Area de interés

Sistemas de Comunicación y Redes, Tecnologías y Aplicaciones
Desarrollo de Sistemas de Información

Autores

María Begoña Rodríguez
Patricia Bazán
bego@ada.info.unlp.edu.ar
pbaz@ada.info.unlp.edu.ar

LINTI
Laboratorio de Investigación en Nuevas Tecnologías Informáticas
Facultad de Informática – UNLP
Universidad Nacional de La Plata
Director: Lic. Francisco Javier Díaz

Calle 50 y 115 – 1er Piso – la Plata (1900) – Buenos Aires – Argentina
fax: 0221-423-0124 – tel: 0221-422-3528

Titulo

"Java y .NET comparación de paradigmas"

Resumen

El desarrollo de software distribuido ha tenido una evolución importante en la última década tanto desde el punto de vista conceptual como tecnológico. Se evidencia la incorporación de estándares a los que adhieren las principales herramientas de desarrollo. De esta manera las nuevas tecnologías marcan una tendencia dentro de la construcción de aplicaciones que podemos decir que definen un paradigma.
Los Servicios Web son una nueva generación de aplicaciones Web. Son componentes de software autocontenidas, autodescriptivas y modulares que pueden ser accedidas, localizadas e invocadas desde cualquier lugar sobre la Internet. Constituyen un modelo de cómputo distribuido que unifica los criterios de comparación propuestos.
El presente trabajo establece una comparación desde el punto de vista del paradigma de la construcción de aplicaciones distribuidas con tecnología Microsoft Vs. tecnología JAVA.

Introducción

El desarrollo de software distribuido ha tenido una evolución importante en la última década tanto desde el punto de vista conceptual como tecnológico. Se evidencia la incorporación de estándares a los que adhieren las principales herramientas de desarrollo. De esta manera las nuevas tecnologías marcan una tendencia dentro de la construcción de aplicaciones que podemos decir que definen un paradigma.
El presente trabajo establece una comparación desde el punto de vista del paradigma de la construcción de aplicaciones distribuidas con tecnología Microsoft Vs. tecnología JAVA.
Dado que la comparación se establece desde el punto de vista de la computación distribuída, es importante tener conocimiento de cual es el modelo de cómputo distribuido que se está comparando. En este sentido, aparece el concepto de Servicio Web como dicho modelo.
Los Servicios Web son una nueva generación de aplicaciones Web. Son componentes de software autocontenidas, autodescriptivas y modulares que pueden ser accedidas, localizadas e invocadas desde cualquier lugar sobre la Internet.
Pero más allá de este nuevo modelo de cómputo existen características a comparar entre las dos tecnologías mencionadas que existen para un modelo de computación distribuida donde no exista el concepto de Servicio Web. Ejemplos de los mismos son: aplicaciones Cliente/Servidor y aplicaciones Web.
Además, podemos decir que el concepto de Servicio Web es una evolución natural de estos dos modelos computacionales mencionados. Ambos tienen en común el hecho de ser un caso particular de software distribuido. El Servicio Web lleva este concepto a la Internet agregando la idea de ser autocontenido y accesible desde cualquier punto.
Una aplicación Cliente/Servidor es software que se construye sobre un modelo computacional distribuido donde existe una comunicación asimétrica entre un proceso servidor que atiende requerimientos y un proceso cliente que los solicita. Ambas componentes son módulos funcionales con interfases bien definidas y que establecen su comunicación a través de mensajes sincrónicos sin hacer uso de un contexto global. La lógica de la aplicación (lógica del negocio) se encuentra distribuida entre cliente y servidor, mientras que el cliente se hace cargo de la interfase de usuario y el servidor de resolver el acceso a los datos.
Una aplicación Web es, al igual que una aplicación cliente/servidor, un tipo especial de aplicación distribuida que se accede via un navegador Web y cuya lógica de aplicación se encuentra en uno varios servidores accesibles vía un WebServer o un ApplicationServer. Las aplicaciones Web surgieron ante la necesidad de contar con aplicaciones Cliente/Servidor donde los clientes fueran genéricos y livianos y donde su única función sea la de presentar la interfase de usuario, llevando la lógica del negocio fuera de esta componente.
En este sentido, hablaremos de plataforma .NET y de plataforma J2EE, que dan soporte al modelo de computación distribuida basada en servicios pero pueden utilizarse de manera tradicional.
El aporte de este trabajo es comparar dichas plataformas de desarrollo a lo largo de toda la evolución de la computación distribuida, desde el cliente/servidor en dos capas tradicional hasta los nuevos conceptos de servicios Web.

