Aplicación de gestión para estaciones de usuario basada en el estándar WBEM y su aplicación a la Red de Datos de la Universidad del Cauca

Guefry Agredo Méndez
Universidad del Cauca
Popayán, Cauca, Colombia
gagredo@unicauca.edu.co

Natalia Maya Ortiz
Universidad del Cauca
Popayán, Cauca, Colombia
nmaya@unicauca.edu.co

Eva Juliana Maya Ortiz
Universidad del Cauca
Popayán, Cauca, Colombia
emaya@unicauca.edu.

RESUMEN

Actualmente las redes de comunicaciones son el eje fundamental de trabajo en la mayoría de las empresas, por tanto los administradores deben procurar el buen desempeño de la red para garantizar el correcto funcionamiento de la organización.

Cuando una red falla, o tiene un bajo desempeño, los costos pueden ser enormes, la productividad de los empleados disminuye, y la insatisfacción entre usuarios es evidente. Debido a esto, la gestión ha llegado a ser un aspecto muy importante, y al mismo tiempo complejo, ya que las redes son cada vez más grandes, heterogéneas, y los requerimientos de confiabilidad, disponibilidad y desempeño más altos.

Por esto, es indispensable tener herramientas que permitan gestionar las redes de una forma integrada y efectiva, pero la mayoría de ellas son costosas, propietarias, e incluso complejas.

Teniendo en cuenta lo anterior, el proyecto del que trata este documento, tiene por objetivo diseñar e implementar una aplicación de gestión para estaciones de usuario con Interfaz Web, que sea portable, escalable, fácil de usar, económica y se adapte a las necesidades de cualquier red. Esto se va a lograr a través del uso de tecnologías estándar, tanto para la gestión de redes, como para el desarrollo de la aplicación, como son respectivamente: la arquitectura de Gestión Empresarial Basada en Web – WBEM (Web Based Enterprise Management) y el Modelo Genérico de Información – CIM (Common Information Model), y la Plataforma de Desarrollo que ofrece Java.

INTRODUCCIÓN

Algunos de los estándares de gestión que están siendo utilizados actualmente, aunque en diferentes áreas, son SNMP (Simple Network Management Protocol) y CMIP (Common Management Information Protocol). SNMPv1 ha sido el más difundido en el mundo de Internet, y CMIP ha estado intentando ganar aceptación en el mundo de las Telecomunicaciones. SNMPv1 no ofrece seguridad y CMIP no ha tenido gran acogida debido a su complejidad. Además, las diferentes versiones del mismo estándar pueden ser incompatibles, como es el caso de SNMPv1 y SNMPv3. Las desventajas de estos estándares han hecho surgir nuevas alternativas, que puedan representar todos los elementos de red posibles en un solo modelo de información y puedan unificar los estándares actuales. Un intento de alcanzar este objetivo es el estándar WBEM del DMTF (Distributed Management Task Force).

El DMTF [1], es una organización de industrias que está liderando el desarrollo, adopción y unificación de estándares de gestión e iniciativas para ambientes de escritorio, empresa e Internet. Sus principales metas son:

- Acelerar la adopción de estándares de gestión.
- Unificar las iniciativas de gestión.
- Promover la interoperabilidad entre los proveedores de soluciones de gestión.

Existen varias razones por las cuales WBEM ha tenido un gran impacto, y podría llegar a ser el estándar definitivo para la gestión de redes. La primera de ellas, es el apoyo que ha recibido de empresas líderes del sector de las telecomunicaciones como: 3COM, Cisco, Dell, Hewlett Packard, IBM/Tivoli, Intel, Microsoft, Sun, entre otras. La segunda razón por la que la popularidad de WBEM está creciendo, es el uso del modelo de información CIM. Con CIM se puede modelar cualquier objeto gestionable, en una forma estandarizada y fácil de entender. Esto hace que WBEM pueda ser aplicado en casi cualquier área de gestión. Una tercera razón para creer en el futuro de WBEM, es que este estándar no intenta reemplazar los estándares actuales tales como SNMP, sino que coexiste con ellos y los complementa.

ESTÁNDAR WBEM

WBEM [2] es una iniciativa y una tecnología. Como una iniciativa, WBEM incluye estándares para la gestión de sistemas, redes, usuarios, aplicaciones, bases de datos, dispositivos, eventos, entre otros, utilizando las tecnologías de Internet. Como una tecnología, WBEM proporciona una forma para que las aplicaciones de gestión compartan datos independientemente del vendedor, protocolo, sistema operativo o estándar de gestión.

