Componentes! Una visión para el desarrollo de software

Sandro Javier
Bolaños Castro

RESUMEN

Uno de los problemas mas complejos que existen es el desarrollo de software por el alto grado de ingeniería que este debe tener. Son muchas las tendencias que existen y filosofías que plantean formas de concebir y solucionar un problema que finalmente se plasme en un producto de software. Dentro de estas tendencias existe la concepción del software como componente, filosofía acogida por las características interesantes que tiene y que en gran medida puede ser una solución bastante técnica del problema. En el presente artículo se presentan tres tecnologías que soportan el desarrollo basado en componentes COM, JavaBeans y CORBA sus características y pautas fundamentales en las que se pueden comparar y finalmente una visión crítica de lo que son las tres filosofías de componentes.

Palabras Clave: Software basados en componentes desarrollo de software con Java Beans, Corba

COMPONENTS! APPROACH TENDED IN THE SOFTWARE DEVELOPMENT

ABSTRACT

Software development is one of the more complex problems by the high level of engineering that it must have. In this moment there are many tendencies and philosophies that introduce forms of conceiving and solving this problem, to generate a software product. Inside this tendencies exist the software conception like a component. This philosophy is accepted by its interesting characteristics and because it must be a good technique solution to the problem. This paper explain three technologies based on components: COM, JavaBeans and CORBA, their characteristics, fundamental guidelines and finally a vision criticizes.

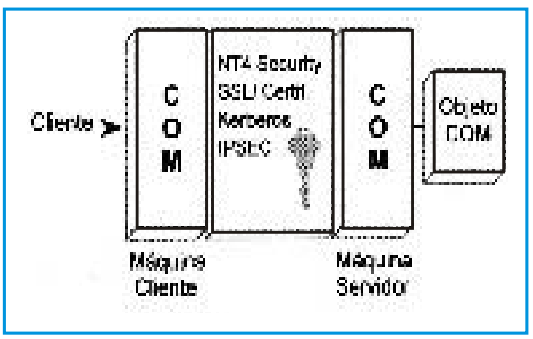
Key Words: Software as component, software, com Java Beams Corba.

PERSPECTIVA DEL DESARROLLO DE SOFTWARE POR COMPONENTES

Cuando se piensa en desarrollar software se piensa en un grupo de trabajo en el que definitivamente como factor clave debe imperar un amplio conocimiento tecnológico y filosófico. Un modelo lógico que funcione, no necesariamente está bien formado, y es precisamente este el punto que hace la diferencia entre un producto software bien desarrollado y un producto con deficiencias, que difícilmente crecerá. Tecnológicamente existen diversas herramientas de soporte a un buen desarrollo, sin embargo no muchas filosofías.

La propuesta orientada a objetos ha madurado lo suficiente para ser aceptada y por su puesto confiable, sin embargo existen enfoques que sin ser radicalmente nuevos, y que de hecho están basados en el modelo de objetos, pueden contribuir a mejores resultados, o al menos esta es la propuesta de los componentes, cuya abstracción del mundo la basan en unidades de carácter específico con alta cohesión e independencia. Existen tres grandes alternativas para la construcción de software basado en componentes, estas son, los COM los JavaBeans y la filosofía CORBA teniendo en cuenta que CORBA es ante todo una filosofía de componente distribuido mientras COM y Beans pueden solventarse como componentes cliente.

II. COM



COM Figura 1 visión

El modelo de objeto componente COM mostrado en la figura 1 fue creado bajo la filosofía de reusabilidad y facilidad para el desarrollo de software. Provee un amplia posibilidad en servicios, herramientas, lenguajes y aplicaciones.

COM ofrece una extensa gama en integración de herramientas de desarrollo y trasportabilidad en red. Ofrece gran flexibilidad y seguridad. COM permite el manejo de transacciones y transferencia de datos de manera transparente y segura.

Una de las características interesantes de COM es la compatibilidad con protocolos como TCP, UDP, IPX, SPX, HTTP lo cual permite tanto a nivel de clientes como de servidores compartir aplicaciones que hablan un mismo idioma y cuya ejecución en ambos tendrán comportamientos similares.

Esta flexibilidad hace juego con el alto nivel de seguridad que se puede manejar directamente en COM con protocolos como SSL, IPSEC y NT4 "security". Bajo este esquema de red el COM se extiende con la concepción de DCOM o modelo de objeto componente distribuido.

Un objeto COM es muy parecido a una instancia de clase C++ o un paquete ADA. De hecho, COM fue pensado básicamente en una sintaxis muy parecida a la empleada en C++. COM soporta encapsulación, polimorfismo y reusabilidad.

Es importante tener en cuenta que COM fue diseñado para ser compatible a nivel binario, por tanto es independiente de la arquitectura, a diferencia de lenguajes que como el C++ deben ser compilados, factor que los hace dependientes de la plataforma en particular sobre la cual se compile.

