PROPUESTA DE COMPATIBILIDAD TECNOLÓGICA ENTRE MICROSOFT OLE/DCOM Y PLATAFORMAS DE DESARROLLO ABIERTAS PARA LA CREACIÓN DE APLICACIONES ORIENTADAS A LOS SERVICIOS".

COMPATIBILITY TECHNOLOGICAL PROPOSAL BETWEEN OLE/ DCOM AND OPEN PLATFORM DEVELOPMENT APPLICATION FOR THE CREATION OF A CUSTOMER ORIENTED SERVICES.

Luis Alejandro Santana Valadez [1]
Wendy Daniel Martínez [2]

[1] Maestro en Tecnologías de la Información. Profesional Certificado por Microsoft desde el año 2006. Ingeniero en Sistemas Computacionales. Docente investigador de la escuela de Ingeniería de la Universidad La Salle Pachuca, México, Presidente de la academia de programación, redes y base de datos y coordinador de las certificaciones MCP y OCA de Microsoft y Oracle respectivamente.
asantana@lasallep.mx

[2] Profesional Certificado por Microsoft desde el año 2006. Ingeniero en Sistemas Computacionales. Docente investigador de la escuela de Ingeniería de la Universidad La Salle Pachuca, México e instructora de la certificación MCP de Microsoft.
wdaniel@lasallep.mx

Resumen
Actualmente los sistemas de software que brindan servicios informáticos vía web o móvil han estado en constante actualización en su infraestructura
tecnológica, llegando a tener una alta complejidad en su diseño debido a que sus plataformas empresariales ahora tienen múltiples componentes que las conforman, generando diversas problemáticas de compatibilidad durante su desarrollo.

Es necesario medir la compatibilidad de la tecnología de Microsoft (por tener gran influencia en el desarrollo de software) con herramientas abiertas como Java, para demostrar sus ventajas y limitaciones. Se aplicó la metodología RAD para generar dos prototipos middleware que permitan a los desarrolladores replicar esta propuesta tecnológica para aprovechar herramientas abiertas y cerradas en la construcción de software. Este trabajo se terminó en marzo del 2010.

Palabras Clave: Compatibilidad, componente, middleware, prototipo, RAD.

Abstract
At the moment the software systems that offer computer services via web or mobile have been in constant up to date in their technological infrastructure, but having a high complexity in their design because their enterprise platforms now have multiple components that conform them, generating diverse problems of compatibility during their development.

It is necessary to measure the compatibility of the Microsoft's technology (to have great influence in the software development) with open tools as Java, to demonstrate their advantages and limitations. The RAD methodology was applied to generate two middleware prototypes that allow to the developers to reply this technological proposal to take advantage of open tools and closed in the software construction. This work ended in March, 2010.

Key words: *Compatibility, component, middleware, prototype, RAD*

Introducción

Dentro de una institución educativa, empresa pública, privada o inclusive dentro de una casa, se maneja información digital que es manipulada por diversos dispositivos: computadoras, celulares, laptop, entre otros. El internet y el ambiente *web* permiten comunicarnos con diferentes servicios que hoy en día se convierten en una necesidad para cualquier tipo de actividad humana, haciendo del desarrollo de aplicaciones de *software* una actividad de gran impacto en la sociedad en diversos ámbitos: económico, social y tecnológico.

Es importante mencionar que actualmente hay una gran competencia entre los proveedores de herramientas de desarrollo de *software* cerrado (tienen costo de adquisición) y las abiertas (son gratuitas o con un costo simbólico) por ser el estándar comercial en la construcción de aplicaciones, pero más que acercarse a un fallo a favor o en contra por parte de los desarrolladores profesionales, estudiantes o empresas en general, se han encontrado con un problema inevitable: La expansión de las tecnologías de la información.

