Aplicación de gestión para estaciones de trabajo de usuario basada en los estándares del DMTF y su aplicación a la Red de Datos de la Universidad del Cauca

Natalia Carolina Maya Ortiz
Eva Juliana Maya Ortiz

Director:
Ing. Esp. Guefry Agredo

Universidad Del Cauca
Facultad De Ingeniería Electrónica Y Telecomunicaciones
Departamento De Telecomunicaciones
Grupo I+D Nuevas Tecnologías En Telecomunicaciones - GNTT
Popayán
2004

CONTENIDO

pág.

INTRODUCCIÓN

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.

La gestión de redes no es fácil debido a los múltiples y variados problemas que se pueden presentar, debido a que la mayoría de las empresas tienen instalaciones ubicadas en zonas alejadas y sus redes cuentan con un gran número de estaciones de usuario. Por esto, es indispensable tener herramientas que permitan gestionar las redes de una forma integrada y efectiva, pero la mayoría de ellas son complejas y costosas.

Teniendo en cuenta lo anterior, el grupo de I+D de Nuevas Tecnologías en Telecomunicaciones GNTT, específicamente su área Gestión Integrada de Sistemas de Telecomunicaciones, busca obtener una "Plataforma Integral para la Gestión de Redes de Datos con Interfaz Web" que sea fácil de usar, económica y se adapte a las necesidades de cualquier red. Con el fin de contribuir a la consecución de este objetivo, el presente trabajo se enfoca en la obtención de una aplicación para gestionar estaciones de usuario en una forma óptima.

Actualmente los estándares del DMTF por ser flexibles y potentes son una excelente opción para la administración de redes, además, han sido desarrollados por empresas líderes del sector de las telecomunicaciones, como 3COM Corporation, Cisco, Dell Computer Corp., Hewlett Packard Company, IBM/Tivoli Systems Inc., Intel Corporation, Microsoft Corporation, Sun Microsystems Inc., entre otras, para unificar la gestión.

En este documento se realiza una descripción detallada de los estándares del DMTF, que muestra que WBEM y CIM son la mejor opción para realizar gestión. Además, con el fin de demostrar las ventajas de utilizar estos dos estándares, se establece una comparación de ellos con las infraestructuras de gestión más importantes y utilizadas actualmente.

También se describen los principales aspectos de las implementaciones WBEM existentes, y se realiza un estudio de opciones, teniendo en cuenta que el principal objetivo del proyecto es desarrollar una aplicación flexible, escalable, portable e interoperable, para gestionar en forma remota equipos Windows, de los que consta en su mayoría cualquier empresa.

Finalmente se presentan los requerimientos y las consideraciones de diseño e implementación que se consideraron para el desarrollo de la aplicación, teniendo en cuenta que su campo de aplicación es la Red de Datos de la Universidad del Cauca, lo que sirve tanto de requerimiento como de validación funcional. Además se muestran las posibles extensiones de este proyecto.

Cabe anotar que con base en este trabajo de grado se realizaron tres artículos que fueron aceptados para ser presentados en tres eventos internacionales, lo que demuestra la trascendencia e importancia de los temas tratados en este proyecto.

1. ESTÁNDARES DEL DMTF

En este capítulo se hace una descripción de uno de los principales organismos de estandarización, el DMTF, y de los principales aspectos de sus diferentes estándares, los cuales son de gran importancia actualmente, ya que están ganando cada vez más, gran aceptación en el mercado. Finalmente se muestra el campo de aplicación y la integración de estos estándares en un entorno gestionado.

1.1 DMTF

El Grupo de Trabajo de Gestión Distribuida, DMTF (Distributed Management Task Force) es un cuerpo colaborativo sin ánimo de lucro, neutral al vendedor, que está dirigiendo el desarrollo, la adopción y la unificación de estándares e iniciativas de gestión para escritorio, entornos de empresa e Internet. El DMTF se caracteriza por adoptar, crear y mantener las especificaciones y las tecnologías que proporcionan herramientas de gestión con capacidad de descubrir, desplegar y controlar datos de gestión en una manera estándar. Al trabajar con los vendedores de tecnologías claves y los grupos de estándares afiliados, el DMTF está permitiendo una aproximación integrada, efectiva en costos para la gestión a través de soluciones interoperables.

La misión del DMTF es dirigir el desarrollo de estándares de gestión para entornos de escritorio, de red, de empresa e Internet, distribuidos.

Las metas del DMTF son:

1. Acelerar la adopción de estándares de gestión.
2. Unificar las iniciativas de gestión de la industria.
3. Promover la interoperabilidad entre los proveedores de soluciones de gestión.

El DMTF está compuesto por un gran número de compañías involucradas en la escenario de la gestión de redes. Algunos de los miembros del DMTF son Sun Microsystems Inc., Microsoft Corporation, Cisco Systems Inc., Intel Corporation, 3Com Corporation y Compaq Computer Corporation.

Los estándares del DMTF son CIM, WBEM, DEN, DMI, ASF y SMBIOS, y de describen a continuación.

1.2 CIM

El Modelo de Información Común, CIM (Common Information Model) es un modelo de información para describir entidades computacionales y de negocios, es decir, es un modelo para describir toda la información de gestión en un ambiente de red/empresa. Debido a que CIM es un modelo conceptual, no está unido a una implementación específica, por tanto permitirá a las aplicaciones de diferentes desarrolladores en diferentes plataformas, describir datos de gestión en un formato estándar para que se puedan compartir entre una variedad de aplicaciones de gestión. CIM intenta además, unificar y extender la instrumentación y los estándares de gestión existentes (SNMP, DMI, CMIP, etc.).

CIM está conformado por una Especificación y un Esquema. La Especificación CIM describe el lenguaje, el nombramiento, el Meta Esquema y las técnicas de mapeo a otros modelos de gestión tales como MIBs SNMP. Mientras que el Esquema CIM proporciona las descripciones del modelo real.

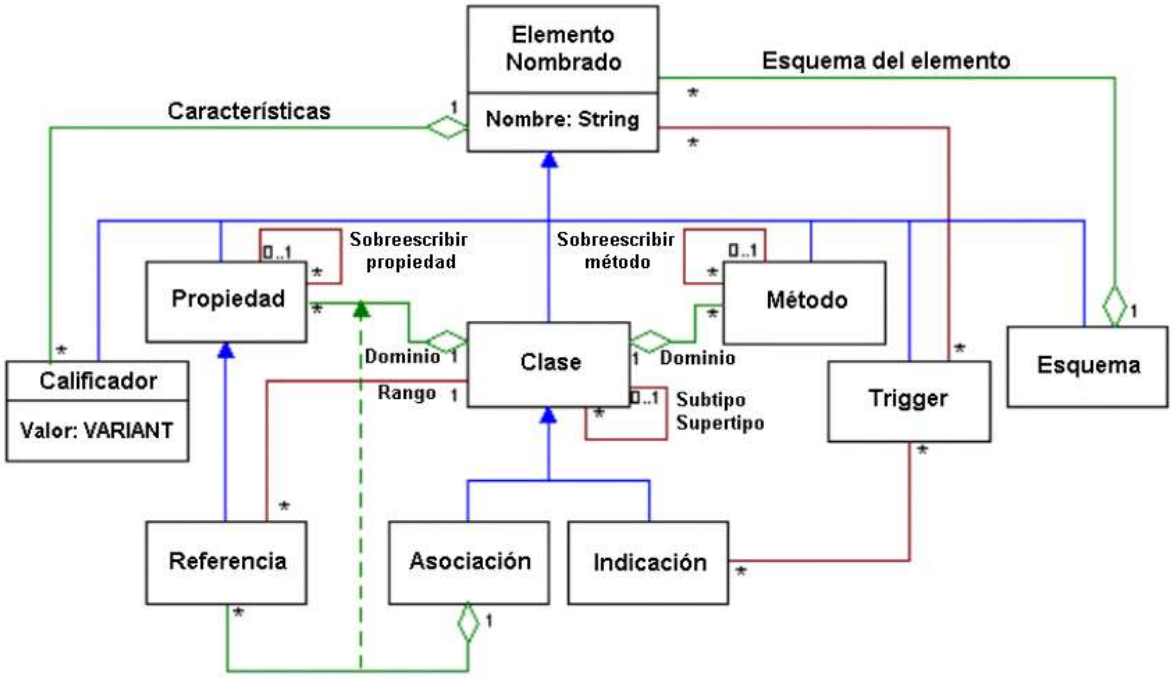
CIM utiliza técnicas orientada a objetos. El lenguaje estándar utilizando para definir los elementos de CIM es el Formato de Objetos Gestionados, MOF (Managed Object Format), además los modelos CIM se representan en UML (Uniform Modelling Language) y en XML (Extensible Markup Language).

1.2.1 La especificación CIM. Describe un meta-modelo orientado a objetos basado en UML. Este modelo incluye expresiones para elementos comunes que se deben presentar claramente a las aplicaciones de gestión (por ejemplo, clases, propiedades, métodos y asociaciones).

El Meta Esquema CIM. Es una definición formal del modelo. Define los términos utilizados para expresar el modelo, su uso y semánticas.

Los elementos del modelo son: Esquemas, Clases, Propiedades y Métodos. El modelo también soporta Indicaciones y Asociaciones como tipos de Clases y Referencias como tipos de Propiedades, como se muestra en la figura 1.1.

Figura 1.1. Estructura del Meta Esquema CIM



- **Esquema.** Es un grupo de clases con un solo propietario. Los Esquemas se utilizan para la administración y el nombramiento de clases. Los nombres de las clases deben ser únicos dentro de sus propios esquemas. Cada nombre de clase incluye el nombre del esquema.
- **Clase.** Es un modelo o prototipo, que define las propiedades y los métodos comunes a una clase particular de objeto. Cada clase CIM es un modelo para un tipo de elemento gestionado. Las clases contienen propiedades que describen los datos de la clase, y métodos, que describen el comportamiento de la clase. Una clase está limitada por el esquema al cual pertenece. Una clase debe pertenecer a solamente un esquema y el nombre de la clase debe ser único para ese esquema. Un nombre de clase calificado totalmente incluye el nombre del esquema. Las clases pueden ser jerárquicas pero una jerarquía no soporta herencia múltiple.
- **Propiedad.** Es un valor utilizado para denotar una característica de una clase. Las propiedades están limitadas por la clase que tiene la propiedad y deben ser únicas dentro de la clase. Una propiedad tiene un nombre, un tipo de dato, un valor y un valor por defecto opcional.
- **Método.** Es una operación que se puede invocar. Los métodos están limitados por la clase que tiene el método y deben ser únicos dentro de la clase. Una clase puede tener cero o más métodos. La declaración de un método incluye un nombre, un tipo de retorno, parámetros de entrada opcionales y parámetros de salida opcionales.
- **Calificador.** Proporcionan información adicional acerca de clases, asociaciones, indicaciones, métodos, parámetros de métodos, propiedades o referencias. Un calificador no se puede utilizar sin una definición de tipo de calificador y el calificador debe concordar con su tipo de calificador, esto

es, el tipo de dato y el valor deben corresponder con el tipo de calificador. Los calificadores están limitados por la "namespace"[1] en la cual ellos están presentes y la definición de tipo de calificador debe ser única dentro de esa "namespace". Todos los calificadores tienen un nombre, un tipo, un valor, un alcance, un "flavor"[2] y un valor por defecto opcional.

El flavor define un comportamiento adicional para los calificadores. Por ejemplo, los calificadores se pueden transmitir automáticamente desde las clases a las clases derivadas o se pueden restringir a la clase para la cual ellos fueron definidos. Los calificadores también se pueden definir para permitir o no, que las clases derivadas puedan sobrescribir el valor del calificador, o se puede determinar si el calificador se debe fijar para una jerarquía completa de clases.

El alcance define los meta-elementos a los que el calificador se puede aplicar. El alcance debe contener al menos un meta-elemento y puede contener una combinación de meta elementos, o se puede aplicar a todos los meta-elementos. El alcance puede incluir los siguientes meta-elementos: Clase, Asociación, Indicación, Propiedad, Referencia, Método y Parámetro.

- **Referencia.** Es un tipo de dato especial de propiedad que indica que es un puntero a otras instancias. Una referencia define el rol que cada objeto cumple en una asociación. Las asociaciones soportan múltiples instancias de relación para un objeto dado. En otras palabras, un sistema se puede relacionar con muchos componentes de un sistema en muchos modos diferentes. Una referencia también se puede utilizar como un parámetro de un método.

- **Asociación.** Es un tipo de clase que contiene dos o más referencias. Las asociaciones representan relaciones entre dos o más clases. Debido a que las asociaciones son clases, ellas establecen una relación entre clases sin afectar ninguna de las clases relacionadas. En otras palabras, la adición de una asociación no tiene efecto sobre ninguna de las clases relacionadas. Una asociación no puede ser una subclase de una clase no asociación. En CIM una asociación es un objeto separado con referencias unidas a ella.

- **Trigger.** Es un reconocimiento del cambio del estado de la instancia de una clase.

- **Indicación.** Una indicación es la representación activa de la ocurrencia de un evento. Debido a que las indicaciones son tipos de clases, pueden tener propiedades y métodos, y se pueden arreglar en una jerarquía. Las instancias de una indicación son transitorias y no se pueden obtener al utilizar operaciones CIM. Las indicaciones solamente se pueden recibir al suscribirse a ellas.

  Existen dos tipos de indicaciones:

  - **Indicaciones de Ciclo de Vida**. Eventos de ciclo de vida de una clase o instancia CIM.
    - ✓ **Clases.** Creación, eliminación y modificación de una clase.
    - ✓ **Instancias.** Creación, eliminación, modificación, invocación de un método y acceso de lectura a una instancia.
  - **Indicaciones de proceso.** Notificaciones de alerta asociadas con objetos que pueden o no estar completamente modeladas en CIM, o no corresponden a un evento de ciclo de vida simple, como alertas de instrumentación de bajo nivel, alertas DMI, traps SNMP y eventos TMN.

  Una suscripción se expresa por la creación de una instancia de una asociación que referencia una instancia de un Filtro y una instancia de un Manejador de Indicación. El Filtro contiene la "query"[3] que selecciona una clase o clases Indicación. El tamaño y la complejidad del resultado entregado al suscriptor se determina por la query.

---

[1] Namespace, conjunto de clases. Se prefiere su uso en Inglés.
[2] Flavor, propiedad de los calificadores. Se prefiere su uso en Inglés.
[3] Query, requerimiento. Se prefiere su uso en Inglés.

MOF. Existen potencialmente muchos modos en los cuales la información de gestión CIM se podría representar para poder intercambiarla. La especificación CIM define un lenguaje basado en el Lenguaje de Definición de Interfaces, IDL (Interface Definition Language), llamado Formato de Objeto Gestionado, MOF (Managed Object Format).

Los principales componentes de una especificación MOF son descripciones textuales de clases, asociaciones, propiedades, referencias, métodos y declaraciones de instancias y sus calificadores asociados. También se permite comentarios. El archivo MOF básicamente está compuesto por una serie de declaraciones de clases e instancias. MOF no es requerido por la especificación CIM pero es aceptado como el lenguaje de la especificación CIM en la mayoría de implementaciones CIM y WBEM disponibles hoy.

El modo regular de definir nuevas clases o instancias CIM es primero escribirlas en MOF y en seguida utilizar un compilador MOF que coloca esta nueva información en CIM para que pueda ser utilizada por un CIMOM o una aplicación cliente WBEM

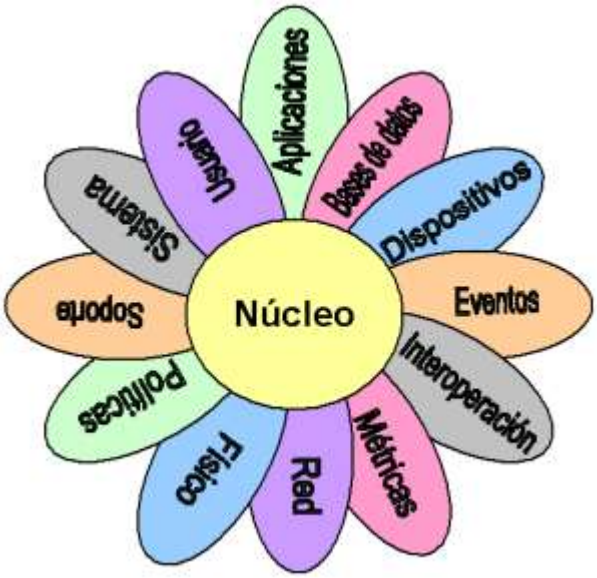
UML. El DMTF utiliza el Lenguaje de Modelado Unificado, UML (Unified Modeling Language) para respresentar el modelo núcleo y el modelo común de CIM. UML es una especificación estándar del OMG y es el lenguaje estándar de la industria para especificar, visualizar, construir y documentar modelos, incluyendo su estructura y diseño.

En UML, una clase se representa por un rectángulo que contiene el nombre de la clase. Una clase con propiedades se representa por un rectángulo dividido en dos regiones, una que contiene el nombre de la clase y la otra una lista de propiedades. Los métodos se representan por un tercera región que contiene la lista de métodos. La herencia, o una relación subclase/superclase, se representa por una línea dibujada entre la subclase y la superclase con una flecha indicando la superclase. Las asociaciones se representan por líneas con el nombre de la asociación usualmente colocado cerca del centro de la línea, y las agregaciones se representan por líneas con un diamante en el extremo que está agregando.

Los documentos del modelo CIM generalmente siguen la convención de utilizar líneas azules para herencia, líneas rojas para asociaciones y líneas verdes para agregaciones. La codificación con colores hace a los diagramas grandes mucho más fáciles de leer pero no es parte de UML estándar.

1.2.2 Esquema CIM. Los esquemas de gestión son los bloques de construcción para las plataformas de gestión y las aplicaciones de gestión, tales como configuración de dispositivo, gestión de desempeño y gestión de cambio. CIM está estructurado en un modo tal que el entorno gestionado se puede ver como una colección de sistemas interrelacionados, cada uno de los cuales está compuesto por un número de elementos discretos. El esquema CIM proporciona un conjunto de clases con propiedades y asociaciones que proporcionan una infraestructura conceptual entendible dentro de la cual es posible organizar la información disponible acerca del entorno gestionado. El Esquema CIM es la combinación del Modelo Núcleo y del Modelo Común como se muestra en la Figura 1.2.

Figura 1.2 Esquema CIM



Modelo núcleo. El modelo núcleo captura nociones que son aplicables a todas las áreas de gestión y está compuesto por un conjunto de clases, asociaciones y propiedades que proporcionan un vocabulario básico para describir los sistemas gestionados. El modelo núcleo representa un punto de partida para determinar cómo extender el esquema común.

Se supone que el Esquema Núcleo es estable y no se espera que DMTF haga modificaciones o eliminaciones de este esquema. Puede haber adiciones al Esquema Núcleo pero no se espera que tengan efectos muy graves en los elementos WBEM existentes.

El modelo común. Es realmente un conjunto de modelos de información que capturan nociones que son comunes a áreas de gestión particulares, pero independientes de cualquier tecnología o implementación particular. Ejemplos de modelos comunes son: sistemas, aplicaciones, redes y dispositivos. Se pretende que las clases, propiedades, asociaciones y métodos en los modelos comunes proporcionen una vista del área lo suficientemente detallada, como para que se pueda utilizar como una base para el diseño de un programa y en algunos casos la implementación.

Debido a que los modelos comunes se pueden extender, ellos pueden ofrecer cada vez más, un rango mayor de información. Los modelos núcleo y común en conjunto se llaman esquema CIM.

El DMTF está trabajando constantemente en nuevos modelos comunes Los modelos comunes para la versión actual del Esquema, CIM 2.8 son: Aplicaciones, Base de datos, Dispositivos, Eventos, Interoperación, Métricas, Red, Físico, Política, Soporte, Sistemas y Usuario

No se requiere que los desarrolladores tomen los modelos comunes completos, ellos pueden utilizar las jerarquías apropiadas a sus necesidades de desarrollo.

A continuación se describen los modelos comunes del Esquema CIM.

- **Modelo de Aplicaciones.** Describe la información comúnmente requerida para desplegar y gestionar productos y aplicaciones software. Este modelo está basado en la necesidad de gestionar el ciclo de vida y la ejecución de aplicaciones. Puede describir cualquier tipo de aplicaciones, desde aplicaciones de escritorio aisladas, hasta una aplicación sofisticada, multi-plataforma, distribuida, basada en Internet. Además, se puede modelar tanto un producto software simple como un grupo de productos software.

- **Modelo de Bases de Datos.** Define los componentes de gestión para un entorno de base de datos. Conceptualmente, existen tres entidades importantes que se modelan.

  - El sistema de la base de datos, que representa los aspectos de las aplicaciones software del ambiente de la base de datos.
  - La base de datos común, que es una entidad lógica que representa la unidad de datos interrelacionados, organizados.
  - El servicio de la base de datos, que representa el proceso o los procesos que realizan tareas para la base de datos, como proporcionar acceso al usuario.

  Además, hay un número de clases de soporte que representan parámetros de configuración, recursos y estadísticas.

- **Modelo de Dispositivos.** Describe la funcionalidad proporcionada por el hardware, y proporciona datos de estado y configuración. El modelo cubre una amplia variedad de elementos y maneja conceptos de bajo nivel tales como sensores, baterías, ventiladores, y abstracciones de alto nivel tales como volúmenes de almacenamiento.

- **Modelo de Eventos.** Típicamente se asume que un evento es un cambio en el estado del entorno o un registro del comportamiento de algún componente del entorno. En el contexto de esta especificación, la indicación concreta de la ocurrencia de un evento se representa por una instancia de la clase indicación.

  Los tipos de indicaciones (que representan diferentes tipos de eventos) se denotan por subclases de la clase indicación. Estas incluyen:

  - **Indicación de instancia.** Para modelar eventos del ciclo de vida CIM, como creación, eliminación, modificación, invocación de métodos y acceso de lectura a una instancia.
  - **Indicación de clase.** Para eventos del ciclo de vida del Esquema CIM, como creación, eliminación y modificación de una clase.
  - **Indicación de proceso.** Para notificaciones de alerta asociadas con objetos que pueden o no estar completamente modelados en CIM, o no corresponden a un evento de ciclo de vida simple, tales como alertas de instrumentación de bajo nivel, alertas DMI, traps SNMP y eventos TMN.

  Una idea fundamental en CIM para la representación de indicaciones es la separación de la publicación de la indicación y la suscripción de la indicación. La publicación de eventos implica la creación de instancias filtro. Una suscripción se expresa por la creación de una instancia de una asociación de indicación de la suscripción, que asocia una instancia filtro de indicación, y una instancia manejadora de indicación. El filtro contiene la query que selecciona una clase o clases indicación. El tamaño y la complejidad del resultado entregado al suscriptor se determina por la "query". Una instancia de una subclase manejadora de indicación se utiliza para especificar el destino que va a recibir el flujo de indicación asociado.

- **Modelo de Interoperación.** Define la gestión de componentes que describen la infraestructura WBEM y cómo otros componentes WBEM, tales como proveedores y adaptadores de protocolo, interactúan con la infraestructura. La infraestructura WBEM se puede describir como sigue:

- **Cliente CIM.** Interactúa con un Servidor CIM al emitir Requerimientos Mensaje Operación CIM y recibe y procesa Respuestas Mensaje Operación CIM.
- **Servidor CIM.** Un servidor que recibe y procesa Requerimientos Mensaje Operación CIM y emite Respuestas Mensaje Operación CIM.
- **CIMOM.** El Manejador de Objetos CIM, CIMOM (CIM Object Manager), es el componente central del Servidor CIM, responsable de la comunicación entre los componentes del Servidor CIM.
- **Proveedor.** Instrumenta uno o más aspectos del esquema CIM

El Modelo de Interoperación CIM se divide en los siguientes submodelos. Cada submodelo describe un área de interés particular.

- **Modelo del CIMOM.** Describe la infraestructura WBEM y sus relaciones, además, de los mecanismos de acceso que soporta el CIMOM, las capacidades del CIMOM e incluso proporciona datos estadísticos básicos basados en Operaciones CIM.
- **Modelo de Namespace.** El Modelo de Namespace define las "namespaces" que soporta el CIMOM así como también la información de tipo que está en cada "namespace".
- **Modelo del Proveedor.** Describe un proveedor y sus capacidades, las cuales incluyen la clase, las propiedades y los métodos que el proveedor soporta. El modelo también define el mecanismo en el cual se requiere que un proveedor se registre con el CIMOM.
- **Modelo de Adaptador de Protocolo.** Un adaptador de protocolo es algo que acepta información utilizando un protocolo particular y convierte esa información de tal forma que se pueda utilizar nativamente, por ejemplo CIM-XML o CIM-SOAP.

  El Modelo de Adaptador de Protocolo describe la información del adaptador de protocolo y permite al administrador hacer "queries" a adaptadores de protocolos, configurarlos (por ejemplo información del puerto), iniciarlos y detenerlos.

❖ **Modelo de Métricas.** Define los componentes de gestión que permiten la definición y la recuperación dinámica de información sobre métricas. Utiliza un patrón basado en una clase CIM de definición de métrica, que especifica las semánticas y el uso de una métrica (sus meta-datos), y otra clase (una clase "valor" de métrica CIM), que contiene los valores de los datos capturados por una instancia particular de la clase de definición de métrica.

Originalmente, este patrón de definición/valor de métrica fue definido para gestionar la información del tiempo de respuesta de una transacción (por ejemplo, tiempos de respuesta al cliente y cómo una transacción fluye a través de un sistema), y proporcionar información adicional acerca de la transacción (por ejemplo, identificación, procesamiento o información de la utilización de recursos). El concepto de tiempo de respuesta de una transacción fue generalizado al concepto de unidad de trabajo. Las clases unidad de trabajo miden el tiempo para que alguna acción se realice, y pueda unir otras métricas para proporcionar información adicional. Aunque este concepto fue definido originalmente para la medida del tiempo de respuesta de una transacción, el concepto de unidad de trabajo es bastante general para manejar una variedad de entidades de "runtime" que requieren la medida del tiempo entre el inicio y el fin de una actividad (por ejemplo, tiempos de procesamiento por lotes).

❖ **Modelo de Red.** Describe y gestiona la conectividad de comunicaciones y la de red, así como también los servicios y protocolos individuales en la red. Las entidades gestionadas en el modelo se pueden agrupar en categorías amplias que describen:

1. Sistemas de red (con un inventario asociado, información de usuario y de seguridad, etc.).
2. Servicios de red (por ejemplo, enrutamiento).
3. Interconexión y acceso lógicos (por ejemplo, endpoints de protocolos, rutas y conductos de red). Aplicable a tanto los sistemas de red como de cliente.
4. Protocolos de red (tales como OSPF y BGP).

5. Tecnologías de networking (por ejemplo, Switching/Bridging y VLANs).
6. Tecnologías de calidad de servicio (QoS) (tales como medidores, marcadores y colas).
7. Otras definiciones de soporte (por ejemplo, diferentes criterios de filtrado de paquetes de red).

En conclusión, el modelo de red describe y gestiona ampliamente la conectividad general entre sistemas, así como también detalles específicos de la tecnología de red y del protocolo. Él cubre no solamente los aspectos de configuración y de estado, sino que también define las estadísticas que se pueden colectar de los elementos de red como soporte a las aplicaciones de gestión de desempeño.

- **Modelo Físico.** Describe la información relacionada con el inventario físico y la gestión activa, describiendo tarjetas, componentes físicos e información de cableado. Los elementos físicos ocupan espacio y cumplen las leyes elementales de la Física. Ellos representan cualquier elemento que tenga una identidad física –por ejemplo, que se pueda tocar o ver. Las relaciones entre elementos físicos se definen como asociaciones en el modelo y tratan principalmente con contención y localización.

  Es importante recordar que las abstracciones en el modelo físico típicamente representan la estructura física de un sistema. Ellas no representan la funcionalidad que los ítems físicos son capaces de proporcionar. Esta funcionalidad está representada por las abstracciones del lado lógico del modelo –usualmente como subclases de dispositivos lógicos o como servicios en un sistema computacional.

- **Modelo de Políticas.** Cuando se utiliza en el contexto de sistemas computacionales, política es un término que se utiliza frecuentemente para describir cualquier configuración de un sistema, que controla sus comportamientos, como “políticas de seguridad” o “políticas de calidad de servicio”. El Modelo de Políticas, desarrollado en conjunto por el IETF y el DMTF, es un modelo que permite construir reglas de políticas de la forma condición-acción.

  El Modelo de Políticas del DMTF proporciona una estructura común para especificar comportamientos de un sistema, que son lo suficientemente abstractos como para ser independientes de los detalles específicos de la implementación y lo suficientemente escalables como para configurar grandes complejidades de los sistemas computacionales.

- **Modelo de Soporte.** El Estándar de Resolución de Problemas, PRS (Problem Resolution Standard), es la fusión de dos estándares previos del DMTF y el CSI (Consortium for Service Innovation) conocidos como SES (Solution Exchange Standard) y SIS (Service Incident Exchange Standard). El principal propósito de PRS es definir un estándar abierto que facilite el intercambio de soluciones entre las partes cooperantes, tanto dentro de una organización como a través de las fronteras organizacionales.

- **Modelo de Sistema.** Define las abstracciones relacionadas con computador-sistema. Muchos de los conceptos relacionados con el sistema computacional se derivan de la abstracción de sistema en el Modelo Núcleo. El sistema CIM describe la agregación de ‘partes’ (o componentes) dentro de un ‘todo’ gestionable simple (el sistema).

  Además del concepto de sistema computacional por si mismo, el modelo de sistema también maneja los componentes y la funcionalidad asociada con la mayoría de sistemas computacionales. Estos incluyen conceptos tales como sistemas de archivos y archivos, sistemas operativos, tareas, procesos e hilos, y diagnósticos. Además, tanto los sistemas de propósito general como los dedicados se pueden describir. No hay subclases específicas para describir la funcionalidad del sistema (enrutamiento, almacenamiento, arreglo de almacenamiento, etc.). La funcionalidad de estos sistemas se describe por los servicios que se tienen o se pueden tener.

- **Modelo de Usuario/Seguridad.** Los enfoques de este modelo son:

- Contacto general e información de páginas blancas para organizaciones, unidades organizacionales y personas.
- Usuarios de servicios y la información de seguridad relacionada para autenticar y autorizar esos usuarios.

Esquema de extensión. Los esquemas de extensión representan extensiones de los modelos comunes específicas a la tecnología. Estos esquemas son específicos a entornos, tales como sistemas operativos.

Estos esquemas dan a CIM una jerarquía de tres niveles donde el nivel central es estable, el nivel de la mitad solamente puede ser cambiado por el DMTF y el tercer nivel es específico al vendedor.

1.3 WBEM

Uno de los modos en los cuales el DMTF está logrando la meta de interoperabilidad, se llama WBEM (Web Based Enterprise Management). WBEM es el acrónimo para la Gestión de Empresa Basada en Web, un conjunto de tecnologías estándar de gestión e Internet desarrolladas para unificar la gestión de los ambientes computacionales de empresa. WBEM proporciona a la industria la capacidad de entregar un conjunto bien integrado de herramientas de gestión basadas en estándares, impulsando la utilización de tecnologías web emergentes. En otras palabras, WBEM es un estándar extremadamente versátil y generalizado que se puede aplicar casi en cualquier área de gestión de redes, como por ejemplo sistemas operativos, redes de computadores, dispositivos hardware, aplicaciones software etc., además ha recibido el respaldo de la mayor parte de la industria de gestión de redes.

Típicamente, diferentes protocolos e interfaces de gestión se han utilizado para diferentes tareas de gestión de empresa. Por ejemplo, SNMP se ha utilizado para la gestión de redes, DMI para la gestión de sistemas de escritorio, y así sucesivamente. La meta de WBEM es proporcionar la habilidad para gestionar todos los sistemas independientemente del tipo de instrumentación. Una interfaz de gestión común para todos los sistemas simplifica mucho la gestión y también reduce sus costos significativamente. Sin embargo, reemplazar todos los estándares de gestión anteriores con WBEM no es realista. Por lo tanto, WBEM puede trabajar junto con los estándares existentes actualmente y no hay necesidad de reemplazarlos.

1.3.1 Historia. WBEM fue iniciado por primera vez en 1996 por un grupo de compañías encabezadas por Microsoft, Compaq, Cisco Systems, BMC Software e Intel. La visión fue definir un entorno abierto para la gestión, donde todos los sistemas y las aplicaciones de gestión pudieran acceder, controlar y compartir información de gestión con otras. Esto se debería hacer utilizando tecnologías y estándares existentes.

