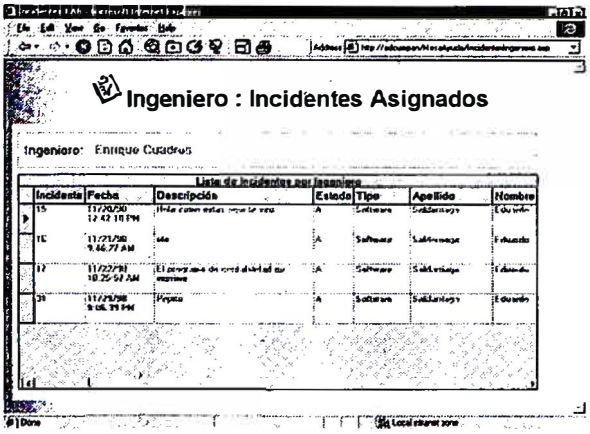
- **Pantalla de Confirmación de Ingreso de Incidentes.** Esta pantalla confirma que se ha agregado un nuevo Incidente de Soporte Técnico.



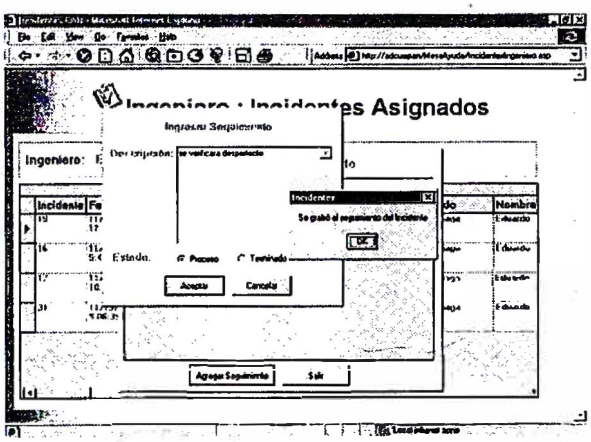
- **Pantalla de Ingreso de los Ingenieros.** En esta pantalla los Ingenieros ingresan la clave y password respectivos para poder ingresar al sistema.



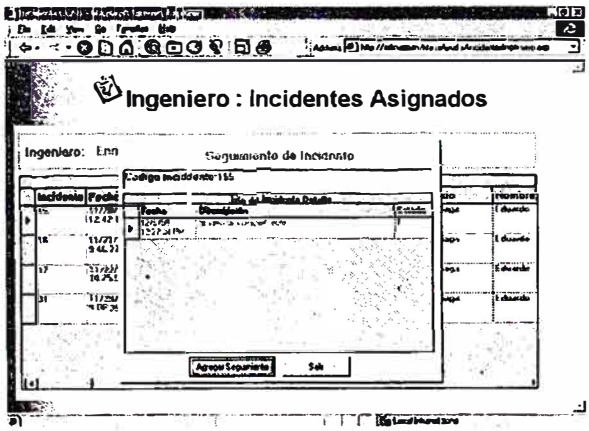
- **Pantalla de Incidentes asignados a un Ingeniero.** En esta pantalla los Ingenieros pueden ver cuales son los Incidentes que tienen pendientes.



- **Pantalla de Ingreso de Seguimiento a los Incidentes asignados a un Ingeniero.** Mediante esta pantalla los Ingenieros pueden ingresar mensajes y comentarios a los Usuarios, sobre la forma de solucionar un Incidente en particular.



- **Pantalla de Seguimiento a los Incidentes asignados a un Ingeniero.** Mediante esta pantalla los Ingenieros pueden hacer seguimiento a los Incidentes, y enviar mensajes y comentarios a los Usuarios, sobre la forma de solucionar el Incidente.



14.2 Anexo 2: Windows® DNA

El siguiente artículo extraído de la pagina web de Microsoft (http://www.microsoft.com). Dichos artículos son publicados por Microsoft para servir como referencia rápida a sus diferentes productos y tecnologías.