Por otro lado, la situación actual sobre el grado de usabilidad de herramientas de desarrollo abiertas como Java o cerradas como *Microsoft Visual Studio.Net* ha generado una división importante tanto en los desarrolladores que generan las soluciones tecnológicas como en las empresas que establecen la demanda de sistemas de *software* y servicios en el mercado (Woo, 2005). Las razones principales son muy específicas:

- La infraestructura requerida (*software, hardware*, formas de comunicación, estándares) es muy diferente en ambos tipos de plataformas (abierta o cerrada), esto define la relación de costos de construcción e implantación (Seongki y Sangyong ,2006).
- La curva de aprendizaje sobre herramientas abiertas o cerradas es muy distinta, ya que la existencia de manuales, simuladores, ayuda y soporte técnico (con costo o gratuito) varía de una plataforma a otra drásticamente.
- Cada plataforma tiene su propio modelo de generación de componentes de *software* (librerías dinámicas) reusables, lo cual genera una barrera de compatibilidad. En la figura 1 se muestra el modelo que permite la reusabilidad entre componentes de diferentes plataformas, donde se aplica el concepto de puerto, que es un bloque de *software* intermedio que brinda cada componente permitiendo la comunicación de datos y procesos con otros componentes.

Figura 1. Reusabilidad de componentes, **B** brinda el puerto P y **A** lo usa (Malony, Shende, Trebon, Ray, Armstrong, Rasmussen y Sottile 2005, p.4)

- Actualmente el mecanismo para generación de estándares y medios de compatibilidad entre las diferentes plataformas de desarrollo que existen es muy lento y complicado, además de la poca difusión que se le da entre la comunidad de desarrolladores, fábricas de software y proveedores de tecnología (Woo, 2005).

Metodología

El objetivo principal de este trabajo es proponer una estrategia tecnológica que permita integrar la plataforma de *Microsoft Visual Studio .Net* sobre el desarrollo de aplicaciones *COM/DCOM* con el entorno de desarrollo abierto de Java, utilizando dos escenarios específicos muy utilizados en la construcción de aplicaciones orientadas a los servicios, para generar nuevas estructuras del conocimiento aplicado en las tecnologías de la información que beneficien a la comunidad de desarrolladores de cualquier nivel profesional y experiencia en el desarrollo de soluciones de *software*.

Con los elementos anteriores, surgen diversos cuestionamientos que acotan esta investigación: ¿Qué beneficios y desventajas tiene un desarrollador al aplicar los componentes *COM/DCOM* de *Microsoft* en la construcción de sistemas? ¿Se puede integrar *COM/DCOM* al desarrollo de *software* con plataformas abiertas? Al desarrollar una aplicación de *software* ¿Cuáles son los requerimientos de *hardware*, *software* y red necesarios para crear un entorno tecnológico que utilice *Microsoft COM/DCOM* de manera abierta?

¿Cuál es el futuro de *COM/DCOM* respecto a la aplicabilidad y compatibilidad del modelo? Para responder estas interrogantes se debe seleccionar y aplicar una metodología que permita analizar el diseño y construcción de componentes de *software* y así de esta manera probar sus ventajas y limitaciones, a diferencia de las metodologías que tienen el objetivo de generar aplicaciones completas o nuevas tecnologías de información.

"El desarrollo basado en componentes es una aplicación de la técnica de divide y vencerás para manejar la complejidad, donde el principal objetivo que se persigue con la introducción de este paradigma es el reuso" (González, 2005).

Se utilizó el Diseño Rápido de Aplicaciones (RAD) como metodología base debido a que se requiere diseñar y desarrollar prototipos (*middleware*) bajo escenarios bien definidos, esto implica que no se construirá un *software* formal para ser liberado a usuarios finales, ya que esta metodología está enfocada solo para desarrolladores de aplicaciones (Pressman, 1997). En la figura 2 se muestran las etapas que integran la metodología RAD, ahí se observa cómo se aplica el reuso en las actividades: demostrar, refinar y construir para generar el prototipo con las características deseadas para su posterior implementación.

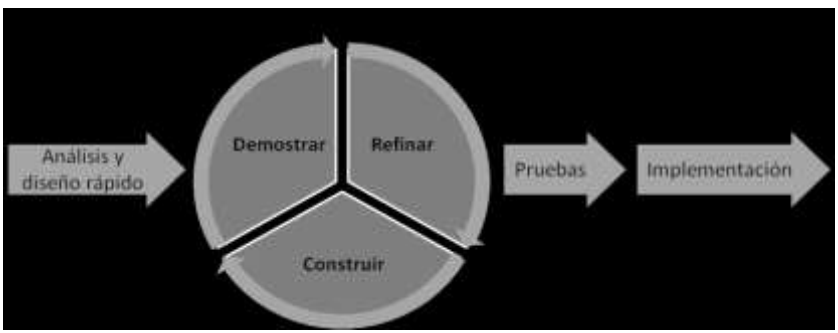

Figura 2. Diagrama funcional de RAD. Fuente propia.