Plataformas de desarrollo

- **Plataforma J2EE**

  La plataforma J2EE (Java 2 Enterprise Edition) fue diseñada para simplificar el desarrollo, distribución y gerenciamiento de problemas complejos en más de dos capas. Surgió como una evolución a la plataforma J2SE (Java 2 Standard Edition) que permite desarrollar aplicaciones para entornos gráficos, orientadas a objetos y basadas en ventanas.

  J2EE es un estandar y no una herramienta [1]. Su arquitectura está basada en Java lo cual permite escribir el código una vez y desplegarlo en cualquier plataforma.

  J2EE es una aplicación de Java. Las componentes J2EE son transformadas en “bytecode” e interpretadas por el JRE (Java Runtime Environment) en ejecución.

- **Plataforma .NET**

Microsoft .NET es una suite de productos que permite a las organizaciones construir aplicaciones empresariales basadas en servicios web o aplicaciones tradicionales en dos o más capas [1].

Es una reescritura de Windows DNA que fue la plataforma de Microsoft anterior para escribir aplicaciones empresariales. Windows DNA incluía MTS (Microsoft Transaction Server) y COM+, MSMQ (Microsoft Message Queue) y SQL Server como base de datos.

Microsoft .NET ofrece independencia del lenguaje e interoperabilidad, pero no portabilidad. Una componente .NET puede ser escrita en cualquier lenguaje (VB.NET, C #) ya que el codigo fuente es traducido a un lenguaje intermedio (IL – Intermediate Language) análogo al bytecode de Java, que es interpretado por CLR (Common Language Runtime) análogo al JRE. Esto es lo que se denomina .NET Framework.

Ninguna de estas características ofrecen portabilidad, o sea la capacidad de escribir en una plataforma y desplegar en cualquier otra. Microsoft argumenta que la interoperabilidad que ofrecen los servicios web “reemplazarian” el concepto de portabilidad de los desarrollos Java.

A su vez, la plataforma .NET incluye los siguientes servidores:

  - ◊ *SQL Server 2000*. Base de datos relacional
  - ◊ *Exchange Server 2000.* Servidor de mensajería y trabajo colaborativo.
  - ◊ *Commerce Server 2000*. Para desarrollo de aplicacines e-commerce.
  - ◊ *Aplication Center Server 2000*. Para administrar servidores clustered.
  - ◊ *Host Integration Server 2000*. Da acceso legal a otras plataformas (principalmente basadas en IBM).
  - ◊ *Internet Security and Acceleration (ISA) Server 2000*. Firewall y caching Web
  - ◊ *BizTalk Server 2000.* Solución basada en XML para integrar aplicaciones.

- **Plataformas J2EE y .NET por analogía**

Se establece el siguiente cuadro comparativo que permite ubicar con que se implementa cada una de las características con las que cuentan las plataformas para desarrollo de aplicaciones distribuidas.