Windows® DNA

"Building Windows Applications for the Internet Age"

Microsoft Corporation
October 1998

For some time now, both small and large companies have been building robust applications for personal computers that continue to be ever more powerful and available at ever-lower costs. While these applications are being used by millions of users each day, new forces are having a profound effect on the way software developers build applications today and the platform in which they develop and deploy their application.

The increased presence of Internet technologies is enabling global sharing of information. Not only from small and large businesses, but individuals as well. The Internet has sparked a new creativity in many, resulting in many new businesses popping up overnight, running twenty-four hours a day, seven days a week. Competition and the increased pace of change are putting ever increasing demands for an application platform that enables application developers to build and rapidly deploy highly adaptive applications in order to gain strategic advantage.

It is possible to think of these new Internet applications needing to handle literally millions of users – a scale difficult to imagine a just a few short years ago. As a result, applications need to deal with user volumes of this scale, reliable to operate 24 hours a day and flexible to meet changing business needs. The application platform that underlies these types of applications must also provide a coherent application model along with a set of infrastructure and pre-built services for enabling development and management of these new applications.

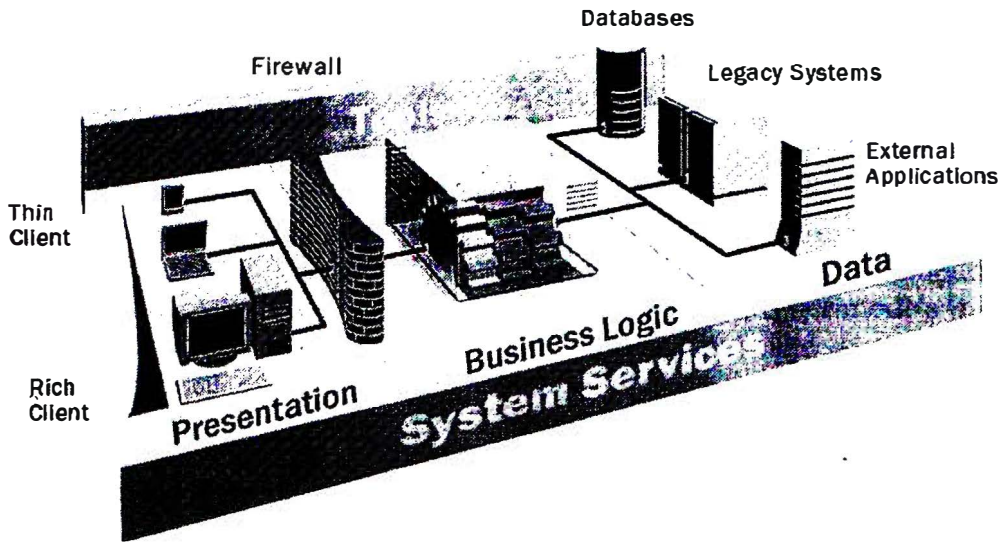


Figure 1. Windows DNA

The Microsoft® application platform consists of a multi-tiered distributed application model called Windows DNA (Figure 1) and a comprehensive set of infrastructure and application services. Windows DNA (http://www.microsoft.com/dna) unifies the best of the services available on personal computers, application servers and mainframes today; the benefits inherent in client-server computing and the best of Internet technologies around a common, component based application architecture. Simply put, guiding principles of Windows DNA are:

Internet Ready. Develop solutions that fully exploit the application platform's flexibility and the Internet's global reach and on-demand communication capabilities.

Faster Time to Market. Develop and deploy solutions rapidly without requiring developer reeducation or a paradigm shift in how software is built. Expose services and functionality through the underlying "plumbing" to reduce the amount of code developers must write.

True Interoperability. Build interoperability into all tiers so functionality can be added to existing systems. Adhere to open protocols and standards so other vendor solutions can be integrated.

Reduced Complexity. Integrate key services directly into the operating system and expose them in a unified way through the components. Reduce the need for IT professionals to function as system integrators so they can focus on solving the business problem.