Análisis de prototipos

Un prototipo bien construido necesita tener una definición clara del ambiente donde será utilizado para desarrollar una serie de pruebas técnicas con un alto grado de eficiencia y veracidad. En el cuadro 1 se comparan las características técnicas que poseen cada ambiente considerado para el desarrollo de componentes: el cerrado que genera componentes COM /DCOM y el abierto que genera componentes EJB.

	COM / DCOM, .NET	**EJB**
Desarrolladores	*Microsoft*	SUN *MicroSystems*, IBM, BEA
Componentes	Módulos con múltiples clases u otras implementaciones	Módulos que pueden contener múltiples clases
Interfaz	OLE IDL (interfaces como una colección de funciones)	Lenguaje Java
Conexión	Vía punteros a interfaces	Vía eventos y escuchadores
Mecanismos de variabilidad	Genericidad, agregación	Herencia y agregación
Plataforma	Windows (.Net es multiplataforma)	Multiplataforma con máquina virtual de java (JVM)
Lenguaje de implementación	C++, C#, J#, VB.Net, Delphi.Net	Java
Mecanismos de distribución	DCOM, internet	EJB, internet, RMI (*Remote Method Invocation*)

Cuadro 1. Comparativa COM / DCOM, .NET y EJB (Vicente, 2005; 58)

En base a los factores de comparación de (Vicente, 2005; 58) y (Woo, 2005;
10), las pruebas de prototipos que serán tomadas en cuenta son:

- Pruebas de requerimientos de *hardware* y *software.*
- Pruebas de enlace entre componentes durante la construcción del
- *middleware.*
- Pruebas de ejecución y tiempo de respuesta.

- Pruebas de compatibilidad entre lenguajes de programación abierto
- (Java) y cerrado (C# de *Microsoft*).
- Pruebas de implementación del *middleware* en cada escenario definido.

Considerando las pruebas de compatibilidad que se necesitan realizar a los componentes cerrados de *Microsoft Visual Studio .Net* y los componentes abiertos de Java aplicando RAD, se contemplaron dos prototipos como escenarios de evaluación, con el objeto de aplicar los dos ambientes más utilizados para construir sistemas orientados a los servicios: *Web* y móvil. Los escenarios son:

1) ***Web*** (desarrollado con *Microsoft Visual Studio .Net*) utilizando un *middleware* con componentes de Java EJB.

2) **Móvil** (desarrollado en Java) utilizando un *middleware* con componentes de *Microsoft* (COM/DCOM).

Para los dos escenarios se creará el mismo servicio: **Administración y control de ventas de productos,** el primero estará enfocado a ventas por internet y el segundo mediante dispositivos móviles, de esta forma se podrá evaluar y comparar qué tan compatibles son los componentes abiertos y cerrados al utilizarlos en ambientes totalmente diferentes (EJB con Microsoft y COM/DCOM con Java).

En la etapa de diseño se mostrarán a detalle los componentes aplicados en cada escenario, para generar dos prototipos que incluyan el mayor número de factores de prueba para lograr que haya una evaluación precisa.

Diseño de prototipos

En el cuadro 2 se muestran a detalle los requerimientos técnicos de *software* de sistemas y de aplicaciones para los dos escenarios, de tal forma que puedan construirse adecuadamente los prototipos y el *middleware* correspondiente:

Requerimiento	**Escenario Móvil (Java)**	**Escenario *Web* (*Visual Studio .Net*)**
Sistema Operativo	Ubuntu 9.10	*Windows* XP SP 2
Servidor *Web*	No aplica al escenario	IIS
Lenguaje de programación	Java	C#
Marco de trabajo del lenguaje	*NetBeans* IDE	*Visual Studio .Net* 2005
Versión del lenguaje	6.7.1	*Framework* 2.0
Tecnología de componentes utilizado	COM/DCOM	EJB
Nombre del componente	clsOperaciones.dll	EJBDCom.jar
API's de compatibilidad	JNI	JBImp.dll, ikvm.dll
Manejador base de datos	MySql	SQL *Server* 2005
Versión del manejador	5.1.37	9.00.1399.00
Drivers de conexión a base de datos	JDBC	ADO.Net
Formato de páginas *web*	No aplica al escenario	Aspx
Formato de formularios móviles	JavaBeans (.jar)	No aplica al escenario
SOA Construido	Ventas móviles	Ventas por internet
Medio de ejecución	*Pocket* pc	Navegador de internet