Característica	J2EE	.NET
Tecnología	Estándar	Producto
Plataforma de corrida	+ de 30	Microsoft
Lenguaje	Java	C #
Interprete	JRE + CORBA + ORB	CLR
Paginas dinámicas	JSP (Java)	ASP+ (VB / C #)
Lenguaje Visual	Java Swing	Win Forms y Web Forms
Componentes	EJB	.NET Managed Components
Acceso a BD	JDBC	ADO.NET
SOAP,WSDL,UDDI	Si	Si

Web Services

Los Web Services son una nueva generación de aplicaciones Web. Son componentes de software autocontenidas, autodescriptivas y modulares que pueden ser accedidas, localizadas e invocadas desde cualquier lugar sobre la Internet [2]. Se construyen sobre estándares como UDDI, WSDL y SOAP.

Los Web Services son:

- ◊ Publicados y localizados vía UDDI (Universal Description, Discovery and Integration).
- ◊ Descriptos usando WSDL (Web Service Description Language).
- ◊ Invocados vía SOAP (Simple Object Access Protocol) sobre HTTP.

- **Historia**

La primer compañía en publicar esta idea fue HP. Ellos desarrollaron la especificación para e-speak, proponiendo lo que fue la próxima generación de intercambio de información en Internet. Microsoft prometió continuar esta propuesta con su producto .NET, e IBM lo hizo con su Web Service Toolkit (WSTK) y luego el Web Service Development Environment (WSDE).

Oracle anuncio en su momento el lanzamiento de Dynamic Services integrado en Oracle 8i Release2.

Por su parte Sun lanzo, cerca de 2001, su iniciativa en Web Services dentro de ll ambiente J2EE.

- **Estructra de los Web Services**

Los Web Services son aplicaciones que utilizan una compleja estructura de protocolos para poder funcionar. Estos se dividen en: Wire Stack (vinculada al mecanismo de comunicaciones), Description

Stack (relacionada con la manera de describir el servicio) y Discovery Stack (relacionada con la manera en que el servicio es descubierto por otros servicios) [2]

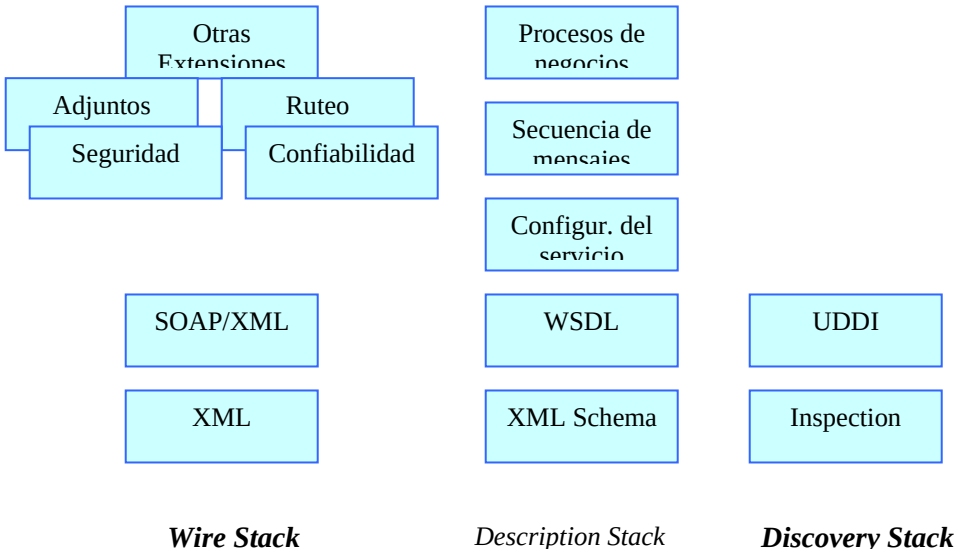

Figura 1

- **J2EE y Web Services**

J2EE ha sido históricamente una arquitectura para construir software del lado del servidor en Java. Puede usarse para sitios web tradicionales, componentes de software o aplicaciones empaquetadas. En 2001 extendió su especificación para construir servicios web basados en XML [1]

La figura 2 muestra el modelo de desarrollo de web services en tecnología J2EE.

◊ Las *aplicaciones J2EE* residen en un contenedor que provee la calidad de servicio necesaria para aplicaciones empresariales como transacciones, seguridad y persistencia.