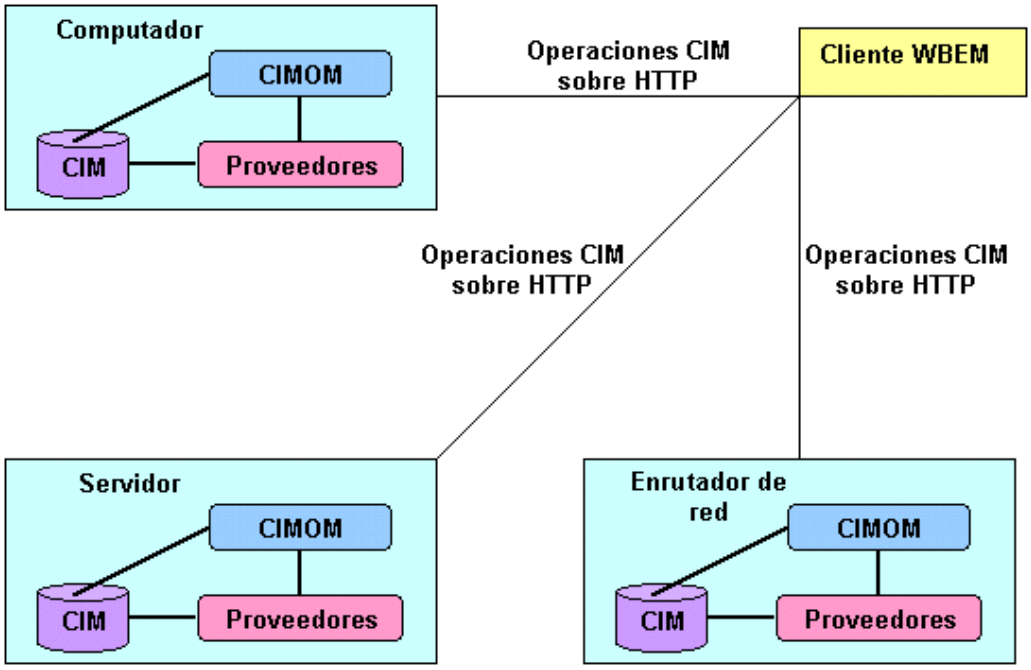
Se desarrollaron algunas especificaciones independientes del entorno para describir y acceder datos de gestión, teniendo en mente estándares como SNMP y DMI. HMMS (HyperMedia Management Schema) fue la especificación para modelar y describir datos de gestión, la cual es ahora el estándar CIM del DMTF. También se propuso una especificación para transportar los datos de gestión, HMMP (HyperMedia Management Protocol), un protocolo de transporte completamente nuevo para utilizar con WBEM, pero fue rechazado por el Grupo de Trabajo de Ingeniería en Internet, IETF (Internet Engineering Task Force), y se decidió utilizar los encabezamientos de extensión de HTTP, así que no hubo necesidad de un protocolo específico para WBEM. Esto se hizo con el fin de asegurar una difusión más grande de las aplicaciones WBEM, ya que la industria ya está familiarizada con HTTP y no tiene que aprender un nuevo protocolo. En el mismo modo, XML, el lenguaje que utiliza WBEM, está difundiéndose rápidamente y ya existen muchas aplicaciones soportándolo.

En 1998 el DMTF se encargó del desarrollo de la iniciativa WBEM, y la especificación HMMS (anterior a CIM) se mantuvo como su parte principal. Entonces, el DMTF acordó mantener la neutralidad de las iniciativas WBEM y por lo tanto se abstuvo de especificar cualquier dependencia de la implementación con respecto a lenguajes de programación y sistemas operativos. La especificación del DMTF para

mapear CIM a XML se lanzó en septiembre de 1998 y un año más tarde se publicó la Especificación de Operaciones CIM sobre HTTP.

1.3.2 Componentes de WBEM. Los principales componentes de WBEM son el CIMOM, los Proveedores WBEM y los Clientes WBEM, que se pueden ver en la Figura 1.3. Un proveedor WBEM se puede ver como la interfaz entre los recursos gestionados y el CIMOM, y el Cliente WBEM se puede ver como la interfaz entre el gestor y el CIMOM. El CIMOM es responsable de llamar el(los) proveedor(es) correcto(s) en respuesta a los requerimientos del cliente y también de llamar al(los) cliente(es) correcto(s) cuando un evento ocurre.

Figura 1.3 Arquitectura WBEM



CIMOM. Es la parte central de WBEM. Tiene un repositorio donde almacena todos los Esquemas CIM, así que puede verificar que los datos enviados por los clientes o proveedores sean correctos. El repositorio también se puede utilizar para almacenar datos de instancias CIM creadas por los clientes o proveedores. Éstos datos se llaman datos estáticos. De otro lado, los datos dinámicos son datos que los proveedores consiguen de los recursos gestionados directamente. Si un cliente quiere modificar o acceder datos estáticos, el CIMOM solo modifica o accede su repositorio, pero si los datos son dinámicos el CIMOM llama al proveedor correcto, el cual a su vez modifica o accede el recurso.

Proveedores WBEM. Se pueden ver como las interfaces entre el recurso gestionado y el CIMOM. Los datos que son proporcionados por un proveedor se llaman datos dinámicos. Cuando el CIMOM requiere datos dinámicos del proveedor, el proveedor consigue los datos del recurso gestionado y los retorna al CIMOM.

Normalmente los proveedores residen en el mismo computador que el CIMOM, y a diferencia de la comunicación entre el Cliente WBEM y el CIMOM, no hay una interfaz estándar entre los proveedores y el CIMOM. Esto se ve como un problema y se están haciendo esfuerzos por estandarizar una interfaz proveedor.

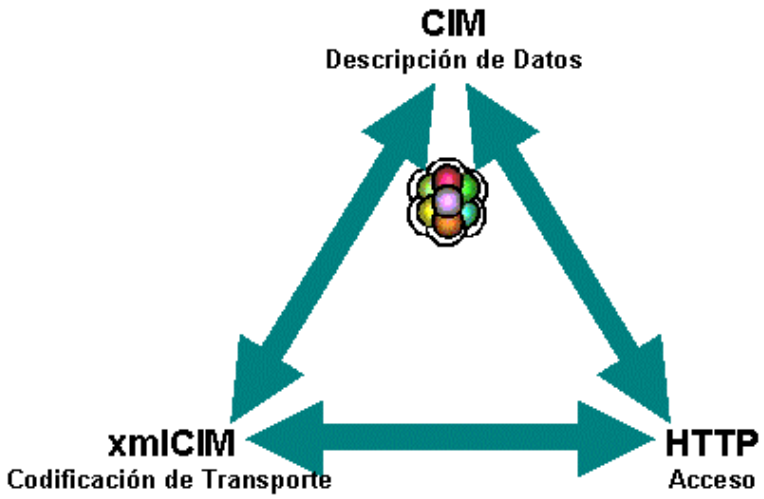
Clientes WBEM. Se puede ver como la interfaz entre el gestor y el CIMOM. Operaciones CIM sobre HTTP, es el estándar entre el cliente y el CIMOM, definido por el DMTF. Sin embargo, la mayoría de

implementaciones también soportan otros mecanismos para la comunicación, como por ejemplo, RMI (Remote Method Invocation) para implementaciones Java, DCOM para la implementación de Microsoft e IPC (Inter Process Communication) para implementaciones UNIX. Sin embargo, el uso de Operaciones CIM sobre HTTP garantiza la compatibilidad entre cualquier cliente y cualquier CIMOM.

Un Cliente WBEM también se puede suscribir por si mismo con el CIMOM para un evento especial, entonces el CIMOM notificará al cliente cuando un evento ocurre. Este mecanismo se soporta en la especificación Operaciones CIM sobre HTTP.

1.3.3 Estándares WBEM. WBEM tiene tres componentes principales que se muestran en la Figura 1.4. CIM (Common Information Model) proporciona un formato, un lenguaje y una metodología comunes para colectar y describir datos de gestión, la Especificación de Codificación xmlCIM, define elementos XML, escritos en DTD (Document Type Definition), que se pueden utilizar para representar clases e instancias CIM, y la Especificación de Operaciones CIM sobre HTTP, define un mapeo de operaciones CIM sobre HTTP que permite a las implementaciones WBEM interoperar de una manera abierta y estandarizada.

Figura 1.4 Estándares WBEM



xmlCIM. XML es un lenguaje para la representación de información estructurada y no tiene un conjunto de tags estándar como HTML, sino que en su lugar permite al usuario definir sus propias tags. El estándar XML (eXtensible Markup Language) está difundiéndose rápidamente a través del mundo de Internet donde está siendo utillizado en más y más áreas. Existe también una versión XML de HTML llamada XHTML (Extensible HyperText Markup Language).

Debido a la estructura de XML, la información codificada en XML es siempre legible por el humano directamente, mientras mantiene una estructura que es adecuada para ser legible por la máquina. Si una aplicación quiere desplegar la información en un modo específico, entonces todo lo que se tiene que hacer es crear una hoja de estilo XSL (eXtensible Stylesheet Language) que le dirá a una aplicación XML cómo la información va a ser desplegada en pantalla.

La razón para representar CIM en XML fue el hecho de que XML fue y es el principal formato para representar datos estructurados en Internet, y la meta con WBEM fue y es utilizar los estándares existentes basados en Web tanto cómo sea posible.

Un documento XML es una colección de datos representados en XML. Un esquema XML es una gramática que describe la estructura de un documento XML. La meta de xmlCIM es crear una

gramática XML que se pueda escribir en DTD (Document Type Definition) y que se pueda utilizar tanto para representar declaraciones CIM (clases, instancias y calificadores) como mensajes CIM que se van a utilizar por los protocolos CIM.

Existen dos modelos fundamentalmente diferentes para mapear CIM en XML:

- **Mapeo de esquema.** Es aquel en el cual el esquema XML se utiliza para describir las clases CIM, y las instancias CIM se mapean a documentos XML para ese esquema.
- **Mapeo de metaesquema.** Es aquel en el cual el esquema XML se utiliza para describir el metaesquema CIM, y tanto las clases como las instancias CIM son documentos XML para ese esquema.

Aunque existen beneficios obvios para emplear un mapeo de esquema (más potencia de validación y una representación de CIM en XML ligeramente más intuitiva), el mapeo de metaesquema fue adoptado, entre otras razones porque requiere solamente un DTD de metaesquema. El mapeo de esquema requiere un DTD por cada esquema CIM.

Operaciones WBEM. Las operaciones WBEM (como se describen en la Especificación de Operaciones CIM sobre HTTP) definen el conjunto de operaciones que implementa un cliente WBEM para operar en una manera abierta y estandarizada. Ellas son independientes del protocolo.

Todos los requerimientos están definidos como invocaciones de uno o más métodos. Un método puede ser intrínseco o extrínseco. Un método intrínseco está definido por la Especificación de Operaciones CIM sobre HTTP con el propósito de modelar una operación CIM. Además, los métodos intrínsecos están caracterizados porque ellos están en un "namespace" CIM. Un método extrínseco está definido como un método de una clase CIM en algún Esquema CIM. Los métodos extrínsecos se invocan sobre una clase CIM (si son estáticos) o sobre una instancia.

Las Operaciones WBEM pueden ser simples (individuales) o múltiples (por bloques). Ellas son:

- **GetClass.** Retorna una clase CIM simple de la "namespace" objetivo.
- **EnumerateClasses.** Enumera las subclases de una clase CIM en la "namespace" objetivo.
- **EnumerateClassNames.** Enumera los nombres de las subclases de una clase CIM en la "namespace" objetivo.
- **GetInstance.** Retorna una instancia CIM simple de la "namespace" objetivo.
- **EnumerateInstances.** Enumerar instancias de una clase CIM en la "namespace" objetivo.
- **EnumerateInstanceNames.** Enumera los nombres de las instancias de una clase CIM en la "namespace" objetivo.
- **GetProperty.** Obtiene el valor de una propiedad simple de una instancia CIM en la "namespace" objetivo.
- **SetProperty.** Establece el valor de una propiedad simple en una instancia CIM en la "namespace" objetivo.
- **CreateInstance.** Crea una instancia CIM simple en la "namespace" objetivo. La instancia no debe existir.
- **ModifyInstance.** Modifica una instancia CIM existente en la "namespace" objetivo. La instancia ya debe existir.
- **DeleteInstance.** Borra una instancia CIM simple de la "namespace" objetivo.
- **CreateClass:** Crea una clase CIM simple en la "namespace" objetivo. Esta clase no debe existir.
- **ModifyClass.** Modifica una clase CIM existente en la "namespace" objetivo. La clase ya debe existir.
- **DeleteClass.** Borra una clase CIM simple de la "namespace" objetivo.
- **Associators.** Enumera los objetos CIM (clases o instancias) que están asociadas a un objeto CIM de una fuente particular.
- **AssociatorNames.** Enumera los nombres de los objetos CIM (clases o instancias) que están asociados a un objeto CIM de una fuente particular.

- **References.** Enumera los objetos asociación que se refieren a un objeto CIM objetivo particular (clase o instancia).
- **ReferenceNames.** Enumera los nombres de los objetos asociación que se refieren a un objeto CIM objetivo particular (clase o instancia).
- **ExecQuery.** Ejecuta una "query" en una "namespace" objetivo.
- **GetQualifier.** Obtiene la declaración de un calificador simple de la "namespace" objetivo.
- **SetQualifier.** Crea o actualiza la declaración de un calificador simple en la "namespace" objetivo. Si la declaración del calificador ya existe se sobrescribe.
- **DeleteQualifier.** Borrar la declaración de un calificador simple de la "namespace" objetivo.
- **EnumerateQualifiers.** Enumera las declaraciones de los calificadores de la "namespace" objetivo.

Operaciones CIM sobre HTTP. Operaciones CIM sobre HTTP es una especificación de cómo intercambiar información CIM sobre el protocolo HTTP. La ventaja al utilizar este estándar, es que todas las implementaciones cliente WBEM que lo utilizan, se pueden comunicar con cualquier CIMOM que cumpla con las Operaciones CIM sobre HTTP.

Toda la información CIM que se intercambia entre los clientes y los servidores, se llama mensajes CIM. Estos mensajes son independientes del protocolo, por lo tanto se podrían enviar sobre cualquier protocolo. Sin embargo, HTTP fue seleccionado para la encapsulación de los mensajes CIM debido su amplio uso en Internet. HTTP corre sobre TCP/IP y por lo tanto también es confiable. Además se puede utilizar HTTPs para una mayor seguridad.

Los mensajes CIM son por ellos mismos documentos XML ordinarios que siguen el DTD CIM-XML. La especificación de Operaciones CIM sobre HTTP define todas las operaciones CIM permitidas en los mensajes CIM, y también define algunos encabezamientos de extensión de HTTP que se utilizan con los mensajes CIM y que les dicen a las aplicaciones cliente y servidor que se está enviando información CIM.

Ya que WBEM envía sus mensajes sobre HTTP, hace uso de la funcionalidad subyacente de TCP. Esto significa que todas las transmisiones son confiables en el sentido de que el que envía sabrá si la transmisión fue recibida correctamente o no. El que envía hace múltiples intentos para conseguir los mensajes si no se recibe ninguna confirmación.

WBEM por si mismo no ofrece ninguna característica de seguridad específica, pero el uso de HTTP proporciona un número de opciones de seguridad que son ampliamente aceptadas y probadas en implementaciones regulares. Estas opciones de seguridad abarcan desde no tener seguridad en lo absoluto, al uso de Secure HTTP. A continuación se presenta una corta descripción de los diferentes modos de autenticación.

- **Autenticación Básica**. Está basado en el paradigma "challenge-response"[4]. En este esquema, cada vez que un usuario intenta acceder un recurso, se le solicita un login y un password. Después de que el usuario proporciona la información requerida, se valida con un registro. Si la información es válida, el sistema considera al usuario autenticado. En este mecanismo de autenticación, el login y el password son codificados pero no cifrados, por lo que no se considera un método seguro, a menos que se emplee con otras formas de seguridad como SSL.
- **Autenticación Digest.** Está basado en el paradigma "challenge-response" al igual que la autenticación Básica, pero es mucho más segura, ya que se cifran el login y el password. Sin embargo este esquema tampoco proporciona cifrado del contenido del mensaje.
- **SSL**. Proporciona autenticación utilizando certificados digitales y cifrado de toda la información intercambiada entre cliente y servidor. SSL es un protocolo independiente de la aplicación, que permite a protocolos como HTTP, FTP y Telnet correr sobre él transparentemente, pero está optimizado para HTTP. Para FTP podría ser preferible IPSec.
- **SHTTP**. Al igual que SSL, permite tanto el cifrado como la autenticación digital. Sin embargo, a diferencia de SSL, S-HTTP es un protocolo de nivel de aplicación.

[4] Challenge-Response, Desafío-Respuesta. Se prefiere su uso en Inglés.

Por otra parte, los eventos son soportados desde la versión 1.1. de la Especificación de Operaciones CIM sobre HTTP.

1.3.4 WQL. WQL (WBEM Query Language) es un subconjunto del lenguaje SQL ANSI y se utiliza para manipular datos CIM. La sintaxis WQL se parece a la sintaxis SQL excepto que las clases CIM corresponden a tablas SQL, las instancias CIM corresponden a filas SQL y las propiedades CIM corresponden a columnas SQL.

1.3.5 Ventajas de WBEM. 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. Ésto, combinado con la gran cantidad de operaciones estándar ofrecidas por WBEM, hace que este estándar se pueda aplicar a casi cualquier área de gestión. Además, como cada parte de un sistema gestionado esta descrito con un único modelo de información, la interoperabilidad está prácticamente garantizada.

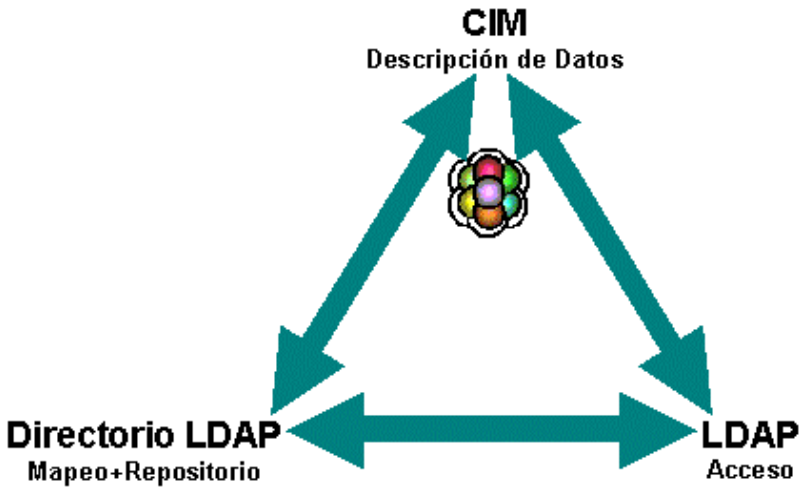
Una tercera razón para creer en el futuro de WBEM, es que este estándar no intenta reemplazar los estándares actuales tales como SNMP, sino que coexiste con ellos y los complementa. Ésto permitirá unificar los estándares actuales, con solo adicionar un proveedor WBEM a los elementos de red existentes, en lugar de tener que reemplazar estos elementos de red con unos completamente nuevos. Otra fortaleza de WBEM son las características de seguridad ofrecidas por HTTP. Los modos más fuertes de seguridad son SSL/SHTTP.

1.4 DEN

Un directorio es una base de datos de propósito especial que contiene información acerca de nodos o dispositivos unidos a la red de una empresa. Los directorios ofrecen una herramienta potencialmente poderosa para ayudar a simplificar y automatizar muchas de las complejas tareas involucradas en la gestión una gran red corporativa. Los servicios de directorio están optimizados para almacenar información que se lee frecuentemente, pero son débiles al gestionar datos que cambian constantemente.

Las tecnologías basadas en Web que se construyen sobre un modelo de información común han llegado a ser penetrantes, y la utilización de directorios permite una integración efectiva y eficiente de estas tecnologías. El uso de un directorio en un ambiente de gestión está estandarizado por la iniciativa de Red Habilitada con Directorio, DEN (Directory Enabled Network). DEN proporciona una utilización extendida de directorios como un repositorio de datos acerca de usuarios, aplicaciones, servicios de gestión, recursos de red y las relaciones entre ellos. Un directorio central de tales datos permite la integración de información de gestión de una variedad de fuentes y proporciona elementos esenciales para la gestión basada en políticas. DEN y WBEM están basados en el estándar CIM del DMTF. Ambos utilizan CIM para definir la organización y las semánticas para la información de gestión. DEN está basado en CIM y X.500, y utiliza LDAP como protocolo de acceso núcleo, como se muestra en la Figura 1.5.

Figura 1.5 Estándares DEN



La iniciativa DEN está diseñada para proporcionar los bloques de construcción para una gestión más inteligente, al mapear conceptos de CIM (tales como sistemas, servicios y políticas) a un directorio, e integrar esta información con otros elementos WBEM en la infraestructura de gestión. DEN utiliza los datos de usuario y de toda la empresa, ya presentes en el directorio de una compañía, permite servicios extremo a extremo y soporta la creación, provisión y gestión de servicios distribuidos por toda la red.

La utilización de CIM al definir un esquema de directorio, asegura que se utilizan un esquema consistente y un entendimiento común de la información de directorio, a través de implementaciones de diferentes vendedores. El esquema y las semánticas comunes son especialmente importantes al definir y descomponer políticas de alto nivel neutrales a la plataforma. La integración dentro de la infraestructura WBEM une contenidos de directorio de alto nivel, estáticos (que no cambian frecuentemente) a componentes de la infraestructura de gestión, de tiempo real, más dinámicos.

Las metas del esfuerzo DEN son:

1. Dirigir clientes a servicios de gestión relevantes.
2. Mantener un subconjunto de datos de gestión.

Los esfuerzos actuales se enfocan en definir un esquema y una utilización de directorio para:

- Administración de identidad y seguridad.
- Entendimiento común de sistemas y servicios gestionados.
- Información relacionada a localidades, agrupamientos y políticas.

Al mapear CIM a un directorio que soporte el protocolo LDAP y un modelo X.500 existen varios aspectos a considerar:

1. **Uso apropiado de un directorio.** Las clases y asociaciones CIM que son buenas candidatas para el mapeo CIM a LDAP son aquellas cuyos tiempos de vida son largos relativos a su número y aquellas cuyas propiedades no son muy volátiles. Muchas área de los modelos núcleo y común pertenecen a esta categoría.
2. **Consideraciones de nombramiento.** Aunque las reglas para unicidad de objetos CIM son estrictas (básicamente, sus propiedades clave deben tener valores correctos), las reglas para los RDNs (Relative Distinguished Names) de los objetos CIM mapeados, son flexibles y varían con la implementación.

3. **Definición del "árbol" de objetos CIM.** Todas las clases CIM se mapean como clases y subclases abstractas como están definidas en el Esquema CIM (existe una excepción menor a esta regla en el caso de la clase de elemento gestionado). Para cada clase concreta, existen dos subclases, la primera es una clase auxiliar y la segunda es una clase estructural.
4. **Falta de asociaciones en el directorio.** Existen pocos modos diferentes en los que las asociaciones CIM se pueden mapear a directorios LDAP. Cada implementación debe considerar cuidadosamente las implicaciones de qué objetos y qué asociaciones CIM se mapearán, dónde se colocarán en la "namespace" y cómo la "namespace" se dividirá.
5. **Versión.** Los nombres de las clases incluyen un número de versión. Esto es necesario porque el mapeo de OID al nombre de la clase del objeto a la descripción de la clase del objeto es 1:1:1. Esto es, una vez la definición de la clase del objeto cambia, se requiere asignar tanto un nombre como un OID nuevos, y un cambio en la definición resulta en un nueva rama del árbol de la clase del objeto, ya que todas las clases hijas también deben tener su información de versión incrementada. Esto hace que encontrar una clase específica sea muy difícil.

1.4.1 Historia. En 1988 la ISO/ITU-T lanzó una especificación de cómo se puede almacenar y acceder información en un directorio global. Esta especificación se llamó X.500 y se publicó una versión actualizada en 1993. X.500 demostró ser demasiado complicado y solamente corría en máquina UNIX altamente potentes. El IETF anunció que no iba a participar en esta especificación, así que en mayor de 1997, Microsoft y Cisco formaron la iniciativa DEN, dirigida a los servicios y las redes de directorio. En septiembre de 1997, los dos socios fundadores anunciaron que más de 20 vendedores de equipos de red habían acordado soportar la iniciativa. En febrero de 1998 la iniciativa pasó al grupo de trabajo Ad Hoc (AHWG) del DMTF (Desktop Management Task Force) (hoy la D es por Distribuido). La idea era lograr que DEN se estandarizara. Finalmente la especificación DEN llegó a ser parte del estándar CIM del DMTF para la gestión de empresa.

1.4.2 X.500. La información del directorio se encuentra en un árbol que está conformado por entradas. Cada elemento en el Árbol de Información de Directorio, DIT (Directory Information Tree) tiene un nombre único, llamado DN (Distinguished Name). Los DNs escritos como cadenas de números decimales separados por puntos, se llaman Identificadores de Objetos, OIDs (Object Identifiers). Cada objeto se representa por una entrada en el DIT y contiene un conjunto de atributos. Los atributos mantienen la información almacenada por cada objeto.

El directorio X.500 es mantenido por un conjunto de Agentes de Sistema de Directorio, DSAs (Directory System Agents). Cada DSA mantiene información de solamente una parte de toda la Base de Información de Directorio, DIB (Directory Information Base). Un DSA dado puede responder o reenviar los requerimientos de un usuario. Esto es lo que une el servicio de directorio distribuido de X.500. Los Agentes de Usuario de Directorio, DUAs (Directory User Agents), son programas independientes dentro de una aplicación, que proporcionan la interfaz entre el escritorio y el DSA local. Un DUA se comunica con un DSA a través de un Protocolo de Acceso a Directorio, DAP (Directory Access Protocol). Entre DSAs, se utiliza un segundo protocolo, un Protocolo de Servicio de Directorio, DSP (Directory Service Protocol). La información que se mantiene en un DSA se puede copiar a otros DSAs con un tercer protocolo, llamado DISP (Directory Information Shadowing Protocol).

1.4.3 LDAP. El software servidor X.500 es complejo y por tanto es difícil trabajar con él. Más aún, está muy relacionado con el stack de protocolos OSI. Debido a esto fue necesaria una solución más centrada en TCP/IP.

Los desarrolladores de aplicaciones necesitaron elegir un sistema de directorio e incluso crear un directorio privado que solamante sería utilizado por su aplicación para almacenar información de usuarios, de configuración, passwords, etc. Pero, actualmente se puede utilizar un protocolo estándar, LDAP (Lightweight Directory Access Protocol), para acceder un directorio de una manera estándar. Actualmente el IETF tiene dos grupos trabajando en modificaciones y actualizaciones. El grupo de trabajo LDUP (LDAP DirectoryUpdate) está trabajando en un servicio de replicación estándar que hará la replicación de directorio más fácil. El Grupo de Trabajo de Extensiones de LDAP se encarga de crear extensiones a LDAP y las APIs de LDAP.

Un directorio LDAP está estructurado como una jerarquía de árbol simple, la cual pone en conformidad el esquema y los modelos de nombramiento LDAP. LDAP realiza dos trabajos críticos, primero que todo, sirve como una base de datos de autenticación. Segundo, una vez la identidad de un usuario se ha establecido controla el acceso a los recursos, aplicaciones y servicios. La versión actual es LDAPv.3. LDAPv.3 tiene una capacidad de remisión así que un servidor de directorio puede enviar una "query" de un cliente, a otro.

LDAP ofrece operaciones funcionales básicas como adicionar, eliminar, buscar, comparar, modificar un DN, entre otras. Para adicionar o modificar entradas a un servidor LDAP, la especificación LDAP describe un formato de datos llamado LDIF (LDAP Data Interchange Format) que es una representación textual ASCII de la base de datos del directorio y sus entradas.

Como se mencionó anteriormente el grupo de trabajo LDUP está desarrollando un protocolo de replicación. La habilidad para distribuir información de directorio por toda la red proporciona un doble beneficio:

- Incrementa la confiabilidad del directorio.
- Trae el contenido de directorio más cerca de los clientes que utilizan los datos.

Los requerimientos de replicación son críticos para el despliegue exitoso y la aceptación de LDAP en el mercado. El principal objetivo es proporcionar un protocolo de sincronización de directorios LDAP que sea simple, altamente eficiente y bastante flexible para soportar tanto operaciones de replicación multi-maestro como multi-esclavo, para cumplir con las necesidades de tanto los entornos de Internet como de empresa.

1.5 DMI

Dentro de un sistema computacional, existe un "gap"[5] entre el software de gestión y los componentes del sistema que requieren gestión. Los gestores deben entender cómo manipular la información de un número de productos constantemente creciente. La Interfaz de Gestión de Escritorio, DMI (Desktop Management Interface) actúa como una capa de abstracción entre estos dos mundos.

DMI ha sido diseñado para ser:

- Independiente de un computador o sistema operativo específicos.
- Independiente de un protocolo de gestión específico.
- Fácil para que los vendedores lo adopten.
- Se puede utilizar localmente, no se requiere red.
- Se puede utilizar remotamente con DCE/RPC, ONC/RPC, or TI/RPC.
- Se puede mapear a protocolos de gestión existentes (como por ejemplo, CMIP, SNMP).

En DMI, sistema significa un sistema computacional. Componentes son entidades físicas o lógicas en un sistema, tales como hardware, software o firmware. Los componentes pueden venir con el sistema o se pueden adicionar a él. El código que lleva a cabo las acciones de gestión para un componente particular se conoce como instrumentación de componentes. Una aplicación de gestión es un programa que inicia requerimientos de gestión. La aplicación de gestión puede ser un programa tal como una aplicación con una interfaz de usuario gráfica o puede ser un agente de un protocolo de gestión de red que traslada requerimientos desde un protocolo de gestión de red estándar (tal como SNMP o CMIP) a DMI y al contrario. El proveedor de servicios DMI es análogo a la capa de servicios DMI de las especificaciones DMI previas. La versión más reciente de esta especificación es la 2.0.

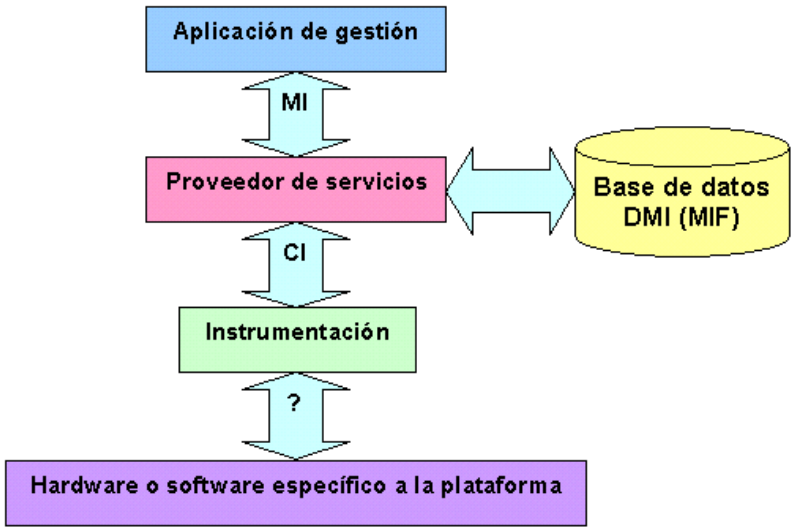
1.5.1 Elementos. DMI tiene cuatro elementos que se muestran en la Figura 1.6.

1. Un formato para describir la información de gestión.

[5] Gap, separación. Se prefiere su uso en Inglés.

2. Una entidad proveedora de servicio.
3. Dos conjuntos de APIs, un conjunto para que proveedores de servicios y aplicaciones de gestión interactúen, y el otro para que proveedores de servicio y componentes interactúen.
4. Un conjunto de servicios para facilitar la comunicación remota.

Figura 1.6 Arquitectura DMI



Las descripciones de componentes están definidas en un lenguaje llamado Formato de Información de Gestión, MIF (Management Information Format). Cada componente tiene un archivo MIF para describir sus caracterísiticas gestionables. Cuando un componente se instala inicialmente en el sistema, el MIF se adiciona a la base de datos MIF (dependiente de la implementación).

Los proveedores de servicios DMI exponen un conjunto de puntos de entradas que pueden ser llamados por la instrumentación de componentes. Estos se llaman colectivamente API Proveedor de Servicios para Componentes. Así mismo, el código de la instrumentación de componentes expone un conjunto de puntos de entrada que pueden ser llamados por el proveedor de servicios DMI. Estos se llaman colectivamente API Proveedor de Componentes. En la Especificación DMI Versión 1.x estas dos APIs fueron incorporadas en la Interfaz de Componentes.

La Interfaz de Componentes, CI (Component Interface), es utilizada por los proveedores de componentes para describir el acceso a la información de gestión y para permitir a un componente ser gestionado. CI y MIF protegen a los vendedores de la complejidad de los estilos de codificación e información de registro de gestión, y no necesitan aprender los detalles de los protocolos de gestión populares y emergentes.

Las versiones previas de esta especificación definieron CI como una interfaz de datos orientada a bloque. La especificación más reciente introdujo una nueva interfaz CI procedimental. Las funciones en la Interfaz de Componentes son específicas al sistema operativo. Algunos sistemas operativos pueden no implementar CI pero proporcionan funcionalidad equivalente utilizando otros mecanismos nativos.