Cuadro 2. Requerimientos técnicos para la construcción de los prototipos (Elaboración propia). Comparativa COM / DCOM, .NET y EJB (Vicente, 2005; 58)

En las figuras 3 y 4 se muestran los dos escenarios con cada una de las partes que los componen: *Middleware*, base de datos, aplicaciones *web*/móvil y los dispositivos que permitieron ejecutar los prototipos para su evaluación:

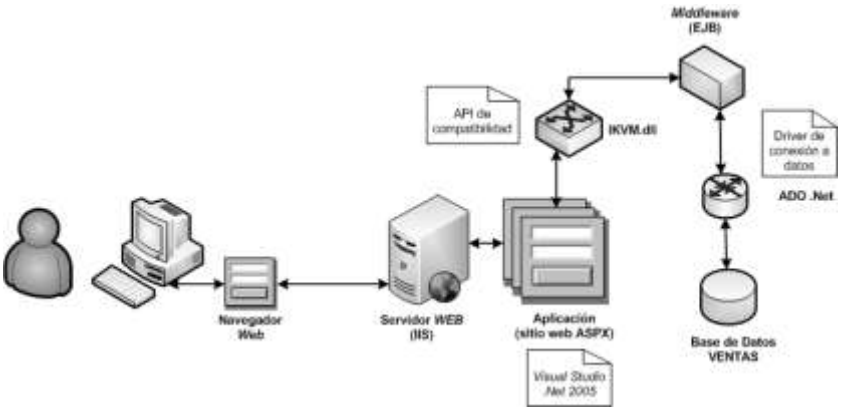

Figura 3. Diseño del escenario web: Visual Studio .Net / EJB (Elaboración propia)

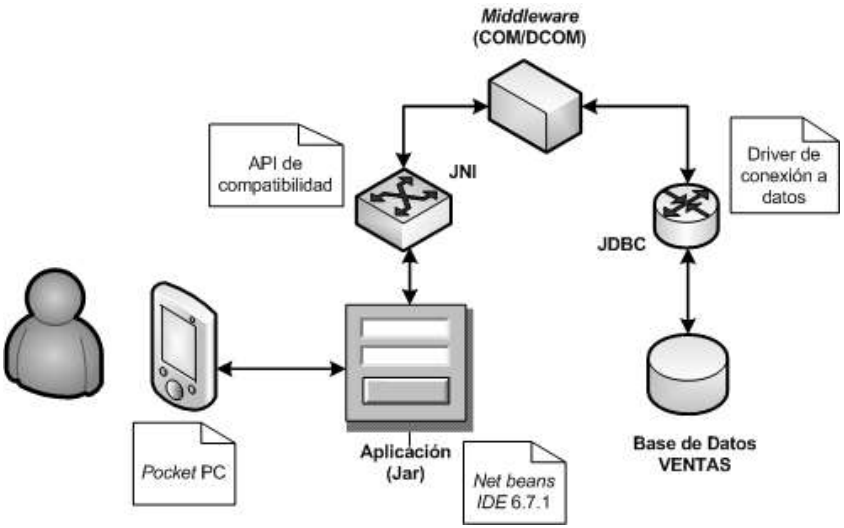

Figura 4. Diseño del escenario móvil: Java/COM/DCOM (Elaboración propia)

En la figura 5 se muestra el diseño de la base de datos utilizada en los dos prototipos, el cual será aplicado en el manejador de base de datos seleccionado para cada escenario, básicamente es el control de ventas de productos de los que se tiene un catálogo descriptivo, de la misma forma se tiene un catálogo de clientes para así poder registrar los datos más representativos de una venta aplicable en cualquier tipo de empresa que tenga
como giro de negocio el control de ventas.