Language, Tool and Hardware Independence. Provide a language-neutral component model so that developers can use task-appropriate tools. Build on the PC

model of computing, wherein customers can deploy solutions on widely available hardware.

Lower the Total Cost of Ownership. Develop applications that are easy to deploy, manage and change over time.

Given the proper underlying infrastructure, the multi-tier model of presentation, business logic and data can physically distribute processing over many computers. However, the core abstractions that have worked for single and two tier models in the past – high-level programming languages, database management systems, and graphical user interfaces – do not fully address the requirements of multi-tier application development. A different level of abstraction is needed to develop scalable, manageable and maintainable multi-user applications and at Microsoft, we believe this abstraction is cooperating components.

Cooperating Components

Microsoft's Windows DNA strategy rests on Microsoft's vision of cooperating components that are built based on the binary standard called the Component Object Model (COM). COM (http://www.microsoft.com/com) is the most widely used component software model in the world available on over 150 million desktop and servers today. It provides the richest set of integrated services, the widest choice of easy-to-use tools, and the largest set of available applications. In addition, it provides the only currently viable market for reusable, off-the-shelf, client and server components.

COM enables software developers to build applications from binary software components that can be deployed at any tier of the application model. These components provide support for packaging, partitioning, and distributed application functionality. COM enables applications to be developed with components by encapsulating any type of code or application functionality, such as a user interface control or line of business object. A component may have one or more interfaces; each exposes a set of methods and properties that can be queried and set by other components and applications. For example, a customer component might expose various properties such as name, address, and telephone number.

With the Microsoft Windows DNA model, components take away the complexity of building multi-tier applications. Applications based on components and the Windows DNA model rely on a common set of infrastructure and networking services provided in the Windows application platform. The Windows NT® security service, for example, provides access control to the Internet Information Server, as well as transaction and message queuing services. Other common services include systems management, directory services, networking, and hardware support.

Client Environments and Presentation Tier

Today many application developers using cooperating components target the development of their applications to the Windows platform to take full advantage of the rich user interface Windows has to offer. Likewise, customers have come to expect a rich, highly functional user interface from their applications. The extended reach of information and services to customers that the Internet has enabled has created a new challenge for the application developer. The application developer today must develop a user interface that is distributable, available on Windows and non-Windows platforms and supports a wide range of client environments, from handheld wireless devices to high-end workstations. Yet, applications must be rich with features to stay competitive and maintain the functionality that customers have come to expect.