El proveedor de servicios también expone un conjunto de puntos de entrada que pueden ser llamados por las aplicaciones de gestión. Estos se llaman colectivamente API Proveedor de Servicio para las Aplicaciones de Gestión. Así mismo, las aplicaciones de gestión exponen un conjunto de puntos de entrada que pueden ser llamados por los proveedores de Servicios DMI. Estos se llaman colectivamente API Proveedor de Gestión. En la Especificación DMI versión 1.x estos se incorporaron en la Interfaz de Gestión. La Interfaz de Gestión MI (Management Interface), es utilizada por las

aplicaciones que desean gestionar componentes. MI protege a los vendedores de las aplicaciones de gestión, de los diferentes mecanismos utilizados para obtener información de gestión de los elementos dentro de un sistema computacional.

Las versiones previas de esta especificación definieron MI como una interfaz de datos orientada a bloque. La especificación más reciente introdujo una nueva interfaz MI procedimental. Todas las nuevas funciones introducidas por esta especificación solamente están disponibles como parte de esta nueva interfaz. La nueva MI procedimental es una interfaz que se puede acceder remotamente y está diseñada para ser utilizada con uno de los RPCs que soporta.

El proveedor de servicios, SP (Service Provider), llamado previamente Capa de Servicio, SL (Service Layer), es una pieza de código activa, residente, corriendo en el sistema computacional que media entre MI y CI, y ejecuta servicios en nombre de cada uno. En otras palabras un proveedor de servicios, es un programa que corre en el sistema gestionado y se comunica con las aplicaciones de gestión por medio de la Interfaz de Gestión y con los componentes gestionados por medio de la Interfaz de Componentes.

1.5.2 Modelo de datos. Los componentes tienen uno o más atributos que colectivamente definen la información disponible a una aplicación de gestión. Los atributos se reúnen en grupos por facilidad de referencia. Los grupos pueden ser escalares o instanciaciones múltiples. Los grupos múltiplemente instanciados se llaman tablas, y una fila (instancia) de una tabla es referenciada por un conjunto de atributos que forman una clave.

Entonces dentro de un sistema, existen muchos componentes, cada uno con uno o más grupos. Cada grupo tiene uno o más atributos y cada grupo puede ser múltiplemente instanciado como una tabla. La instrumentación de componentes presenta esta representación componente/grupo/clave/atributo a la aplicación de gestión.

1.5.3 Caracterísiticas operacionales. La relación entre las aplicaciones de gestión, el proveedor de servicios DMI y la instrumentación de componentes puede existir como una relación muchos a uno a muchos. Puede haber muchas aplicaciones de gestión emitiendo comandos a través de un solo SP DMI para gestionar muchos componentes. Si múltiples aplicaciones de gestión están activas y cada una tiene un lenguaje diferente especificado, se requiere una instrumentación de componentes que soporte múltiples lenguajes simultáneamente.

Para propósitos de identificación, las aplicaciones de gestión se deben registrar con el SP DMI antes de que puedan participar en las funciones de gestión. La instrumentación de componentes se debe instalar dentro del SP DMI cuando se introduce por primera vez al sistema. Los mecanismos para hacer esto pueden diferir entre sistemas operativos e implementaciones SP DMI.

Normalmente el flujo de control se inicia desde la aplicación de gestión al Proveedor de Servicio DMI y a la instrumentación de componentes. También puede haber indicaciones, las cuales son reportes no solicitados que fluyen en la dirección opuesta.

Existen tres categorías generales de comandos de acceso: *Get*, *Set* y *List*. Los comandos *Get* y *Set* permiten a las aplicaciones de gestión leer y escribir entidades gestionables dentro de un sistema.

El comando *List* retorna meta información. Este comando no consigue los valores de atributos reales dentro del componente sino que le permite a la aplicación de gestión, conseguir la información semántica en MIF. Ya que el proveedor de servicios DMI consigue información MIF desde su base de datos MIF, el comando List no requiere la invocación de ningún código de instrumentación de componentes.

1.5.4 Acceso remoto a la interfaz. La interfaz procedimental, además de ser adecuada para acceso remoto a través de uno de los mecanismos RPC soportados, definidos previamente, es muchos más amigable para los programadores y muchos menos propensa a errores.

La interfaz para acceso remoto (DMIv2.0) está diseñada para proporcionar acceso remoto a la funcionalidad y datos DMI, mientras oculta la complejidad de manipular bloques de datos. DMIv1.x a munudo maneja por bloques, funciones relacionadas de algún modo, con comandos simples. Esto hace que los comandos retornen mucha información relacionada y se requiera que el solicitante seleccione lo que desea. En DMIv2.0, las llamadas se dividen funcionalmente para proporcionar información específica. Por lo tanto, un comando DMIv1.x dado puede equivaler a múltiples comandos DMIv2.0, cada uno realizando una función específica.

Una Llamada de un Procedimiento Remoto, RPC (Remote Procure Call), está basado en un arquitectura cliente/servidor. El lado del cliente incluye un conjunto de "stubs"[6] que tienen interfaces con las mismas declaraciones de las llamadas a funciones en el servidor que ellos representan. Los stubs interactúan con el soporte RPC local para intercambiar parámetros de entrada, parámetros de salida y códigos de retorno, y con el procedimiento remoto localizado en el servidor. Un nodo remoto actúa como un cliente para llamadas a funciones MI procedimentales, y como un servidor al recibir indicaciones. El nodo bajo gestión actúa como un servidor para llamadas a funciones MI procedimentales, y como un cliente al entregar indicaciones a un nodo remoto.

1.5.5 Seguridad. DMIv2.0s, que es una versión extendida de la Especificación DMIv2.0, define un mecanismo para controlar el acceso remoto a la Interfaz de Gestión DMI y el acceso local a las interfaces DMI. El mecanismo de control de acceso remoto está definido sobre los mecanismo RPC estándar, mientras el mecanismo de control de acceso local se define sobre los mecanismos del sistema operativo. DMIv2.0s no especifica un formato estándar para identidades ni un criptosistema para verificar esas identidades, pero confía en aquellas proporcionadas por RPC y por el sistema operativo. Las principales características introducidas por DMIv2.0s son autenticación, autorización basada en roles, políticas flexibles, indicaciones de seguridad y logging.

Si una implementación de proveedor de servicios DMI proporciona un mecanismo de control de acceso, tiene que implementar la Extensión de Seguridad DMI como está definido en la especificación.

La seguridad DMIv2.0s está basada en la infraestructura de seguridad proporcionada por RPC y el Sistema Operativo. Por lo tanto, si la seguridad de RPC o del sistema operativo está comprometida, la seguridad DMIv2.0s también estará comprometida.

1.5.6 Especificación de mapeo DMI a SNMP. El objetivo de este estándar de mapeo es hacer un puente en el "gap" de interoperabilidad entre soluciones basadas en SNMP y DMI. Para la interoperación de estas soluciones se puede:

1. Permitir a aplicaciones SNMP gestionar sistemas instrumentados con DMI.
2. Permitir a sistemas instrumentados con DMI soportar MIBs estándar
3. Proporcionar acceso SNMP a sistemas instrumentados con DMI que puedan o no puedan tener un agente de mapeo SNMP residente.

Una colección de clases de grupo DMTF relacionadas, es semánticamente equivalente a una MIB SNMP, en que modela funcionalidad específica independiente del empaquetamiento físico. La principal diferencia es el modo en cual la información de gestión DMI y SNMP se nombra, DMI utiliza nombres de clases mientras que SNMP utiliza identificadores de objetos numéricos. El esquema permite a MIBs estándar ser generadas a partir de MIFs estándar y a MIBs propietarias ser generadas a partir de MIFs propietarias. Estas MIBs entonces pueden ser utilizadas por una aplicación basada en SNMP, en conjunto con un agente de mapeo, para gestionar cualquier sistema intrumentado con DMI cuya MIF está compuesta por estos grupos.

[6] Stub, proxy. Se prefiere su uso en Inglés.

1.5.7 "Fin de la vida" para DMI. Debido al rápido avance de las tecnologías del DMTF, tales como CIM, el DMTF está definiendo un proceso de "Fin de vida" para DMI, el cual tomará lugar en marzo 31 de 2005. El DMTF terminará el desarrollo activo de nuevos estándares DMI en diciembre 31 de 2003 pero continuará aceptando Certificaciones DMI hasta marzo 31 de 2005 y las publicará hasta por 6 meses después de la aceptación.

Las especificaciones CIM y WBEM del DMTF fueron desarrolladas para gestionar un entorno más amplio y la complejidad adicional de los entornos de empresa e Internet de hoy, utilizando técnicas de modelamiento orientadas a objetos. El DMTF está recomendando la transición de DMI a estas tecnologías más nuevas, para proporcionar la próxima generación de gestión, monitoreo y servicios de configuración para sistemas, software y redes.

1.6 ASF

El término "gestionabilidad de un sistema", representa un amplio rango de tecnologías que permiten el acceso y el control de un sistema remoto en entornos tanto de sistema operativo-presente, OS-presente, como de sistema operativo-ausente, OS-ausente. Estas tecnologías se enfocan principalmente en minimizar el mantenimiento en el sitio, maximizando la disponibilidad y el desempeño del sistema al usuario local, así como también la visibilidad remota de (y acceso local a) sistemas locales por los gestores, y minimizando el consumo de potencia del sistema que se requiere para mantener esta conexión remota intacta.

El DMTF define las interfaces DMI y CIM que operan cuando el cliente gestionado es totalmente operacional en su entorno de OS-presente. El Formato Estándar de Alertas, ASF (Alert Estandar Format) define el control remoto y las interfaces de alerta que sirven mejor los entornos de OS-ausente del cliente.

Dos conceptos importantes a tener en cuenta son:

- **Sistema operativo ausente.** Sistema operativo ausente es un término utilizado para describir el estado de un computador cuando su sistema operativo no es totalmente operacional. Ejemplos de estados de OS-ausente incluyen cuando el sistema está arrancando, suspendido, o cuando el arranque ha fallado.

- **Sistema operativo presente.** El sistema operativo presente es un término utilizado para describir el estado de un computador cuando su sistema operativo es totalmente operacional. Este es el estado típico del sistema operativo cuando está ejecutando requerimientos para los usuarios.

El problema de la gestionabilidad de sistemas sin un sistema operativo, históricamente ha sido resuelto con soluciones propietarias y relativamente costosas. Las tecnologías de alerta proporcionan advertencias e indicaciones anticipadas de fallas del sistema de los clientes gestionados a consolas de gestión remotas. Las generaciones iniciales de esta tecnología, como las implementaciones de IBM/Intel, Alerta en LAN, AoL (Alert on LAN), proporcionaron notificaciones remotas de los estados y las fallas del hardware o del software del sistema cliente sin importar el sistema operativo o el estado de potencia del sistema. La iniciativa Interfaz de Gestión de Plataforma Inteligente, IPMI (Intelligent Platform Management Interface), dirigida por Intel y otros, subsecuentemente proporcionó una interfaz de alerta abierta. Los proveedores de consolas de gestión del sistema se enfrentaron con la posibilidad de soportar múltiples interfaces de alerta.

Una vez la alerta de un sistema proporciona su advertencia o reporte de error, el próximo paso en la gestionabilidad de un sistema remoto, es permitir que se tome una acción correctiva, estas acciones incluyen la habilidad para restablecer, encender, apagar o reiniciar el sistema cliente remotamente. Cuando el sistema está en estado de OS-presente, estas acciones pueden ser proporcionadas por las interfaces CIM que interactúan con el sistema local. Esta especificación proporciona una funcionalidad similar cuando el sistema está en estado de OS-ausente, igual a la adicionada por la segunda generación de las tecnologías IBM/Intel AoL.

La principal meta de esta especificación es definir las interfaces basadas en estándares, con las cuales los vendedores de productos de alerta y de acción correctiva, pueden hacer implementaciones de productos y asegurar interoperabilidad.

Los protocolos basados en estándares (como por ejemplo, SNMP y UDP) sobre los cuales se construyen las interfaces de esta especificación, son livianos, portadores de información basada en bit, ya que se anticipa que la mayoría de las implementaciones cliente ASF estarán basadas en hardware y/o firmware. Los métodos de configuración basados en CIM pueden proporcionar la capa de abstracción entre las implementaciones XML de OS-presente y las primitivas de bajo nivel definidas por ASF.

Actualmente existen múltiples soluciones en la industria, lo que produce pérdida de interoperabilidad entre productos de alerta y de acción correctiva del sistema.

Un sistema de alerta consiste de un sistema (o sistemas) cliente y una consola de gestión que monitorea y controla los clientes. Un cliente que cumple con ASF proporciona varios medios para permitir interoperabilidad entre el cliente y su consola de gestión ya que:

1. Permite la transmisión de mensajes del sistema ASF, incluyendo estado del sistema y alertas de seguridad.
2. Permite la recepción y procesamiento de requerimientos de mantenimiento remoto, enviados por la consola de gestión.
3. Tiene la habilidad para describir las capacidades y características específicas del sistema cliente a la consola de gestión.

Adicionalmente se requiere un nivel de interoperabilidad entre los componentes de alerta de un sistema cliente.

1. El firmware del sistema debe comunicar las capacidades del sistema, al software de configuración de OS-presente del componente ASF.
2. Los mensajes entre el componente ASF, el equipo del sistema local y los sensores del sistema local se deben soportar.

Cuando se adiciona un dispositivo de envío de alertas ASF a un cliente gestionado, el dispositivo de envío de alertas debe utilizar la configuración de hardware específica del cliente para que pueda producir alertas apropiadamente y pueda responder a los requerimientos de mantenimiento remoto. Para llevar a cabo esto, el sistema cliente requiere un buen arranque en un entorno de OS-presente, para permitir al software de configuración del dispositivo, correr y almacenar información específica del sistema en el almacenamiento no volátil del dispositivo.

ACPI (Advanced Configuration and Power Interface) es un especificación abierta de la industria co-desarrollada por Hewlett-Packard, Intel, Microsoft, Phoenix y Toshiba. ACPI establece interfaces estándar de la industria, para la gestión de potencia en computadores portátiles, de escritorio y servidores. ACPI se aplica a todas las clases de computadores.

En un entorno de OS-presente que cumple con ACPI, el software de configuración del dispositivo de envío alertas solicita los datos de configuración del cliente, para recuperar la información requerida, con el objetivo de que cualquier mensaje de alerta se envíe, y guarda esa información en el almacenamiento no volátil del dispositivo para utilizarla en el entorno de OS-ausente:

1. La implementación ACPI del cliente contiene sus capacidades ASF, incluyendo el ID del Fabricante de IANA y el ID del Sistema.
2. La estructura-tabla SMBIOS del cliente contiene el Identificador Único Global, GUID (Globally Unique IDentifier), o el Identificador Único Universal, UUID (Universally Unique IDentifier) del sistema.
3. El sistema operativo asigna una dirección TCP/IP al dispositivo de envío de alertas.

4. La cantidad de tiempo que el dispositivo de envío de alertas espera antes de producir una alerta de falla de arranque del sistema.

El software de configuración también proporciona una interfaz para permitir al propietario del sistema, identificar la dirección TCP/IP de la consola de gestión a la se va a enviar cualquier mensaje de alerta.

Durante este proceso de configuración de OS-presente, la configuración ASF opcional del cliente gestionado, también se determina y se coloca en el almacenamiento no volátil del dispositivo de envío de alertas:

1. Si el cliente incluye sensores, se guarda la información de direccionamiento y de configuración para cada sensor.
2. Si el cliente soporta operaciones de control remoto, se almacenan las características específicas definidas para ASF.

Una vez el propietario del sistema ha configurado el dispositivo que envía alertas, el cliente gestionado se habilita para enviar mensajes de alerta y, opcionalmente, responder a requerimientos de control remoto desde una consola de gestión especificada.

1.6.1 Protocolos de red ASF. ASF utiliza dos protocolos de red, el primero es PET (Platform Event Trap) para enviar alertas a consolas de gestión y el segundo es RMCP (Remote Management and Control Protocol) para hacer control remoto del sistema.

PET. El protocolo PET utiliza una PDU de traps SNMP para llevar información de gestión del sistema. Las alertas cubren varias actividades del sistema de bajo nivel. Estas incluyen:

- Eventos del entorno.
- Errores y eventos en progreso, del firmware del sistema.
- Errores de la CPU/eventos DOA (dead on arrival).
- Intrusión al chasis.
- Eventos del sistema operativo.
- Falla en el arranque del sistema.

Los eventos del entorno definidos en este estándar incluyen problemas de temperatura, de voltaje y de ventilación. Los errores del firmware del sistema incluyen problemas con la memoria y el disco duro del sistema, entre otros. Los eventos en progreso del firmware del sistema proporcionan información en el progreso de la operación de arranque. Los errores de la CPU/eventos DOA indican que el procesador del equipo se ha removido o no está funcionando. Intrusión al chasis alerta que el sistema ha sido forzado y/o abierto. Los eventos del sistema operativo incluyen fallas del sistema operativo en el arranque, entre otros.

RMCP. RMCP (Remote Management and Control Protocol) es un protocolo basado en UDP (User Datagram Protocol), para el control de un sistema, cuando un cliente gestionado está en el estado de OS-ausente. En este entorno, los paquetes RMCP se intercambian entre una consola de gestión y un cliente gestionado. Las funciones de control de un cliente típico incluyen operaciones tales como:

- Restablecimiento
- Encendido, Apagado
- Reinicio.

El protocolo es intencionalmente simple, para permitir al firmware de los dispositivos de alerta, convertir fácilmente la información en ausencia de drivers de OS-presente.

Una consola de gestión utiliza métodos RMCP, como parte de un aproximación doblemente estratificada, para gestionar sistemas cliente. La consola siempre debería utilizar métodos de OS-presente como el método principal para restablecer, encender, apagar o reiniciar un cliente gestionado,

así que cualquiera de estas operaciones se maneja en un modo ordenado. Las consolas deberían emplear métodos RMCP solamente si el cliente gestionado falla al responder a los métodos de OS-presente, ya que los métodos RMCP basado en hardware podrían resultar en pérdida de datos en el sistema cliente.

1.6.2 Beneficios de ASF. ASF permite a un administrador responder proactivamente y/o reactivamente a un problema, en un sistema o conjunto de sistema particulares, cuando un sistema operativo no está presente.

Existen muchos métodos disponibles en el mercado de hoy para gestionar hardware. ASF fue diseñado específicamente para llenar el "gap" de la gestión de sistemas de OS-ausente, aunque también opera en estados de OS-presente. El problema de la gestionabilidad de sistemas sin un sistema operativo, históricamente ha sido resuelto con soluciones propietarias y relativamente costosas, pero ASF representa la solución basada en estándares con el costo más bajo por sistemas en el mercado de hoy.

1.6.3 Limitaciones. Después de un cambio en la configuración del hardware del sistema, por ejemplo, adición o remoción de una tarjeta, se requiere al menos un buen arranque en el entorno de OS-presente, para que el subsistema ASF opere apropiadamente. El entorno de OS-presente se utiliza para configurar el dispositivo que envía alertas ASF con información que no se conoce o que no se puede determinar fácilmente dentro del entorno de OS-ausente, por ejemplo, las direcciones TCP/IP de la consola de gestión y del sistema local.

El control en OS presente de las características ASF específicas del sistema, se reduce si se utiliza un sistema operativo que no cumple con ACPI, ya que ACPI proporciona métodos estándar para que el entorno de OS-presente se comunique con el firmware del sistema.

1.6.4 Exposición de ASF a través de DMI. Esta especificación describe los grupos DMI estándar, utilizados para configurar un sistema gestionado, para comunicaciones fuera de banda, mientras está operando con OS presente. Se definen tres grupos DMI que actúan como una interfaz intermedia, para comunicaciones fuera de banda, entre una aplicación de gestión DMI y un sistema gestionado, donde un sistema gestionado que cumple con DMI de OS presente traslada las comunicaciones ASF fuera de banda. Dos grupos DMI están definidos para mapear alertas ASF a indicaciones DMI y el tercer grupo DMI se utiliza para enviar requerimientos ASF a un sistema gestionado y recibir respuestas ASF.

DMI no es la única interfaz que se puede utilizar para realizar la configuración ASF y las comunicaciones fuera de banda. Las aplicaciones de gestión ASF pueden elegir comunicarse directamente a través de las interfaces proporcionadas por el hardware ASF o a través de las interfaces de gestión proporcionadas por las extensiones relacionadas con ASF de CIM. Los grupos DMI estándar se utilizan para realizar las siguientes funciones:

- Configurar con OS presente, un sistema habilitado con ASF.
- Mapear alertas ASF, recibidas desde un sistema habilitado con ASF, a indicaciones DMI
- Acceder fuera de banda a un sistema gestionado utilizando RMCP.

1.7 SMBIOS

La especificación de Referencia SMBIOS (System Management Bios) establece cómo los vendedores de "motherboards" y de sistemas presentan información de gestión relacionada con hardware en un formato estándar, al extender la interfaz BIOS en los sistemas de arquitectura Intel. Se pretende que la información permita a la instrumentación genérica entregar estos datos a las aplicaciones de gestión que utilizan DMI, CIM o acceso directo.

La especificación SMBIOS se inició como un esfuerzo colectivo de los vendedores de BIOS y de los fabricantes de sistemas en 1995, y originalmente se llamó DMI BIOS. El enfoque inicial fue especificar

mecanismos para proporcionar información del sistema a la instrumentación DMI. En 1999 la Especificación SMBIOS fue adoptada por el DMTF y se ha actualizado periódicamente.

La especificación SMBIOS define la estructura de la información, así como también los métodos de acceso.

1.7.1 Estructuras SMBIOS. SMBIOS define aproximadamente 40 estructuras de datos, representando información sobre varios componentes y configuraciones del sistema. Para reportar la información de un sistema a una aplicación o proveedor de instrumentación, una implantación que cumpla con SMBIOS debe (como mínimo) llenar el siguiente conjunto de estructuras base:

- BIOS Information
- System Information
- System Enclosure
- Processor
- Cache
- System Slots
- Physical Memory Array
- Memory Device
- Memory Array Mapped Addresses
- Memory Device Mapped Addresses
- System Boot Information

Además de las anteriores estructuras, las implementaciones SMBIOS también pueden representar información adicional acerca de la configuración del sistema, configuraciones de jumpers, controladores y módulos de memoria, puertos y conectores, dispositivos de apuntamiento, baterías y suministros de potencia, entre otros. También se puede reportar información relativa a controladores y procesadores en el sistema. Además, las opciones de configuración de tales dispositivos (por ejemplo, dispositivos que cumplen con ASF del DMTF o IPMI (Intelligent Platform Management Interface) de la industria) están disponibles en SMBIOS. SMBIOS también define estructuras para presentar información acerca de los sensores en un sistema (por ejemplo, voltaje, temperatura, corriente, etc.) y su estado.

La BIOS típicamente llena las estructuras SMBIOS al momento del arranque del sistema y no está bajo control cuando el sistema operativo está operando. Por lo tanto, los datos que cambian dinámicamente raramente se representan en tablas SMBIOS. En la mayoría de implementaciones, es una práctica común, utilizar interfaces directas a los controladores de gestión para conseguir acceso a su información dinámica.

Las estructuras SMBIOS permiten extensiones a través de un arreglo de cadenas. Esta aproximación puede ser utilizada por los vendedores de sistemas para proporcionar información adicional que no está representada en las estructuras estándar definidas.

1.7.2 Métodos de acceso SMBIOS. La especificación SMBIOS especifica dos métodos de acceso para recuperar las estructuras de la BIOS nombradas anteriormente.

Un método basado en tablas proporciona las estructuras SMBIOS como un lista de datos empaquetada, referenciadas por un punto de entrada en la tabla. El punto de entrada puede ser localizado por una aplicación, al buscar por el valor de cadena apropiado dentro del rango de direcciones de memoria física 000F0000h y 000FFFFFh. Una vez la estructura de entrada se localiza, las aplicaciones pueden acceder esta estructura y conseguir el resto de la información necesaria para acceder las estructuras SMBIOS restantes. Todas las implementaciones SMBIOS deben proporcionar este método de acceso.

Un método opcional, definido en la especificación, proporciona estructuras SMBIOS a través de una interfaz de función Plug-n-play, pero esta interfaz interfaz se desaprobó. Las futuras implementaciones

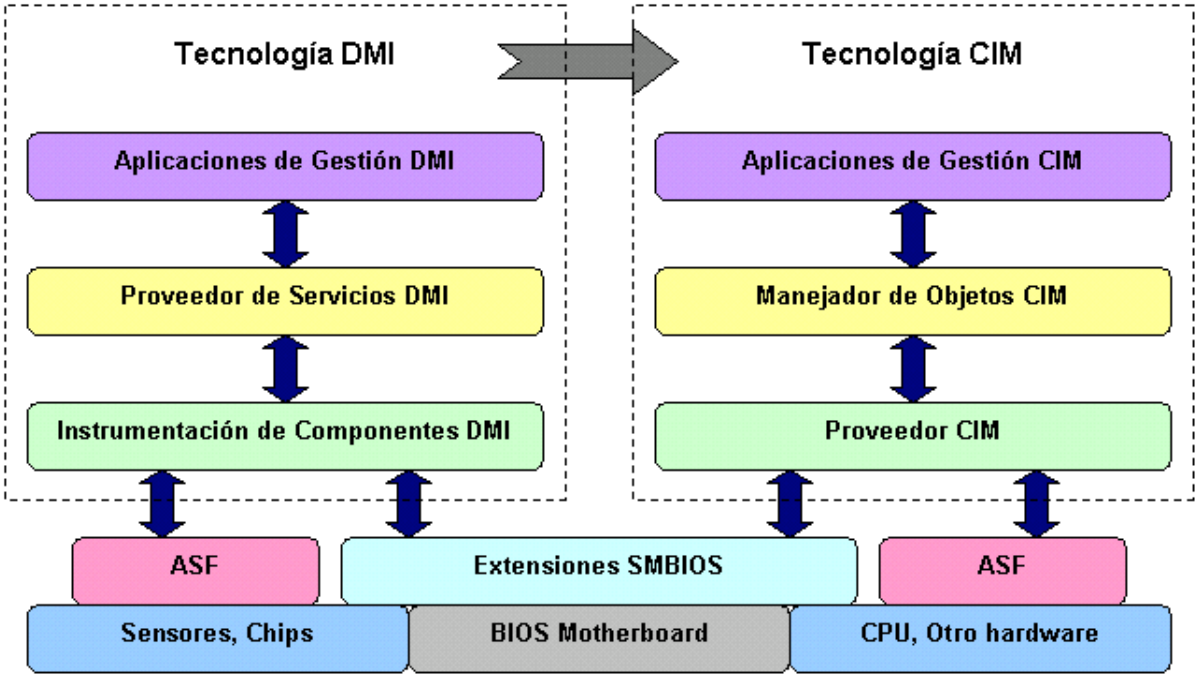
de SMBIOS y los consumidores de datos SMBIOS deberían migrar a utilizar solamente el método basado en tabla.

Para entornos de OS-presente, la especificación SMBIOS actúa como un estándar complementario a CIM. SMBIOS proporciona una abstracción de nivel más bajo, la cual los vendedores de BIOS y de sistemas puede utilizar para adquirir y reportar información de configuración del sistema a través de CIM. Además, para entornos de OS-ausente y pre-OS (fases POST/BOOT), SMBIOS proporciona una herramienta poderosa para el descubrimiento de la configuración del sistema, permitiendo que el software se adapte a las necesidades del sistema.

1.8 CAMPO DE APLICACIÓN DE LOS ESTÁNDARES DEL DMTF

La Figura 1.7 muestra cómo las interfaces de gestión de bajo nivel tales como ASF y SMBIOS interactúan con DMI y tecnología basada en CIM para permitir operaciones de gestión. La estrategia del DMTF es migrar la tecnología basada en DMI a tecnologías basadas en WBEM y CIM para obtener una infraestructura de gestión simple y consistente.

Figura 1.7 Campos de aplicación de los estándares del DMTF



Teniendo en cuenta las características de los diferentes estándares del DMTF y sus entornos de aplicación, los estándares del DMTF más adecuados para desarrollar este proyecto son CIM y WBEM, ya que permiten representar toda la información de gestión en una forma estandarizada y utilizar las más utilizadas e importantes tecnologías de Internet. Estos dos estándares son muy potentes y además se pueden integrar con otros estándares de gestión como por ejemplo SNMP, muy difundido actualmente.

Por lo comentado en este capítulo se puede concluir que de los estándares del DMTF, CIM y WBEM son la mejor opción para gestionar estaciones de trabajo de usuario, ya que permiten realizar aplicaciones de gestión flexibles, escalables e interoperables. Además, en el siguiente capítulo se realiza una comparación de las principales infraestructuras de gestión y WBEM, con el fin de mostrar sus fortalezas y ventajas de utilización.

2. COMPARACIÓN ENTRE LAS PRINCIPALES INFRAESTRUCTURAS DE GESTIÓN

2.1 INFRAESTRUCTURAS DE GESTIÓN

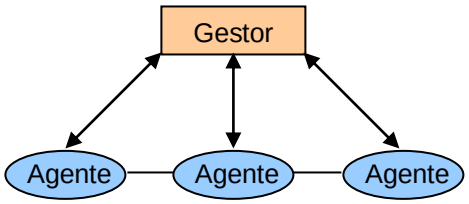
Una infraestructura de gestión consta de una arquitectura, servicios y protocolos de gestión, y un conjunto de APIs que permiten extender las capacidades de la infraestructura a través de aplicaciones. Las infraestructuras son implementadas como plataformas de gestión y son definidas a un nivel de abstracción más alto que las tecnologías de gestión. De hecho, las infraestructuras pueden ser definidas en una forma independiente de la tecnología.

En general, se puede decir que una infraestructura de gestión consta de un modelo organizacional, un modelo de información, un modelo de comunicación y un modelo funcional. A continuación se hace una descripción de cada uno de ellos.

Modelo Organizacional. Describe los componentes de un sistema de gestión, sus funciones y relaciones. Se define en función de los paradigmas de gestión de red o modelos de distribución. Ellos son: centralizado, débilmente distribuido, fuertemente distribuido y cooperativo.

- **Gestión centralizada.** La gestión centralizada consta de dos niveles, un gestor de alto nivel y los agentes. Los agentes recolectan datos de los dispositivos gestionados y los hacen disponibles al gestor central. El gestor está encargado de recuperar los datos de los agentes y presentarlos en una forma útil al administrador de la red. La Figura 2.1 muestra un sistema de gestión centralizado.

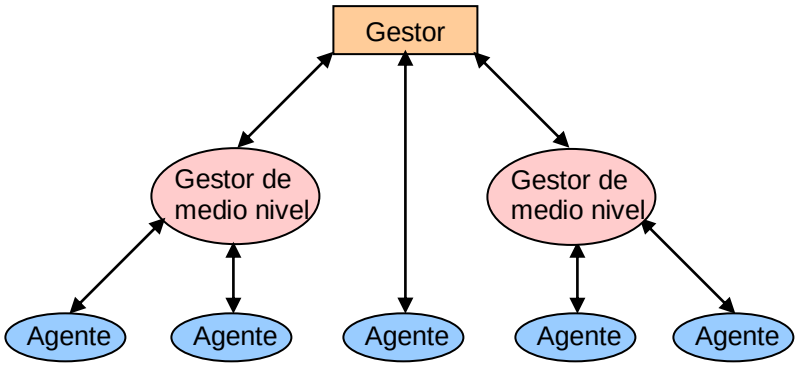
Figura 2.1 Sistema de gestión centralizado



Los sistemas centralizados trabajan bien en redes pequeñas, pero son inadecuados para un entorno con un bajo ancho de banda porque el gestor consume mucha capacidad de red con el tráfico generado por las consultas a los agentes.

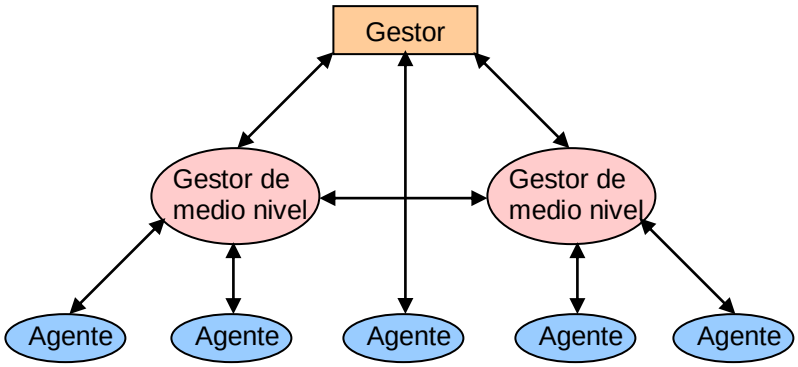
- **Gestión débilmente distribuida.** Estos sistemas se caracterizan por la presencia de entidades de gestión de medio nivel que asumen tanto el rol de agente y gestor. Utilizan delegación de tareas vertical, es decir, los gestores solamente delegan tareas a gestores de más bajo nivel, por esto hay una jerarquía bien definida. Los gestores de medio nivel no se comunican entre ellos. El número de entidades de medio nivel es mucho más pequeño que el número de agentes en el sistema. La Figura 2.2 muestra un sistema de gestión de este tipo.