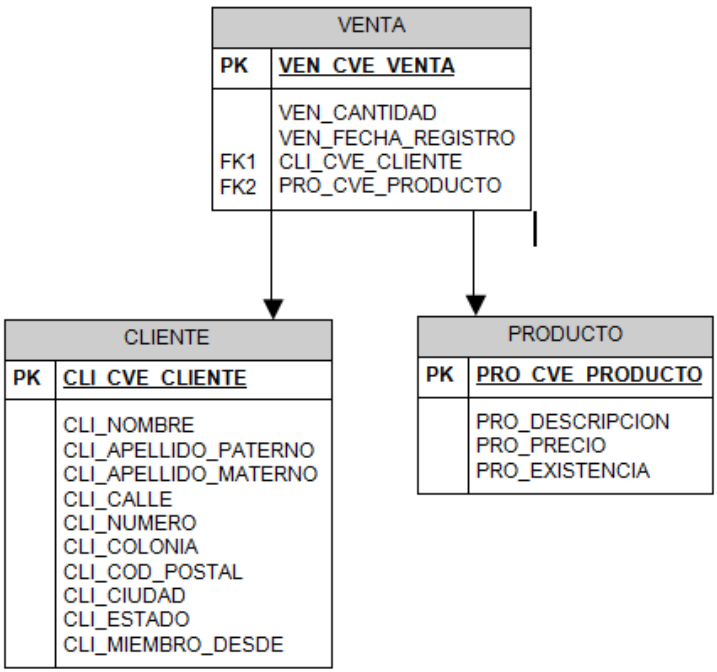

Figura 5. Diseño de la base de datos aplicado en los dos escenarios (Elaboración propia)

Construcción del Middleware

Debido a que la parte más importante de la construcción de prototipos es la creación de los dos *middleware* (llamados *broker* por tener la función de ser intermediarios entre las plataformas de desarrollo de *software*), a continuación se describen los entornos de desarrollo utilizados para la creación de los mismos conforme a los requerimientos técnicos descritos. En la figura 6 se observa el IDE de *Visual Studio .Net* utilizado para la programación del *middleware* de *Microsoft* que produce un objeto COM/DCOM, donde además se muestra la referencia a la API requerida para crear la interfaz entre los componentes de java que serán utilizados y el entorno de desarrollo de *Microsoft*.

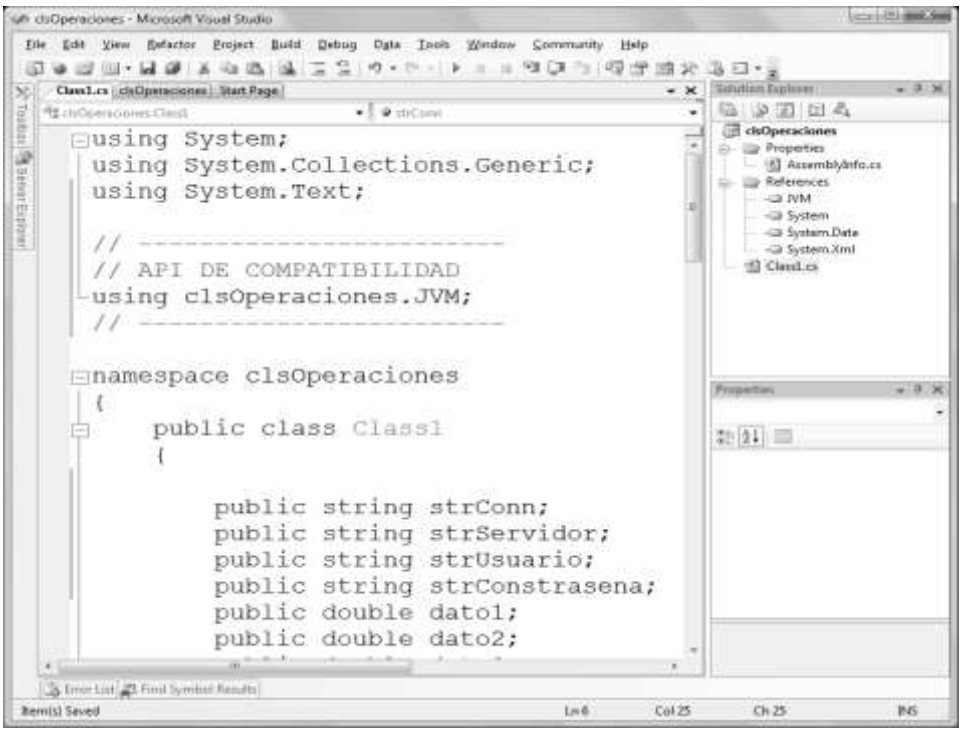

Figura 6. IDE de Visual Studio .Net donde se construyó el componente clsOperaciones.dll

La figura 7 contiene el IDE de *NetBeans* para la creación del *middleware* de
Java que produce un objeto empresarial EJB, donde puede observarse también el entorno abierto proporcionado por el sistema operativo Ubuntu
9.10 de Linux.