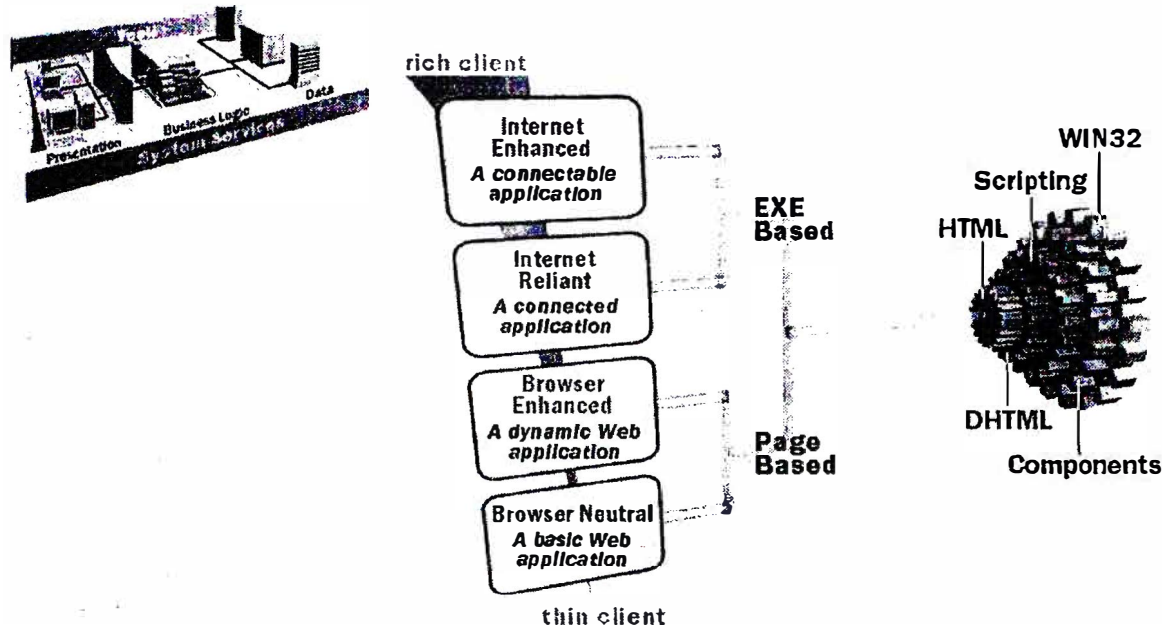


Figure 2. Windows DNA Presentation Approaches

As depicted in Figure 2, Windows DNA offers a broad range of presentation options giving the application developer the choice when developing the best solution. Windows DNA permits the developer to choose the appropriate Windows components and Internet technologies to support the richest possible interface and range of client environments, from handheld wireless devices to high-end workstations.

Maintaining broad reach to a wide range of client environments while achieving the greatest compatibility with all browsers, application developers will generally use standard HTML in developing their Browser Neutral applications. Microsoft tools and application services support the current generation of standard HTML.

The compromise in using static HTML is the reduced amount of functionality and richness in an applications user interface that customers have come to expect. This is

OK for some applications as their application requires broad reach and browser neutrality.

There is a class of applications that don't have a browser neutrality requirement. The reality is that many corporations standardize on a single browser. In addition, application developers who want to provide more functionality in their application than they can achieve with standard HTML write code to determine the browser being used. These Browser Enhanced applications are written to take advantage of the technologies inherent in the browser to gain maximum functionality and richness. With technologies like Dynamic HTML and Scripting, application developers can create actions with functional Web-based interfaces for data entry or reporting without using custom controls of applets.

DHTML is based on the W3C-standard Document Object Model, which makes all Web-page elements programmable objects. Think of DHTML like a "programmable" HTML. Contents of the HTML document, including style and positioning information can be modified dynamically by script code embedded in the page. Thus scripts can change the style, content and structure of a Web page without having to refresh the web page from the web server. By doing so, the client does not have to repeatedly return to the web server for changes in display resulting in increased network performance. Unlike Java applets or ActiveX® controls, DHTML has no dependencies on the underlying virtual machine or operating system. For clients without DHTML support, the content appears in a gracefully degraded form.

There are times when DHTML plus Scripting is not enough. Segments of applications need to leverage the operating system and underlying machine in which it is hosted on, while still maintaining an active connection to the Internet for data or additional services. It is in those instances in which application developers can take advantage of the robust components and Internet services provided by Windows to build Internet Reliant applications. Unlike Page-Based applications that are being run within the context of a browser, an Internet Reliant application is a full fledged Windows Executable that has full access to the broad range of services provided by the Windows client. These applications generally use a combination of HTML, DHTML, Scripting, and ActiveX controls to provide rich integration with the client system as well as full connectivity to remote services on the Internet.

Applications written using the Win32® API offer the most functionality with reach limited to the application platforms that support the Win32 API. Through the use of cooperating components, developers can today have access to Internet technologies in the Windows application platform from within a Win32-based application. Applications written to the Win32 API that takes advantage of system features and leverage Internet connectivity are called Internet Enhanced. Some common examples are Microsoft Office 97 and Visual Studio® 98 development system. These applications support unified browsing, by embedding hyperlinks from within the application, host the browser for the display of documentation written in DHTML and provide the capability to download updates to the products over the Internet seamlessly.