Figura 2.2 Sistema de gestión débilmente distribuida



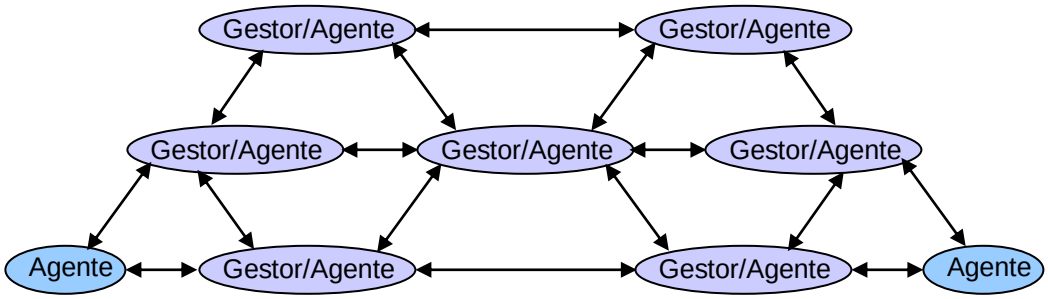
❖ **Gestión fuertemente distribuida.** Estos sistemas se caracterizan por gestores de medio nivel, cumpliendo tanto el rol de agente y gestor, que pueden comunicarse entre ellos. La Figura 2.3 muestra un sistema de gestión de este tipo.

Figura 2.3 Sistema de gestión fuertemente distribuida



❖ **Gestión cooperativa.** Estos sistemas se caracterizan porque casi todos los dispositivos son capaces de cumplir el rol de agente o gestor. La Figura 2.4 muestra un sistema de este tipo.

Figura 2.4 Sistema de gestión cooperativa



Es importante notar que la noción de jerarquía vertical se reemplaza por una estructura horizontal.

La Tabla 2.1 describe como se hace la clasificación de los modelos de distribución en números, donde *m* es el número de gestores y *n* el número total de elementos (gestores de alto y medio nivel más los agentes).

Tabla 2.1 Clasificación de modelos de distribución

$1 = m$	Gestión centralizada
$1 < m << n$	Gestión débilmente distribuida
$1 << m < n$	Gestión fuertemente distribuida
$m \sim n$	Gestión cooperativa

Comparación entre modelos de distribución

- Con gestión centralizada toda la red puede salirse de control si el gestor falla. Además, la gestión centralizada puede ser menos eficiente, por ejemplo, cuando se necesita intercambiar mucha información de gestión se pueden presentar cuellos de botella.

- Con gestión distribuida y cooperativa es difícil cambiar la funcionalidad que toma las decisiones de gestión después que la fase operacional ha comenzado, porque estos cambios requieren la modificación de un gran número de sistemas, lo cual es costoso. Por lo tanto, en algunos casos puede ser mejor utilizar gestión centralizada, que reúne la funcionalidad de gestión que toma las decisiones en un solo sistema.

- Los sistemas centralizados son más fáciles de mantener porque ellos no tienen que tratar con una variedad de sistemas operativos, protocolos y formatos de datos. Sin embargo, los sistemas distribuidos y cooperativos tienen una mejor relación costo-desempeño, son más escalables, flexibles y confiables.

- La gestión centralizada se puede realizar en forma implícita o explícita, mientras que la gestión distribuida y cooperativa se debe realizar en forma implícita debido al gran número de sistemas que pueden tomar decisiones de gestión. Con gestión explícita, la decisión de iniciar funciones de gestión es tomada explícitamente por personas durante la fase operacional. Con gestión implícita, todas las funciones de gestión son ejecutadas por módulos hardware y software, por lo tanto, no es necesario la intervención de operadores. Una ventaja de la gestión explícita es que no es necesario elaborar todas las funciones de gestión durante la fase de diseño, ya que las funciones que falten pueden ser incluidas durante la fase operacional. Esto es muy importante, ya que muchas veces no es posible diseñar todas las funciones de gestión antes de comenzar la fase operacional. Como resultado de esto, el procedimiento de diseño es menos complicado y requiere menos tiempo del que se requiere con gestión implícita. La gestión explícita es particularmente útil para solucionar los problemas inesperados que surgen durante la fase operacional y que requieren nuevas soluciones, por lo tanto la gestión explícita es adecuada para gestión de fallas. Sin embargo, ya que la gestión explícita es ejecutada por seres humanos, el tiempo de respuesta puede ser pobre si se compara al de la gestión implícita. Además la gestión explícita tiene capacidad limitada y es propensa a errores. Si se comparan los costos asociados con ambas formas de gestión, se puede decir que en el caso de la gestión explícita, las funciones de gestión que son elaboradas durante la fase de diseño son menos complejas y por lo tanto menos costosas. De otro lado, la gestión explícita requiere intervención humana durante la fase operacional, así, la gestión explícita es más costosa durante esta etapa. Los ejemplos del mundo real muestran una combinación de ambas formas de gestión: algunos problemas son solucionados en forma implícita, mientras otros requieren la utilización de gestión explícita.

- Los sistemas cooperativos hacen transparente la localización (los usuarios de un sistema tratan con entidades computacionales sin tener en cuenta donde están localizadas físicamente), la migración (las entidades computacionales pueden ser movidas a diferentes localizaciones físicas

sin que sus clientes sean enterados de esto), y la concurrencia (muchos clientes pueden acceder las mismas entidades sin tener en cuenta otros usuarios en el sistema).

Modelo de Información. Proporciona una representación abstracta de todos los objetos gestionados, y describe la estructura y organización de la información de gestión.

Ha habido confusión con la diferencia entre Modelo de Información y Modelo de Datos, sin embargo, ellos son diferentes porque son utilizados para diferentes propósitos. El principal propósito de un Modelo de Información es modelar objetos gestionados a un nivel conceptual, independientemente de cualquier implementación o protocolo. El grado de especificidad o detalle de las abstracciones definidas en el Modelo de Información depende de las necesidades de modelamiento de los diseñadores. Los Modelos de Información pueden ser implementados en diferentes formas y mapeados a diferentes protocolos. Otra característica importante de ellos es que pueden, y generalmente deben, especificar relaciones entre objetos.

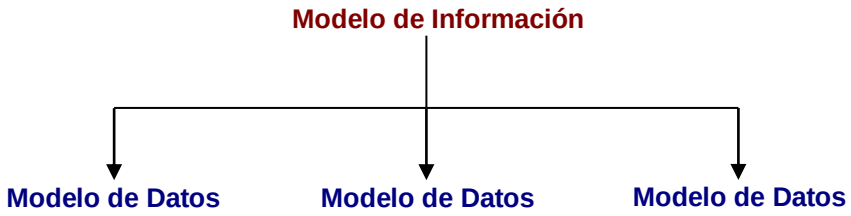
Los términos Modelos Conceptuales y Modelos Abstractos, que se utilizan a menudo en la literatura, se relacionan con los Modelos de Información.

Los Modelos de Información pueden ser definidos utilizando lenguajes naturales como el Inglés, o lenguajes formales. Una de las posibilidades para formalmente especificar Modelos de Información es utilizar diagramas de clase UML. Varias organizaciones como el DMTF, la ITU-T, 3GGP, el TeleManagement Forum y el ATM Forum los emplean para este propósito. Una ventaja importante de los diagramas de clase UML es que ellos representan objetos y sus relaciones en una forma gráfica estándar, y los diseñadores y operadores pueden entenderlos más fácil. Además, UML tiene la ventaja de ser ampliamente adoptado en la industria, enseñado en las Universidades y está estandarizado por el OMG (Object Management Group).

Los Modelos de Datos definen objetos gestionados a un nivel de abstracción más bajo. Ellos incluyen detalles específicos de la implementación y el protocolo.

La relación entre un Modelo de Información y un Modelo de Datos se mostrado en la Figura 2.5.

Figura 2.5 Relación entre un Modelo de Información y un Modelo de Datos



Aunque los Modelos de Información y los Modelos de Datos son para diferentes propósitos, no siempre es posible definir precisamente que clase de detalles deben ser expresados en un modelo u otro. Hay aspectos en los que los Modelos de Información y los Modelos de Datos se traslapan, por lo que en algunos casos es muy difícil determinar a cual modelo pertenece una abstracción.

Modelo de Comunicación. Está relacionado con el intercambio de información a través de la red. Es definido por el protocolo de gestión, e incluye operaciones para monitoreo y control.

Modelo Funcional. Especifica las funciones de gestión que serán soportadas por el protocolo de gestión.

2.2 COMPARACIÓN ENTRE INFRAESTRUCTURAS DE GESTIÓN

Existen muchas infraestructuras de gestión: gestión OSI, TMN, SNMP, WBEM, JMX, CORBA, Jini, Jiro, Agentes Móviles, TINA, entre otras.

En este capítulo se hará una descripción y comparación de las seis primeras infraestructuras, con respecto a los cuatro modelos nombrados anteriormente, teniendo en cuenta que estas infraestructuras son las más conocidas, desarrolladas y utilizadas actualmente en el campo de la gestión.

2.2.1 Gestión OSI. Fue creada por la ISO y está orientada a la gestión de redes de datos (LANs y WANs). CMOT (Common Management Information Services over TCP/IP) fue un intento de utilizar gestión OSI en el entorno de Internet. Sin embargo, CMOT enfrentó los mismos problemas que la gestión OSI: es compleja, las especificaciones no salieron a tiempo y se hicieron muy pocas implementaciones. Como resultado, la gestión OSI y CMOT no tuvieron una gran aceptación.

- **Modelo Organizacional.** Sigue el paradigma gestor-agente fuertemente distribuido: las aplicaciones de gestión pueden actuar como gestores o agentes, en relaciones jerárquicas o punto a punto. En las relaciones jerárquicas los agentes pueden actuar como gestores para los agentes localizados en un nivel más bajo de la jerarquía, y distribuir el trabajo delegado por los gestores de un nivel más alto de la jerarquía a otros agentes.

  El modelo organizacional utiliza el concepto de dominio de gestión. Un dominio puede contener uno o más sistemas de gestión, sistemas gestionados y subdominios. La división del entorno de gestión OSI en dominios de gestión puede hacerse con base en requerimientos funcionales, tecnológicos, geográficos, entre otros.

- **Modelo de Información.** OSI tiene un modelo de información orientado a objetos, es decir, la información de gestión es modelada como objetos (también llamados objetos gestionados). Un objeto tiene atributos, que son las propiedades o características del objeto; operaciones, que pueden ser ejecutadas sobre el objeto; comportamiento, que es exhibido en respuesta a las operaciones; y notificaciones, las cuales son emitidas por el objeto. Un objeto puede representar uno o varios recursos, o un recurso puede ser representado por uno o varios objetos. Sin embargo, no todos los objetos representan recursos, algunos existen para satisfacer necesidades de gestión, como por ejemplo notificaciones y control de acceso.

  Anteriormente existía una diferencia entre objetos gestionados e información de gestión: los objetos gestionados se encuentran en varias capas del modelo de referencia OSI, mientras que la información de gestión reside en la MIB (Management Information Base). La MIB puede ser vista como una base de datos, cuyo contenido no son los objetos gestionados por sí mismos, sino la información asociada con ellos. Los Gestores de Capa, LMs (Layer Managers), son los responsables de mantener la asociación entre la información de la MIB y los objetos gestionados. Sin embargo, posteriormente se decidió remover la definición de información de gestión, cambiar la definición de objetos gestionados y cambiar la descripción de MIB. Como resultado de este cambio, en OSI no existe diferencia entre la información de gestión que se encuentra almacenada en una MIB y los objetos gestionados, porque el conjunto de objetos gestionados dentro de un sistema constituyen la MIB del sistema.

  Algunos autores consideran las MIBs como un Modelo de Datos.

  La SMI (Structure of Management Information) especifica la estructura, sintaxis y semántica de la información de gestión almacenada en la MIB.

  OSI emplea dos tipos de lenguajes de especificación: ASN.1 (Abstract Syntax Notation One) para definir tipos de datos y GDMO (Guidelines for the Definition of Managed Objects) para definir objetos gestionados.

La estructura que forman las instancias de los objetos gestionados se llama árbol de contenimiento. Cada instancia dentro del árbol tiene un RDN (Relative Distinguished Name) que está formado por un atributo de nombramiento de la instancia y su valor. La concatenación de todos los RDNs desde la raíz a una instancia se llama DN (Distinguished name). Los DNs no identifican confiablemente una instancia ya que ellos pueden ser reutilizados cuando una instancia ha sido borrada.

❖ **Modelo de Comunicación.** El modelo de referencia OSI define tres formas para intercambiar información de gestión: gestión de sistemas, gestión de capas y operación de capas.

- **Gestión de sistemas.** Utiliza protocolos de la capa de aplicación para el intercambio de información de gestión. Ellos utilizan servicios subyacentes orientados a la conexión. La decisión de utilizar protocolos de la capa de aplicación está basada en que la información de gestión se debe intercambiar en la misma forma que todas las otras formas de información.

  La capa de aplicación comprende el CMISE (Common Management InformationService Element) que está sobre ACSE (Association Control Service Element) y ROSE (Remote Operation Service Element). El CMISE está compuesto por el CMIS (Commom Management Information Service) y el CMIP (Commom Management Information Protocol). CMIS/CMIP opera sobre un pila de protocolos OSI completa de siete capas. En cuanto a los protocolos de red y transporte se puede utilizar una gran variedad de combinaciones incluyendo TCP/IP (CMOT).

  CMIS se puede descomponer en Entidades de Aplicación de Gestión de Sistemas, SMAEs (Systems Management Application Entities). Estas entidades contienen Elementos de Servicio de Aplicación, ASEs (Application Service Elements) y utilizan el proveedor del servicio de la capa de presentación para transferir sus datos. La interacción entre SMAEs es definido por CMIP (Commom Management Information Protoco).

  Ventajas:

  ✓ Los protocolos de la capa de aplicación son los más potentes. Un solo protocolo de la capa de aplicación será capaz de transferir muchos tipos de información de gestión, y será mucho mejor que definir muchos protocolos de gestión más simples.

  ✓ Todas las capacidades que proporcionan las capas OSI (detección y corrección de errores, segmentación, reensamblado, etc) satisfacen completamente las necesidades de gestión.

  Desventajas:

  ✓ Implementar las siete capas del Modelo de Referencia es costoso y hay muchos dispositivos como enrutadores, switches, entre otros, que para su operación normal no necesitan implementarlas. En estos sistemas puede ser un desperdicio de dinero implementar las capas restantes solo para permitir gestión.

  ✓ Después que una red colapsa se debe realizar una restauración de los servicios de red. Como resultado del colapso, los protocolos de la capa de aplicación no funcionarán, y por lo tanto no será posible realizar funciones de gestión.

  ✓ Los protocolos de la capa de aplicación involucran mucho procesamiento y son relativamente lentos.

- **Gestión de capas.** La gestión de la capa N soporta el monitoreo y control de los objetos gestionados de la capa N. Los protocolos de gestión de la capa N son soportados por protocolos de las capas N-1 e inferiores.

- **Operación de capas.** Al igual que en gestión de capas, la operación de capas utiliza los protocolos subyacentes para el intercambio de información de gestión, pero en este caso no se emplea un protocolo de gestión, sino que la información de gestión es transportada como parte de un protocolo normal de la capa.

❖ **Modelo Funcional.** OSI define cinco áreas funcionales: gestión de fallas, configuración, contabilidad, desempeño y seguridad. Ellas son llamadas Áreas Funcionales de Gestión Específicas, SMFAs (Specific Management Functional Areas) o FCAPS (Fault Configuration Account Perfomance Security)

OSI también define un conjunto de funciones, llamadas Funciones de Gestión del Sistema, SMF (System Management Functions), que son definidas a un nivel de abstracción más bajo que las FCAPS. Algunas de ellas son: función de gestión de objeto, función de gestión de estado, función de reporte de alarmas, función de gestión de reporte de eventos.

Cada área funcional tiene asociada a ella un conjunto de funciones. Las funciones que son aplicables a una SMFA pueden ser aplicables a otra.

2.2.2 TMN. Las recomendaciones TMN fueron desarrolladas por la ITU-T. TMN está orientada a las necesidades de los proveedores de servicio y operadores de red, y está diseñada para la gestión de redes y servicios de telecomunicaciones, como por ejemplo, SONET, SDH, ATM, redes telefónicas públicas conmutadas, redes inteligentes, redes de telefonía móvil, entre otras.

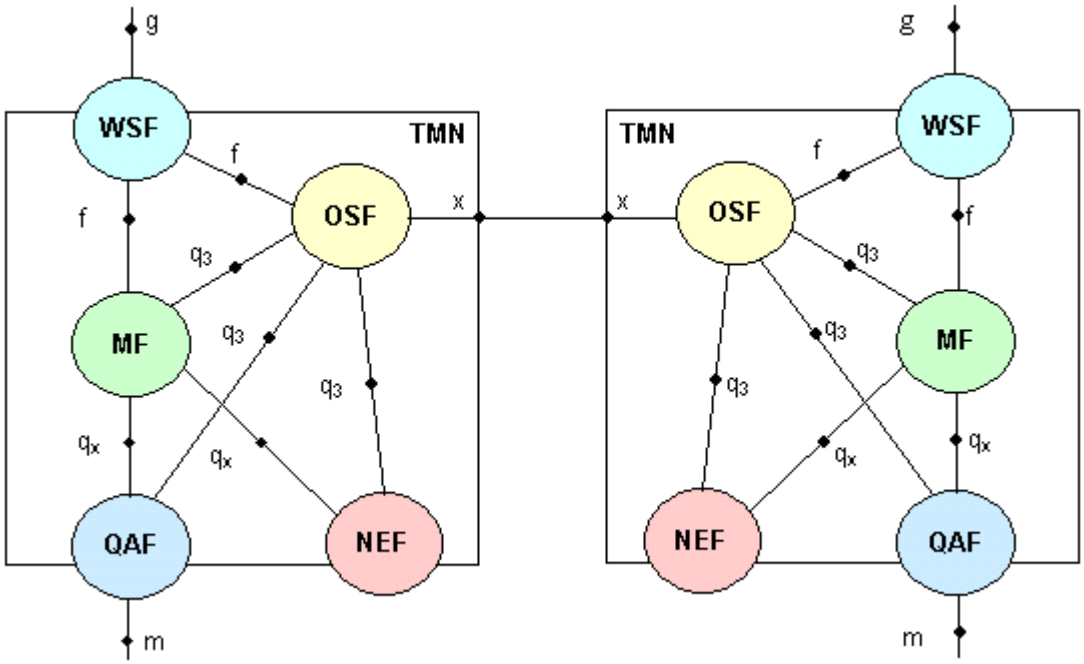
TMN consta de varias arquitecturas más pequeñas: una Arquitectura Funcional, una Arquitectura Física, una Arquitectura de información, y una Arquitectura Lógica que incluye un Modelo de Responsabilidad. Sin embargo, se tratará de relacionar los aspectos tratados en estas arquitecturas con los cuatro modelos definidos anteriormente.

❖ **Modelo Organizacional y de Información.** TMN utiliza gestión OSI como su infraestructura base, y por esto TMN incluye varios aspectos de la gestión OSI, algunos de ellos son: paradigma gestor-agente, el concepto de dominios de gestión, la aproximación orientada a objetos, el concepto de SMI, y la utilización de ASN.1 y GDMO como lenguajes de definición de tipos de datos y objetos gestionados respectivamente. Esto hace que el Modelo Organizacional y de Información de OSI también puedan ser aplicados a la infraestructura TMN.

❖ **Modelo de Comunicación y Funcional.** A pesar de que TMN no define específicamente un Modelo de Comunicación, existen varios aspectos en su Modelo Funcional como por ejemplo puntos de referencia y el componente funcional MCF (Message Communication Function), que definen como se intercambia la información de gestión. Es por esto que estos dos modelos no se describen en forma separada.

En TMN el Modelo Funcional se define en términos de bloques de funciones y puntos de referencia. Los bloques de funciones se pueden comparar con las entidades de protocolo OSI. Los puntos de referencia se utilizan para interconectar bloques de funciones y pueden, en terminología OSI, ser comparados a los proveedores de servicio subyacentes. La Figura 2.6 muestra un ejemplo de puntos de referencia entre bloques de funciones.

Figura 2.6 Puntos de referencia y bloques de funciones TMN



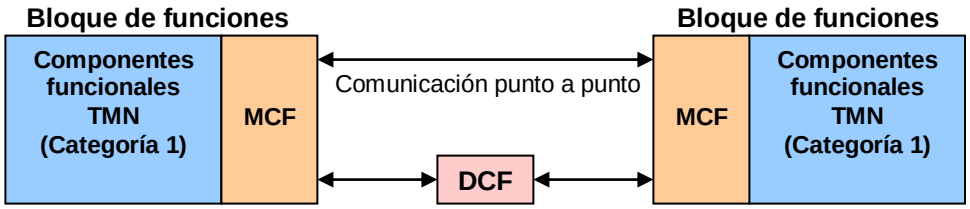
En TMN se definen cinco tipos diferentes de bloques de funciones, pero no es necesario que todos ellos estén presentes en cada configuración de TMN. De otro lado, la mayoría de configuraciones de TMN soportan varios bloques de funciones del mismo tipo.

En la Figura 2.6, los bloques de funciones OSF y MF se dibujan dentro del cuadro TMN. Esto indica que ellos están complemente especificados por las recomendaciones TMN. Los otros tres tipos de bloques de funciones (WSF, NEF y QAF) se dibujan en el límite del cuadro para indicar que TMN solo especifica parte de ellos. Cada bloque de funciones está compuesto por componentes funcionales, ellos son: Función de Aplicación de Gestión, Base de Información de Gestión, Función de Conversión de Información, Adaptación Humano Máquina, Función de Presentación y Función de Comunicación de Mensajes (MCF). Estos componentes funcionales se pueden dividir en dos categorías: los primeros cinco componentes pertenecen a la primera categoría. Ellos realizan las actividades de gestión reales y no tratan problemas relacionados con el intercambio de información de gestión. El último componente pertenece a la segunda categoría. Él está asociado con todos los bloques de funciones que requieren un servicio subyacente para el intercambio de información de gestión. Está compuesto de una pila de protocolos que permite la conexión de bloques de funciones a la Función de Comunicación de Datos, DCF (Data Communication Function). El MCF proporciona las funciones de las capas 4 a 7 del Modelo de Referencia OSI, y la DCF proporciona las capas 1 a 3.

En los drafts iniciales de TMN, el DCF era modelado como un bloque de funciones, por lo tanto era parte de la arquitectura funcional de TMN. Actualmente el DCF no se considera como un bloque de funciones.

La Figura 2.7 muestra la relación entre bloques de funciones, componentes funcionales, MCF y DCF.

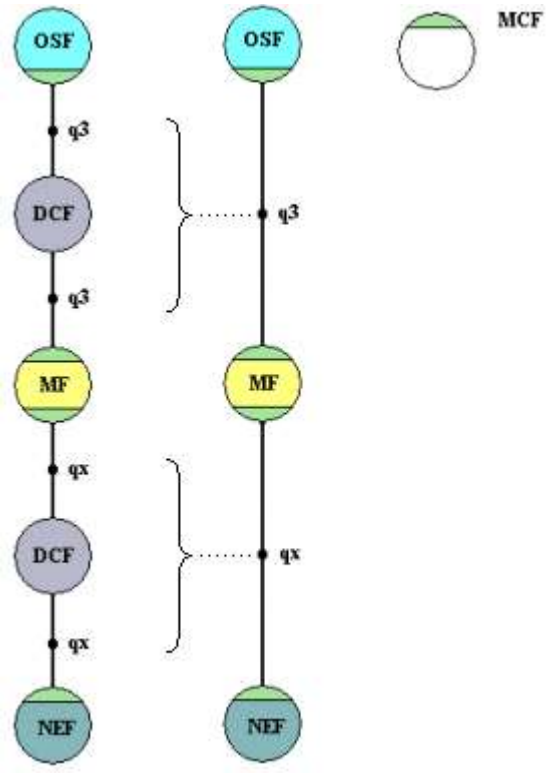
Figura 2.7 Relación entre bloques de funciones, componentes funcionales, MCF y DCF



En cuanto a los puntos de referencia existen cinco clases diferentes. Tres de ellos (q, f y x) se describen complemente en las recomendaciones TMN, las otras clases (g y m) se describen parcialmente. Anteriormente, existían tres puntos de referencia q: q1, q2 y q3. Sin embargo, después de un tiempo, se determinó que no había ninguna distinción entre q1 y q2. Por lo tanto, estos dos puntos de referencia se reemplazaron por el punto de referencia qx.

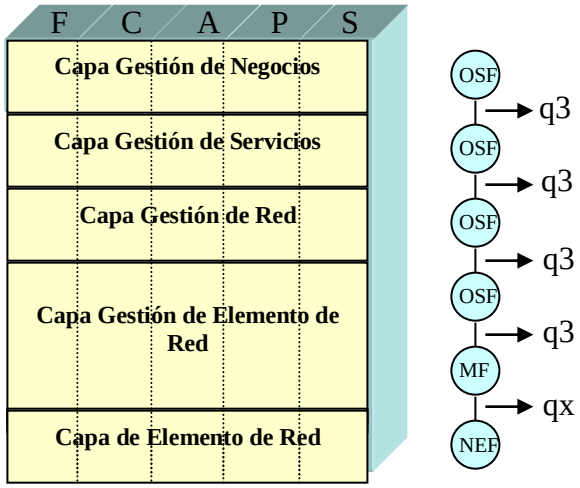
La Figura 2.8 muestra la relación entre bloques de funciones, componentes funcionales, MFC, DFC y puntos de referencia.

Figura 2.8 Relación entre bloques de funciones, componentes funcionales, MCF, DCF y puntos de referencia



El Modelo Funcional TMN cubre las áreas funcionales definidas en OSI (configuración, fallas, desempeño, contabilidad y seguridad). Esta funcionalidad de gestión se puede organizar en las diferentes capas del Modelo de Responsabilidad de gestión de TMN: gestión de elemento, red, servicio y negocios. La Figura 2.9 muestra la relación entre el Modelo de Responsabilidad, bloques de funciones, puntos de referencia y las FCAPS.

Figura 2.9 Jerarquía funcional TMN y FCAPS



- **Arquitectura Física.** Define que los bloques de funciones se deben implementar como equipo físico (o bloques de construcción), y los puntos de referencia como interfaces.

  La Arquitectura Física de TMN define los siguientes bloques de construcción: Elemento de Red, NE (Network Element); Dispositivo de Mediación, MD (Mediation Device); Adaptador Q , QA (Q Adaptor); Sistema de Operaciones, OS (Operations System); Estación de Trabajo, WS (Work Station) y Red de Comunicación de Datos, DCN (Data Communication Network). Los bloques de construcción siempre implementan los bloques de funciones del mismo nombre.

  Es posible implementar varios bloques de funciones (del mismo o de diferente tipo) en un solo bloque de construcción. El Sistema de Operaciones, por ejemplo, se puede utilizar para implementar múltiples OSFs, pero también se puede utilizar para implementar un OSF, un MF y un WSF. En caso que un bloque de construcción implemente varios bloques de funciones de diferentes tipos, el nombre del bloque de construcción se determina por la utilización predominante del bloque.

  La Tabla 2.2 muestra cuales bloques de funciones se pueden implementar en cada bloque de construcción.

Tabla 2.2 Correspondencia entre bloques de funciones y bloques de construcción

	NEF	MF	QAF	OSF	WSF
NE	M	O	O	O	O*
MD		M	O	O	O
QA			M		
OS		O	O	M	O
WS					M

M: Obligatorio
O: Opcional
O*: puede solamente estar presente si OSF o MF también están presentes.

Una clase especial de bloque de construcción es la Red de Comunicación de Datos. A diferencia de los otros, este bloque de construcción no implementa ningún bloque de funciones. De hecho, la DCN es utilizada por otros bloques de construcción para el intercambio de la información de gestión, es decir, la tarea de la DCN es actuar como una red de transporte.

A primera vista parece extraño que TMN defina un bloque de construcción que no implementa ningún bloque de funciones. La existencia de la DCN se entiende cuando se recuerda que los drafts TMN previos modelaron la DCF como un bloque de funciones. De acuerdo a estos drafts, la DCF tenía que ser implementada por una DCN, y en este caso, cada bloque de construcción implementaba al menos un bloque de funciones. Después se decidió que el DCF no seguiría siendo un bloque de funciones, pero después de que se hizo esta decisión, el estándar no fue reescrito en una forma consistente y la DCN siguió siendo un bloque de construcción.

Las interfaces pueden ser consideradas como las implementaciones de los puntos de referencia de TMN. Mientras los puntos de referencia pueden ser comparados con los servicios subyacentes, las interfaces pueden ser comparadas con la pila de protocolos que implementa estos servicios.

En la mayoría de los casos, los puntos de referencia y las interfaces tienen un mapeo uno o uno. Sin embargo, no existen interfaces para los puntos de referencia que:

- Comunican bloques de funciones que son implementados dentro de un solo bloque de construcción.
- Están fuera del alcance de TMN, como es el caso de los puntos de referencia g y m.

Cada interfaz tiene el nombre del punto de referencia relacionado pero escrito en letras mayúsculas.

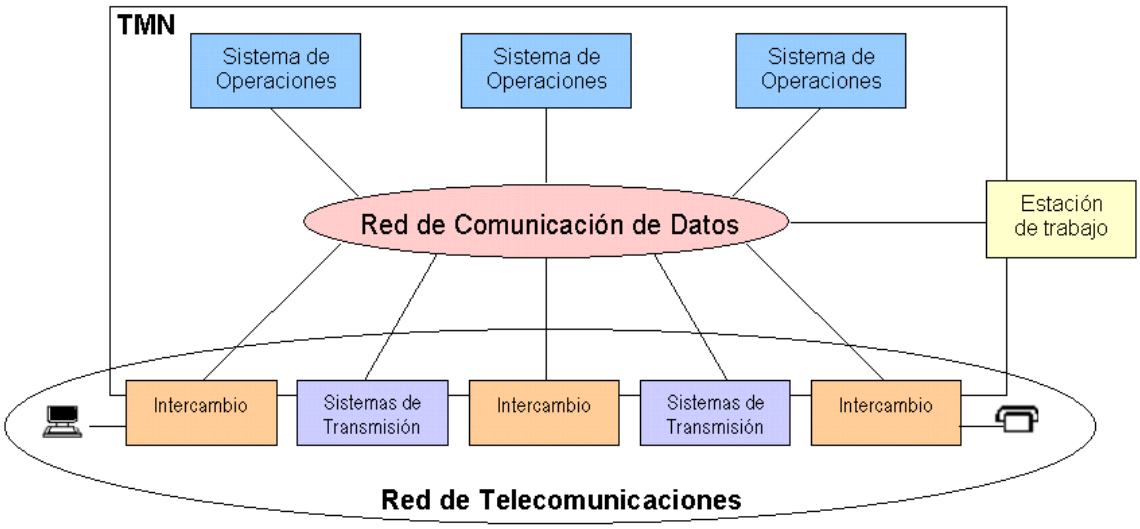
Comparación entre TMN y OSI

- OSI definió una sola arquitectura, mientras que TMN definió varias arquitecturas en diferentes niveles de abstracción. La Arquitectura Funcional de TMN define las funciones de gestión, mientras que la Arquitectura Física muestra como estas funciones pueden ser implementadas en equipo físico. Por lo tanto la Arquitectura Física está definida a un nivel de abstracción más bajo que la Arquitectura Funcional.

- TMN define una Arquitectura Lógica (Modelo de Responsabilidad) que permite entender y distinguir más fácil las diferentes responsabilidades de gestión.

- TMN hace distinción entre la red que está siendo gestionada (la red de telecomunicaciones) y la red que transfiere la información de gestión (la red de comunicación de datos), como se muestra en la Figura 2.10.

Figura 2.10 Relación entre TMN y una red de comunicación



De acuerdo a la Figura 2.10, los puntos de interfaz entre TMN y la red de telecomunicaciones son formados por sistemas de Intercambio y Transmisión. Ellos se conectan a través de una red de comunicación de datos a uno o más Sistemas de Operaciones, que realizan la mayoría de funciones de gestión. La red de comunicación de datos también permite la comunicación entre Sistemas de Operación (es posible que una función de gestión sea ejecutada por varios Sistemas de Operaciones), y la conexión de estaciones de trabajo

Esta separación evita los problemas con la gestión de fallas que tiene la gestión OSI, ya que sin importar los problemas en la red gestionada, siempre se podrá acceder los componentes que están fallando. Por lo tanto TMN tiene mejores capacidades de gestión de fallas que OSI.

Sin embargo, una red de comunicación de datos requiere la introducción de equipos y sistemas de transmisión adicionales. Además, las fallas en la red de comunicación de datos no pueden ser excluidas, lo cual implica que es necesario gestionarla también. Por lo tanto los costos de introducir una red de datos serían altos.