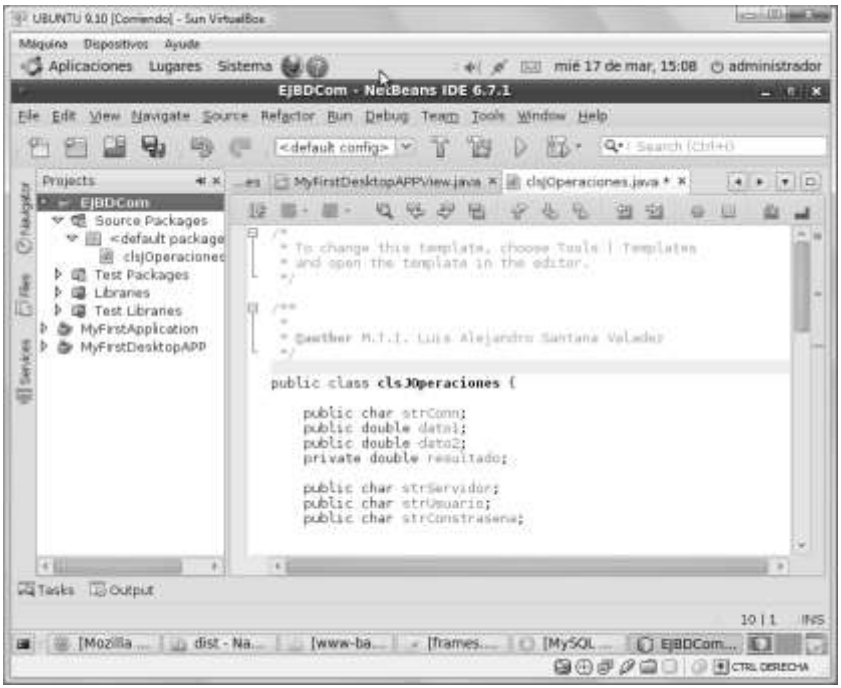

Figura 7. IDE de NetBeans donde se construyó el componente EJBDCom.jar

Construcción de los prototipos

Los dos prototipos, aunque tienen una interfaz muy diferente (web y móvil), deben ser construidos de tal manera que puedan demostrar que los componentes COM/DCOM y EJB puedan ser referenciados y consumidos por plataformas contrarias en su ingeniería interna. En la figura 8 se muestran las dos interfaces de usuario construidas para poder utilizar los componentes descritos.

(a) (b) Figura 8. (a) Interfaz del prototipo web, (b) Interfaz del prototipo móvil

Pruebas y resultados

Al realizar las diferentes pruebas sobre la ejecución de los dos prototipos en conjunto con el *middleware* correspondiente (las cuales iniciaron desde el mismo diseño, construcción y ejecución), se obtuvieron comparaciones sobresalientes que permitieron enmarcar las ventajas, limitaciones, compatibilidad y eficiencia de las plataformas de desarrollo utilizadas. En el cuadro 3 se presentan las características de *hardware* aplicadas en los dos prototipos al momento de utilizar el *middleware*, es evidente que los requerimientos de memoria RAM y disco duro son muy diferentes en ambos prototipos.

Hardware	Mínima (Java)	Recomendada (Java)	Mínima (.Net)	Recomendada (.Net)
Procesador	Pentium III 500 MHz	Cualquiera Superior	Pentium III 733 MHz	Cualquiera Superior
Memoria RAM	256 MB	512 MB	512 MB	1024 MB
Disco Duro	350 MB	350 MB	3GB	4.5GB

Cuadro 3. Comparación de requerimientos de hardware de JAVA y .Net (Elaboración propia)

En el cuadro 4 se muestran los resultados obtenidos de la evaluación de ambos prototipos al utilizar el *middleware* en las etapas de diseño, construcción y pruebas para poder hacer una comparación sobre las actividades realizadas.