Application Services

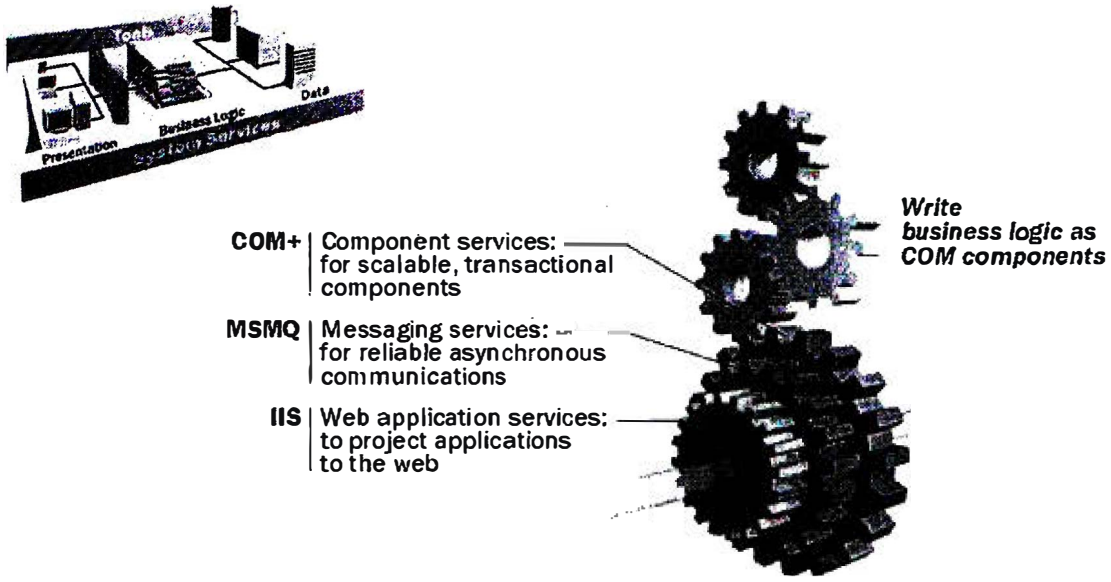


Figure 3. Application Services

The business logic tier is the heart of the application where the application specific processing and business rules are maintained. Business logic placed in components bridge the client environments and the data tiers. The Windows DNA application platform has been developed through years of innovation around supporting high-volume, transactional, large-scale application deployments and provides a powerful runtime environment for hosting business logic components. As depicted in figure 3, the application platform for developing Windows DNA applications include web services, messaging services and component services.

Web Services

Integrated with Microsoft's application platform is a high-performance gateway to the presentation tier. Microsoft's Internet Information Server (http://www.microsoft.com/iis) enables the development of web-based business applications that can be extended over the Internet or deployed over corporate Intranets. With Internet Information Server, Microsoft introduced a new paradigm to the Internet - transactional applications. Transactions are the plumbing that makes it possible to run real business applications with rapid development, easy scalability and reliability.

Active Server Pages (ASP), a component of Internet Information Server, is the language neutral, compile-free, server-side scripting environment that is used to create and run dynamic, interactive Web server applications. By combining DHTML,

scripting, and components, ASP enables application developers to create dynamic interactive Web content and powerful Web-based applications.

With the trend toward distributed computing in enterprise environments, it is important to have flexible and reliable communication among applications. Businesses often require independent applications that are running on different systems to communicate with each other and exchange messages even though the applications may not be running at the same time. Applications built using a combination of Active Server Page scripts communicating with cooperating components can interoperate with existing systems, applications and data.

Component Services

In the early 1990's, the underlying concept that facilitated interoperability was componentization; the underlying technology that enabled interoperability was Microsoft's Component Object Model (COM). As it turns out, componentization is not only a great way to achieve interoperability, but a great way to design and develop software in general. So in the mid-1990's Microsoft broadened COM's applicability beyond the desktop application to also include distributed applications by introducing Microsoft Transaction Server (MTS). MTS was an extension to the COM programming model that provided services for the development, deployment, and management of component-based distributed applications. MTS was a foundation of application platform services that facilitated the development of distributed applications for the Windows platform in a much simpler, more cost effective manner than other alternatives.