Existen razones para introducir una red de comunicación de datos. Una razón importante podría ser, por ejemplo, que la red gestionada no proporciona facilidades adecuadas para transferir información de gestión. Este es el caso de las redes de telefonía, las cuales proporcionan un tipo de servicio isócrono. Este tipo de servicio no corresponde al tipo de servicio asíncrono (orientado a paquete) que es requerido por la mayoría de los protocolos de gestión. En estos casos una red de comunicación de datos sería indispensable para gestionar tales clases de redes.

2.2.3 Gestión basada en SNMP. SNMP fue creado por la IETF (Internet Engineering Task Force) y está orientado a la gestión de Internet. Aunque tiene varias deficiencias, y no es adecuado para hacer tareas de gestión complejas, se ha convertido en el estándar de facto para la gestión de redes debido a su simplicidad.

SNMP define específicamente un Modelo de Información y un Modelo de Comunicación, pero no define un Modelo Organizacional ni un Modelo Funcional. Sin embargo, para realizar la comparación con otras infraestructuras de gestión se tratará de asociar los diferentes aspectos de SNMP a los cuatro modelos definidos anteriormente.

- **Modelo Organizacional.** SNMPv1 sigue el paradigma gestor-agente centralizado, es decir un gran número de sistemas son gestionados por un solo gestor. Además, ya que un agente responde a cualquier sistema de gestión que se comunique apropiadamente con él utilizando SNMP, varios gestores pueden comunicarse con uno a varios agentes.

  SNMP utilizando la MIB de Monitoreo Remoto, RMON (Remote Monitoring), es un ejemplo de gestión débilmente distribuida. En el modelo gestor-agente centralizado, el gestor recibe datos no procesados de los agentes y los procesa. Sin embargo, cuando se monitorea las estadísticas de tráfico, es mejor que el gestor obtenga datos procesados. Para lograr esto, en lugar de que el gestor continuamente monitoree los eventos y procese la información, se inserta un agente intermedio llamado RMON (Remote Monitoring) entre el gestor y el elemento de red para que realice estas tareas.

  En el modelo gestor-agente centralizado, el gestor está limitado en cuanto al número de agentes que puede manejar debido a que debe consultar periódicamente cada sistema bajo su control, lo cual toma tiempo. Para solucionar este problema, SNMPv2 sigue un paradigma fuertemente distribuido: las consultas son realizadas por un conjunto de gestores de nivel intermedio, que pueden actuar como gestores o agentes, bajo control del gestor del nivel más alto, reduciendo el tráfico de red y la carga de procesamiento. Antes de que los gestores de nivel intermedio comiencen a realizar las consultas, el gestor de nivel más alto les dice cuales variables se quieren obtener y en que agentes, y los eventos acerca de los cuales él quiere ser informado. Después de que los gestores de nivel intermedio son configurados, ellos comienzan su tarea. En caso de que un gestor de nivel intermedio detecte en un agente particular, un evento en el cual el gestor de nivel más alto está interesado, genera una PDU Inform. Después de la recepción de la PDU, el gestor opera directamente sobre el agente que causó el evento. En SNMPv2 también son posibles las relaciones gestor-gestor (a través del mensaje Inform).

  SNMPv3 también utiliza el modelo fuertemente distribuido. La arquitectura de SNMPv3 está formada por una colección distribuida de entidades SNMP. Estas entidades implementan una porción de la funcionalidad de SNMP y pueden actuar como gestores, agentes o los dos. Cada entidad SNMP consta de una Máquina SNMP y un conjunto de aplicaciones. La máquina SNMP es una colección de cuatro subsistemas: Subsistema de Transmisión, Subsistema de Procesamiento de Mensajes, Subsistema de Seguridad y un Subsistema de Control de Acceso, que interactúan para proporcionar servicios. Esta arquitectura modular permite mucha flexibilidad en la especificación e implementación de SNMP. En cuanto a las aplicaciones, SNMPv3 define formalmente cinco tipos: Command Generador, Notification Originator, Proxy Forwarder, Command Responder y Notification Receiver.

- **Modelo de Información.** El modelo de información en SNMP no es orientado a objetos, porque no tiene el concepto de clases, solo de variables, y consta de MIBs que describen los recursos gestionados. Una MIB es una colección de información organizada jerárquicamente. Esta información incluye, variables, tablas, valores que estas variables y tablas pueden tomar, y los nombres globales utilizados para acceder ésta información. Algunos autores consideran que las MIBs definidas por la IETF son un Modelo de Datos.

  La Estructura de Información de Gestión, SMI (Structure of Management Information) define las reglas para describir la información de gestión de las MIBs utilizando un subconjunto de ASN.1 (Abstract Syntax Notation One). La SMI especifica que todos los objetos gestionados deben tener asociado a ellos un nombre (OID), una sintaxis (ASN.1) y una codificación (BER: Basic Encoding Rules). El nombre sirve como un identificador de objeto, la sintaxis define los tipos de datos del objeto, y la codificación de datos describe cómo la información se transmite por la red.

  Para garantizar la identificación única de cada variable de gestión, SMI introduce el concepto de árbol de nombramiento. Las hojas de este árbol representan la información de gestión real. Una nueva MIB se crea al definir un número de variables en una posición determinada en el árbol de nombramiento. Estas variables son la información que un agente SNMP proporcionará.

❖ **Modelo de Comunicación.** La comunicación entre gestores y agentes se hace a través de diferentes tipos de mensajes enviados asincrónicamente dentro de diferentes PDUs SNMP. Se pueden incorporar varios requerimientos en una PDU siempre y cuando ellos sean del mismo tipo.

SNMP ha sido diseñado para soportar un estilo de comunicación no orientado a la conexión. Esto se debe a:

- Los protocolos confiables como TCP demandan mucha memoria y recursos de procesamiento para ser soportados por dispositivos de red simples como enrutadores y concentradores. Sin embargo los avances en capacidades de procesamiento, y los bajos costos de memoria sugieren que el argumento de simplicidad en los recursos gestionados no sigue siendo válido.
- Mantener conexiones de transporte requiere información de estado, y ya que SNMPv1 sigue un modelo centralizado, es imposible mantener simultáneamente miles de conexiones desde un centro de gestión a los dispositivos gestionados.
- Los protocolos no orientados a la conexión están diseñados de acuerdo a la aproximación del mejor esfuerzo. Esto permite que la gestión sea posible, aunque en forma limitada, cuando se presentan problemas con el proveedor de servicio de transporte, porque aún en caso de fallas, algunos datos pueden llegar al destino. Con un protocolo orientado a la conexión, eso no es posible, porque ellos son diseñados de acuerdo a una aproximación todo o nada: o todos los datos son entregados o nada es entregado. Si los datos no pueden ser entregados, la conexión será liberada.

No existen requerimientos para el protocolo sobre el que está SNMP, pero está diseñado para correr sobre UDP, y por lo tanto éste es el más utilizado. Sin embargo hay RFCs que definen SNMP sobre otros protocolos, como por ejemplo Ethernet, IPX e incluso protocolos OSI.

Una característica de UDP es que los paquetes no pueden exceder cierto tamaño. Para garantizar que solamente se generen paquetes de tamaño limitado, el protocolo SNMP ha definido un conjunto de reglas. Una de ellas es que si la respuesta a un cierto requerimiento SNMP excede el tamaño máximo de paquete, la información no es retornada. Los gestores deben tener en cuenta esto, y en lugar de hacer un solo requerimiento abarcando todo, deben hacer varios requerimientos más pequeños para conseguir la información pieza por pieza. Desafortunadamente, los gestores en muchos casos no serán capaces de predecir la cantidad de información que se puede obtener a través de un solo requerimiento. Para mejorar el desempeño, SNMPv2 introdujo el requerimiento GetBulk. Al contrario de Get y GetNext, la respuesta al requerimiento GetBulk siempre retorna tanta información como sea posible. Si la información requerida excede el máximo tamaño de un paquete UDP, solamente será retornada la parte de información que permita el paquete.

Ya que los gestores no pueden determinar el tamaño preciso de los paquetes de respuesta, usualmente requieren por PDU solo una pequeña cantidad de datos. Para obtener toda la información, los gestores pueden necesitar realizar un gran número de requerimientos consecutivos.

La utilización de un protocolo no orientado a la conexión tiene las siguientes implicaciones:

- No hay garantía que los PDUs alcancen su destino, por esto el gestor tiene la responsabilidad de detectar pérdida de datos e iniciar retransmisión. Sin embargo, la experiencia ha mostrado que es muy difícil optimizar la retransmisión en la misma forma que es hecho por los protocolos de transporte confiables.
- Los gestores deben realizar alguna clase de consulta para detectar si los agentes son todavía operacionales. Con proveedores orientados a la conexión (por ejemplo, el servicio de presentación de OSI) esto no sería necesario, porque los proveedores ya incluyen funciones de control de tiempo de vida. En OSI, estas funciones son parte del protocolo de transporte orientado a la conexión. Ellas revisan periódicamente si los sistemas remotos (agentes, en este

caso) están operando. En caso de que no, el proveedor (gestor, en este caso) toma la iniciativa de liberar la conexión e informa al usuario.

- **Modelo Funcional.** Como se dijo anteriormente, SNMP no define un modelo funcional, la funcionalidad de gestión está implícitamente contenida en las MIBs. Por ejemplo, la primera MIB estandarizada, también considerada la más importante, la MIB-II (RFC 1213), se utiliza principalmente para gestión de configuración, fallas y desempeño.

  Ahora que se han definido miles de variables de gestión, tanto en MIBs estandarizadas como en MIBs específicas de empresas, la falta de una estructura funcional para clasificar estas variables ha llegado a ser un problema. Esto se debe a que SNMP sólo permite acceder las variables de gestión, y no define nada con respecto a cuales de ellas deben ser monitoreadas o modificadas, estas decisiones deben ser tomadas por el administrador. Por esto, sin una estructura adecuada, los administradores deben entender el significado preciso de un gran número de variables, para determinar cuales de ellas se deben monitorear y que modificaciones se deben hacer. Como consecuencia, los administradores necesitan bastante tiempo antes de tomar una decisión con respecto a qué hacer.

Comparación entre OSI, TMN y SNMP

- La gestión basada en SNMP es simple y poco costosa, por lo que se ha convertido en el estándar de facto para la gestión de redes. OSI y TMN son complejas, y por lo tanto difíciles de implementar y costosas. Ésto, unido a su lento proceso de estandarización, han hecho que OSI y TMN no hayan ganado la misma aceptación que SNMP.

- En SNMP existe diferencia entre objetos gestionados e información de gestión. Los objetos gestionados son todos los recursos de procesamiento y comunicación de datos que pueden ser gestionados a través del protocolo SNMP. La información de gestión consta de variables que tienen un valor que puede ser leído o modificado. Las MIBs SNMP no almacenan información, solo definen que datos se pueden obtener desde un dispositivo. Cuando el gestor solicita información al agente, éste utiliza hardware o software del dispositivo para obtener los datos. Esta información no se almacena en la MIB. En OSI no existe diferencia entre la información de gestión que se encuentra almacenada en una MIB y los objetos gestionados, porque el conjunto de objetos gestionados dentro de un sistema constituyen la MIB del sistema.

- La gestión basada en SNMP tiene un Modelo de Información no orientado a objetos, utiliza protocolos no orientados a la conexión y está basada en consultas. La gestión OSI y TMN utiliza una aproximación orientada a objetos, protocolos orientados a la conexión y está basada en eventos.

- SNMP fue diseñado para colocar la carga de gestión en el gestor, con el fin de mantener el agente simple. En OSI y TMN se distribuye la carga de una forma más uniforme a través de agentes más inteligentes y por lo tanto más complejos, pero como consecuencia los dispositivos necesitan más recursos y capacidades.

- SNMP, a diferencia de la gestión OSI y TMN, no define un estándar arquitectural separado para describir los conceptos que existen detrás de SNMP, porque estos conceptos son parecidos a los que ya están descritos en los drafts de la arquitectura de gestión OSI y fueron consideraros obvios.

- SNMP sólo permite acceder las variables de gestión, y no define nada con respecto a cuales de ellas deben ser monitoreadas o modificadas. OSI, a través de CMIS, especifica los servicios básicos necesarios para realizar varias funciones de gestión.

- CMIS/CMIP está diseñado para correr sobre la pila de protocolos OSI, lo que produce más carga de procesamiento en comparación a SNMP.

2.2.4 WBEM. La infraestructura WBEM fue desarrollada por el DMTF y puede ser aplicada a casi cualquier área 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. Para realizar la comparación con otras infraestructuras de gestión, se tratará de asociar estos estándares a cada uno de los modelos definidos anteriormente. Sin embargo, debido a que WBEM fue descrito en el capítulo 1, solo se hará una pequeña descripción de cada uno de ellos.

- **Modelo Organizacional.** WBEM sigue un paradigma gestor-agente fuertemente distribuido.

- **Modelo de Información.** El estándar CIM es el Modelo de Información de WBEM. Los esquemas CIM son orientados a objetos y son publicados en dos formas: gráfica y textual. La forma gráfica utiliza diagramas UML, y no son obligatorios porque no todos los detalles pueden ser representados gráficamente. La forma textual, considerada más como un Modelo de Datos, está escrita en un lenguaje llamado Formato de Objetos Gestionados, MOF (Managed Object Format).

- **Modelo de Comunicación.** Estándar Operaciones CIM sobre HTTP y xmlCIM.

- **Modelo Funcional.** WBEM divide la gestión en varias áreas, definidas en el Modelo Común del Esquema CIM. Hasta ahora se han establecido las siguientes: Aplicaciones, Bases de Datos, Dispositivos, Eventos, Modelo de Interoperabilidad, Métricas, Red, Físico, Políticas, Soporte, Sistemas y Usuarios.

  Desde el punto de vista OSI, se puede decir que WBEM cubre las FCAPS.

Ya que WBEM ha sido la infraestructura seleccionada para realizar la aplicación de gestión, y SNMP se ha convertido en el estándar de facto para la gestión de redes, a continuación se hará una comparación de estos dos estándares.

Comparación entre WBEM y SNMP

- **Modelos de Información**

  - SNMP tiene un Modelo de Información que describe prácticamente todo como lo hace CIM, pero la diferencia es que el Modelo de Información de SNMP está distribuido en miles de diferentes MIBs, mientras CIM es un solo modelo. Esto garantiza interoperabilidad, porque todos los desarrolladores de aplicaciones y servicios, y fabricantes de dispositivos, utilizarán un solo modelo para describir sus productos.
  - Las MIBs de SNMP permiten representar los recursos, pero no las relaciones que existen entre ellos. CIM ofrece la habilidad para definir asociaciones entre componentes, y su aproximación orientada a objetos, hace más fácil identificar las relaciones e interdependencias entre recursos gestionados.
  - SNMP solo puede hacer lo que MIB permite y la MIB es estática. Por lo tanto la flexilibilidad no es buena. Además las MIBs SNMP no aseguran compatibilidad entre equipos similares de diferentes fabricantes, lo cual significa que es difícil adaptar aplicaciones de gestión SNMP a diferentes entornos. CIM es un modelo extensible que se puede adaptar fácilmente a nuevas necesidades de gestión.

- **Integración con otras infraestructuras de gestión.** WBEM tiene la ventaja de soportar mapeos a otros estándares de gestión, y esto significa que por ejemplo, los dispositivos habilitados con SNMP pueden ser gestionados a través de WBEM. Como SNMP no tiene un modelo de información como CIM, el no sería capaz de mapear todas las operaciones de WBEM, para gestionar elementos WBEM. Aunque las conversiones de mensajes al formato CIM adicionarán procesamiento en los recursos gestionados (porque ellas tienen que hacerse en el proveedor, antes de pasar la información al CIMOM), ésto permite a WBEM unificar los estándares de gestión

existentes, con solo adicionar un proveedor WBEM a los recursos existentes, en lugar de reemplazarlos por unos complemente nuevos.

- **Nombramiento**. SNMP utiliza un sistema de nombramiento basado en OIDs, el cual puede rápidamente llegar a ser ilegible, y son difíciles de recordar. WBEM emplea un trayecto de objeto para identificar una instancia particular que es mucho más fácil de mantener en mente. Está compuesto por el "namespace" en el cual reside la instancia, el nombre de la clase de la cual el objeto es una instancia, y los pares clave-valor que forman la clave única para la instancia.

  Una ventaja con el sistema OID es que garantiza un unicidad global, pero con CIM se puede conseguir una unicidad similar, siempre y cuando los desarrolladores registren su esquema de extensión con el DMTF.

- **Codificación de mensajes.** Todos los mensajes SNMP son codificados en BER (Basic Encoding Rules), un esquema de codificación simple, construido con base en el esquema TLV (Tag, Length, Value). Esto significa que cada valor es codificado con una etiqueta que dice qué clase de dato es representado, un campo de longitud indicando el tamaño de los datos, y finalmente los datos. La codificación BER de los mensajes SNMP es muy simple, ya que solo es necesario conocer el tipo, tamaño y valor de un objeto para codificarlo; y compacta, porque la etiqueta y el campo longitud son de un byte cada uno, produciendo solamente dos bytes de carga de procesamiento a cada valor. La información CIM codificada en XML, requiere buen conocimiento de CIM y XML, producirá más carga de procesamiento que SNMP, y demandará más recursos computacionales a los servidores y aplicaciones de gestión. Sin embargo, XML se está convirtiendo en el formato estándar para el intercambio de datos debido a que es extensible y permite que aplicaciones desarrolladas en diferentes lenguajes, y máquinas con diferentes sistemas operativos interactúen.

- **Transporte de datos.** SNMP corre sobre UDP, mientras HTTP corre sobre TCP. Esto significa que SNMP no puede garantizar entrega confiable de los mensajes enviados, mientras WBEM si. Sin embargo, con UDP, por ser un protocolo no orientado a la conexión y utilizar la aproximación del mejor esfuerzo, se pueden realizar tareas de gestión aunque en forma limitada, cuando se presentan problemas con el proveedor de servicio de transporte.

- **Desempeño.** En SNMP, un cliente debe tener conocimiento previo del modelo de objetos utilizado por el elemento gestionado. Para hacer esto, el elemento gestionado debe tener acceso a una MIB describiendo este modelo. El cliente puede encontrar su estructura a través de la utilización repetida de la operación Get-next o Get-bulk. Un cliente WBEM no necesita tener conocimiento del modelo de objetos, porque éste se puede adquirir a través de llamadas a métodos de enumeración de clases e instancias. Esto tiene la ventaja que el cliente solamente tiene que realizar una llamada a través de la red, mientras que con SNMP se tendría que realizar numerosas llamadas para conseguir toda la información requerida.

- **Ancho de banda.** WBEM requiere un ancho de banda mucho mayor que SNMP, debido a la utilización de HTTP y XML. Sin embargo, el impacto sobre la red depende de las tareas de gestión que se estén realizando. Por ejemplo, ya que la gestión de configuración se realiza a intervalos irregulares, la carga que pone sobre la red es casi imperceptible, mientras que la gestión de alarmas podría forzar un incremento en la capacidad en muchas redes. Esto se debe al hecho que las alarmas a menudo llegan en ráfagas, porque una sola falla en un elemento de red podría generar nuevas fallas que también generan alarmas, y esto podría producir mucha congestión en la red. Sin embargo WBEM reduce el número de alarmas enviadas a través de la configuración de filtros en el elemento gestionado, y el problema de la congestión se puede evitar.

- **Complejidad.** La simplicidad de SNMP es su fortaleza más grande. Por lo tanto, en esta área, WBEM no puede competir con SNMP.

  Existen siete operaciones de protocolo diferentes en SNMPv2: Get, GetNext, Set, Trap, GetBulk, Inform y Reply. SNMPv3 adiciona un nuevo tipo de Trap y un mensaje Report. WBEM tiene

solamente cuatro operaciones de protocolo: SIMPLEREQ, SIMPLERSP, MULTIREQ y MULTIRSP. Considerando esto, WBEM es menos complejo en términos del número de operaciones de protocolo que tienen que ser implementados. Pero HTTP es más complejo que UDP, por lo que la comunicación en WBEM es de todas formas más compleja que la comunicación en SNMP. En cuanto a los tipos de mensajes, SNMP tiene los mismos tipos de mensajes que las operaciones de protocolo nombradas anteriormente, mientras que WBEM soporta 23 métodos intrínsicos más un número arbitrario de métodos extrínsecos. Esto significa que tanto el cliente y el servidor WBEM, deben ser capaces de entender un número mucho más grande de tipos de mensajes que SNMP. Esto hace que el desarrollo de las aplicaciones WBEM sea más complicado y que requieran más procesamiento.

El soporte de WQL también adiciona complejidad al CIMOM, porque él tiene que realizar la conversión de los requerimientos WQL. Para reducir la complejidad, WQL tiene cuatro niveles, que soportan diferentes métodos SQL, con el fin de que el servidor WBEM pueda decidir qué nivel quiere soportar.

- **Seguridad.** SNMPv2 proporciona la misma seguridad que SNMPv1, la cual está basada en el concepto de comunidad. Este esquema da el mismo nivel de seguridad que proporciona la autenticación básica de HTTP, es decir, esto protege en un entorno de red seguro, pero no tiene mucho que ofrecer si una persona con alguna forma de acceso a la red realiza un ataque. SNMPv3 ofrece grandes mejoras en el área de seguridad de SNMP a través del Modelo de Seguridad Basado en Usuario, USM (User-based Security Model). Este modelo soporta autenticación y cifrado de mensajes. SNMPv3 ofrece la opción de seleccionar el nivel de seguridad que sea más adecuado para una operación específica, puede utilizarse sin seguridad, con autenticación y sin cifrado, o con autenticación y cifrado. Comparando el USM de SNMPv3 a SSL y SHTTP, los dos últimos ofrecen más flexibilidad para la selección de algoritmos que se pueden emplear. Esta flexibilidad, hace mucho más difícil a una persona realizar un ataque, porque el método de atacar debe cambiar para cada posibilidad. Además SNMPv3 no ofrece soporte para firmas digitales, mientras que SSL y SHTTP si.

  Otro aspecto es que el USM de SNMPv3 no tiene soporte para compresión de datos, mientras que SSL si, lo que reduce el consumo de ancho de banda. Sin embargo, al utilizar la codificación BER, los datos en SNMP ya están lo suficientemente compactos y no se tendría mucha ganancia al utilizar compresión de datos.

  El modelo de seguridad en SNMPv3 también proporciona un mecanismo de control de acceso llamado Modelo de Control de Acceso basado en Vista, VACM (View-based Access Control Model), que permite la configuración de diferentes derechos de acceso para los usuarios. Por ejemplo, el VACM permite dar acceso de escritura a un administrador, y acceso de lectura a un grupo de otros usuarios. WBEM no proporciona ningún modelo estándar para los diferentes niveles de acceso. Es tarea de la persona que implementa el CIMOM, decidir como modelar los diferentes niveles de acceso que tienen los diferentes usuarios.

Se puede observar que WBEM es superior en la mayoría de aspectos a SNMP, el estándar más difundido actualmente para la gestión de redes. Además, la mayor fortaleza de SNMP, su simplicidad, se ve reflejada en menores capacidades funcionales con respecto a otros estándares como WBEM. Adicionalmente, los dispositivos SNMP pueden gestionarse a través de WBEM (utilizando un proveedor SNMP), por lo que se puede decir que SNMP no es reemplazado por WBEM sino que extiende sus capacidades.

2.2.5 CORBA. CORBA es un estándar abierto creado por el OMG (Object Managemet Group) para crear, distribuir y gestionar objetos distribuidos. CORBA permite la interconexión de objetos y aplicaciones, independientemente del lenguaje de las aplicaciones que utilizan los objetos, el sistema operativo de los equipos involucrados y la localización geográfica de los mismos.

CORBA se puede utilizar para gestionar un sistema, esto se puede lograr adicionando objetos CORBA al sistema cuya tarea es manejar otros objetos, o adicionando operaciones a interfaces CORBA existentes para hacerlas gestionables.

- **Modelo Organizacional.** CORBA sigue el modelo cliente-servidor cooperativo. Los clientes pueden ser vistos como programas que invocan métodos (o hacen requerimientos) sobre objetos CORBA, mientras los servidores son programas que implementan estos objetos y ejecutan las operaciones asociadas con ellos. CORBA no tiene el concepto de jerarquía pero asume relaciones punto a punto simétricas entre objetos. En términos de gestión, esto resulta en una extensión del modelo de roles: no solamente son posibles relaciones gestor-agente o gestor-gestor, sino también relaciones agente-agente. Esto simplifica la distribución y la delegación de la funcionalidad de gestión y soporta cooperación entre diferentes componentes de la infraestructura de gestión. Por lo tanto, no solamente es posible delegar funcionalidad desde sistemas gestores a sistemas gestionados, sino también entre sistemas gestionados.

  Otro aspecto importante es que los roles de los sistemas participando en actividades de gestión están sujetos a cambios dinámicos: los roles como sistema gestor y sistema gestionado son temporales y dependen de las tareas que tengan que realizarse. Esto tiene implicaciones de seguridad, porque mientras en la arquitectura cliente-servidor tradicional, un cliente puede confiar en el servidor, con objetos distribuidos esto no es posible debido al frecuente cambio de roles entre clientes y servidores CORBA. Por lo tanto, es muy importante proporcionar mecanismos de seguridad en el lado del cliente y servidor para evitar ejecución no autorizada de funcionalidad, es decir, un objeto debe tener la habilidad para revisar si un cliente está autorizado para ejecutar funciones sobre el objeto.

- **Modelo de Información.** La descripción de objetos CORBA se establece al especificar sus interfaces en el Lenguaje de Definición de Interfaces, IDL (Interface Definition Language). IDL es un medio de presentar en una manera uniforme las interfaces de todos los objetos conectados a un ORB.

  IDL permite definir interfaces, junto con sus atributos y operaciones, pero no puede utilizarse para implementar una interfaz. Para esto, el código IDL tiene que ser traducido a un lenguaje de implementación como C++ o Java. IDL fue diseñado solamente como lenguaje de especificación, con el fin de permitir la utilización de varios lenguajes de implementación, y lograr de esta forma independencia del lenguaje. Por lo tanto es posible que un cliente acceda un objeto desde un lenguaje diferente del cual el objeto fue implementado.

  Si se introducen nuevas implementaciones de objetos al sistema de Runtime del ORB, sus interfaces se registran en el Repositorio de Interfaces. El Repositorio de Interfaces se puede pensar como una base de datos distribuida, la cual contiene las descripciones de interfaz de cualquier objeto conocido por el ORB.

- **Modelo de Comunicación.** En contraste a las infraestructuras de gestión existentes, CORBA no tiene la necesidad de un protocolo de gestión específico porque los objetos cooperan invocando métodos entre ellos. La infraestructura de comunicación es el ORB (Object Request Broker), el cual transmite una llamada cliente a un objeto servidor independientemente de su localización. Los servicios del ORB son accedidos a través de APIs estándar.

  Las llamadas de los clientes son aceptadas por el ORB a través de Stubs Cliente o la Interfaz de Invocación Dinámica, DII (Dynamic Invocation Interface). Un compilador IDL acepta código IDL y genera Stubs cliente y Skeletons para el lenguaje especificado. Los Stubs envían los requerimientos de un cliente a un servidor a través del ORB. Para esto, los Stubs convierten los requerimientos de un cliente, implementados en un algún lenguaje de programación, en una representación adecuada para el envío de información a través del ORB (esto se conoce comúnmente como "marshaling"). Los Skeletons, en el lado del servidor, se encargan de recibir los requerimientos provenientes del ORB y enviarlos a la implementación de objetos CORBA. Para

esto, el Skeleton hace una conversión de un formato de transmisión a un formato en un lenguaje de programación dado (esto es conocido comúnmente como unmarshaling). Además, a través de los Skeletons, se envían las respuestas al ORB, las cuales son recibidas por el cliente por intermedio del Stub. Al conjunto de envíos a través del Stub y el Skeleton se le conoce con el nombre de invocación estática (tanto el cliente como el servidor tienen pleno conocimiento de las interfaces IDL que están siendo invocadas).

El otro tipo de invocación que existe en CORBA es la invocación dinámica, que permite en tiempo de ejecución descubrir las operaciones de un objeto, sin tener un conocimiento previo de sus interfaces (sin un stub). Para llevar a cabo la invocación dinámica, se definen dos tipos de interfaces: la Interfaz de Invocación Dinámica, DII (Dinamic Invocation Interface) y la Interfaz de Skeleton Dinámica, DSI (Dinamic Skeleton Interface). La DII se encarga de hacer peticiones a un objeto del que no se conocen sus interfaces. Ésta petición se hace a través de un pseudo objeto llamado Requerimiento (Request), sobre el cual el cliente especifica el nombre de la operación y sus argumentos, que pueden ser obtenidos del Repositorio de Interfaces(IR). Cuando el Requerimiento esté completo se envía al servidor. Este envío puede hacerse de tres formas, en la primera el objeto se envía y todos los procesos se bloquean hasta que el servidor emita una respuesta (Invocación Sincrónica), en la segunda, cuando se envía el objeto, el cliente sigue procesando y más tarde recoge la respuesta (Invocación Sincrónica Aplazada), y en la última forma cuando el objeto se envía, el cliente sigue procesando y la respuesta del servidor se recoge por algún otro medio (por ejemplo a través de otro proceso). En el lado del servidor, cuando se recibe un Requerimiento, la DSI es quien lo toma y envía alguna respuesta al cliente.

Como parte de la especificación DII, se introdujeron los objetos Contexto. Ellos son una lista de propiedades (pares nombre valor) que permiten pasar información adicional con el requerimiento al servidor, por ejemplo información acerca del entorno del cliente. Ellos se pueden ver como parámetros dinámicos, ya que normalmente solamente se pueden pasar parámetros estáticos definidos en la especificación IDL.

La transparencia de la localización se logra gracias a que los ORBs corriendo en diferentes sistemas puedan interoperar. La interoperabilidad se alcanza a través de la especificación formal de interfaces, las APIs ORB y los protocolos inter-ORB subyacentes.

La arquitectura de interoperabilidad de ORBs está basada en el Protocolo Inter-ORB General, GIOP (General Inter-ORB Protocol), que especifica un conjunto de formatos de mensajes y representaciones de datos para la interacción entre ORBs. GIOP está diseñado para trabajar sobre cualquier protocolo de transporte orientado a conexión, por ejemplo, el Protocolo Inter-ORB de Internet, IIOP (Internet Inter-ORB Protocol), especifica como se intercambian mensajes GIOP sobre redes TCP/IP. Gracias a IIOP, es posible utilizar Internet como un backbone ORB al cual otros ORBs pueden conectarse.

Por otra parte, la arquitectura de interoperabilidad entre ORBs define un conjunto de protocolos llamados Protocolos Inter-ORB Específicos del Entorno, ESIOPs (Environment-Specific Inter-ORB Protocols) que hacen posible la interacción de ORBs sobre redes específicas, por ejemplo uno de los primeros ESIOPs especificados fue el utilizado por el Entorno de Computación Distribuida, DCE (Distributed Computing Environment), llamado Protocolo Inter-ORB Común del DCE, D-CIOP (DCE Common Inter-Operability Protocol) con el ánimo de que el mundo de las aplicaciones CORBA y DCE interoperen en forma transparente.

- **Modelo funcional.** El Modelo Funcional de CORBA se divide en tres capas. Debido a la naturaleza orientada a objetos de la infraesctructura OMG, los servicios contenidos en las capas más altas pueden heredar funcionalidad desde capas subyacentes.

  - **Servicios CORBA (CORBAservices):** es una colección de interfaces que forman la capa más baja del modelo funcional, y aumentan y complementan la funcionalidad del ORB. Son obligatorios para la operación de cualquier implementación basada en CORBA. Los servicios

de Externalización, Propiedades, Gestión de Cambio y Licenciamiento, son ejemplos de servicios CORBA particularmente útiles para propósitos de gestión.

- **Facilidades CORBA (CORBAfacilities):** son servicios multipropósito útiles para una gran parte de aplicaciones. Clases típicas de facilidades CORBA son: Gestión de Información, Gestión de Tareas y Gestión de Sistemas. La última proporciona la base para introducir funcionalidad de gestión en la infraestructura OMG.

- **Interfaces de Dominio:** son los servicios de alto nivel adecuados para dominios de aplicación específicos. Debido a que la gestión de sistemas es un aspecto importante en todas las clases de dominio de aplicación, esta capa no define ninguna funcionalidad de gestión. Ella es heredada desde los capas subyacentes (Servicios CORBA y Facilidades CORBA).

Los objetos de aplicación se encuentran sobre estas tres capas y obtienen una parte importante de su funcionalidad al heredar de ellas. Un ejemplo típico para un objeto de aplicación es una aplicación de gestión distribuida.

La aproximación basada en servicios asume que la funcionalidad de gestión no está atada al sistema gestor sino que permite que sea distribuida y delegada: estos servicios pueden ser utilizados por sistemas gestores o sistemas gestionados, ya que estos servicios son implementados como objetos y cualquier sistema gestor o gestionado puede derivar la funcionalidad necesaria desde estos servicios.