El DMTF ha desarrollado un conjunto núcleo de estándares que componen a WBEM, el cual incluye un modelo de datos, el estándar CIM; una especificación de codificación, xmlCIM; y un mecanismo de transporte, Operaciones CIM sobre HTTP, como se muestra en la Figura 1.

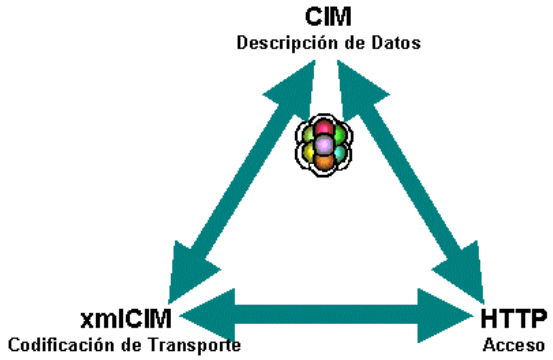

Figura 1. Conjunto Núcleo de Estándares WBEM

Estándares WBEM

1. CIM

CIM [3] tiene dos partes: la Especificación CIM y el Esquema CIM.

La Especificación CIM describe el lenguaje, nombramiento, meta-esquema y técnicas de mapeo a otros modelos de gestión. El Meta-Esquema define los términos usados para expresar el modelo, su uso y semánticas. Los elementos del Meta-Esquema son Clases, Propiedades y Métodos.

El Esquema CIM del DMTF permite a las aplicaciones de diferentes desarrolladores en diferentes plataformas, describir datos de gestión en un formato estándar, para que puedan ser compartidos entre una variedad de aplicaciones.

El Esquema CIM está estructurado en tres capas distintas, como se muestra en la Figura 2.

Figura 2. Estructura en Capas del Esquema CIM

El Modelo núcleo es un modelo de información aplicable a todas las áreas de gestión. Incluye un pequeño conjunto de clases, asociaciones y propiedades que proporcionan un vocabulario básico para analizar y describir sistemas gestionados. El Modelo común es un modelo de información aplicable a áreas de gestión particulares, tales como sistemas, dispositivos y aplicaciones, pero independientes de una tecnología o implementación particular.

El Modelo núcleo y común conforman al Esquema CIM.

El Esquema de extensión, es un modelo de información que soporta el esquema CIM y representa una plataforma específica.

2. xmlCIM

El DMTF ha desarrollado un DTD (Document Type Definition) para CIM, el cual define cómo los elementos CIM pueden ser representados por elementos XML. La razón para representar CIM en XML, es el hecho de que XML se ha convertido en el principal formato para representar datos sobre Internet, y el fin de WBEM es usar estándares basados en Web tanto como sea posible.

3. Operaciones CIM sobre HTTP

Es una especificación de cómo intercambiar información CIM sobre el protocolo HTTP. Toda la información CIM que es intercambiada entre clientes y servidores son mensajes CIM. Estos mensajes son independientes del protocolo, por tanto pueden ser enviados sobre cualquiera de ellos. Sin embargo, HTTP fue seleccionado como el protocolo para la encapsulación de mensajes CIM debido a su amplio uso en Internet.

Arquitectura WBEM

La arquitectura del estándar WBEM consta de lo siguientes elementos: el CIMOM (CIM Object Manager), los proveedores y los clientes, como se muestra en la Figura 3.

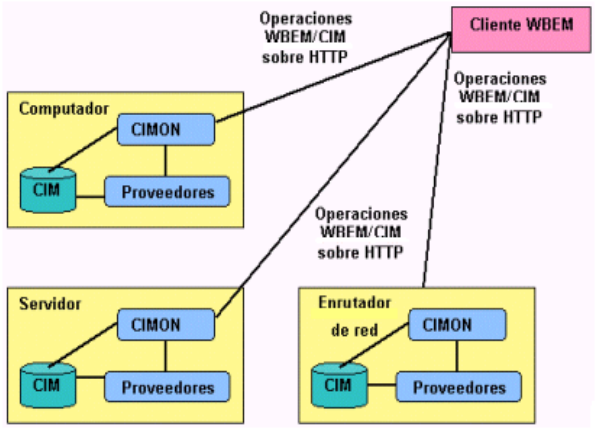

Figura 3. Arquitectura de WBEM

CIMOM: es la parte central de WBEM. Tiene un repositorio donde almacena todos los esquemas CIM. Es el responsable de manejar la interacción entre las aplicaciones de gestión y los proveedores de objetos. Cuando una aplicación hace una petición, el CIMOM se comunica con el repositorio (información estática), o con el proveedor apropiado (información dinámica), dependiendo del tipo de información requerida.