◊ La *capa de negocio* es implementada por EJB en aplicaciones de mediana a gran envergadura. Este nivel resuelve la lógica de negocios y se conecta vía JDBC a las bases de datos. Los sistemas preexistentes se conectan vía JCA (Java Connector Architecture)

◊ Los "*business partnerts"* conectan con aplicaciones J2EE a través de la tecnología de Web Services (SOAP,UDDI,WSDL y XML). Un servlet (objeto Java orientado al request/reply) puede aceptar requerimientos desde los business partners.

◊ Los *clientes tradicionales* tales como applets y aplicaciones basadas en ventanas, conectan con la capa de EJB a través de IIOP (protocolo Inter-ORB) ya que generalmente estos clientes están escritos en la misma tecnología J2EE con lo cual no necesitan la infraestructura de web services

◊ Los *web browsers y dispositivos sin cable*, conectan directamente JSP via http

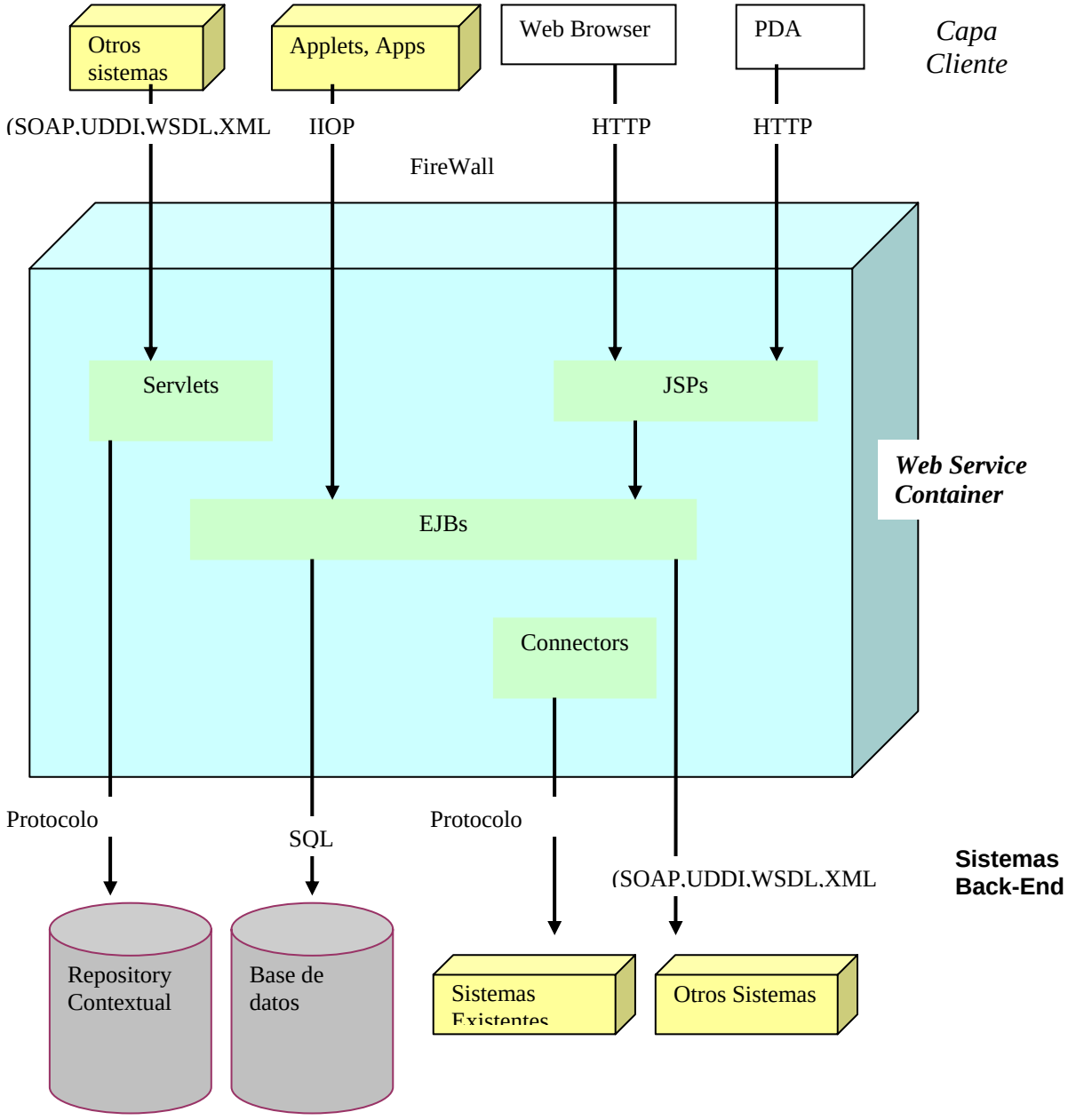
Otros sistemas
Applets, Apps
Web Browser
PDA
Capa Cliente
(SOAP,UDDI,WSDL,XML
IIOP
HTTP
HTTP
FireWall
Servlets
JSPs
Web Service Container
EJBs
Connectors
Protocolo
SQL
Protocolo
(SOAP,UDDI,WSDL,XML
Sistemas Back-End
Repository Contextual
Base de datos
Sistemas Existentes
Otros Sistemas