Como un objeto binario un objeto COM se preocupa por la interacción con otros objetos. Cuando COM no se utiliza en el entorno de su creador se expone una interfaz visible desde otros entornos no nativos.

Debe tenerse presente que COM no es un lenguaje aunque se asocie a C++ y su sintaxis. COM es ante todo un estándar que permite a los componentes de software interactuar entre sí con todas las propiedades de los objetos, brindando también la posibilidad de trabajar con cualquier lenguaje que soporte la estructura binaria de un objeto COM.

Tanto objetos OLE como su tecnología sucesora ActiveX están construidas basadas en objetos COM a través de interfaces que brindan un manejo robusto pero flexible y transparente para el usuario.

Mediante un conjunto de interfaces bien diseñadas es posible construir un documento ActiveX que pueda operar dentro de cualquier contenedor ActiveX, sin saber nada mas sobre éste, a excepción de la existencia de un conjunto de interfaz COM, que permiten al usuario combinar componentes de maneras quizá no generadas por el mismo desarrollador de una aplicación.

Es importante recordar, que la tecnología ActiveX es básicamente una escala mas allá de la tecnología OLE y pensada fundamentalmente como tecnología orientada a Internet, con elementos y características propias como hipervínculos, conferencias ActiveX, extensión ActiveX para servidores, firma de código, ActiveMovie y otra serie de conceptos que hacen pensar no solo en utilización de objetos componentes, sino también en objetos componentes distribuidos DCOM, los cuales se constituyen como una extensión de Network OLE y que nos permiten vincular objetos a través de la red [1].

III. Java Beans

Figura No 2 visión JavaBean

JavaBean cuya visión se muestra en la figura No.2 es un modelo de componente portable e independiente de la plataforma, escrito en Java, que habilita a los desarrolladores escribir componentes reutilizables que corran bajo cualquier arquitectura.

Un JavaBean puede actuar como puente entre modelos de componentes. También ofrece un poderoso significado a la construcción de componentes que actúen junto con la tecnología

ActiveX para la cual ya existen en el mercado productos con APIs prometedoras.

Cuando se habla de Bean "JavaBean" es importante no confundirlo con la mínima aplicación en java creada pensando en el web, esto es el Applet, pues Bean ante todo tiene un significado de componente en el que podemos distinguir características tan interesantes como la personalización, manejo de eventos, persistencia y otras que hacen de un Bean una filosofía robusta y consistente para el desarrollo de software.

IV. Corba

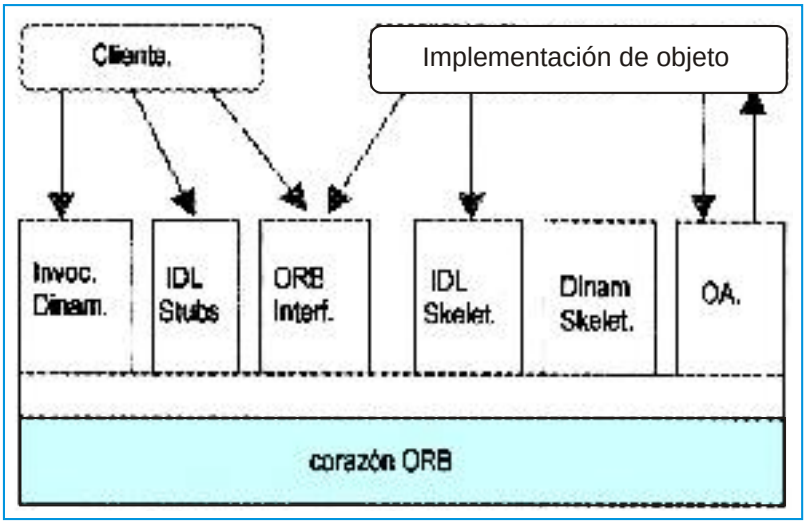


Figura No. 3 Arquitectura CORBA

Usar componentes como JavaBeans permite a los desarrolladores construir código portable y reusable, dos características fuertes y deseadas en todo buen software y que dan la posibilidad de atacar rápidamente nuevas oportunidades de mercado y desarrollo, con nuevas formas de vender pequeños paquetes de software, que hoy por hoy constituyen un campo de acción de grandes dividendos.

JavaBean es un modelo de componente en toda su extensión, soportando el estándar de la arquitectura y ofreciendo un ambiente ideal para el desarrollador, quien desea extender el concepto de reusabilidad de componentes más allá de la plataforma.

Sun Microsystems ha pensado en la tecnología JavaBean con gran acierto, viendo un mercado de componentes, bastante amplio y atractivo e incluso pensado en la filosofía BeanBox que aunque planeada como un contenedor de prueba de JavaBean, puede apuntarse como un ambiente de desarrollo que permite en gran medida evaluar el comportamiento de componentes java.