Hay dos formas de introducir servicios de gestión en CORBA. La primera posibilidad es implementar nuevos servicios e introducirlos como nuevas Facilidades CORBA. La segunda forma consiste en reutilizar funcionalidad de gestión definida en las infraestructuras de gestión tradicionales y hacerla accesible a objetos CORBA.

Es importante considerar que la frontera entre Servicios CORBA y Facilidades CORBA no es estática sino que puede cambiar con el tiempo. La consecuencia de esto es que los servicios de gestión de sistemas no están necesariamente asociados a Facilidades CORBA sino que también pueden ser definidos como Servicios CORBA, e incluso pueden ser proporcionados como parte del sistema de Runtime del ORB.

Comparación entre CORBA y otras infraestructuras de gestión

- Las infraestructuras de gestión nombradas anteriormente, están basadas en el concepto de agentes (proveedores en el caso de WBEM), que representan un conjunto de recursos a través de MIBs (esquemas CIM almacenados en el repositorio en el caso de WBEM), y gestores accediendo agentes a través de un protocolo bien definido. En CORBA, los clientes acceden instancias directamente, independientemente del servidor en el cual ellas están localizadas. En este sentido, los servidores CORBA se pueden comparar en su funcionalidad a los agentes, porque ellos contienen los objetos, pero la presencia de los servidores CORBA es transparente a las aplicaciones de gestión. Además, y en contraste a las infraestructuras de gestión existentes, CORBA no define un protocolo de gestión específico, sino que especifica la API de acceso al ORB, a través de la cual los objetos cliente pueden realizar operaciones sobre objetos servidor.

- Los modelos de información de CORBA, OSI y WBEM son orientados a objetos, mientras que el modelo de información de SNMP solamente tiene variables para representar información. Sin embargo, todos los modelos son muy generales, y a pesar de sus diferencias, cualquier cosa que pueda ser modelada en uno puede ser también modelada en otro.

- OSI y TMN utilizan GDMO y ASN.1 como lenguajes de especificación de objetos gestionados y tipos de datos respectivamente, SNMP utiliza SMI y un subconjunto de ANS.1 para la definición de información de gestión (variables), WBEM utiliza MOF como lenguaje formal para la definición de

las clases de su Modelo de Información, y CORBA utiliza IDL para definir las interfaces de los objetos.

- Los objetos CORBA son comúnmente referidos por tipo y no por nombre. Esto se debe a la naturaleza de los sistemas distribuidos, donde típicamente las instancias del mismo tipo ofrecen exactamente el mismo servicio. Por supuesto, esto no significa que no existen mecanismos para distinguir entre instancias del mismo tipo, sino que la infraestructura está optimizada para un estilo de operación de acceso a un solo objeto referenciado por tipo, en contraste con las otras infraestructuras de gestión que están optimizadas para un estilo de operación de acceso a múltiples objetos, referenciados por nombre.

- CORBA identifica instancias a través de referencias a objetos, mientras que en OSI los objetos gestionados se identifican por su DNs. En SNMP, se utilizan identificadores de objetos para referirse a variables SNMP de una forma única, y WBEM utiliza nombres de clases y propiedades clave como sistema de nombramiento.

- CORBA define objetos independientemente del protocolo subyacente porque solamente las interfaces de objeto se exponen al cliente. Por supuesto, para interactuar con un objeto remoto a través de su interfaz se necesita un protocolo para enviar el requerimiento, pero este hecho se esconde a los clientes, ofreciendo de esta forma un nivel de abstracción más alto. En las otras infraestructuras de gestión, se define un protocolo con el que los clientes tienen que tratar para comunicarse con una entidad.

- Una diferencia clave entre CORBA y las otras infraestructuras de gestión existentes, es que en CORBA, los protocolos subyacentes se pueden reemplazar, ya que los aspectos de comunicación se esconden dentro del ORB. Por supuesto, por motivos de interoperabilidad se establece un protocolo, pero el resto de la infraestructura no tiene una dependencia fuerte de él. El mayor beneficio de esto es la portabilidad de objetos a través de diferentes implementaciones CORBA.

- En CORBA, las operaciones remotas se soportan en un protocolo RPC (Remote Procedure Call) orientado a la conexión que utiliza TCP e IP como protocolos de transporte y red respectivamente. Por lo tanto, las aplicaciones que utilizan CORBA tienen un transporte confiable, al igual que OSI, TMN y WBEM, pero a diferencia de SNMP.

- El protocolo RPC de CORBA no ofrece facilidades especiales para descubrimiento de objetos y acceso a múltiples objetos, como las que ofrece SNMP con get-next y get-bulk, OSI a través de su servicio de requerimiento (scoping y filtering), y WBEM a través de las operaciones de enumeración de clases e instancias. En CORBA estas facilidades son proporcionadas por el ORB.

- En entornos CORBA, SNMP y WBEM, la funcionalidad de gestión no se especifica en términos de un modelo dedicado, como es el caso de OSI y TMN.

- SNMP y WBEM tienen un Modelo de Información complejo y funciones simples para manipularlo. CORBA, OSI y TMN, tienen Modelos de Información simples, pero funciones más complejas para acceder y manipular los datos, haciendo que los protocolos sean más potentes.

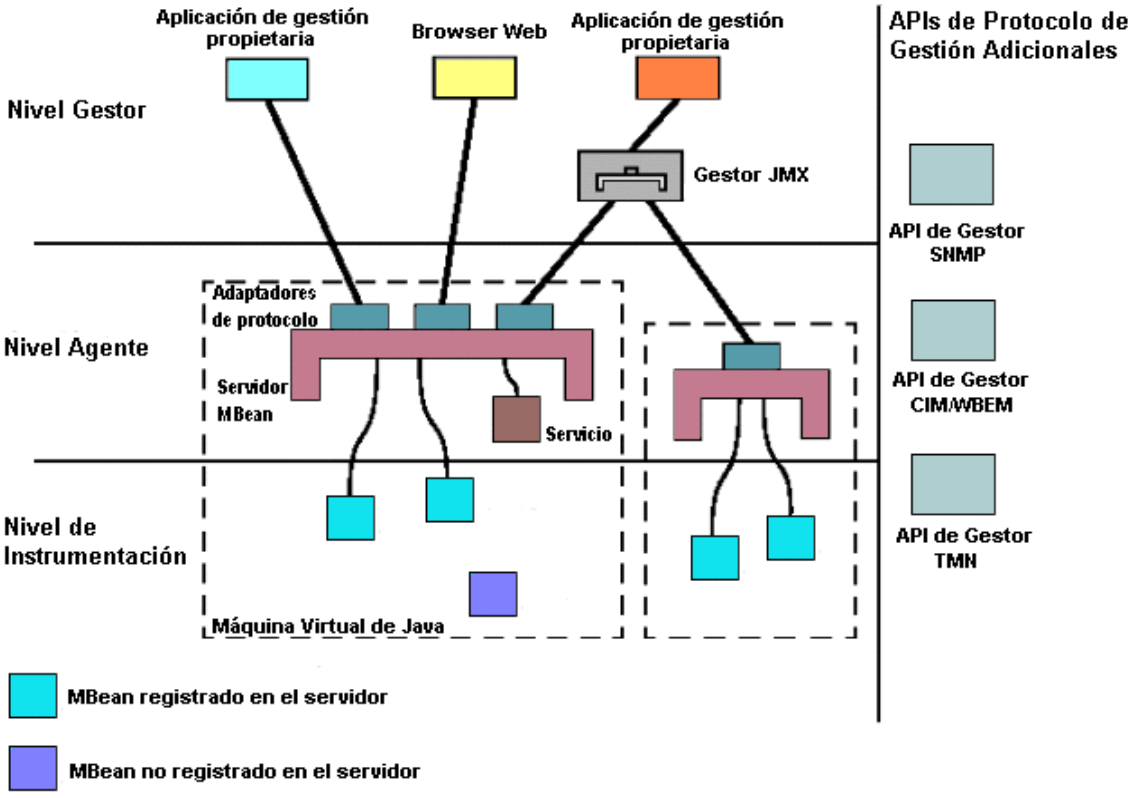
- La DII y el Repositorio de Interfaz de CORBA desacoplan el sistema gestor de los cambios ocurridos en los sistemas gestionados. Por lo tanto no se tienen que hacer modificaciones al sistema gestor si ocurren cambios en los sistemas gestionados. Esto no ocurre en las otras infraestructuras de gestión, donde es necesario que el gestor tenga conocimiento de las MIBs, y por lo tanto los cambios en ellas (agentes) también afectan a los aplicaciones de gestión (gestores). En el caso de WBEM, cualquier modificación o adición a los esquemas de extensión también implica un cambio en las aplicaciones de gestión.

2.2.6 JMX. Esta infraestructura de gestión, desarrollada por Sun Microsystems, permite la gestión de objetos Java a través de aplicaciones de gestión basadas en Java. Casi todo puede ser gestionado a través de JMX: aplicaciones, servicios, dispositivos, etc.

JMX define una arquitectura, APIs y servicios para gestión de red, en el lenguaje de programación Java. Sin embargo se tratará de asociar estos aspectos con los cuatro modelos definidos anteriormente con el fin de realizar la comparación.

- **Modelo Organizacional.** JMX sigue un paradigma gestor-agente fuertemente distribuido. Su arquitectura es dividida en tres niveles: nivel de instrumentación, nivel de agente y nivel del gestor, como se muestra en la Figura 2.11.

Figura 2.11 Arquitectura de JMX



- **Nivel de instrumentación:** la instrumentación de un recurso dado es proporcionado por uno o más Beans Gestionables o MBeans (Managed Beans). MBeans son objetos Java que siguen ciertos patrones de diseño derivados del modelo de componentes de los JavaBeans. Si el MBean de un recurso dado es registrado con el servidor MBean (en el nivel agente) entonces el recurso puede ser gestionado por una aplicación.

- **Nivel agente:** los agentes controlan los recursos y los hacen disponibles a las aplicaciones de gestión remota. Un agente JMX está compuesto de un sevidor MBean, al menos un adaptador o conector de protocolo, y también puede contener servicios de gestión. El servidor MBean es un registro de MBeans en el agente, y proporciona los servicios que permiten su manipulación. Todas las operaciones de gestión realizadas sobre los MBeans se hacen a través de las interfaces Java del servidor MBean.

- **Nivel de gestor:** proporciona las interfaces para implementar gestores JMX que interactúen transparentemente con un agente y sus recursos gestionables JMX, a través de un conector o adaptador de protocolo. Además distribuye la información de gestión entre agentes y gestores JMX, resume la información de gestión que viene de varios agentes JMX y proporciona seguridad.

Todos los niveles, instrumentación, agente y gestor, son aislados, y sus interacciones son definidas a través de la API JMX. Esto hace posible tener un desarrollo modular, porque cada nivel es diseñado e implementado independientemente.

❖ **Modelo de Información.** Los recursos se representan a través de MBeans. Un MBean es una clase Java que implementa una interfaz de gestión, mapea las variables y métodos de sus recursos, a atributos y operaciones de la interfaz, y puede emitir notificaciones a las aplicaciones de gestión que se hayan registrado para recibir este tipo de eventos. Los MBeans no necesitan conocer nada acerca de la comunicación con el mundo exterior.

Los MBeans no se pueden considerar como un Modelo de Información, sin embargo, se describen aquí porque son definidos por la especificación JMX como la forma de representar los recursos gestionados.

Hay cuatro tipos de MBeans: MBeans Estándar, MBeans Dinámicos, MBeans Modelo y MBeans Abiertos.

❖ **Modelo de Comunicación.** Los gestores utilizan adaptadores de protocolo o conectores para comunicarse con el servidor MBean. Ellos sirven para el mismo propósito: exponer los agentes JMX a las entidades gestoras, pero son diferentes. Los conectores están formados por dos componentes: un componente reside en el agente JMX, y el otro es utilizado por las aplicaciones cliente. Los clientes utilizan el componente conector para contactar el componente del lado del servidor y comunicarse con un agente JMX. De esta manera, el conector esconde el protocolo real que se está utilizando para contactar el agente, todo el proceso pasa entre los dos componentes del conector. Ejemplos de conectores son el conector RMI y el conector JINI. Los adaptadores de protocolo escuchan mensajes de un protocolo particular como HTTP o SNMP. Por lo tanto, los adaptadores están compuestos por solo un componente que se encuentra en el agente. Se pueden desarrollar nuevos adaptadores y conectores para responder a protocolos propietarios, sin rescribir los MBeans o las aplicaciones de gestión.

Los adaptadores y los conectores pueden manejar varias conexiones, entonces el agente solo necesita uno de ellos para cada protocolo que quiera soportar. El servidor MBean también soporta requerimientos simultáneos pero los MBeans son responsables de la sincronización. Esto hace que los MBeans se puedan acceder simultáneamente a través de cualquier número de protocolos. Adicionalmente, los adaptadores y conectores ofrecen autenticación y cifrado.

La especificación JMX incluye la definición de varias APIs de Protocolo de Gestión. Ellas permiten que los gestores JMX accedan agentes de otras infraestructuras de gestión, y se comuniquen con ellos a través de un protocolo existente. Por ejemplo, la API del gestor SNMP proporciona los servicios necesarios para escribir aplicaciones que manejen agentes SNMP o actúen como proxies SNMP.

❖ **Modelo Funcional.** Aunque JMX no define específicamente un Modelo Funcional, es importante tener en cuenta que la funcionalidad está definida por servicios. Tanto los agentes como los gestores integran servicios que les dan autonomía e inteligencia. Los servicios permiten a los agentes manejar sus recursos, y permiten a los gestores enviar información entre agentes y aplicaciones de gestión.

En la arquitectura JMX se puede registrar o cancelar el registro de cualquier MBean en cualquier momento, incluso mientras el agente está corriendo. Ya que los servicios son también MBeans,

ellos pueden adicionarse o removerse según las necesidades. Esto da extensibilidad y escalabilidad a los agentes y a los gestores, y es algo esencial cuando se despliegan en clientes con recursos limitados.

Comparación de JMX con otras infraestructuras

- A diferencia de las otras infraestructuras de gestión, JMX:
  - Permite a los desarrolladores de aplicaciones, servicios o dispositivos hacer a sus productos gestionables en una forma estándar, a través de código Java, sin tener que entender o invertir en sistemas de gestión.
  - Es independiente del Modelo de Información.
  - Permite a las aplicaciones de gestión cambiar de protocolo de acuerdo a condiciones de tiempo real.
  - Proporciona independencia de las aplicaciones de gestión gracias a la utilización de conectores y adaptadores de protocolo.
- WBEM y JMX proporcionan los mecanismos para la integración con infraestructuras de gestión existentes, porque permiten integrar cualquier protocolo existente o propietario, a través del desarrollo de proveedores en el caso de WBEM, y al desarrollo de nuevos adaptadores y conectores en el caso de JMX, sin necesidad de rescribir los MBeans, los agentes o las aplicaciones de gestión.
- A diferencia de las otras infraestructuras de gestión, JMX y CORBA proporcionan independencia del protocolo utilizado en la comunicación cliente-servidor.
- JMX está fuertemente ligada a la plataforma Java, mientras que CORBA y WBEM proporcionan independencia del lenguaje en el están implementados los objetos.
- El modelo de eventos de SNMP es muy simple y consiste de traps que son enviados desde los agentes a los gestores o desde gestores a gestores. Los traps llevan información acerca del evento ocurrido en un agente.

  En el modelo de eventos de OSI, cada vez que los objetos gestionados necesitan informar al agente acerca de la ocurrencia de una nueva situación, envían una notificación a los discriminadores de envío de eventos de los agentes. Ellos evalúan la notificación según filtros configurados por los gestores y envían los eventos que pasan el filtro al gestor adecuado en la forma de reportes de eventos.

  En WBEM, los eventos se pueden originar del CIMOM o de los proveedores. El CIMOM puede generar eventos programados o eventos intrínsecos. Un proveedor puede realizar notificaciones de eventos intrínsecos y extrínsecos. Este tipo de proveedor es llamado un proveedor de eventos. Cuando una clase no tiene un proveedor de eventos, se utiliza un mecanismo de consulta para monitorear eventos. Con este mecanismo, el repositorio se revisa cada cierto intervalo de tiempo, para ver si se han hecho cambios. Un consumidor de eventos recibe los eventos y realiza una tarea específica de acuerdo al evento generado. Ellos se registran para recibir notificaciones de los eventos en los que estén interesados. Para registrarse, estos consumidores especifican un filtro que se crea utilizando WQL. Éste filtro describe las condiciones bajo las cuales el consumidor quiere recibir notificación de eventos. El CIMOM actúa como un intermediario, enviando los eventos generados a los consumidores apropiados.

  En CORBA los eventos pueden ser enviados directamente o a través de canales. Un consumidor puede ser notificado en cualquier momento acerca de la adición de un nuevo evento al canal de

eventos (modelo push), o el consumidor puede recuperar eventos activamente desde el canal (modelo pull).

La especificación JMX define un modelo de notificación basado en el modelo de eventos Java. Las notificaciones pueden ser emitidas por instancias MBean, así como también por el servidor MBean. Esta especificación describe los objetos de notificación, y las interfaces broadcaster y listener que se deben implementar para transmitir y recibir eventos.

Debido a que WBEM y JMX, se consideran las infraestructuras de gestión basadas en Web más importantes, a continuación se hará una comparación entre estas tecnologías para explicar por qué WBEM fue finalmente seleccionada para el desarrollo de este proyecto.

Comparación entre WBEM y JMX

- Con JMX no se necesario entender o invertir en sistemas de gestión. Esto no es una ventaja con respecto a WBEM, si se tiene en cuenta que ésta es una infraestructura simple gracias a la utilización de tecnologías de Internet y a las implementaciones WBEM ya existentes, que proporcionan CIMOM, proveedores y APIs cliente y proveedor. Además, no requiere inversión porque las implementaciones WBEM son complemente gratis (y algunas de código abierto), e incluso algunas ya vienen incluidas en los sistemas operativos y dispositivos.

- JMX proporciona independencia del Modelo de Información, y aunque la utilización de MBeans para la representación de recursos, permite que ésta especificación sea amplia y extensible, la hace menos interoperable, ya que cada implementador puede decidir qué variables quiere monitorear y controlar en un recurso específico. WBEM, por su parte, a través de los esquemas núcleo y común de CIM, permite que exista una representación común de la información de gestión, y por lo tanto permite compartir datos independientemente del vendedor, proporcionando así mayor interoperabilidad.

- CIM es definido a un nivel de abstracción más alto que los MBeans, y esto da a WBEM independencia del lenguaje de implementación. JMX por el contrario, está fuertemente ligada a la plataforma de desarrollo Java.

- La flexibilidad de JMX en cuento al protocolo utilizado para la comunicación gestor-agente, trae como consecuencia mayor complejidad, porque es necesario desarrollar adaptadores o conectores de protocolo, y menor interoperabilidad, porque un gestor solo se puede comunicar con agentes que tengan el adaptador o componente del conector apropiado. WBEM al proporcionar un método estandarizado para acceder recursos gestionados, simplifica el desarrollo de aplicaciones de gestión y garantiza compatibilidad entre cualquier cliente y cualquier CIMOM.

- Desarrollar una plataforma de gestión utilizando JMX es más complejo que utilizar WBEM, porque es necesario realizar la instrumentación de los recursos, desarrollar la aplicación la gestión, y en algunos casos, desarrollar adaptadores o conectores de protocolo. Con WBEM, únicamente se debe implementar la aplicación de gestión, porque cada vendedor hace a sus productos gestionables a través de CIMOMs y proveedores. Además, no es necesario preocuparse por los aspectos de comunicación entre aplicaciones y servidores, porque esto ya está definido y estandarizado.

Como se puede observar, WBEM es mejor que JMX en aspectos como interoperabilidad y complejidad. Además, la integración de tecnologías de Internet en WBEM permite una amplia y rápida adopción de ésta infraestructura, porque la industria ya es familiar con HTTP y no tiene que aprender un nuevo protocolo. XML, por su parte, tiene una gran aceptación y cada vez hay más aplicaciones soportándolo. Además, la utilización de tecnologías de Internet también facilita el desarrollo de implementaciones WBEM porque por ejemplo, se pueden emplear implementaciones ya existentes de HTTP para construir las parte de comunicación de WBEM.

De acuerdo a lo anterior se puede concluir que WBEM es la infraestructura de gestión con mayores posibilidades de convertirse en el nuevo estándar de facto para la gestión de redes, debido principalmente a que proporciona todos los elementos necesarios para superar los mayores problemas que existen en la gestión: la falta de interoperabilidad y la falta de integración entre estándares de gestión.

En el siguiente capítulo se hace una descripción de las principales implementaciones WBEM, con el fin de determinar y comparar sus capacidades y alcances, y definir cual de ellas es la más apropiada para el desarrollo de este proyecto.

3. DESCRIPCIÓN DE LAS IMPLEMENTACIONES WBEM

Actualmente existen varias plataformas de desarrollo WBEM, algunas de fuente abierta y otras comerciales. En este capítulo se hace una descripción de las principales características de las implementaciones más utilizadas.

3.1 IMPLEMENTACIONES WBEM DE FUENTE ABIERTA

La iniciativa WBEMsource es una organización encargada de coordinar implementaciones WBEM de fuente abierta, con el fin de lograr interoperabilidad y portabilidad entre ellas.

Algunas de las compañías que están involucradas en la iniciativa son: BMC Software, Caldera International, Cisco Systems, Compaq, DMTF, Evidian, Hewlett-Packard, IBM, The Open Group, SNIA (Storage Networking Industry Association) y Sun Microsystems.

Los principales proyectos que están participando en la iniciativa son:

- WBEM Services
- Pegasus
- OpenWBEM
- SNIA Open Source CIMOM

Antes de comenzar a describir las diferentes implementaciones WBEM, es necesario entender qué es SBLIM (Standards Based Linux Instrumentation for Manageability) y cuáles son sus principales objetivos.

3.1.1 El proyecto SBLIM. SBLIM es un proyecto de fuente abierta de IBM, que busca mejorar la gestionabilidad de sistemas Linux a través de WBEM. Los principales objetivos del proyecto son:

- Proporcionar una implementación de fuente abierta del esquema CIM para los sistemas Linux.
- Proporcionar un completo conjunto de proveedores Linux escritos en C.

Al hacer esto los sistemas Linux serán habilitados con WBEM.

Los paquetes SBLIM disponibles pueden ser divididos en cuatro categorías:

- **Interfaces de programación.** Debido a que cada implementación de CIMOM define su propia interfaz proveedor (la comunicación entre CIMOM y proveedores aún no ha sido estandarizada por el DMTF), los programadores tienen que escribir proveedores para cada CIMOM que se quiera soportar. Por esto, el propósito de estas interfaces es definir una API C para la comunicación entre CIMOM y proveedores, que pueda ser soportada por cualquier manejador de objetos CIM. Para lograr esto, estas APIs realizan las conversiones (traducciones) a las diferentes implementaciones de CIMOM, para ocultar las características específicas de ellos (como por ejemplo el lenguaje de programación) a los proveedores.

  Actualmente existen dos interfaces: Interfaz de Proveedor Nativa, NPI (Native Provider Interface), e Interfaz de Programación Genérica de Gestionabilidad, CMPI (Common Manageability Programming Interface). Debido a que ellas se pueden portar a cualquier CIMOM, los proveedores de una plataforma específica que se escriban utilizando alguna de estas interfaces se pueden comunicar con cualquier CIMOM. Esto evita la necesidad de escribir la misma funcionalidad más de una vez.

  El lenguaje de programación es C porque aunque C++ es un lenguaje orientado a objetos y por lo tanto más adecuado para el esquema CIM, tiene una compatibilidad binaria muy frágil. Cuando se

utilizan clases C++ para escribir código de instrumentación (proveedores), el código binario creado durante la compilación puede variar de compilador a compilador. Por lo tanto, el proveedor tendría que ser recompilado utilizando el mismo compilador que se utilizó para las interfaces. Esto no es deseable, porque no todas las personas pueden estar dispuestas a que su código fuente sea disponible.

Sin embargo las interfaces siguen una aproximación orientada a objetos, definiendo funciones como punteros en estructuras C regulares, que son equivalentes independiente del compilador.

- **NPI.** La arquitectura de NPI consta de una clase Java llamada NativeProvider, que se tiene que extender para cada proveedor nativo, y una librería proveedor nativa escrita en C++. Estas dos partes son conectadas a través de JNI. La librería de proveedor nativo tiene la habilidad para enlazar dinámicamente las librerías de proveedor escritas en C y pasar las llamadas de funciones y retornar valores al CIMOM. Además, la librería proporciona un conjunto de funciones para el proveedor.

  El objetivo de NPI es proporcionar una forma fácil de escribir proveedores, haciendo que JNI sea invisible para el programador de C.

  NPI soporta los siguientes tipos de proveedores: instancia, asociación y métodos (a partir de la versión 0.7)

  NPI ya no seguirá siendo desarrollada.

- **CMPI.** Esta API es la sucesora de NPI y está en proceso de estandarización por la iniciativa WBEMsource. Está siendo extendida con una interfaz de compatibilidad con CMPI, para permitir que los proveedores existentes que utilizan NPI se comuniquen con CIMOMs con soporte para CMPI.

  La principal diferencia entre NPI y CMPI, es que CMPI adiciona el concepto de Adaptador de Proveedor o Adaptador CMPI, el cual es parte del CIMOM. Él permite la integración de CMPI y sus proveedores con el CIMOM. Los requerimientos que sean soportados por proveedores CMPI son enrutados a través del adaptador, que se encarga de cargar el proveedor CMPI, y transformar y pasar todas las llamadas del CIMOM a las equivalentes CMPI. Cada clase que tenga un proveedor CMPI registrado en el CIMOM, tiene una librería de instrumentación de gestión y las capacidades de proveedor (o tipos de proveedor) asociados con ella. De esta forma el adaptador CMPI conoce cual librería cargar.

  Teniendo en cuenta que existen tres CIMOMs soportados actualmente por CMPI, el adaptador CMPI está implementado en tres versiones también: dos en C++ para Pegasus y OpenWBEM, y una utilizando JNI para integrar CMPI con el CIMOM de SNIA. Sin embargo este código es invisible para los proveedores CMPI.

❖ **Paquetes de instrumentación CIM.** Contienen los proveedores para acceder los recursos del sistema. La Tabla 3.1 muestra los principales paquetes desarrollados hasta el momento.

Tabla 3.1 Principales paquetes CIM para Linux

Nombre del paquete	Dependencia	Clases instrumentadas	Descripción
Cmpi-base	cmpi-adpater	Linux_ComputerSystem, Linux_OperatingSystem, Linux_UnixProcess, Linux_Processor,Linux_RunningOS, Linux_OSProcess, Linux_CSProcessor	Este paquete proporciona acceso a información básica, como sistema operativo, procesos, procesador y características del sistema. Además, ofrece funciones de librería a otros módulos.
Cmpi-network	cmpi-adpater, cmpi-base	Linux_IPProtocolEndpoint, Linux_LocalLoopbackPort, Linux_EthernetPort, Linux_TokenRingPort, Linux_CSNetworkPort, Linux_NetworkPortImplementsIPEndpoint	Este paquete proporciona acceso a configuraciones de red básicas, como tipo de puerto de red, dirección IP, máscara de red, entre otras.
Cmpi-fsvol	cmpi-adpater, cmpi-base	Linux_Ext2FileSystem, Linux_Ext3FileSystem, Linux_ReiserFileSystem, Linux_NFS, Linux_HostedFileSystem, Linux_BootOSFromFS	Este paquete proporciona acceso a información acerca de archivos.
Cmpi-smbios	cmpi-adpater, cmpi-base	Linux_BIOSElement, Linux_BIOSFeature, Linux_BIOSProduct, Linux_FeatureBIOSElements, Linux_ProductBIOSFeatures	Este paquete proporciona acceso a información de la BIOS, como la versión, el ID del producto, número serial, entre otras.
Base	Npi	Linux_ComputerSystem, Linux_OperatingSystem, Linux_Process, Linux_RunningOS, Linux_OSProcess	Este paquete proporciona acceso a información básica como sistema operativo, procesos, y características del sistema. Además, ofrece funciones de librería a otros módulos.

Fsvol	Npi, base	Linux_Directory, Linux_Ext2FileSystem, Linux_Ext3FileSystem, Linux_ReiserFileSystem, Linux_NFS, Linux_HostedFileSystem, Linux_BootOSFromFS, Linux_Mount, Linux_NFSExport, Linux_ResidesOnExtent, Linux_NativeLogicalDisk, Linux_DiskPartition, Linux_StorageVolume, Linux_DiskPartitionBasedOnVolume, Linux_LogicalDiskBasedOnPartition, Linux_LVMLogicalDisk, Linux_LVMVolumeGroup, Linux_LVMLogicalVolume, Linux_LVMDiskPartition, Linux_LVMLogicalDiskBasedOnLV, Linux_LVMLVBasedOnDP, Linux_LVMLVInVG, Linux_LVMDPInVG, Linux_LVMDPIsDiskPartition, Linux_LVMSystemVolumeGroup	Este paquete proporciona acceso a información acerca de archivos
service	Npi, base	Linux_Service, Linux_HostedService, Linux_ServiceProcess, Linux_ServiceComponent	Permite monitorear y manejar servicios remotamente.
Rpm	Npi, base	RPM_Package, RPM_FileCheck, RPM_AssociatedFile	Estos proveedores listan los paquetes software instalados, y proporcionan información detallada acerca de ellos.
Mail	npi, base	Mail_MailService, Mail_HostedMailService, Mail_MailProcess, Mail_MailServiceMailProcess, Mail_MailDomain, Mail_MailServiceForDomain, Mail_MailGroup, Mail_MailAccount, Mail_MailAccountOnSystem, Mail_PersonalAccountInfo, Mail_GroupMembership, Mail_UsersMailDomain, Mail_UsersActiveOnMailService, Mail_MailAlias, Mail_HostedMailAlias, Mail_MailAliasForMailAccount, Mail_MailContainer, Mail_Mailbox, Mail_MailboxReplication, Mail_Folder, Mail_FolderParentChild, Mail_PersonalMailbox, Mail_MailboxServer, Mail_FolderLocation, Mail_DedicatedFolder, Mail_DedicatedFolderOf, Mail_FolderTrash, Mail_FolderTrashOfMailbox, Mail_FolderInbox, Mail_FolderInboxOfMailbox, Mail_FolderSent, Mail_FolderSentOfMailbox, Mail_FolderDrafts,	Este paquete contiene un esquema de gestión para un servidor de email, donde se utilizan varios tipos de software configurados en una forma diferente.

		Mail_FolderDraftsOfMailbox, Mail_FolderACI, Mail_HostedFolderACI, Mail_FolderACIOnFolder, Mail_FolderACIForGroup, Mail_FolderQuota, Mail_HostedFolderQuota, Mail_FolderQuotaOnFolder, Mail_QuotaForGroup	

- **Aplicaciones cliente CIM.** Incluyen una aplicación cliente y una infraestructura para gestionar sistemas Linux, una herramienta cliente de línea de comando y aplicaciones ejemplo para el manejo de eventos.
- **Herramientas de desarrollo.** Ofrecen soporte para automatizar la creación de skeletons de proveedores para ciertos tipos de interfaces y herramientas para automatizar las pruebas de verificación de función de los proveedores.

3.1.2 WBEM Services

- **Desarrollador:** Sun
- **Lenguaje de implementación:** Java
- **Sistemas operativos soportados:** todos
- **Licencia:** Sun Industry Standards Source License (SISSL) v1.2
- **Versión actual:** 2.5
- **Componentes software:**

  - **CIMOM:** implementación Java del manejador de objetos CIM.

    - ✓ Soporta el estándar xmlCIM para codificación.
    - ✓ Soporta Operaciones CIM sobre HTTP y RMI (Remote Method Invocation) para transporte. Escucha conexiones XML/HTTP por el puerto 5988 y conexiones RMI por el puerto 5987.
    - ✓ Soporta el modelo de eventos CIM.

  - **Repositorio:** implementación Java del repositorio.
  - **Compilador MOF:** convierte archivos MOF a clases Java y las adiciona al Repositorio.
  - **Generador de JavaBeans:** genera JavaBeans que representan las clases CIM definidas en los archivos MOF. Estos JavaBeans definen las interfaces, es tarea del usuario adicionar el código de implementación. Para cada clase CIM el compilador MOF genera una interfaz Java que contiene lo siguiente:

    - ✓ Métodos set y get para las propiedades que son definidas en el archivo MOF.
    - ✓ Métodos que están definidos en el archivo MOF.