COM+ is the next evolutionary step of COM and MTS. The unification of the programming models inherent in COM and MTS services make it easier to develop distributed applications by eliminating the tedious nuances associated with developing, debugging, deploying, and maintaining an application that relies on COM for certain services and MTS for others. The benefits to the application developer is to make it faster, easier, and ultimately cheaper to develop distributed applications by reducing the amount of code required to leverage underlying system services.

To continue to broaden COM and the services offered today in MTS 2.0, COM+ consists of enhancements to existing services as well as new services to the application platform. They include:

Bring Your Own Transaction. COM components are able to participate in transactions managed by non-COM+ transaction processing (TP) environments that support the Transaction Internet Protocol (TIP).

Expanded Security. Support for both role-based security as well as process access permissions security. In the role-based security model, access to various parts of an application is granted or denied based on the logical group or *role* that the caller has been assigned to (e.g. administrator, full-time employee, part-time employee).

COM+ expands on the current implementation of role-based security by including method-level security for both custom and IDispatch(Ex)-based interfaces.

Centralized Administration. The Component Services Explorer, a replacement for today's MTS Explorer and DCOMCNFG presents a unified administrative model, making it easier to deploy, manage, and monitor n-tiered applications by eliminating the overhead of using numerous individual administration tools.

In-Memory Database. For maintaining consistent durable state information and transient state information in a consistent manner. The In-Memory Database is an in-memory fully transactional database system designed to provide extremely fast access to data on the machine that it resides.

Queued Components. For asynchronous deferred execution when cooperating components are disconnected. This is in addition to the session based, synchronous client/server-programming model, where the client maintains a logical connection to the server today.

Event Notification. For times when a loosely coupled event notification mechanism is desirable. COM+ Events is a unicast/multicast, publish/subscribe event mechanism that allows multiple clients to "subscribe" to events that are "published" by various servers. This is in addition to the existing event notification framework delivered with connection points.

Load balancing. Allowing component-based applications to distribute their workload across an application cluster in a client-transparent manner.

Interoperating with Existing Systems, Applications and Data

As companies extend to embrace the Internet, offering new services and making information more readily available to it's customers, partners and employees, it is important that an application architecture provide the necessary means to extend inside the corporation to existing applications and data. Some software vendors have advocated that by wrapping your existing applications and data in a common component model in one language that runs all platforms you can achieve interoperability between applications and data. These are lofty goals that the industry has been promised over the past 30 years that fail to come to fruition. Microsoft's approach has been to reach out to existing applications and data on other platforms without disturbing the platform that the application or data resides on. Microsoft's application platform enables application developers to interoperate with applications and data through the use of Messaging Services, COM Transaction Integrator and through a framework called Universal Data Access.

Messaging Services

Microsoft Message Queue Server (MSMQ) (http://www.microsoft.com/msmq) provides loosely coupled and reliable network communications services based on a

messaging queuing model. MSMQ makes it easy to integrate applications, by implementing a *push-style* business event delivery environment between applications, and build reliable applications that work over unreliable but cost-effective networks. The simple application based on the Component Object Model lets developers focus on business logic and not on sophisticated communications programming. MSMQ also offers seamless interoperability with other message queuing products, such as IBM's MQSeries, through products available from Microsoft's ISV partners.

Extending to the Mainframe Transaction Processing World

Using Microsoft's COM Transaction Integrator (TI), application developers can extend and expose CICS and IMS transaction programs through the use of COM components. COM Transaction Integrator consists of a set of development tools and runtime services that automatically "wrap" IBM mainframe transaction and business logic as COM components that run in a Windows DNA environment. All COM TI processing is done on a Windows NT Server; host communication is accomplished through the SNA Server (http://www.microsoft.com/sna). COM TI does not require the mainframe to run any executable code or be programmed in any special way.

Universal Data Access

Universal Data Access is Microsoft's strategy for providing access to information across the enterprise (http://www.microsoft.com/data). Today, companies building database solutions face a number of challenges as they seek to gain maximum business advantage from the data and information distributed throughout their corporations. Universal Data Access provides high-performance access to a variety of information sources, including relational and non-relational data, and an easy to use programming interface that is tool and language independent.