Proveedores: pueden ser vistos como una interface entre el recurso gestionado y el CIMOM. Los datos proporcionados por un proveedor son llamados datos dinámicos. Cuando el CIMOM requiere datos dinámicos, el proveedor los obtiene desde el recurso gestionado y los retorna al CIMOM.

Usualmente los proveedores residen en el mismo computador que el CIMOM, y a diferencia de la comunicación entre clientes y CIMOM, la comunicación entre proveedores y CIMOM no está estandarizada.

El proyecto SBLIM (Standards Based Linux Instrumentation for Manageability) [4] de IBM, desarrolló dos APIs escritas en C, para lograr la comunicación entre proveedores y cualquier CIMOM. Ellas son: NPI (Native Provider Interface) [5] y CMPI (Common Manageability Programming Interface) [6]. NPI ya no está siendo desarrollada, mientras que CMPI, su sucesora, está en proceso de estandarización por la iniciativa WBEMsource [7]. La iniciativa WBEMsource es una organización encargada de coordinar implementaciones WBEM de fuente abierta, con el fin de lograr interoperabilidad y portabilidad entre ellas

Cliente: puede ser visto como una interfaz entre el administrador y el CIMOM. Las operaciones CIM sobre HTTP es el estándar de comunicación entre el cliente y el CIMOM. Sin embargo, la mayoría de implementaciones también soportan otros mecanismos para comunicación, como RMI (Remote Method Invocation) para Java, DCOM (Distributed Component Object Model) para Microsoft, e IPC (Inter Process Communication) para implementaciones Unix. Pero únicamente el uso de operaciones CIM sobre HTTP garantiza compatibilidad entre cualquier cliente y cualquier CIMOM.

Implementaciones WBEM

1. Implementaciones de uso libre

Actualmente existen pocas implementaciones gratis. Ellas difieren en el lenguaje de implementación y el sistema operativo para el cual fueron hechas. A continuación se describen cuatro de las implementaciones más conocidas. Todas ellas soportan requerimientos XML/HTTP, tienen implementación de CIMOM, y proporcionan API cliente.

WBEM Services [8]

- Desarrollador: Sun Microsystems
- Lenguaje de implementación: Java
- Lenguaje de programación: Java

Sistemas operativos soportados: Todos

Pegasus [9]

- Desarrollador: The Open Group
- Lenguaje de implementación: C++
- Lenguaje de programación: C++
- Sistemas operativos soportados: Linux, Unix y Windows

OpenWBEM [10]

- Desarrollador: Caldera
- Lenguaje de implementación: C++
- Lenguaje de programación: C++
- Sistemas operativos soportados**:** Linux, Unix y Solaris

SNIA [11]

- Desarrollador: SNIA (Storage Networking Industry Association)
- Lenguaje de implementación: Java
- Lenguaje de programación: Java
- Sistemas operativos soportados: Todos

SNIA lanzó en el 2002 la iniciativa de gestión de almacenamiento SMI (Storage Management Initiative) [12], para crear e impulsar la adopción de una interfaz de gestión interoperable y funcional, para SANs (Storage Area Network).

La especificación de SMI, SMI-S, es la interfaz entre los objetos de almacenamiento que deben ser gestionados y las aplicaciones de gestión.

SMI-S está basada en Bluefin, una especificación de gestión estándar para SANs, que usa los estándares CIM y WBEM para gestionar dispositivos sobre SANs. Además, Bluefin proporciona una interfaz proxy, que permite a los equipos ya adquiridos, interoperar con productos nuevos, habilitados con Bluefin.

2. Implementaciones comerciales

WMI [13]

- Desarrollador: Microsoft
- Lenguaje de implementación: C++
- Lenguaje de programación: Visual Basic, Visual C++, lenguajes Scripting.
- Sistemas operativos soportados: Windows

ESTUDIO DE ALTERNATIVAS

Después de estudiar las implementaciones WBEM nombradas anteriormente, la mejor opción por usar Java como lenguaje de implementación y programación, y por ser la más completa y clara en la documentación, es WBEMServices de Sun Microsystems. Aunque solo proporciona proveedores específicos para Solaris, es posible desarrollar proveedores propios. Además, puede ser considerada como una implementación WBEM de referencia, y es utilizada por otras como SNIA y OpenWBEM. Además, OpenWBEM no es soportada por equipos Windows (un aspecto esencial, teniendo en cuenta que la aplicación es para estaciones de usuario, que son en su mayoría Windows), y Pegasus está aún en proceso de desarrollo.