Figura 2

- **.Net y Web Services**

Microsoft .NET es la evolución de Windows DNA para la inclusión de un nivel de Web Services. Los mismos se encuentran dentro del .NET Framework que también cuenta con un lenguaje de programación mejorado respecto de la tecnología de origen [1]

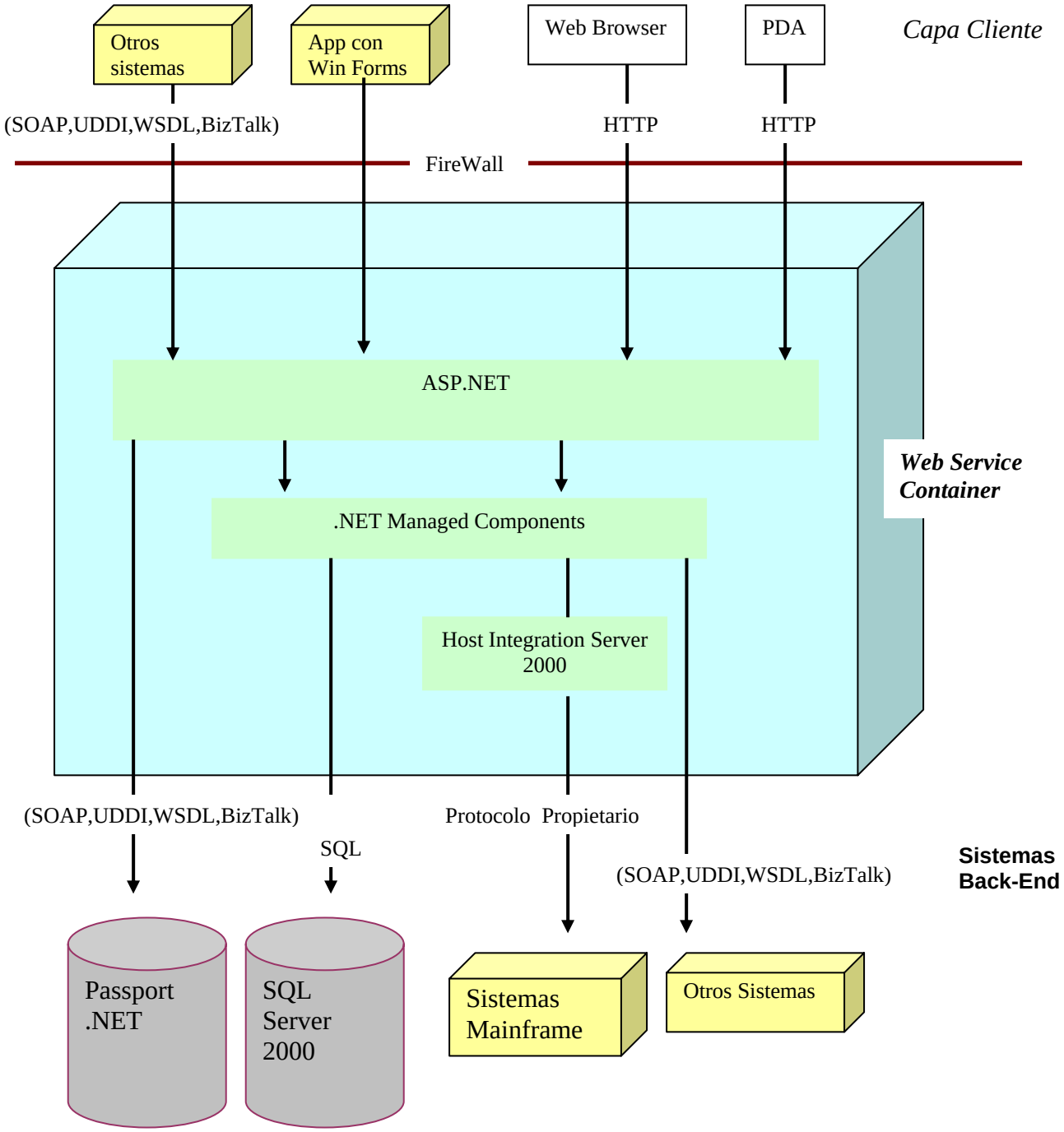

Figura 3

◊ Las *aplicaciones .NET* residen en un contenedor que provee la calidad de servicio necesaria para aplicaciones empresariales como transacciones, seguridad y persistencia.

◊ La *capa de negocio* de las aplicación .NET se construye con ".NET managed components". Este nivel resuelve la logica de negocios y se conecta via ADO.NET a las bases de datos. Los sistemas preexistentes se conectan via Microsoft Host Integration Server 2000

◊ Los "*business partnerts"* conectan con aplicaciones .NET a través de la tecnología de Web Services (SOAP,UDDI,WSDL y BizTalk).

◊ Los *clientes tradicionales, web browsers y dispositivos sin cable* conectan directamente a ASP.NET via HTTP.

Patrones de comparación

Se establece en este punto una serie de patrones de comparación de ambas plataformas, algunos de los cuales cobran sentido únicamente considerando la tecnología Web Service, mientras que otros son inherentes exclusivamente a cada plataforma desde el punto de vista conceptual de las misma, más allá de la inclusión o no del concepto de Web Service.

- **Similitudes**

La principal similitud tiene que ver con la normalización / estandarización del intercambio de información a través de la adopción del XML para tal fin.

- **Diferencias**

La principal diferencia entre ambas plataformas radica en la tecnología utilizada para realizar la lógica de la aplicación y la manipulación de datos. Java como producto posee un marco muy sólido que puede ser adecuadamente extendido para alcanzar los requerimientos de las aplicaciones sin alterar la estructura del framework. Esto facilita la adopción de esta tecnología por parte de otras compañías. Microsoft, por su parte, si bien aporta .NET framework con CLR para facilitar la corrida en plataformas no Windows, no ofrece un especificación estándar que puedan adoptar otras compañías.

- **Impacto en el mercado futuro**

El impacto a futuro de estas tecnologías será más notorio en función del tipo de negocio del que se trate. Contar con la capacidad de centralizar información globalmente impacta sobre el mercado de agencias de viajes, servicios de comidas, teatros. Por otra parte, también constituye un valor agregado para la intranet de una compañía, donde la factorización y dispersión de la información ha sido siempre un problema.