Existen además dos especificaciones de nombres de códigos como son Glasgoew [2] y Edinburgh de especificación de JavaBeans que ofrecen características como extensibilidad en los protocolos de servicios, modelos de agregación-delegación, capacidad de plataforma nativa "drag and drop" con lo que se puede extender con gran fuerza el alcance, que de por sí desde sus inicios tiene el componente java [3].

CORBA "Common Object Request Broker Architecture" mostrada en la figura No. 3 especifica un sistema que provee interoperabilidad entre objetos dentro de un ambiente heterogéneo y distribuido, brindando un método transparente de desarrollo. Se trata de un estandar de sistemas de objetos distribuidos. Esta arquitectura está diseñada con base en el modelo de objeto de la OMG (Object Managemet Group)que ofrece una semántica de objeto común para especificar externamente las características visibles de los objetos en forma estándar e implementación independiente.

En este modelo los clientes requieren servicios de los objetos a través de las interfaces definidas, especificadas por la OMG como IDL (Interface Definition Language). El elemento central de CORBA es el ORB "Object Request Broker", el cual abarca toda la infraestructura de comunicación necesaria para identificar y localizar objetos.

En general ORB no se utiliza para hacer un componente simple, éste es definido para realizar sus interfaces.

Existen en el mercado diversos productos ORB, dirigidos a suplir las necesidades de ambientes operacionales específicos, lo cual exige necesariamente formas de interoperabilidad entre ellos, normalizados por la arquitectura CORBA. Es por lo cual CORBA introduce conceptos de alto nivel de un dominio, en el que se circunscribe un conjunto de objetos de

implementación o de administración separados de los otros objetos.

Esto hace necesario a su vez mecanismos de puenteo entre dominios en orden a interactuar adecuadamente. El aprovechamiento de la interoperabilidad se puede dividir en puenteo inmediato y puenteo intermediario. Cuando se habla de puenteo intermediario los elementos intermediarios de un dominio son transformados al límite de cada dominio entre la forma específica interna a este dominio y algunas otras formas mutuamente agregadas sobre el dominio. Esta forma común podría ser estándar o un acuerdo privado entre las dos partes.

Con puenteo inmediato, los elementos de interacción son transformados directamente entre la forma interna de un dominio y el otro. La segunda forma tiene un gran potencial a ser más rápida pero es menos general. Pero quizá sea adecuado usar las dos formas.

El puenteo puede ser implementado inmediato e intermediario internamente en un ORB o en su nivel superior. Si estos son implementados con un ORB estos son llamados puentes en línea, de otro modo son llamados puenteo de nivel de petición.

El puenteo en línea puede ser implementado a través de requerimientos que el ORB provee o a través de la introducción adicional de código "stub code o skeleton code".

Para hacer posible puenteo es necesario especificar alguna clase de sintaxis. Esta función es suministrada por General Inter - ORB Protocol (GIOP) definido por la OMG, el cual ha sido específicamente diseñado para cumplir con la necesidad de interacción de ORB a ORB.

Aparte de la definición de la sintaxis de transferencia general la OMG también especifica, como va a ser implementada usando transporte TCP/IP definido como Internet-Inter-ORB Protocol (IIOP).

En orden a ilustrar la relación entre GIOP y IIOP. OMG apunta a que ello sea lo mismo que la relación entre IDL y su mapeo concreto, por ejemplo el mapeo sobre C++. IIOP está diseñado para proveer interoperabilidad con otros ORBs compatibles [4].

Una ventaja importante de Corba es la portabilidad de objetos entre sistemas distribuídos.

V. CONFLUENCIA DE LOS MODELOS DE COMPONENTES

Uno de los aspectos interesantes en los que confluyen los modelos de objetos componentes es el de las transacciones para lo cual COM, JavaBeans y CORBA tienen sus propias tecnologías.

Cuando hablamos de transacciones bajo COM debemos referirnos a los objetos MTS "Microsoft® Transaction Server" los cuales están típicamente diseñados para encapsular algún conjunto de funcionalidades del negocio. Por ejemplo un objeto MTS puede permitirle a un cliente realizar transferencias monetarias entre dos cuentas con gran flexibilidad y seguridad. Un objeto MTS básicamente puede ser invocado con la llamada directa de un objeto COM, en otras palabras usando un DCOM.

Por su parte JavaBean implementa para efectuar el servicio de transacción los EJB "Enterprise JavaBeans". Una especificación EJB toma dos tipos de EJB diferentes, uno de ellos llamado la sección Bean la cual es muy similar al objeto MTS y el otro un objeto persistente llamado entidad Bean, con los cuales se suministra por un lado comunicación entre componentes y por otro seguridad. Ambos tipos de EJB pueden ser accedidos con JRMP "Java Remote Method Protocol" o a través de IIOP "Internet Inter-ORB Protocol" definido por la OMG.