WMI viene incluido en el sistema operativo, excepto en Windows 95 y 98, para los cuales se puede descargar sin ningún costo. Utiliza CIM como lenguaje de modelamiento, pero usa DCOM en lugar de operaciones CIM sobre HTTP. Sin embargo, la mayoría de las operaciones son soportadas por DCOM. Proporciona varios proveedores específicos de Windows y también es posible desarrollar otros. Debido a que utiliza un protocolo propietario (DCOM), solo permite la comunicación dentro de un entorno Windows, por lo que se pierde interoperabilidad.

Debido a esto, para el desarrollo del Proyecto sobre el que trata este documento, se pensó utilizar la API cliente de WBEMServices, con el fin de realizar la aplicación en Java, y el CIMOM de WMI que ya viene instalado en los equipos Windows (excepto Windows 95 y 98). Sin embargo, es importante tener en cuenta que ya que el CIMOM de WMI no soporta requerimientos XML/HTTP, no se puede lograr la comunicación entre éste y una aplicación cliente realizada con el SDK de Sun (WBEMServices), que si soporta estos estándares.

Prodexnet [14], una importante empresa en el sector de las telecomunicaciones, desarrolló una solución para que una aplicación cliente desarrollada, ya sea con la API Cliente del SDK de SNIA o Sun, se comunique con el CIMOM WMI. Esta solución consiste en un "Wrapper" WBEM/WMI y un proveedor "proxy" que se debe instalar

solamente en el caso en el que la aplicación esté sobre un sistema no Windows. Esto demuestra la complejidad de una solución de este tipo.

Entonces se pensó utilizar el CIMOM y la API cliente de Sun para evitar estos problemas de comunicación, pero esto tiene varios inconvenientes. Uno de ellos es la comunicación entre el CIMOM de Sun y un proveedor de Windows, ya que la comunicación entre CIMOM y proveedor aún no está estandarizada por el DMTF. Para solucionar esto, se estudió la posibilidad de utilizar JNI como un puente de comunicación entre el CIMOM de Sun y el proveedor de Windows, que consiste de un archivo .mof y una DLL. Esto implica un conocimiento profundo de la DLL y una programación a bajo nivel. Además, con esta solución se debe instalar el CIMOM de Sun en cada una de las máquinas que van a ser gestionadas, algo no tan viable sobre una red.

La plataforma .NET de Microsoft proporciona una nueva forma de programación para la plataforma Windows, y una de las principales características es el soporte para XML, HTTP y otros protocolos relevantes. Además, WMI se expone a .NET a través de la librería system.management. Así que, mientras WMI no proporciona soporte para XML y HTTP, .NET si soporta estos dos estándares, y proporciona una forma de usar la API de WMI a través de XML y Servicios Web. Sin embargo esta solución también tiene algunas desventajas, la primera de ellas es que .NET no es una plataforma muy difundida en la actualidad (menos en la Universidad del Cauca), y además ésta infraestructura sigue siendo muy cerrada, es decir la comunicación se limita a entornos Windows.

Hasta este punto, existía un conflicto entre el deseo de utilizar Java, que permite obtener aplicaciones portables y reducir enormemente los costos de desarrollo y despliegue, y la necesidad de utilizar la API de WMI, que además de ser la más apropiada y potente para gestionar estaciones de usuario Windows, es la única que se puede comunicar con el CIMOM WMI, ya instalado en los equipos Windows de los que constan casi en su totalidad la mayoría de las empresas en la actualidad (como la Universidad del Cauca). Para solucionar este problema se decidió utilizar un puente Java-COM [15], que permitiera manejar la API WMI desde Java. Existen herramientas gratis, como JACOB, Bridge Between Java and Windows, Jawin, Jcom, JWindows, etc., y otras comerciales, como J-Integra, R-JAX., WebLogic jCOM, jacoZoom, Java2COM, Bridge2Java, entre otras. De estas herramientas las más apropiadas para desarrollar este proyecto son JACOB [16] y J-Integra [17]. A continuación se muestra una comparación entre ellas.