Universal Data Access does not require expensive and time-consuming movement of data into a single data store, nor does it require commitment to a single vendor's products. Universal Data Access is based on open industry specifications with broad industry support, and works with all major established database platforms.

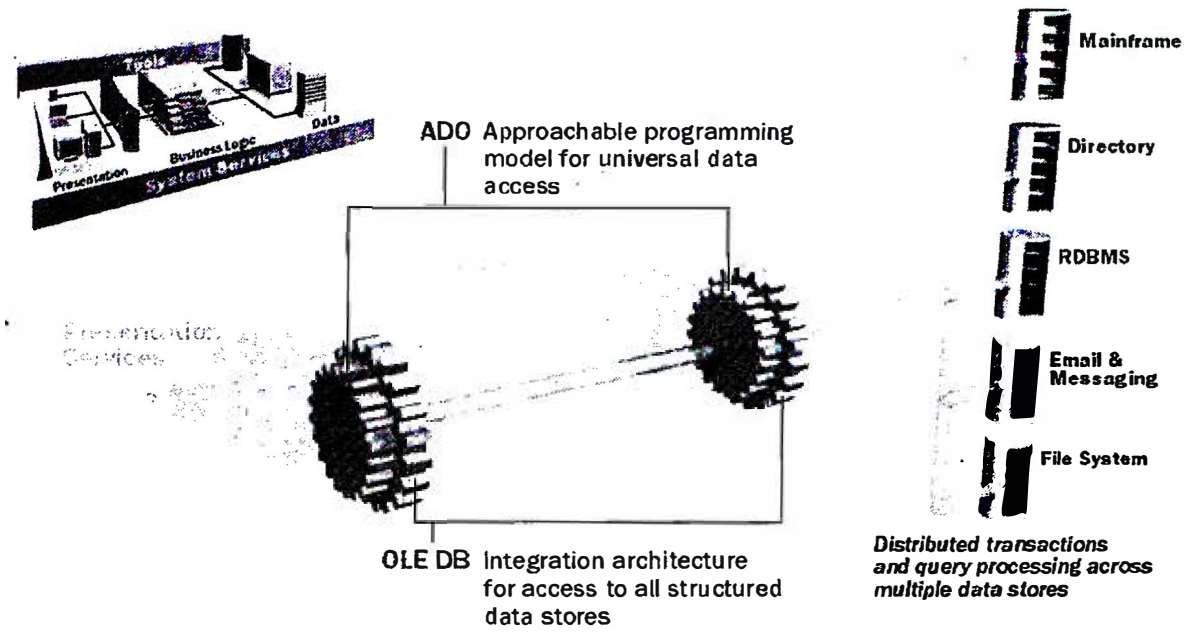


Figure 4. Data Access

As depicted in figure 4, the Universal Data Access based framework operates at two levels. At the systems level, OLE DB defines a component-based architecture specified as a set of COM based interfaces that encapsulate various database management system services. The OLE DB architecture does not constrain the nature of the data source; as a result, Microsoft and Independent Software Vendors have introduced providers for a wide variety of indexed sequential files, groupware products and desktop products (http://www.microsoft.com/data/oledb/products/product.htm). At the application level, ActiveX Data Objects (ADO) provides a high level interface to enable developers to access data from any programming language.

Conclusion

The Microsoft Windows DNA Architecture and the Windows NT platform offer many distinct advantages to customers and its ISV partners. Its key benefits are: Provides a comprehensive and integrated platform for distributed applications freeing developers from the burden of building the required infrastructure or assembling it using a piecemeal approach. Easily interoperates with existing enterprise applications and legacy systems to extend current investments. Makes it faster and easier to build distributed applications by providing a pervasive component model, extensive pre-built application services and a wide choice of programming language and tools support.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. Microsoft makes no warranties, express or implied, in this document.