Cuando hablamos de CORBA dirigido sustancialmente a aplicaciones distribuidas debemos mencionar entonces el conjunto de protocolo de la OMG bajo los cuales se normaliza una transacción, estos son IIOP y GIOP.

Otro aspecto importante a mencionar es el lenguaje sobre el cual se formulan cada una de las tecnologías de componentes citadas.

Cuando hablamos de COM se habla de una tecnología con posibilidades de implementación bajo diferentes lenguajes como C, C++, VB, Java, Fortran, Cobol, Perl, REXX, Javascript y muchos otros.

Cuando hablamos de JavaBean estamos hablando exclusivamente de una tecnología implementada sobre Java.

Si se trata de CORBA estamos hablando de una

tecnología con un marco de implementación en lenguajes como C, C++, Java, Small Talk, cobol y ADA, no se puede implementar bajo lenguajes script.

La anterior consideración nos permite ver como COM puede ser un estándar de implementación con un rango mas amplio de lenguajes lo cual puede ser de alguna forma favorable, teniendo en cuenta lógicamente que la cantidad no necesariamente implica la calidad.

Sin lugar a dudas es necesario también hablar de otro aspecto importante a tener en cuenta cuando de componentes se trata, este es la seguridad.

En el modelo componente de objeto COM hablamos de SSL clave pública, DCE o MIT Kerberos.

En el modelo de componente JavaBean podemos hablar de SSL.

En la arquitectura CORBA hablamos de múltiple variaciones de SSL para encriptar los datos que son enviados por internet.

En cuanto a protocolos de transporte sobre los cuales pueden los modelos de componentes desempeñarse podemos mencionar:

Para COM TCP IP, IP, IPX, SPX, HTTP. Para JavaBean; TCP IP, IP, IPX, HTTP. Para CORBA, TCP/IP. Esta característica nos deja entrever la flexibilidad de COM y JavaBean en cuanto a protocolos de transporte se refiere en comparación con la arquitectura CORBA la cual corre exclusivamente sobre TCP IP.

Desde luego se debe tener en cuenta la concepción bajo la cual fue creado CORBA con su énfasis distribuido el cual tiene grandes limitaciones no por su filosofía que de hecho es muy buena sino más bien sobre la infraestructura tecnológica que de alguna manera impone barreras importantes.

VI. CONCLUSIONES

Siempre que aparece en el mercado una nueva tendencia de desarrollo de software se crea una ansiedad hacia ella, mezclada con la incertidumbre de saber si será la más adecuada para un problema en particular. Surge ahí el gran trabajo de conocer su estructura y testificar si realmente es confiable, flexible y robusta.

Las características mencionadas sobre COM, JavaBean y CORBA nos presentan tres perspectivas de desarrollo de componentes, teniendo en cuenta lógicamente que CORBA ante todo hace una propuesta de componente pero con énfasis en objetos distribuidos.

Las tres con fortalezas significativas pero quizá con una característica común, la solidez no solo en el mercado sino en cuanto a la solución de problemas de software se refiere.

Las tres tendencias presentadas. COM, JavaBean y CORBA reafirman la tendencia que en este momento el mercado del software tiene, "El desarrollo de componentes" los cuales reúnen características singulares que los hacen tan atractivos entre las que podemos mencionar: portabilidad, compatibilidad, transportabilidad, seguridad, escalabilidad, y reusabilidad.

Otra de las consideraciones importantes a tener en cuenta, es el mercado de fabricantes de tecnologías, pues ello puede dar una visión de lo que se debe enfrentar. Esto para hacer referencia a lo que muchos expertos podrían llamar el monopolio de los fabricantes.

Por un lado, se puede ver el sello Microsoft con COM y por otro Sun Microsystem con JavaBean, además de CORBA del grupo de investigación OMG. Se tendrá que escoger quizá como parámetro principal de elección, la tecnología que cumpla con una mayor estandarización y solidez, tarea que puede resultar un poco complicada sino se efectúa un estudio a fondo de la tecnología en cuestión.

BIBLIOGRAFIA

[1] *COM "Component Object Model "* www.microsoft.com
[2] www.javasoft.com
[3] *Hong Youngrae, Real-Time Operating System Lab., KyungHee Uniersity*
[4] *Kate Keahey* kksiazek@cs.indiana.edu

Sandro Javier Bolaños Castro
Ingeniero de Sistemas, Universidad Distrital.
Candidato a Maestría en Teleinformática, Universidad Distrital.
Profesor Facultad de Ingeniería, Universidad Distrital.