  - **Esquema Solaris:** colección de clases que se extienden del esquema CIM y representan recursos gestionados en un entorno Solaris.
  - **API CIM:** clases y métodos comunes utilizados por las aplicaciones para representar todos los elementos CIM básicos.
  - **APIs Cliente:** métodos utilizados por las aplicaciones para transferir datos a y desde el CIMOM.
  - **APIs Batching:** un subconjunto de las APIs cliente, que permite realizar múltiples requerimientos en una llamada, reduciendo el retardo introducido por el intercambio de mensajes.
  - **APIs Proveedor:** interfaces utilizadas por el CIMOM y los proveedores para comunicarse entre si. WBEMServices soporta las siguientes interfaces:

- ✓ Interfaz proveedor Java: soporta los siguientes tipos de proveedores:

  - **Proveedor de instancias:** se utilizan para crear, enumerar, modificar y borrar instancias CIM. Las clases que tengan proveedores de instancias, no tienen sus instancias almacenadas en el Repositorio porque el CIMOM llama al proveedor de instancias para obtener o manipular estas instancias. Por lo tanto los proveedores de instancias se pueden ver como programas que convierten recursos reales en instancias CIM.
  - **Proveedor de métodos:** se utilizan para ejecutar métodos sobre instancias o clases (métodos estáticos). En las implementaciones WBEM, la única forma de ejecutar un método es implementar un proveedor de métodos, porque no es posible almacenar implementaciones de métodos en el Repositorio.
  - **Proveedor de asociaciones:** se utilizan para obtener relaciones entre instancias. Un proveedor de asociaciones se puede ver como un programa que proporciona asociaciones dinámicas. Por lo tanto, un proveedor de asociaciones solo tiene sentido si al menos una instancia en el relación es dinámica. Si ambas instancias fueran estáticas, el CIMOM podría encontrar las relaciones por sí mismo.
  - **Proveedor de indicaciones:** se utilizan para enviar eventos. Cuando se lanza una indicación en el CIMOM, ella se puede enviar en varias formas dependiendo del proveedor de indicación que sea cargado. Por ejemplo, un proveedor podría convertir el evento a un trap SNMP, mientras que otro podría enviar un email.
  - **Proveedor de autorización:** permite implementar mecanismos propios de autorización.

  Un proveedor puede pertenecer a uno o más de estos tipos.

- ✓ Interfaz proveedor NPI.

- **API "Query":** contiene clases y métodos que permiten realizar y manipular "queries" utilizando WQL.

Todas las APIs nombradas anteriormente han sido propuestas para estandarización a través del JCP (Java Community Process).

- **Proveedores Solaris:** obtienen y configuran información de los recursos gestionados en el entorno Solaris. Han sido desarrollados para una variedad de áreas: usuarios, grupos, roles, sistemas de archivos, discos, procesos, configuración de red, registro del producto, y monitoreo de desempeño de sistemas y dispositivos.

- **Aplicaciones WBEM:** procesan y despliegan datos de los recursos gestionados. Ellas son:

  - ✓ Sun WBEM User Manager y Solaris Management Console User: permiten al administrador del sistema adicionar y borrar usuarios, y configurar privilegios de acceso a los recursos gestionados.
  - ✓ Solaris Management Console Log Viewer: despliega archivos de registro (log files), incluyendo el nombre del usuario logeado y el computador cliente sobre el cual ocurrió el evento.
  - ✓ CIM workshop: permite a los desarrolladores crear, borrar y buscar clases e instancias CIM.
  - ✓ Programas ejemplo: código fuente que muestra la utilización típica de las APIs proveedor y cliente.

- Desde la versión 2.4, WBEMServices incluye los servicios SNMP para WBEM, que proporcionan dos componentes: el adaptador SNMP para WBEM y el proveedor SNMP con un compilador MIB a MOF.

El adaptador SNMP para WBEM permite a las aplicaciones de gestión SNMP acceder información de gestión proporcionada por los servicios WBEM de Solaris, convirtiendo las variables SNMP en objetos CIM y viceversa, a través de archivos de mapeo. Un archivo de mapeo contiene el OID , nombre de la clase, nombre de la propiedad y el tipo ASN.1 para cada objeto. Hasta el momento, solamente son soportados los requerimientos Get y variables escalares.

El proveedor SNMP permite a las aplicaciones de gestión WBEM acceder información SNMP. Para lograr esto se utiliza el archivo de una MIB para generar un archivo MOF. El proveedor SNMP mapea operaciones CIM que son ejecutadas sobre clases CIM (que se encuentran en archivos MOF) a operaciones SNMP.

El proveedor SNMP soporta traps, los cuales son reportados como eventos. Cuando un cliente se suscribe para recibir estos eventos, el proveedor escucha por puerto 162. La información es copiada desde el trap a la indicación y finalmente es entregada al cliente.

- **Mecanismos de seguridad.** WBEM emplea varios mecanismos para garantizar acceso seguro a sus datos, ellos son:

  - **Autenticación:** cuando un cliente se conecta al CIMOM debe autenticarse. El servidor WBEM requiere un login y un password. Estos parámetros son verificados por el servidor, que retorna una excepción si el permiso fue negado.

  - **Autorización:** WBEMServices soporta dos tipos de autorización:

    - ✓ Autorización basada en Listas de Control de Acceso, ACLs (Access Control Lists): las ACLs son mantenidas por el servidor WBEM para "namespaces" específicos. Todos los usuarios autenticados sin una entrada ACL solamente tienen permisos de lectura.
    - ✓ Autorización basada en Control de Acceso basado en Roles, RBAC (Role-Based Access Control): RBAC utiliza el principio de seguridad del menor privilegio, en el cual cada usuario tiene los privilegios necesarios para realizar su trabajo. Para esto, RBAC permite separar las capacidades de una cuenta de administrador y asignarlas a cuentas de usuario especiales llamadas roles. Los roles pueden asignarse a individuos específicos, de acuerdo a sus necesidades de trabajo. RBAC es soportado por RMI solamente.

    En general, si una clase es soportada por un proveedor, la implementación del proveedor usualmente utiliza RBAC. Si la clase no es soportada por un proveedor, es decir las instancias son almacenadas en el repositorio, el CIMOM utiliza ACLs. Sin embargo, el mecanismo de autorización depende de cómo el proveedor esté implementado y de la operación particular que se ejecute.

  - **Mensajería segura (Secure messaging):** es el proceso de adicionar un autenticador de mensaje a cada uno de los requerimientos del cliente. El autenticador se construye a partir de los parámetros de los datos en el mensaje. El servidor WBEM verifica el autenticador del mensaje para garantizar que el requerimiento proviene desde el mismo cliente que fue autenticado y que el mensaje no fue modificado durante su entrega al servidor. El transporte seguro es soportado solamente por RMI.

  - **Auditoría (Auditing):** es el proceso de escribir un archivo de auditoría de una operación específica que fue ejecutada por el servidor WBEM. Estos registros rastrean los cambios que un usuario autenticado hace a los datos de gestión sobre el sistema servidor WBEM. Es necesario habilitar el Módulo de Seguridad Básico, BSM (Basic Security Module) de Solaris para garantizar que la información es almacenada.
  - **Registro (Logging):** es el proceso de registrar los eventos relacionados con seguridad. El registro se puede ver utilizando la herramienta Solaris Management Console Log Viewer. La mayoría de mensajes incluyen la identidad del usuario y el nombre del equipo cliente.

3.1.3 Pegasus. También es llamado OpenPegasus. Su arquitectura está basada en módulos estandarizados que tienen interfaces bien definidas entre ellos. Esto proporciona un alto grado de especialización y personalización, debido a que los módulos permiten extender y modificar la funcionalidad de acuerdo a las necesidades.

OpenPegasus utiliza solamente componentes de uso libre y está diseñado para tener alta portabilidad.

- **Desarrollador :** The Open Group
- **Lenguaje de implementación:** C++
- **Sistemas operativos soportados:** la Tabla 3.2 muestra los sistemas operativos soportados por Pegasus.

Tabla 3.2 Sistemas operativos y compiladores soportados por Pegasus

Plataforma	Compiladores
AIX 5.2	VisualAge C++ Versión 5.0.2.3
HP-UX	HP aC++ B3910B
Linux Itanium	Gcc
Linux IA-32	gcc (versions 2.9x and 3.xx)
Windows 2000	Microsoft Visual C++ Ver 6 y Microsoft. Net compiler v7 y v 7.1
Windows XP	Microsoft Visual C++ Ver. 6 y Microsoft. Net compiler v7 y v 7.1

- **Licencia:** MIT open source license. La intención de esta licencia es hacer a Pegasus tan ampliamente disponible como sea posible, sin restricciones o limitaciones.
- **Versión actual:** 2.3
- **Componentes software:**

  - **CIMOM:** implementación C++ del manejador de objetos CIM.

    - ✓ Soporta el estándar xmlCIM para codificación.
    - ✓ Soporta el estándar Operaciones CIM sobre HTTP (puerto 5988) y HTTPS (puerto 5989) para transporte.
    - ✓ Soporta indicaciones de proceso, pero no incluye soporte para indicaciones de ciclo de vida.
    - ✓ El lenguaje “query” está basado en WQL, y es implementado y utilizado por filtros. Sin embargo, la operación executeQuery no ha sido implementada, esperando la disponibilidad de un lenguaje “query” estándar del DMTF.
    - ✓ Soporte para el Protocolo de Localización de Servicios, SLP (Service Location Protocol). SLP es un protocolo estándar del IETF que permite a las aplicaciones descubrir la existencia, localización y configuración de servicios en redes empresariales. Tradicionalmente, para localizar servicios en una red, los usuarios deben proporcionar el nombre y la dirección de red de la máquina que proporciona el servicio deseado. Con SLP, el usuario solo necesita conocer la descripción del servicio en el que está interesado. Basado en esta descripción, SLP puede obtener la URL del servicio deseado.

  - **Repositorio de clases y Repositorio de instancias:** estos repositorios fueron creados por funcionalidad, no por eficiencia. El repositorio por defecto es el de instancias.
  - **Compilador MOF Pegasus:** se puede utilizar para almacenar archivos MOF dentro del repositorio y también para chequear sintaxis.
  - **WMI Mapper:** una versión de la plataforma Pegasus que específicamente se comunica con WMI.
  - **APIs cliente:** Pegasus ha implementado una interfaz WBEM y también una interfaz basada en las APIs C++, por tanto el cliente puede comunicarse con el Servidor Pegasus, a través de requerimientos XML/HTTP o directamente a través de las interfaces C++ cliente de Pegasus.

Las APIs C++ implementan las operaciones como son definidas dentro de la documentación WBEM HTTP, utilizando los mismos parámetros pero con llamadas C++.

- **APIs proveedor:** Pegasus soporta las siguientes interfaces proveedor:

  ✓ Interfaz proveedor C++: soporta los siguientes tipos de proveedores:

  - Instancias.
  - Propiedades: se utilizan para obtener y establecer los valores de la propiedades de las instancias CIM. Las clases que tienen proveedores de propiedad no tienen los valores de las propiedades para sus instancias en el Repositorio, aún aunque las instancias estén almacenadas en el repositorio. El beneficio con los proveedores de propiedad, es que es posible almacenar las instancias en el Repositorio, mientras los valores de las propiedades son dinámicos.
  - Métodos.
  - Asociaciones.
  - Indicaciones.
  - "Query": permiten al usuario utilizar su propia versión de un lenguaje "query".

  Un proveedor puede pertenecer a uno o más de estos tipos.

  ✓ Interfaz proveedor NPI.
  ✓ Interfaz proveedor CMPI.

- **Proveedores Pegasus:**

  ✓ Proveedores internos:

  - __Namespace Provider: soporta la clase __Namespace para permitir manipulación de "namespaces".
  - Proveedores de clases de Suscripción: proporcionan suscripción, filtrado y procesamiento de manejadores de indicación.
  - Proveedores de registro de proveedor: proporcionan registro de proveedores.
  - ConfigProvider: maneja información de configuración de Pegasus.
  - UserAuthProvider: maneja logins, passwords, etc.
  - ShutdownProvider: proporciona soporte para apagar el CIMOM Pegasus.
  - Interop Provider: soporta muchas de las clases del esquema de Interop del DMTF.

  ✓ Proveedores genéricos.
  ✓ Proveedor genérico del sistema operativo.
  ✓ Proveedores de prueba y muestra.
  ✓ Proveedores del sistema gestionado: ComputerSystem, DNSAdmin, DNSService, Operating System, Process, Processor.
  ✓ Proveedores de estadísticas del CIMOM.
  ✓ Proveedores específicos de Linux: algunos de ellos son: DiskDrive, CDROMDrive, Interrupt, IOPOrt, IPROute, NetworkAdapter, Operating System, PCI Controller, PRocessor, ProviderData, ProviderSupport, SoftwareElement.

- **CIM Servlet:** proporciona una interfaz para que las aplicaciones cliente accedan información CIM a través de un servidor Web, es decir a través del puerto 80. Esto se hace porque en algunos casos no es deseable que los clientes se conecten directamente al puerto 5988 debido a restricciones de seguridad.

  El CIM Servlet toma los requerimientos XML/HTTP enviados por el servidor Web, los procesa y los envía al CIMOM a través de sockets Java. Sin embargo esta idea no se probó y no continuó siendo desarrollada.

- **Clientes de prueba Pegasus:** clientes simples de prueba que se están desarrollando como parte del proceso de desarrollo de Pegasus.
- **Cliente de prueba HTML Pegasus:** utiliza un servidor Web con un conjunto de módulos CGI y HTML, para permitir la entrada de operaciones Pegasus desde un browser Web y la recepción de la respuesta como páginas Web. Esto ha sido útil como una herramienta de prueba y puede ser utilizada para una amplia variedad de demostraciones.
- **SDK Cliente de Pegasus:** herramientas para desarrollar clientes Pegasus.
- **SDK Proveedor de Pegasus:** herramientas para construir proveedores Pegasus utilizando las interfaces C++.
- **Extensiones del Servicio Pegasus:** permiten extender las capacidades del CIMOM Pegasus.
- **Herramientas cliente**

  ✓ Administración del Servidor CIM de Pegasus:

  - cimuser: maneja los usuarios del servidor CIM de Pegasus.
  - cimconfig: maneja los parámetros de configuración del servidor CIM de Pegasus.
  - cimauth: maneja las características de autorización del servidor CIM. Actualmente esto consiste en autorizar los usuarios a "namespaces" específicos.
  - cimprovider: maneja información de los proveedores.

  ✓ Herramientas cliente de información: estas herramientas proporcionan información de los objetos manejados por el servidor.

  ✓ Herramientas de soporte de pruebas:

  - Pegasus Unit tests.
  - The Pegasus Test Suite.
  - TestClient: cliente básico para probar la operación de toda la plataforma.
  - CGICLient : este cliente no ha sido completamente actualizado a la versión 2.2.
  - wbemexec: herramienta de prueba que envía y recibe XML desde una entrada XML.
  - CLI: interfaz de línea de comando.

❖ **Mecanismos de seguridad**

- **Soporte para SSL:** SSL es implementado en Pegasus utilizando las librerías OpenSSL. Pegasus no proporciona estas librerías, pero SSL puede implementarse sobre cualquier plataforma que soporte OpenSSL.
- **Autenticación:** Pegasus soporta únicamente autenticación Básica, la autenticación Digest no está implementada.
- **Autorización:** el CIMOM concede permisos de lectura o escritura dentro de un "namespace" basado en ACLs. Estas listas son mantenidas en el "namespace" root/security. Debido a que las operaciones CIM se hacen dentro del contexto de un "namespace", estas ACLs determinan si una operación debe permitirse o no. Para operaciones que finalmente sean manejadas por un proveedor, el proveedor apropiado puede reemplazar el esquema de autorización. Esto permite a los proveedores implementar un control más estricto si se desea, es decir puede reemplazar el esquema de autorización. En este caso no se hacen llamadas al CIMOM.

3.1.4 OpenWBEM. Este proyecto comenzó en Caldera a comienzos del 2001. Su objetivo es ayudar a incrementar la compatibilidad de productos de gestión, especialmente para Linux. El proyecto fue transferido a Center 7 Inc en Noviembre de 2002. Esta empresa proporciona soporte, entrenamiento y consultoría para OpenWBEM.

Esta implementación WBEM es adecuada para aplicaciones comerciales y no comerciales. Puede ser vista como una versión C++ de WBEM Services de Sun, porque proporciona interfaces similares y soporta gran parte de la funcionalidad de esta plataforma.

OpenWBEM es configurable y muchos módulos se pueden cargar como librerías compartidas, las cuales pueden ser reemplazadas por el usuario.

- **Desarrollador:** Caldera
- **Lenguaje de implementación:** C++
- **Sistemas operativos soportados:** Linux (RedHat, SuSE, Debian, entre otros), SCO UnixWare , SCO OpenServer , Solaris y Darwin
- **Licencia:** BSD Style License
- **Versión actual :** 2.0.8

- **Componentes software:**

  - **CIMOM:** implementación C++ del manejador de objetos CIM.

    - ✓ Soporta dos tipos de codificación: xmlCIM (también conocida como codificación WBEM), y binaria, que es específica a OpenWBEM. La codificación binaria implementada por OpenWBEM no está basada en estándares. Fue desarrollada para optimizar operaciones con clientes WBEM que corren sobre la máquina local. Esto se logró serializando los objetos C++ sobre un socket de dominio Unix , con información adicional para identificar la API llamada, lo que eliminó la necesidad de la carga útil XML y la sobrecarga de procesar el requerimiento XML/HTTP. Así se logró un incremento bastante grande en el desempeño. Este protocolo requiere que el cliente tenga un conocimiento específico del formato de los datos binarios que representan la operación CIM, y la API cliente de OpenWBEM es la única que tiene conocimiento de este formato. Los desarrolladores de OpenWBEM han extendido la funcionalidad para que la representación binaria propietaria pueda ser enviada sobre HTTP y HTTPS, lo que permite al CIMOM utilizar el protocolo binario cuando se comunica con clientes corriendo en máquinas remotas que están utilizando la API cliente WBEM de OpenWBEM.
    - ✓ Soporta Operaciones CIM sobre HTTP a través del puerto 5988, Operaciones CIM sobre HTTPS a través del puerto 5989 (requiere OpenSSL), e IPC (Inter Process Communication).
    - ✓ Soporta todos los aspectos del modelo de eventos CIM.
    - ✓ Soporta SLP sobre el lado del cliente y del CIMOM utilizando OpenSLP. El proyecto OpenSLP comenzó en Caldera y es una implementación de fuente abierta de SLP adecuada para aplicaciones comerciales. La versión actual (1.0.x) es completamente estable y puede ser utilizada en aplicaciones.
    - ✓ La mayoría de las características son cargadas como librerías compartidas. Esto permite escoger las que realmente sean necesarias y no consumir recursos. Ellas son:

      - Soporte para indicaciones.
      - Librería WQL.
      - Módulos de Autenticación.
      - Interfaces proveedor.
      - Proveedores: son descargados de memoria si ellos no son utilizados por el tiempo configurado por el usuario.
      - Soporte para SLP, que es implementado como un proveedor.
      - Manejadores de requerimientos (CIM/XML o binario): pueden ser cargados y descargados por demanda y con tiempo de descarga configurable.

  - **Repositorio.**
  - **Compilador MOF.**
  - **API Cliente WBEM:** es casi idéntica a la API Cliente WBEM de WBEMServices.
  - **APIs proveedor**: extensibles a través de plug-ins.

    - ✓ Interfaz proveedor C++: es similar a la interfaz proveedor Java de WBEMSevices. Esta interfaz tiene soporte para los siguientes tipos de proveedores:

- Métodos
- Instancias
- Asociaciones
- Propiedades
- Indicaciones
- Consulta: son ejecutados a intervalos regulares (especificado por el proveedor) por el CIMOM, o ellos pueden comenzar su propio hilo y correr continuamente. Se pueden emplear como un proveedor de Indicación. Los proveedores de consulta no están asociados ninguna clase, propiedad o método especial.
- Autenticación: permiten al usuario escribir su propio mecanismo de autenticación.
- WQL

Un proveedor puede ser de múltiples tipos.

✓ Interfaz Proveerdor NPI.

- **Utilidad de línea de comando WQL:** utilizada para enviar "queries" WQL al CIMOM.
- **Implementación cliente WBEM.**

❖ **Mecanismos de seguridad**

- **Soporte para SSL.**
- **Autenticación:** es extensible a través de módulos plug in de autenticación.

✓ Módulo de autenticación PAM (Pluggable Authentication Modules): es el método de autenticación en el que el CIMOM utiliza la autenticación del sistema, es decir el cliente utiliza el login y el password con el que tiene acceso al sistema. Se emplean los encabezados de autenticación HTTP, o sea que el password es enviado en una forma insegura. Para evitar que sniffers obtengan las credenciales, se debe utilizar SSL junto con el módulo de autenticación PAM.

✓ Módulo de autenticación SIMPLE: esta forma de autenticación se basa en un archivo donde cada línea contiene un nombre de usuario y un password. En este caso también se emplean los encabezados de autenticación HTTP, por lo que también se recomienda utilizar SSL junto con este esquema de autenticación.

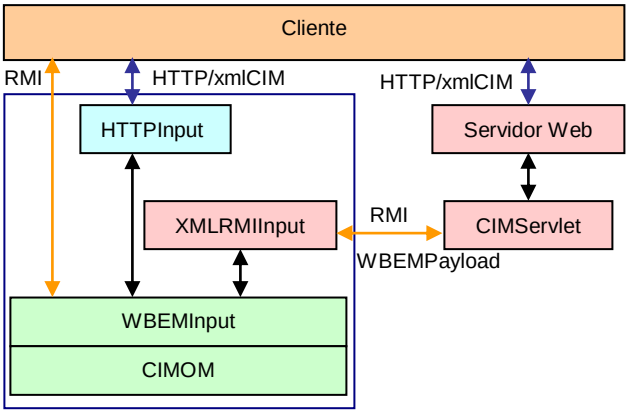
✓ Modulo de autenticación Digest: este mecanismo de autenticación utiliza cifrado para prevenir descubrimiento de passwords. Debido a esto, no puede integrarse con passwords del sistema, sino que es necesario crear un archivo de passwords utilizando una utilidad específica. Este esquema de autenticación es segura sobre una conexión no SSL.

- **Autorización:** listas de control de acceso

3.1.5 SNIA. También es llamada openCIMOM. Este proyecto ha sido trasladado al Open Group.

❖ **Desarrollador:** SNIA (Storage Networking Industry Association).
❖ **Lenguaje de implementación:** Java.
❖ **Sistemas operativos soportados:** todos.
❖ **Licencia:** actualmente el código fuente esta bajo la licencia SNIA Public License. En un futuro ella será cambiada a la licencia MIT, utilizada también por el proyecto Pegasus.
❖ **Componentes software:** la Figura 3.1 muestra los componentes WBEM que proporciona SNIA.

Figura 3.1 Componentes WBEM de SNIA



- **CIMOM:** implementación Java del manejador de objetos CIM.

  - ✓ Soporta xmlCIM.
  - ✓ Soporta Operaciones CIM sobre HTTP y HTTPS, y RMI.
  - ✓ Soporta el modelo de eventos CIM.

- **Repositorio.**
- **Compilador MOF.**
- **API cliente Java.**
- **APIs proveedor:** la implementación CIMOM de SNIA soporta las siguientes interfaces proveedor:

  - ✓ Interfaz proveedor Java: soporta los siguientes tipos de proveedores: instancias, asociaciones, métodos, propiedades e indicaciones.
  - ✓ Interfaz proveedor NPI.
  - ✓ Interfaz proveedor CMPI.

- **HTTPInput:** acepta conexiones y crea un hilo por cada una de ellas, para pasar y procesar el requerimiento xmlCIM.

- **CIMServlet:** la implementación actual SNIA proporciona comunicación entre su CIMOM y un cliente a través de una conexión socket utilizando el puerto 80. Sin embargo, si un servidor Web está corriendo sobre el mismo sistema que el CIMOM, también querrá utilizar el puerto 80. El CIMServlet permite que el CIMOM SNIA coexista con un servidor Web instalado en la misma máquina (debido a que el servidor Web estará escuchando por el puerto 80, el puerto del CIMOM debe cambiarse). La Figura 3.1 muestra como los requerimientos WBEM que llegan al servidor Web son enrutados al CIMOM a través del CIMServlet. El CIMServlet obtiene los datos del encabezado HTTP y pasa estos datos junto con los datos XML al CIMOM. Esta información es encapsulada en un objeto WBEMPayload. El CIMServlet se comunica con el CIMOM sobre una conexión RMI. El paradigma de entrada al CIMOM ha sido modificada para permitir la adición de un modo de transporte XML/RMI. La arquitectura propuesta proporciona un solo método remoto que toma un objeto WBEMPayload que contiene información del encabezado HTTP y el flujo de entrada XML, ejecuta el requerimiento XML y retorna la salida XML sobre el flujo de salida HTTP. Sin embargo esta idea no se probó y no continuó siendo desarrollada.
- **XMLRMIInput:** procesa requerimientos provenientes desde el CIMServlet.

❖ **Mecanismos de seguridad**

- **Soporte para SSL.**
- **Autenticación:** en adición a los esquemas de autenticación Basic y Digest, el módulo de autenticación se puede configurar para manejar otros esquemas de autenticación no muy complicados (por ejemplo mecanismos de autenticación que sigan el modelo "challenge-response" descrito en las especificaciones del protocolo HTTP). Si se requieren esquemas más sofisticados se puede desarrollar un módulo de autenticación.

  La autenticación puede ser realizada por el servidor Web o por el CIMServlet. Si WBEM quiere mantener una lista de usuarios con ciertos accesos a ciertos recursos, WBEM tendrá su propia base de datos de usuarios. Esto diferenciará los usuarios WBEM de los usuarios del servidor Web y no habrá que preocuparse por actualizar dos o más archivos.

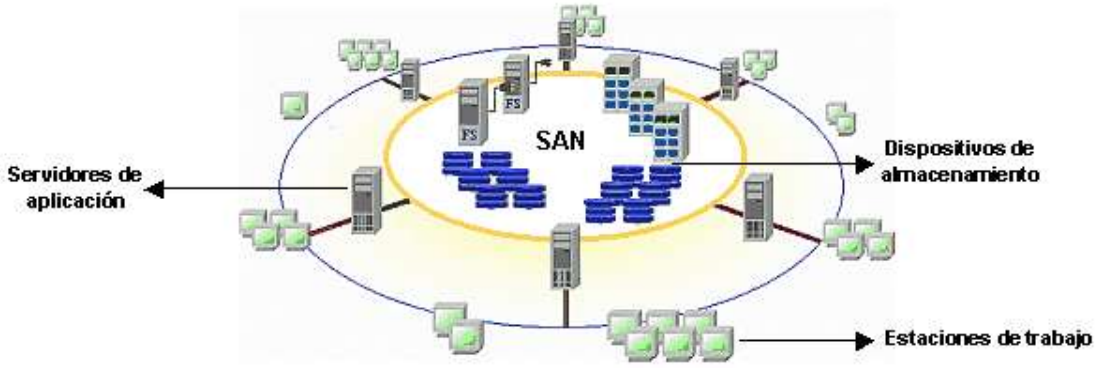
3.1.6 La Iniciativa de Gestión de Almacenamiento. SNIA lanzó en el 2002 la Iniciativa de Gestión de Almacenamiento SMI (Storage Management Initiative), para crear e impulsar la adopción de una interfaz de gestión interoperable y funcional para SANs.

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 utiliza los estándares CIM y WBEM para gestionar dispositivos sobre redes de almacenamiento. Además, Bluefin proporciona una interfaz "proxy", que permite a los equipos ya adquiridos, interoperar con productos nuevos, habilitados con Bluefin. Las compañías que contribuyeron al desarrollo de Bluefin son: BMC Software, Brocade, Computer Associates, Dell, EMC, Emulex, GadzooxNetworks, HP, Hitachi Data Systems, IBM, JNI, Prisa Networks, QLogic, Storage Technology, Sun Microsystems y VERITAS Software.

Las Redes de Área de Almacenamiento SANs (Storage Area Networks) se basan en el concepto de separar los dispositivos de almacenamiento de los servidores, y colocarlos directamente sobre una red de fibra óptica. Es decir, el almacenamiento ya no está en los servidores sino que es trasladado a su propia red. Esto permite conexiones muchos a muchos entre servidores y dispositivos de almacenamiento y entre dispositivos de almacenamiento. Además, las copias de datos se pueden hacer sin afectar toda de la red, porque el tráfico se transporta a través de la SAN, no de la LAN.

Los protocolos utilizados en las SANs son el protocolo FC (Fibre Channel) que sirve para la comunicación entre dispositivos hardware, y el protocolo SCSI (Small Computer System Interface) que es utilizado por las aplicaciones software para comunicarse con los dispositivos de almacenamiento. La Figura 3.2 muestra una forma de visualizar una SAN.

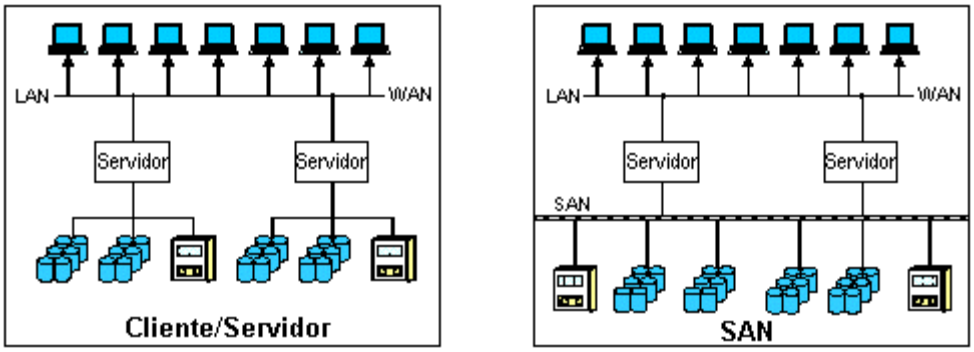
Figura 3.2 SAN



La SAN es donde se encuentran todos los datos y está compuesta por un conjunto de dispositivos de almacenamiento conectados con fibra óptica. Los servidores de aplicación actúan como un punto de acceso a los datos.

En Figura 3.3 se muestra la arquitectura de una red cliente-servidor y una SAN.

Figura 3.3 Arquitectura Cliente/Servidor y Arquitectura SAN



Últimamente el concepto de SANs se ha expandido para incluir otras posibilidades además de fibra óptica, como por ejemplo, Gigabit Ethernet y SCSI sobre IP. Sin embargo, algunos autores argumentan que esto no constituye una SAN.

Es posible que la próxima generación de productos SAN regrese a los protocolos de red tradicionales TCP/IP. Aunque FC y SCSI son más eficientes, los switches FC son más costosos comparados con los switches Ethernet, además 10 Gigabit Ethernet permite a TCP/IP sobrepasar a FC en velocidad de transferencia a pesar de la mayor carga de procesamiento en la transmisión de datos.

Esta nueva arquitectura proporciona los siguientes beneficios:

- **Incrementar la flexibilidad:** las fuentes de datos pueden estar disponibles a múltiples servidores. El límite está determinado por la infraestructura de fibra. Esto permite la reutilización de los recursos de almacenamiento en una forma más eficiente.
- **Mejorar la escalabilidad:** los servidores y sus recursos de almacenamiento ya no están juntos. Ahora se puede adicionar dinámicamente nuevos dispositivos de almacenamiento, sin tener que adicionar un nuevo servidor.

- **Mejorar el desempeño:** en lugar de transportar datos de un servidor a otro a través de LANs o WANs, los datos se trasladan de un dispositivo de almacenamiento a otro a través de fibra óptica. De esta forma los servidores solo se encargan de las aplicaciones, y la capacidad de la redes LAN y WAN es solo para los usuarios.
- **Mejorar la disponibilidad:** cualquier servidor puede reemplazar a otro en caso de fallas, mientras ellos tengan acceso a los mismos dispositivos de almacenamiento y sirvan los mismos usuarios. No hay restricciones por cuellos de botella, porque ahora se pueden obtener los datos a través de cualquier servidor de aplicación. Además los datos se pueden duplicar. Cuando se hacen cambios sobre los datos, todas las copias se actualizan simultáneamente (replicación síncrona). Sin embargo, para la mayoría de datos es aceptable que las copias estén algunos segundos o minutos desactualizadas (replicación asíncrona). Una utilidad fundamental de las SANs es el conjunto de facilidades de réplica, ellas deben ser completamente exactas, confiables y tener un alto desempeño. Por lo tanto, el modelo de SAN proporciona disponibilidad de datos, gracias a la utilización de redundancia de datos automática, copias de seguridad automáticas y manteniendo copias para recuperación ante desastres.
- **Mejorar la gestión:** debido a la centralización de los datos y la confiabilidad que proporcionan las SANs, la gestión es más segura y simple.
- **Mejorar la confiabilidad:** esto se logra con la fibra óptica, además, en un futuro, la tecnología SAN proporcionará la habilidad para reemplazar o reparar cualquier componente durante la operación normal, reconfigurar el sistema y adicionar nuevos componentes.

Sin embargo la mayoría de las SANs que existen actualmente son soluciones propietarias porque muchos vendedores tienen su propia forma de interpretar el estándar FC. Debido a esto, la incompatibilidad entre dispositivos es el mayor problema cuando se implementan redes de este tipo.

Afortunadamente, Fibre Channel Industry Association (FCIA), Storage Networking Industry Association (SNIA) y otras organizaciones están impulsado la tecnología y desarrollando SMI para solucionar la falta de interoperabilidad entre implementaciones. Compaq, EMC, Hewlett-Packard, Sun Microsystems y otros vendedores han adoptado rápidamente la tecnología y ya ofrecen soluciones completas.