Factor	Prototipo Java	Prototipo .NET
Formato *web* dinámico	JSP	ASP.Net
Acceso a base de datos usado	JDBC	ADO.Net
Independencia de plataforma	Si	Aun no (solo en plataformas *Microsoft*)
Lenguajes compatibles	Solo Java	Se usó C#, pero hay opciones como: VB.Net, J#, Delphi.Net
Modelo de componentes	EJB	COM/ DCOM
Compatibilidad del componente con los datos	Complicación con el manejo de versiones	Compatibilidad configurable por versiones del EJB y ADO .Net
Líneas de código generadas	Código con gran cantidad de líneas necesarias para manejo de componentes	Código resumido, basado en XML
Costo de la construcción de prototipos	Todas las herramientas usadas son *freeware*	No tiene *freeware,* todas las herramientas tienen costo
***Brokers* de compatibilidad**	JNI	JBImp.dll, ikvm.dll
Eficiencia de *brokers*	Requiere soporte y configuración	Fácil uso e implementación
Enlace entre componentes	Fallas en conectividad y configuración	Fallas en versiones
Ejecución del prototipo	Respuesta alta	Respuesta alta
Ejecución de *middleware*	Respuesta media-alta	Respuesta alta

Cuadro 4. Resumen comparativo de resultados entre Java y .NET (Elaboración propia)

Conclusiones

Al investigar con diversos autores (González, 2005; Seongki y Sangyong, 2006; Vicente 2005; Woo, 2005) sobre la reusabilidad de componentes de *software* en plataformas abiertas o cerradas, quedó demostrado que dan por hecho que la compatibilidad de COM/DCOM con plataformas abiertas como Java no puede darse de forma tan simple actualmente, es por esto que sus propuestas van enfocadas a explotar al máximo cada plataforma bajo escenarios bien definidos: *Web*, móvil o cliente-servidor.

De manera constante los proveedores de tecnología tales como *Canonical* Ltd, *Microsoft*, *Sun Microsystems* entre otros, hacen comparaciones sobre las ventajas de usar una plataforma abierta o una cerrada, pero al hacer un consolidado de ventajas y limitaciones de ambas mediante la observación de los resultados de las pruebas realizadas, se concluye que hay un equilibrio entre plataformas sobre sus requerimientos de *hardware* y *software*, tiempo de respuesta y compatibilidad, lo cual no se ha aprovechado aun ya que si COM/DCOM y Java son tecnologías de punta con actualización constante, actualmente hay poca investigación enfocada a tratar de integrarlas y aprovechar sus capacidades para eliminar las desventajas que tienen de forma aislada.

Se observó que la metodología RAD utilizada para construir los prototipos *middleware* fue una elección eficiente siempre que se requiera desarrollar aplicaciones de *software* basadas en componentes. Algunos autores (González, 2005; Malony, Shende, Trebon, Ray, Armstrong, Rasmussen y Sottile, 2005) incluso la aplicaron para generar un *middleware* que posteriormente fuera utilizado en otros proyectos bajo diferentes metodologías de desarrollo más complejas e incluyentes.

Cualquier desarrollo tecnológico debe estar normalizado por conceptos teóricos válidos y actualizados por autores o proveedores de tecnología de la información reconocidos en el campo y por estándares técnicos aplicados por arquitecturas bien definidas. Actualmente en el campo de desarrollo de componentes de *software* existe una normalización y estandarización a medio camino, ya que los proveedores de tecnología (*Microsoft, Sun Microsystems*, entre otros) estandarizan procesos, *freeware* y código fuente en ciertas áreas (manejo de errores, instanciamiento de objetos, configuración de componentes), pero no dejan claro cómo puede implementarse un canal libre de comunicación entre plataformas distintas y ahí es donde surge la necesidad de información, ejemplos y pruebas de compatibilidad.

El soporte técnico efectivo sobre las herramientas de desarrollo de *software* puede llegar a ser tan vasto como el de *Microsoft Visual Studio .Net* o tan limitado como el de Java. Esto también es determinante para la aceptación de las herramientas y su actualización permanente, ya que por un lado *Microsoft* provee soporte por distintas líneas: Bibliografía, sitios de ayuda específica, soporte vía telefónica o por mail, descarga de manuales, ayuda y ejemplos de forma gratuita; mientras que de Java

también hay bibliografía y ayuda por internet, pero resultan poco eficientes debido a la gran cantidad de versiones y plataformas que utiliza, lo cual nos hace depender de foros de ayuda que son útiles, pero consume mucho tiempo debido a la dependencia a las respuestas de los expertos.