- **Impacto en el mercado actual**

El principal impacto de estas dos tecnologías de middleware en el mercado actual es justamente la polarización que produjeron y la necesidad de elegir una u otra. Esta polarización ha sido mayor que la que produjeron los sistemas operativos UNÍX y Windows con anterioridad.

- **Lenguaje**

Es ampliamente conocido que el 80% del costo de un proyecto de software se encuentra en el mantenimiento de los sistemas. Por lo tanto, contar con un lenguaje simple asegura mantenibilidad.

.NET cuenta con varios lenguajes. El sentido común indica que el mercado cuenta con mas programadores VB que C#. Por lo tanto sería razonable utilizar VB.NET que es orientado a objetos y posee la característica de CLR. Por otra parte, y justamente por lo enunciado ante, existe un gran “gap” entre VB6 y VB.NET, lo cual hace que los programadores requieran un entrenamiento adicional.

De este modo, se debe evaluar el costo del nuevo entrenamiento y allí surge la disyuntiva si se aplica ese entrenamiento sobre VB.NET o Java.

En USA, el 78% de las universidades enseñan Java. Por lo tanto, dado que un nuevo entrenamiento es un hecho indiscutible, ya existe una cierta facilidad en los profesionales para aprender Java. Por otra parte, un programador VB no cuenta con experiencia en complejidades tales como programación concurrente y arquitecturas con tolerancia a fallas.

- **Migración de desarrollos preexistentes**

Los desarrollos que actualmente se ejecutan en plataforma Windows deberán migrar a .NET. Esto implica varios cambios tanto a nivel de tipos de datos como también de objetos COM. Las aplicaciones VB deberán rescribirse.

Por otra parte, los desarrollos Java que incorporen conceptos de Web Services no sufrirán un gran impacto, ya que las nuevas APIs se ubicaron en componentes que facilitan el cambio de las aplicaciones tradicionales a Web Services.

- **Soporte para sistemas existentes**

Las organizaciones invierten mucho tiempo y dinero en el desarrollo de sistemas y no siempre se requiere reemplazar los mismos ante los cambios tecnológicos. La integración es uno de los desafíos màs importantes a la hora de construir un Web Service.

Con tecnología J2EE hay varias formas de realizar la integración: JMS (Java Message Service) para integrar con sistemas existentes, web service para integrar con cualquier sistema, CORBA para interactuar con código escrito en otros lenguajes que existen en maquinas remotas y JNI para cargar librerías nativas y luego invocarlas localmente.

.NET ofrece integración a través del Host Integration Server 2000. COM para transacciones colaborativas a través de mainframes. MSMQ para integrar con sistemas IBM MQSeries.

Las características de integración ofrecidas por J2EE son superiores a las ofrecidas por .NET. Este último se circunscribe a lo ofrecido por su producto Host Integration Server 2000.

- **Madurez**

Ambas plataformas han madurado al punto de contar con aplicaciones empresariales ya desarrolladas. Los Web Services son nuevos en ambas plataformas y esto hace que las similitudes se separen: en .NET hubo un cambio tecnológico sustancial respecto de sus productos antecesores, incorporando la orientación a objetos y el entorno de ejecución CLR. Para la tecnología J2EE simplemente significó una ampliación de la especificación anterior y una mejora en el manejo de persistencia de los objetos.

- **Portabilidad**

Como ya se explicó anteriormente la portabilidad es un concepto diferente de la interoperibilidad. Si bien .NET ha incorporado ventajas para mejorar la manera de interoperar con otras plataformas, sigue dejando cautivos a desarrolladores y usuarios de la plataforma subyacente.

- **Aplicaciones desconectadas**

En los próximos años se espera un gran crecimiento de la tecnología alrededor de los dispositivos móviles, lo cual implica la necesidad de contar con aplicaciones “stand-alone” que sincronicen la información cuando establecen una conexión.

.NET cuenta con tecnología para tal fin como WinForms y MobiliForms pero, nuevamente, se limita a plataforma Windows.

Java ha avanzado más en ese sentido contando con una serie de capacidades y herramientas que realizan en forma automática la actualización cuando se establece la conexión.