1. JACOB es para Windows solamente. J-Integra soporta cualquier plataforma con una JVM compatible.

2. JACOB no es bidireccional, mientras que J-Integra si lo es.

3. JACOB solo ha sido probado sobre Windows NT con el JDK 1.1.6 y 1.2.2 de Sun. J-Integra ha sido probada sobre muchas plataformas y con muchos JDKs

4. JACOB no ha sido actualizado desde Septiembre de 2001. El último lanzamiento de J-Integra fue el 15 de agosto de 2003, y aproximadamente cada dos meses, son lanzadas nuevas versiones.

5. El soporte de JACOB se limita a un grupo de discusión de Yahoo. J-Integra, además del grupo de Yahoo, tiene personal técnico.

6. JACOB no soporta DCOM, solo COM, por lo que la comunicación entre máquinas no es posible. J-Integra si soporta DCOM.

Por estas razones se decidió utilizar J-Integra, que aunque es un producto comercial, proporciona soporte para DCOM, WMI, y su documentación es clara y completa .

A continuación se hace una descripción más detallada de WMI y el puente Java-COM J-Integra, que fueron las herramientas seleccionas para el desarrollo del proyecto.

WMI

La comunicación entre el CIMOM y los clientes se puede hacer a través de las siguientes formas [18]:

- API COM, que puede ser accedida desde C y C++.
- API Scripting, que puede ser accedida desde Visual Basic, JScript, Perl, o cualquier otro lenguaje scripting soportado por Windows.
- Internet Explorer y ASPs (Active Server Pages), que pueden almacenar scripts WMI.
- Adaptador ODBC de WMI, que proporciona una API estándar que permite a las aplicaciones basadas en ODBC usar los datos CIM como si ellos estuvieran en una base de datos.
- Interfaces ADSI, las aplicaciones de servicios directorio pueden usar la extensión ADSI (WMI Active Service Directory) para integrar los servicios de directorio y los datos de gestión.

WMI tiene diferentes proveedores [19], algunos de los más importantes son: Proveedor Win32, Proveedor SNMP, Proveedor de Registro del Sistema, Proveedor de Registro de Eventos, Proveedor de Directorio Activo, Proveedor WDM y Proveedor Windows Installer.

Además de usar los proveedores estándar, un desarrollador puede crear uno propio, que atienda requerimientos relacionados con objetos gestionados, específicos de una empresa.

WMI tiene las siguientes clases [20]: Clases de Sistema, Clases Win32, Clases Consumidor Estándar, Clases MSFT, Clases MSMCA y Clases C++ WMI.

J-INTEGRA

A continuación se mencionan los aspectos más importantes de J-Integra.

J-Integra es un puente Java-COM que puede ser utilizado para acceder componentes COM como si fueran objetos Java, y objetos Java como si fueran componentes COM.

El runtime Java de J-Integra se comunica con los objetos COM a través de DCOM, como se muestra en la Figura 4.

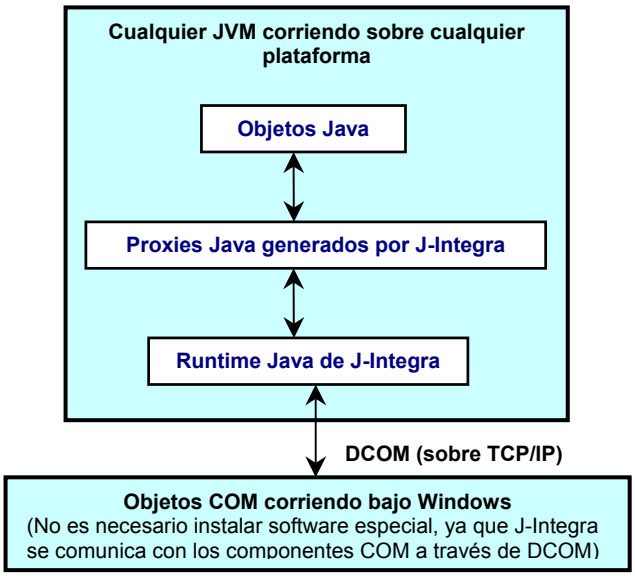

Figura 4. Runtime Java de J-Integra.