Se logró realizar pruebas suficientes en plataformas de desarrollo abiertas y cerradas para comprobar que *Microsoft* y sus herramientas tienen un alto grado de compatibilidad con Java, lo cual no puede decirse en sentido contrario, ya que Java presentó complicaciones en el uso de componentes originados en OLE/DCOM de *Microsoft*.

Al generar los *middleware* para ser utilizados en sus correspondientes plataformas, se tuvo que utilizar herramientas de *software* como soporte para el logro del funcionamiento de un *broker*. De lado de *Microsoft* hubo dos opciones para ser utilizadas (JBImp.dll, Ikvm.dll), ambas construidas por proveedores de *Microsoft* con una interfaz de fácil uso dentro de las aplicaciones y no se requirió soporte o ayuda extra para su aplicación; de lado de Java solo hay una opción viable: El uso de JNI (Interfaz Nativa de Java), la cual presentó dificultades para su configuración y aplicación posterior, exigiendo la investigación sobre su aplicación correcta en foros de discusión y portales de ayuda en línea para la realización de las pruebas requeridas.

Se puede constatar que la herramienta abierta Java no es tan abierta como comercialmente se difunde y la herramienta cerrada *Visual Studio .Net* no es tan cerrada como sus competidores remarcan dentro del mercado global de Tecnologías de la Información. Esto está permitiendo que el proveedor de *software* *Microsoft* gane mercado en el ámbito del desarrollo de componentes, pero sobre todo, la aceptación de los desarrolladores que son (o eran) fieles a las plataformas de desarrollo de *software* abierto.

La metodología RAD aplicada en la comparación de componentes nos permitió conocer una parte importante de las plataformas de tecnologías de información que no se conocía y estaba enmascarado por la comercialización de herramientas de *software*: La compatibilidad de componentes.

Finalmente, los prototipos construidos lograron la meta: El uso de los componentes abiertos de Java en conjunto con componentes cerrados de *Microsoft*, esto permitirá abrir la visión de los desarrolladores de *software* para poder construir aplicaciones más robustas, explotando de la mejor manera los recursos de las computadoras y logrando la tan ansiada compatibilidad de plataformas de desarrollo (*Visual Studio .Net* y Java) que hasta ahora no ha tenido un avance pleno debido a razones mercadotécnicas a nivel mundial, pero con prototipos como los presentados en este trabajo, permitirá que los estudiantes y profesionales con perfiles afines al desarrollo de Tecnologías de Información comprueben la compatibilidad aquí propuesta y la apliquen en la construcción de sus sistemas e incluso, puedan crear

nuevos caminos de compatibilidad para el beneficio de toda la comunidad de ingenieros y licenciados en tecnologías de la información, en los ámbitos de desarrollo de aplicaciones web, móvil y cliente servidor en general para múltiples plataformas.

FUENTES DE CONSULTA.

- GONZALEZ, Antonio. (2005) *Aplicación de la tecnología de componentes Java-Beans para realizar el control de un robot trepador.* Colombia: Universidad Politécnica de Cartagena.

- MALONY, Allen; Shende, Sameer; Trebon, Nicholas; Ray Jaideep; Armstrong, Robert; Rasmussen, Carl; Sottile, Matthew. (2005) *Performance technology for parallel and distributed component software*. Oregon: Wiley InterScience.

- PRESSMAN, Roger. (1997). *Software Engineering: A Practitioner's Approach.* New York: McGraw-Hill.

- SEONGKI, Kim; Sangyong Han. (2006) *Performance comparison of DCOM, CORBA and Web service*. Seoul: School of Computer Science and Engineering.

- VICENTE, Cristina. (2005) *Desarrollo integral de sistemas de procesamiento de información visual: Un enfoque multiparadigma basado en líneas de producto, componentes y generación automática de software*. Colombia: Universidad Tecnológica de Cartagena.

- WOO, Jongwook. (2005) *The Comparison of J2EE and .NET for e-Business*. Los Angeles: California State University.