- **Herramientas**

La plataforma J2EE cuenta con una variedad de IDE basados en Java anteriores tanto a J2EE como a .NET. Muchos de ellos comerciales y otros tantos son productos de terceras partes o de software libre. Podemos mencionar en general Visual Age for Java de IBM, Jbuilder de Borland, entre los comerciales.

Por su parte Microsoft ya contaba con una ambiente de desarrollo maduro que mejoro ante el lanzamiento de .NET y que es VisualStudio .NET que soporta todos los lenguajes, excepto Java.

El problema que presentan los IDE para Java es que no son totalmente interoperables, justamente porque no provienen de un único vendedor.

- **Contexto Compartido**

El concepto de contexto compartido tiene que ver con la naturaleza “sin estado” del protocolo de comunicación de las aplicaciones Web y que generalmente esta basado en HTTP.

Se requiere un contexto compartido que permita que el usuario ingrese la información una vez y pueda ser luego utilizada. La información debe estar a disposición del usuario y estará protegida por las reglas de seguridad que este defina.

Sun J2EE es la de contextos compartidos descentralizados y distribuidos que viven en la Internet.

Microsoft .NET lleva a cabo el contexto compartido a través del servicio de Passport.NET. Es un repositorio alojado por Microsoft que contiene información de la identidad del usuario.

El aspecto descentralizado y distribuido de J2EE permite contar con repositorios especializados para diferentes necesidades. No existe el concepto de “hermano mayor” en el sentido que los individuos no necesitan confiar sus datos a cualquiera. No hay un único punto de falla.

El aspecto centralizado de Passport .NET no deja duda acerca de cual es el contexto compartido “oficial” y reduce el riesgo de la proliferación de contextos que implica un esquema descentralizado.

- **Performance**

Una plataforma tiene buen rendimiento cuando el tiempo de respuesta ante un requerimiento es “aceptable”, donde la definición de este término cambia depende de la naturaleza del problema.

Más allá de esto, sin lugar a dudas el principal cuello de botella en estas arquitecturas se encuentra los sistemas de bases de datos back end.

J2EE reduce este tráfico y la cantidad de conexiones a la base de datos permitiendo procesos sin estado o “desconectados” y “caching” de memoria por largos periodos de tiempo.

Estas particularidades deben ser utilizadas por programadores experimentados que sepan manejar adecuamente el “tradeoff”.

Microsoft.NET no ofrece estas características liberando al programador de la toma de decisiones pero impidiendo también manejos de mas bajo nivel que podrían mejorar la perfomance.

- **Escalabilidad**

Una plataforma es escalable si un incremento en los recursos de hardware resulta en una correspondencia lineal al incremento de carga de usuario que soporta, manteniendo el mismo tiempo de respuesta.

En este sentido ambas plataformas son escalables. La única diferencia es que .NET soporta Win32 solamente.

- **Costo**

Como ya se menciono, si una organización no se encuentra ya desarrollando bajo tecnología Java, la adopción de la misma requiere un nuevo entrenamiento de los programadores. La ventaja en cuanto a costos tiene que ver con que existen IDE libres y para todos los sistemas operativos.

Por otra parte, si ya se está dentro de la tecnología Microsoft, el re-entrenamiento sigue siendo necesario por los cambios conceptuales que se introducen en .NET Framework.

Referencias

1- Chad Vawter-Ed Roman. *“J2EE Vs Microsoft.NET. A comparison of building XML-based web services”*. Sun Microsystem. Junio 2001.

2- Ben van Eyle. *“Web Services – A Business Perspective on Platform Choice”.* Agosto 2001. www.theserverside.com.

3- Jim Farley. *“Microsoft .NET vs J2EE: How do The Stack Up?”.* Enero 2000. www.oreilly.com.

4- Carol Silwa “.Net vs Java: Five Factors to Consider”. Mayo 2002. ComputerWorld