Modos de acceder objetos COM desde clientes Java [21]

Modo DCOM (Java Puro)
En este escenario un cliente Java puro corriendo sobre cualquier plataforma accede un servidor COM. En este modo la autenticación se establece en el código cliente, además no se requiere software de J-Integra en la máquina Servidor COM.

Modo DCOM con autenticación nativa
Este escenario es idéntico al anterior, excepto que en lugar de establecer explícitamente el dominio, usuario y password en el código cliente Java, J-Integra selecciona automáticamente la identidad del usuario que corre el cliente Java. Este modo sólo trabajará cuando el cliente Java se corra bajo Windows.

Modo Nativo
En este escenario, J-Integra usa código nativo (DLLs), en lugar de DCOM para invocar métodos de componentes COM desde un cliente Java.

Por defecto, J-Integra usa modo DCOM, el modo Nativo debe ser habilitado explícitamente.

IMPLEMENTACIÓN

La Aplicación de *Gestión para Estaciones de Usuario basada en el estándar WBEM*, está siendo desarrollada usando el lenguaje de programación Java, la API WMI de Microsoft, y la herramienta J-Integra.

WMI ofrece dos APIs: la API Scripting y la API COM, como se mencionó anteriormente. Es posible acceder ambas APIs usando J-Integra. Sin embargo,

solamente la API Scripting es accesible usando modo DCOM, por lo tanto, esta fue la API seleccionada.

J-Integra tiene una gran variedad de herramientas, y para el desarrollo del Proyecto fue necesario usar dos de ellas: com2java [22] y setdllhost [23]. Estas herramientas facilitan su utilización por parte del desarrollador y solo es necesario correrlas una vez.

Com2java lee información desde una librería de tipo, en este caso la librería de WMI, y genera archivos Java (proxies) que pueden ser utilizados para acceder las interfaces y clases COM definidas en la librería de tipo. Estos archivos junto con el runtime Java de Jintegra (jintegra.jar) deben ser configurados en el JDK seleccionado para manejar la API Scripting desde Java y utilizar las clases WMI.

La selección de las clases y los proveedores WMI, a utilizar en la aplicación, se hizo teniendo en cuenta las necesidades de la Administración de la Red de Datos de la Universidad del Cauca, lo cual sirve tanto de requerimiento, como de validación funcional. Las clases utilizadas pertenecen a las categorías de hardware y sistema operativo de las Clases Win32. Además se utilizaron algunas Clases Consumidoras Estándar. Los proveedores escogidos fueron el Proveedor Win32, Proveedor de Registro del Sistema, y Proveedor de Registro de Eventos.

La Figura 5 muestra el esquema de la arquitectura de la aplicación.

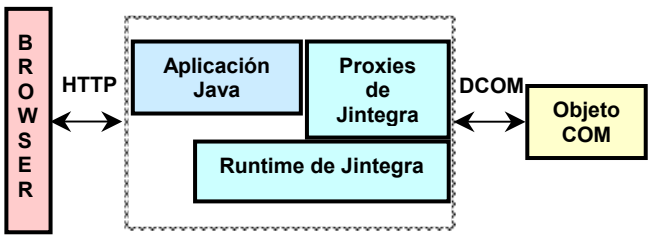

Figura 5. Esquema de la arquitectura de la aplicación.

La Aplicación Java se está desarrollando utilizando el modelo de diseño MVC (Model/View/Controller) [24]. Con este modelo la lógica o procesamiento es dividido en tres partes:

- Model (Beans): corresponde a la lógica y a los datos.
- View (Servlets o JSPs): corresponde al presentación.
- Controller (Servlets o JSPs): corresponde al procesamiento de requerimientos.

Esta arquitectura tiene varias ventajas, como claridad en el diseño, modularidad, escalabilidad, y permite separar la lógica de la presentación.

La aplicación permite:

- Descubrir los puertos de switches o hubs gestionables, y proporciona las direcciones MAC e IP y el nombre de los dispositivos conectados a esos puertos.

- Proporciona información del sistema operativo, BIOS, memoria, motherboard, puertos, teclado, tarjeta de red, procesos, servicios, archivos, cuentas, particiones, usuarios, grupos, protocolos, etc.
- Permite configurar direcciones IP, DNSs, puertas de enlace y proxies.
- Permite habilitar y deshabilitar seguridad IP.
- Habilitar DHCP.
- Configurar los puertos permitidos.
- Restringir las aplicaciones que los usuarios pueden correr y restringir a los usuarios de correr aplicaciones específicas.
- Apagar o reiniciar un equipo.
- Obtener el tamaño que ocupan los archivos de un determinado tipo, y borrar archivos.
- Instalar y desinstalar productos a través de Windows Installer.
- Enviar un email o ejecutar un scritps cuando ocurren eventos.

Todo esto es posible sobre máquinas remotas. Además la aplicación permite establecer todas las restricciones que sean necesarias para que los usuarios no cambien las configuraciones realizadas.

Esta aplicación podría ser fácilmente convertida en un Servicio Web, lo que permitiría a otro tipo de usuarios, como los de dispositivos móviles, acceder a toda la funcionalidad de esta herramienta de gestión.

Se puede usar un toolkit de servicios web cualquiera para tomar ventaja del puente java-COM de J-Integra. Primero se crea un Servicio Web java que use COM, y después una aplicación cliente java que use SOAP para comunicarse con el servicio web.

Además, ya que J-Integra proporciona un plugin JCA-COM [25], con el que se pueden acceder Enterprise Java Beans (EJB) desde COM, la aplicación podría ser desarrollada usando EJBs, como se muestra en la Figura 6. Así se puede tomar ventaja de la potencia de EJBs, sin sacrificar la inversión existente en tecnología COM.

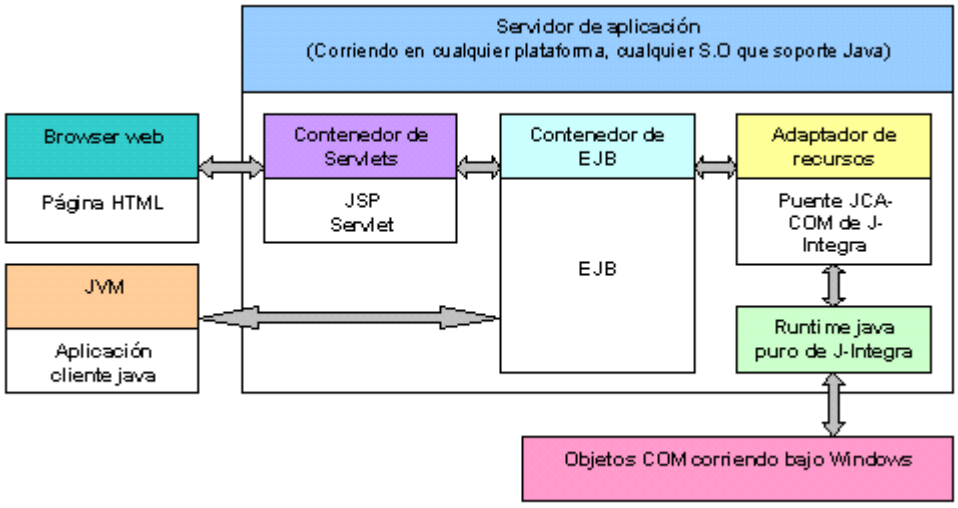

Figura 6. EJB accediendo COM

REFERENCIAS

[1] http://www.dmtf.org/home
[2] http://www.dmtf.org/standards/wbem
[3] http://www.dmtf.org/standards/standard_cim.php
[4] http://www-124.ibm.com/sblim/index.html
[5] http://www-124.ibm.com/sblim/npi.html
[6] http://www-124.ibm.com/sblim/cmpi.html
[7] http://www.wbemsource.org/
[8] http://wbemservices.sourceforge.net/
[9] http://www.openpegasus.org
[10]http://openwbem.sourceforge.net/
[11]http://www.opengroup.org/snia-cimom
[12]http://www.snia.org/smi/home
[13]http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/wmi_start_page.asp
[14]http://www.prodexnet.com/nsm/wbem_products.htm
[15]http://staff.develop.com/halloway/JavaWin32.html
[16]http://danadler.com/jacob/
[17]http://j-integra.intrinsyc.com/j-integra/doc/
[18]http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/wmi_management_applications.asp
[19]http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/wmi_providers.asp
[20]http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32_classes.asp
[21]http://www.intrinsyc.com/support/j-integra/doc/deployment/javatocom.htm
[22]http://www.intrinsyc.com/support/j-integra/doc/com2java/
[23]http://j-integra.intrinsyc.com/j-integra/doc/tools/SetDllHost.html
[24]http://developer.java.sun.com/developer/onlineTraining/JSPIntro/contents.html
[25]http://j-integra.intrinsyc.com/j-integra/doc/jca_com/Intro.htm