CIM-SAN es el primer entorno multi-vendedor de desarrollo y demostración para acelerar la implementación de los estándares CIM y WBEM en productos de red de almacenamiento. El programa CIM-SAN se originó por el gran éxito de la demostración CIM-SAN-1 en la conferencia Storage Networking World en Octubre del 2002 donde 17 vendedores integraron 32 productos. CIM-SAN-1 probó que CIM y WBEM están acelerando el desarrollo y la integración de más productos de gestión de almacenamiento. La fase actual del programa es CIM-SAN-2. En ella, se están incorporando nuevas características de seguridad, descubrimiento y gestión, que mejorarán la eficiencia de la red. El mejoramiento más significativo entre CIM-SAN-1 y CIM-SAN-2 es el incremento en la colaboración por parte de los vendedores. Además CIM-SAN-2 ha llegado a ser el entorno para solucionar el problema de interoperabilidad entre las diferentes implementaciones WBEM. Hoy, la mayoría de implementaciones de uso libre, como WBEM Services, Pegasus, SNIA y Open WBEM, pueden comunicarse, y esto muestra que CIM es el mejor modelo para lograr la interoperabilidad en la gestión.

Cuando SMI-S se adopte ampliamente, reemplazará todos los modelos y protocolos con un modelo y un protocolo común. Los desarrolladores de aplicaciones de gestión podrán abarcar dispositivos de varios vendedores rápidamente, los costos de desarrollo y pruebas disminuirán, la satisfacción del usuario incrementará y por lo tanto las ganancias.

3.2 IMPLEMENTACIONES WBEM COMERCIALES

3.2.1 WMI. WMI es un componente clave de los servicios de gestión de Windows, que también incluyen: Directorio Activo, MMC (Microsoft Management Console) y WSH (Windows Script Host).

- **Desarrollador:** Microsoft
- **Lenguaje de implementación:** C++

- **Sistemas operativos soportados:** Windows. A partir de Windows 2000, WMI viene incluido en el sistema operativo. Para Windows 95 y 98 se puede descargar sin ningún costo.
- **Versión actual:** 1.5
- **Componentes software:**

  - **CIMOM:** implementación C++ del manejador de objetos CIM. Sobre computadores Windows 98 el CIMOM corre como un archivo ejecutable estándar. Sobre computadores Windows NT, Windows 2000 y Windows XP corre como un servicio. En todos los casos, el CIMOM comienza cuando una aplicación cliente hace la primera llamada y se conecta exitosamente, y continua corriendo mientras las aplicaciones de gestión estén requiriendo sus servicios.

    - ✓ No soporta xmlCIM.
    - ✓ No soporta Operaciones CIM sobre HTTP, en su lugar se utiliza DCOM (Distributed Component Object Model).

  - **Repositorio.**
  - **Compilador MOF.**
  - **APIs cliente:**

    - ✓ **API COM:** puede ser accedida desde C y C++.
    - ✓ **API Scripting:** puede ser accedida desde Visual Basic, VBScript, JScript, Perl, o cualquier otro lenguaje scripting soportado por Windows. Está implementada en un solo componente de automatización llamado wbemdisp.dll que se encuentra en el directorio *systemroot*\System32\Wbem.Las aplicaciones en Visual Basic y los scripts pueden hacer la mayoría de las operaciones WMI a través de la API Scripting, pero las aplicaciones escritas en C++ pueden hacer más. Por ejemplo, los scripts pueden recibir eventos pero no pueden enviarlos. Además, un proveedor puede solo se se puede escribir en C++. Los scripts y aplicaciones en Visual Basic deben acceder clases cuyas instancias sean proporcionadas por proveedores existentes.

    Aunque aprender y utilizar la API Scripting es más fácil, C++ tiene la flexibilidad y potencia que le falta a los lenguajes scripting. Por lo tanto, la API COM es más adecuada para aplicaciones sofisticadas y es requerida para escribir proveedores.

    Además de estas APIs, WMI ofrece otras formas de comunicación entre cliente y CIMOM, ellas son:

    - ✓ Internet Explorer y ASPs (Active Server Pages): pueden almacenar scripts WMI.
    - ✓ Adaptador ODBC de WMI: proporciona una API que permite a las aplicaciones basadas en ODBC utilizar los datos del Repositorio CIM como si él fuera una base de datos. Sin embargo, el adaptador ODBC de WMI tiene las siguientes desventajas:

      - Se puede utilizar solamente con Access.
      - La información que se obtiene con este adaptador es de solo lectura.
      - Soporta solamente declaraciones (statements) SELECT de SQL.

    Es soportado en Windows NT versión 4.0 con Service Pack 4 o superior y Windows 2000 como una extensión de valor agregado al sistema operativo. El adaptador ODBC no está disponible en la familia Windows Server 2003.

    - ✓ Extensión ADSI de WMI: permite integrar WMI y los servicios de directorio. Mientras que la fortaleza de WMI está en proporcionar control sobre un computador específico, la fortaleza del Directorio Activo está en proporcionar información general a través de la toda la red. El adaptador ADSI no está disponible en la familia Windows Server 2003.

- **APIs proveedor:** la API COM es la interfaz utilizada por los proveedores para interactuar con el manejador de objetos CIM. Soporta los siguientes tipos de proveedores:

  - ✓ Instancias.
  - ✓ Clases.
  - ✓ Métodos.
  - ✓ Propiedades.
  - ✓ Eventos o Indicaciones.
  - ✓ Consumidor de eventos: proporcionan información de la localización de aplicaciones que reciben los eventos.

  Los proveedores pueden pertenecer a uno o más de estos tipos.

- **Esquema Windows:** colección de clases que se extienden del esquema CIM y representan objetos gestionados en un entorno Windows. Incluye las siguientes clases: Clases de Sistema, Clases Win32, Clases Consumidoras Estándar, Clases MSMCA y Clases C++ WMI.

  - ✓ Clases del Sistema: son una colección de clases predefinidas basadas en CIM. Algunas de ellas se incluyen en cada "namespace". Los objetos de estas clases se utilizan para soportar las actividades WMI, como registro de proveedores y eventos, seguridad y notificación de eventos. Algunos objetos son temporales y algunos son almacenados en el repositorio como instancias de las Clases del Sistema. Estas clases siguen una convención de nombramiento que consiste de dos underscore (__) seguidos por el nombre de la clase. Debido a que __ está reservado para uso del sistema, no se pueden crear clases que comiencen con esta secuencia.

  - ✓ Clases Win32: estas clases permiten manipular una variedad de objetos. Las categorías de clases Win32 son las siguientes:

    - Clases hardware del sistema: agrupa clases que representan objetos de hardware, como dispositivos de entrada, discos duros, tarjetas de expansión, dispositivos de video, dispositivos de red, entre otros. Las clases hardware se dividen en: Clases de Dispositivos de Refrigeración, Clases de Motherboard, Controladores y Puertos, Clases de Dispositivos de Networking, Clases de Potencia, Clases de Impresión, Clases de Telefonía y Clases de Video y Monitor

    - Clases del sistema operativo: agrupa clases que representan la configuraciones de un entorno computacional, como configuraciones de inicio, de COM, de escritorio, de controladores, de usuario y del registro. Esta categoría tiene las siguientes subcategorías:COM, Desktop, Drivers, File system, Job objects, Memory and page files, Multimedia audio/visual, Networking, Operating system events, Operating system settings, Processes, Registry, Scheduler jobs, Security, Services, Shares, Start menu, Storage,Users, Windows NT event log, y Windows Product Activation.

    - Clases de las aplicaciones instaladas: agrupa clases que representan objetos relacionados con software. El acceso a estos objetos es soportado por el Instalador de Windows. Algunos ejemplos de objetos en esta categoría son Productos Instalados, Especificaciones de Archivos, Acciones de Registro, entre otros.

    - Clases de gestión del servicio WMI : estas clases se utilizan para configurar el servicio WMI y manejar las operaciones WMI. Esta categoría incluye las siguientes subcategorías:

      - ➢ Clases de configuración WMI: proporcionan métodos que configuran el servicio WMI.

- Clases de gestión WMI: representan parámetros operacionales para el servicio WMI.

- Clases de Contador de Desempeño (Performance Counter): permiten acceder los datos de desempeño calculados por los proveedores de alto desempeño. Las diferentes versiones del sistema operativo tienen diferentes conjuntos de clases de desempeño. Cada clase describe la versión de Windows en la cual ella es soportada. Las clases derivadas de clase abstracta Win32_PerfFormattedData contienen datos procesados. Ellos son proporcionados por el proveedor Cooked Counter. Las clases derivadas de la clase abstracta Win32_ PerfRawData contienen los datos de desempeño sin procesar. Estos datos son proporcionados por el proveedor Performance Counter. Las clases de desempeño, de datos procesados y sin procesar, se generan en el repositorio WMI cuando el sistema inicia, a partir de las librerías de desempeño en el registro. Cuando se registra una nueva librería de desempeño por el proceso ADAP (AutoDiscovery/AutoPurge), también se adicionan las clases correspondientes al repositorio. Las librerías de desempeño se pueden crear a partir de las clases de desempeño de WMI por el proceso Reverse Adapter.

  El formato del nombre de la clase de desempeño indica si ella proporciona datos procesados o sin procesar, la librería de desempeño desde la cual fue cargada y el objeto que representa en la librería de desempeño. Por ejemplo, las propiedades de la clase Win32_PerfFormattedData_PerfDisk_PhysicalDisk representan las propiedades de un objeto PhysicalDisk en la librería de desempeño PerfDisk.

✓ Clases consumidoras estándar: se puede crear instancias de estas clases para proporcionar la clase consumidora permanente, que responde cuando se producen los eventos especificados en un filtro. Ellas son:

- ActiveScriptEventConsumer: ejecuta un script predefinido en un lenguaje scripting arbitrario cuando se presenta un evento específico. Esta clase está disponible en Windows 2000 y XP.
- ScriptingStandardConsumerSetting: proporciona datos de registro comunes a todas las instancias de la clase ActiveScriptEventConsumer.
- LogFileEventConsumer: escribe cadenas personalizadas a un archivo de texto cuando se producen eventos. Este consumidor está disponible en Windows XP.
- NTEventLogEventConsumer: registra un mensaje específico al registro de eventos de Windows NT cuando se produce un evento. Este consumidor está disponible en Windows XP.
- SMTPEventConsumer: envía un mensaje de email utilizando SMTP cada vez que se produce un evento. Esta clase no soporta archivos adjuntos y la codificación del mensaje debe ser ASCII. Este consumidor está disponible en Windows 2000 y XP.
- CommandLineEventConsumer: lanza un proceso arbitrario en el sistema local cuando se produce un evento. Cuando se utiliza esta clase es importante que se asegure el archivo que se quiere ejecutar. Si el archivo no está en una localización segura, o asegurado con una ACL, un usuario no autorizado puede reemplazarlo. Este consumidor está disponible en Windows XP.

Estas clases tienen un proveedor WMI asociado a ellas, además, no están disponibles para utilizarlas, hasta que sean registradas en el repositorio.

✓ Clases MSMCA (Microsoft Machine Check Architecture): vienen incluidas en el sistema operativo y exponen MCA (Machine Check Architecture). Las versiones de 64 bits de Windows XP Professional y Windows Server 2003 soportan esta arquitectura, que es definida por Intel para el procesador Itanium. MCA es una arquitectura hardware y software

que intenta mejorar las características (confiabilidad, disponibilidad y escalabilidad) del sistema. MCA inluye:

- Componentes hardware responsables de identificar y entregar eventos causados por errores en hardware al software.
- Componentes software responsables de identificar, analizar y si es posible, resolver estos errores.

Casi todos los eventos MCA son errores hardware, pero no todo error hardware es reportado como un evento MCA.

- ✓ Clases C++: incluyen clases que se utilizan para crear proveedores.

- **Proveedores:** los proveedores son servidores COM y DCOM estándar que funcionan como mediadores entre los objetos gestionados y el CIMOM. Los proveedores pueden ser implementados como DLLs, servicios o archivos ejecutables estándar.

  En WMI un proveedor consta de un archivo MOF y una DLL (dynamic-link library). La DLL actúa como un intermediario entre la infraestructura WMI y el recurso gestionado (ella llama las APIs nativas del recurso gestionado). El archivo MOF contiene las definiciones de clase que describen todos los aspectos del recurso gestionado. Cada clase define las propiedades y los métodos que pueden ser ejecutados sobre el recurso gestionado. Cuando se instala un proveedor, la DLL se registra con el sistema operativo y WMI, y el archivo MOF es compilado, para que las clases se carguen en el repositorio.

  WMI proporciona varios proveedores preinstalados (también llamados proveedores estándar o built-in) para el sistema operativo Windows. Ellos son:

  - ✓ Proveedor Win32: proporciona acceso a información del sistema operativo, dispositivos periféricos, sistemas de archivos e información de seguridad. Este proveedor obtiene la información haciendo llamadas al sistema operativo y pidiendo información al registro. Es un proveedor de instancias y métodos.

  - ✓ Proveedor del Registro: permite recuperar y modificar datos en el registro del sistema, y recibir notificaciones cuando ocurren eventos. Es un proveedor de instancias, propiedades y de eventos.

  - ✓ Proveedor SNMP: permite que las variables de la MIB sean leídas y modificadas, y que los traps puedan ser automáticamente convertidos a eventos a través de WMI. Viene con un compilador que convierte variables MIB en objetos WMI.

    En Windows 2000 y XP el proveedor configura un nombre de comunidad y el agente puede aceptar paquetes SNMP desde otros computadores. Después de Windows XP SP1 el proveedor SNMP no configura un nombre de comunidad y el agente es configurado para aceptar paquetes SNMP solamente desde el equipo local.

    En Windows NT el proveedor está disponible y viene preinstalado. En Windows 2000 y XP el proveedor está disponible pero no viene preinstalado. El proveedor SNMP está incluido en el CD como un componente Windows opcional que se puede instalar. Esto asegura compatibilidad hacia atrás con el conjunto de características de Windows 2000, pero no significa que el proveedor esté disponible en un futuro. No está disponible en Windows 95, 98 y ME.

  - ✓ Proveedor del Instalador de Windows (Windows Installer): hace a las funciones del Instalador de Windows disponibles a través de WMI. La tecnología Windows Installer organiza el software en una jerarquía de partes que tienen subpartes. Cada parte contiene

propiedades del producto software instalado. Además, permite instalar y obtener información de las aplicaciones habilitadas con MSI. En Windows XP y anteriores este proveedor es un componente del sistema operativo y no requiere instalación desde el CD. Es un proveedor de instancias y métodos.

- ✓ Proveedor de Directorio Activo o proveedor DS (Directory Services): permite que la información del Directorio Activo sea accedida utilizando WMI. Para lograr esto, este proveedor realiza la conversión de los objetos del Servicio de Directorio a WMI. En el "namespace" LDAP se puede referenciar cualquier objeto del Directorio Activo. Es un proveedor de clases e instancias. No está disponible en Windows NT, ME, 98 y 95.

- ✓ Proveedor WPA (Windows Product Activation): el proveedor WPA de WMI soporta administración WPA a través de interfaces WMI. WPA es una tecnología anti-piratería que requiere que los usuarios contacten un servidor de licencia Microsoft Clearinghouse online o por teléfono, para crear una asociación entre el ID del producto (PID) y el ID del hardware del computador. Esta asociación se requiere para activación del sistema. El PID es derivado de la clave del producto y el código del producto de Microsoft. El ID del hardware es derivado de un determinado número de características de la máquina. No está disponible en Windows 2000 y anteriores. Es un proveedor de clases e instancias.

- ✓ Proveedor de Contador de Desempeño (Performance Counter): proveedor de alto desempeño que proporciona datos de desempeño no calculados o sin procesar, como el tiempo que un disco gasta escribiendo datos. Es necesario aplicar fórmulas a estos datos para que ellos sean útiles al usuario. Cualquier contador de desempeño instalado en el sistema será automáticamente visible a través de este proveedor.

  Los contadores de desempeño son parte de los microprocesadores modernos. Ellos miden su desempeño sin afectar las aplicaciones. Una librería de desempeño permite acceder contadores de desempeño sobre muchos microprocesadores a través de una interfaz uniforme. El proveedor de Contador de Desempeño almacena los datos en clases WMI derivadas de la clase Win32_PerfRawData. Cada clase de desempeño corresponde a un objeto de desempeño en una librería de desempeño. Las propiedades de estas clases representan los contadores para el objeto. El calificador CounterType en cada propiedad señala la fórmula utilizada para calcular los resultados. WMI coloca los objetos contador en el "namespace" \root\cimv2. El nombre de la clase para un objeto contador es de la forma Win32_PerRawData_<nombre del servicio>_<nombre del objeto>. Por ejemplo, el nombre de la clase WMI que contiene los contadores de un disco es Win32_ PerRawData_PerfDisk_LogicalDisk. No está disponible en Windows NT, Me, 98 y 95

- ✓ Proveedor Cooked Counter: proveedor de alto desempeño que proporciona datos procesados, como por ejemplo el porcentaje de tiempo que un disco gasta escribiendo datos. El proveedor Cooked Counter procesa los datos obtenidos desde el proveedor Performance Counter y los almacena en clases derivadas de Win32_PerfFormattedData. WMI coloca los objetos derivados de estas clases en el "namespace" \root\cimv2. La fórmula que utiliza el proveedor Cooked Counter para procesar los datos se puede determinar buscando el valor del calificador Cooking Type en los tipos de Performance Counter. Este calificador es el mismo que el calificador CounterType en las clases Win32_ PerfRawData. Este proveedor no está disponible en Windows 2000 y anteriores.

- ✓ Proveedor Disk Quota: permite a los administradores controlar la cantidad de datos que cada usuario almacena en un sistema de archivos NTFS. El proveedor puede registrar eventos cuando los usuarios están cerca de pasar el límite y negar espacio adicional de disco a usuarios que excedieron su espacio. Es disponible solo para Windows 2000. Es un proveedor de instancias, métodos y eventos.

- ✓ Proveedor DFS (Distributed File System): proporciona la habilidad para agrupar lógicamente particiones que se encuentran en múltiples servidores y enlazarlas transparentemente en una estructura de árbol. DFS es para los servidores y particiones, lo que los sistemas de archivos son para los discos duros. Los sistemas de archivos proporcionan un acceso uniforme a los sectores de los discos, DFS proporciona una convención de nombramiento uniforme para servidores, particiones y archivos. De esta forma, DFS hace posible organizar servidores de archivos y sus particiones en una jerarquía lógica, haciendo considerablemente más fácil a una organización manejar y utilizar sus recursos de información. Debido a que DFS convierte el almacenamiento físico en una representación lógica, el principal beneficio es que la localización física de los datos es transparente a los usuarios y aplicaciones.

  Este proveedor puede crear un nodo DFS y utilizar una clase asociación para representar conexiones entre un nodo y las particiones y servidores enlazados a ese nodo. Este proveedor no está disponible en Windows XP y anteriores, solo en sistemas Windows 2000 Advanced Server y la familia Windows Server 2003. Es un proveedor de instancias y métodos.

- ✓ Proveedor Trustmon: crea clases con información acerca de las relaciones de confianza entre dominios. No está disponible en Windows XP y anteriores. Es un proveedor de instancias.

- ✓ Proveedor Session: permite a los desarrolladores manejar sesiones de red y conexiones a través de comandos, scripts, páginas Web o lenguajes de programación estándar. No está disponible en Windows NT, Me, 98 y 95. Es un proveedor de instancias y propiedades.

- ✓ Proveedor Shadow Copy: proporciona funciones de gestión para las copias de archivos que están en un recurso compartido, como un servidor de archivos. No está disponible en Windows XP y anteriores. Es un proveedor de instancias y propiedades.

- ✓ Proveedor Storage Volume: proporciona funciones de gestión de volumen de almacenamiento. Se puede utilizar para manejar letras de drives y espacio de almacenamiento. Este proveedor tiene métodos para defragmentar, revisar, montar, desmontar y formatear un volumen. No está disponible en Windows XP y anteriores. Es un proveedor de instancias y métodos.

- ✓ Proveedor Event Log: proporciona acceso a datos y notificaciones de eventos del Registro de eventos de Windows 2000. Este proveedor no está disponible en Windows 95, 98 y ME. Es un proveedor de instancias, métodos y eventos.

- ✓ Proveedor IP Route: proporciona información de enrutamiento de red, incluyendo, pero no limitado a, la información disponible a través del comando route print. Este proveedor permite examinar cómo los paquetes son enrutados a través de la red, a y desde una máquina particular. Los datos proporcionados se obtienen de las tablas de enrutamiento de IPv4 y las tablas de enrutamiento almacenadas en el registro. El proveedor permite la actualización de una entrada de la tabla de enrutamiento, asumiendo que todos los valores escritos son válidos. También proporciona eventos para cambios en la tabla de enrutamiento. No está disponible en Windows 2000 y anteriores. Es un proveedor de instancias y eventos.

- ✓ Proveedor Kernel: permite ver los eventos del kernel en creación de procesos, terminación de procesos, creación de hilos, terminación de hilos y carga de módulos. Es un proveedor de eventos.

- ✓ Proveedor Ping: proporciona acceso a la información de estado proporcionada por el comando ping estándar. No está disponible en Windows 2000 y anteriores. Es un proveedor de instancias

- ✓ Proveedor Power Management Event: permite modelar los protocolos de gestión de potencia de Windows 2000, como APM (Advanced Power Management) y ACPI (Advanced Configuration and Power Interface). Es un proveedor de eventos. No está disponible en Windows 95, 98, ME y NT.

- ✓ Proveedor Security: permite recuperar o cambiar las configuraciones de seguridad que controlan los derechos de acceso a los archivos, directorios y particiones del sistema NTFS de Windows NT y 2000. No está disponible en Windows ME, 98 y 95. Es un proveedor de instancias y métodos.

- ✓ WDM Provider: permite acceder información, configurar dispositivos, y proporcionar notificación de eventos de los drives de los dispositivos habilitados con WDM (Windows Driver Model). WDM es una interfaz del sistema operativo a través de la cual los componentes hardware proporcionan información y notificación de eventos. Es un proveedor de clases, instancias, eventos y métodos.

- ✓ Proveedor de Vista: permite crear nuevas clases de clases existentes. Se puede tomar las propiedades de diferentes clases, diferentes "namespaces", o diferentes computadores y combinarlas en una sola clase. Es un proveedor de instancias y métodos.

Además de los proveedores estándar, también es posible desarrollar proveedores específicos de la aplicación. Ellos son creados por vendedores para proporcionar información de sus productos al CIMOM.

- **Compilador del Módulo de Información SNMP:** compila información del esquema SNMP nativo al formato utilizado por CIM.

- **Browser de objetos WMI:** sirve para desplegar el árbol de objetos del Repositorio, ver y editar detalles como propiedades, métodos, calificadores, asociaciones y ejecutar métodos.

- **WMI CIM Studio:** sirve para desplegar el árbol de herencia de clases del Repositorio, ver y editar información de instancias y clases, como propiedades, métodos, calificadores y asociaciones, y para ejecutar operaciones. Además, proporciona asistentes para generar y compilar archivos MOF y para generar código de proveedor.

- **Registro de eventos WMI:** sirve para desplegar y modificar consumidores de eventos, filtros y timers para un "namespace" específico, y asociaciones entre filtros y consumidores, a través de una interfaz gráfica.

- **Visor de eventos WMI:** despliega los eventos generados por el CIMOM para todas las instancias de los consumidores registrados (cuando el CIMOM genera eventos, los objetos representándolos son enviados automáticamente a los consumidores registrados para ese tipo de objeto). Además esta aplicación sirve para ver información de eventos, como la hora y fecha del evento, punto de origen y una descripción.

- **Visor HTML de WMI:** sirve para desplegar el árbol de herencia de clases, buscar una clase específica, desplegar la definición de una clase existente, incluyendo propiedades, calificadores y métodos, y desplegar instancias de una clase existente.

- **Manejador de usuario WMI:** permite crear usuarios WMI y configurar sus permisos de acceso.

- **Administrador WMI:** permite realizar copias de seguridad del Repositorio y habilitar y configurar el acceso de usuarios.

❖ **Mecanismos de seguridad.** La seguridad de WMI es una extensión del subsistema de seguridad en los sistemas operativos Windows. La seguridad WMI incluye:

- **Seguridad de usuario:** soporta el modelo de seguridad de Windows NT, Windows 2000 y posteriores. WMI utiliza un Descriptor de Seguridad, SD (Security Descriptor) para controlar el acceso a servicios WMI. El SD contiene una ACL que describe cuáles usuarios tienen acceso a qué recursos, incluyendo modificación del Repositorio WMI, acceso y ejecución remota, y permisos de entrada. WMI almacena estos ACEs (Access Control Entry) en el Repositorio WMI. Una ACL vacía en el descriptor de seguridad concede acceso ilimitado a todos los usuarios.

- **Seguridad a nivel de "namespace":** al igual que la infraestructura WMI, cada "namespace" tiene un SD para determinar los usuarios que pueden acceder un "namespace" específico. La ACL de un "namespace" es heredada por los "namespcaces" que se deriven de él. Sin embargo, es posible escoger los "namespaces" desde los cuales se deriva uno nuevo, a través de una bandera (flag) que es configurada para todos los "namespaces" que permiten que los "namespaces" hijos hereden la ACL. Si no se desea que un "namespace" herede desde su padre, se debe deshabilitar la bandera, y la ACL debe ser configurada para los "namespaces" hijos. Esto permite que los "namespaces" tengan configuraciones de seguridad únicas.

  La Tabla 3.3 lista los permisos WMI disponibles para "namespaces", los cuales son configurados a través de la aplicación Control WMI (esta herramienta se encuentra en el directorio systemroot\System32\Wmimgmt.msc). Sobre computadores corriendo Windows NT 4.0 SP4, Windows 98 y Windows 95, la aplicación Control WMI es llamada Wbemcntl.exe (esta herramienta se encuentra en el directorio systemroot\System32\Wbem en Windows NT 4.0 SP4).

Tabla 3.3 Permisos de "namespace" en WMI

Permiso	Descripción	Administradores	Todos
Ejecutar métodos	Permite llamar métodos en un "namespace" específico. Sin embargo, el proveedor revisa si el usuario tiene el derecho para ejecutar estas tareas. Por ejemplo, un usuario no puede correr un script que detenga un servicio a menos que el usuario tenga el privilegio correspondiente.	X	X
Escritura completa	Permite crear o modificar un "namespace", una clase del sistema o una instancia.	X	
Escritura parcial	Permite crear o modificar cualquier clase estática o cualquier instancia, excepto instancias de las clases del sistema.	X	
Escritura de proveedor	Permite a los usuarios escribir clases e instancias para proveedores WMI.	X	X

Habilitar cuenta	Concede permisos de lectura a un “namespace” WMI. Esto permite correr programas que recuperan datos pero solamente sobre el computador local.	X	X
Llamada remota habilitada	Permite acceder un “namespace” desde un computador remoto. Por defecto, este permiso solo es concedido a los administradores, los usuarios regulares no pueden obtener ningún tipo de información de un computador remoto.	X	
Seguridad de lectura	Permite leer pero no modificar el descriptor de seguridad para un “namespace”.	X	
Modificar seguridad	Permite modificar el descriptor de seguridad para un “namespace” WMI.	X	

Los permisos WMI se aplican a nivel de “namespace” y a todas las clases dentro de él. Por defecto, los permisos son aplicados solamente al “namespace” root y heredados por todos los “namespaces” hijos.

Por defecto, el grupo de seguridad Administradores estándar tiene completo control de WMI y el Repositorio, sobre el computador local y computadores remotos. Todos los otros usuarios tienen permiso de Ejecutar métodos, Escritura de proveedor y Habilitar cuenta sobre el computador local solamente. En Windows XP y Windows 2000 se puede utilizar una cuenta diferente de una de administrador pero con privilegios para acceder un “namespace”, para conectarse a un computador Windows 2000 remoto.

La seguridad es chequeada solamente cuando un usuario se conecta al CIMOM. Por esto, cualquier cambio que se haga a los permisos WMI mientras el usuario este conectado, no tendrán efecto hasta que el usuario establezca una nueva conexión. Por defecto, las aplicaciones WMI corren en el contexto de seguridad del usuario corriendo la aplicación.

- **Seguridad DCOM:** comprende la configuración para autenticación y representación (impersonation), que WMI utiliza para acceder la infraestructura WMI en computadores remotos.

  - ✓ Autenticación*:* cuando se hacen llamadas a otro proceso o a un servicio WMI remoto, WMI utiliza DCOM. Estas llamadas se hacen a través de “proxies”, los cuales requieren autenticación de las credenciales del proceso llamante. Un “proxy” es un objeto que representa un proceso remoto, como WMI o un proveedor remoto. COM utiliza “proxies” para permitir a los desarrolladores acceder datos remotos como si ellos fueran locales. Típicamente la autenticación se realiza con passwords. La Tabla 3.4 muestra los niveles de autenticación, que varían desde no hacer autenticación hasta autenticación cifrada por paquete.

Tabla 3.4 Niveles de Autenticación en WMI

Valor	Descripción
None	No utiliza ninguna autenticación. Todas las configuraciones de seguridad son ignoradas.
Default	Utiliza una negociación de seguridad estándar para seleccionar un nivel de autenticación. Esta es la configuración recomendada porque el cliente involucrado en la transacción negociará el nivel de autenticación especificado por el servidor. DCOM no selecciona el valor None durante una sesión de negociación.

Connect	Autentica las credenciales del cliente solamente cuando él intenta conectarse al servidor.
Call	Autentica las credenciales del cliente al comienzo de cada llamada, cuando el servidor recibe el requerimiento. Los paquetes de datos intercambiados entre el cliente y el servidor no son cifrados.
Pkt	Verifica que todos los paquetes de datos sean recibidos del cliente esperado. Al igual que Call, los paquetes de datos no son cifrados.
PktIntegrity	Verifica que ninguno de los paquetes de datos transferidos entre el cliente y el servidor hayan sido modificados durante la transmisión. Ninguno de los paquetes de datos es cifrado.
PktPrivacy	Cifra cada paquete de datos. Esto asegura que la comunicación entre cliente y servidor es confidencial.

La autenticación se configura cuando se realiza la conexión a un computador y a un "namespace" específico, y depende de los sistemas operativos que se estén conectando. Por esto se recomienda dejar que DCOM lo negocie.

- ✓ Representación (impersonation): DCOM proporciona un mecanismo conocido como representación, que permite especificar como quién debería actuar el servicio WMI cuando lleve a cabo una tarea. La representación ocurre cuando WMI pasa la información de seguridad del usuario que está corriendo una aplicación de gestión, a un proveedor. El proveedor puede retornar solamente la información a la cual el usuario tiene derecho de acceder. Es decir el proveedor está representando el usuario. DCOM soporta los niveles de representación mostrados en la Tabla 3.5

Tabla 3.5 Niveles de representación en WMI

Valor	Descripción
Anonymous	Esconde las credenciales del que llama. WMI no lo soporta; si una aplicación especifica este nivel de representación, WMI lo mejorará al nivel Identify. Sin embargo, con Identify las aplicaciones tienden a fallar.
Identify	Permite a los objetos requerir las credenciales del que llama, pero no actuar en su nombre. No es posible correr aplicaciones sobre computadores remotos utilizando Identify.
Impersonate	Permite que WMI actúe en nombre del usuario, es decir utilizando sus credenciales. Por lo tanto, WMI será capaz de ejecutar cualquier tarea que el usuario sea capaz de hacer según sus derechos. Es el nivel por defecto y el recomendado.
Delegate	Permite que WMI utilize las credenciales de usuario sobre un computador remoto, y permite que éste las utilize sobre otro. Este nivel de representación se debe utilizar sólo si es necesario porque tiene riesgos de seguridad. Es soportado a partir de Windows 2000.

Por defecto DCOM soporta solamente representación de un solo salto. Es decir, si una aplicación corre sobre un computador A, solo puede recuperar información del computador B. Sin embargo, con el nivel de representación Delegate, es posible que el computador A obtenga información del computador C a través del computador B. Esto es un riesgo de seguridad porque con un solo salto, el usuario está limitado a trabajar máximo con dos computadores, con Delegate, es posible acceder muchos computadores. Es decir, el nivel de impersonation Delegate es requerido por cualquier operación que involucre más de un salto de red.

- **Servicio de autenticación (Authority Setting):** permite definir el protocolo de autenticación que se utiliza en la conexión WMI. Se puede especificar autenticación NTLM, Kerberos o Negociate. Kerberos no es necesario a menos que se utilize el nivel de representación Delegate. Debido a que los sistemas difieren en los servicios de autenticación que utilizan, se recomienda que no se especifique ninguno, sino que el sistema operativo y DCOM lo seleccionen.

- **Privilegios:** para realizar ciertas operaciones como crear una nueva cuenta, apagar computadores, modificar el tiempo del sistema, entre otros, es necesario habilitar los privilegios correspondientes. De esta forma, para que una aplicación WMI realmente ejecute las operaciones requeridas, un usuario no sólo debe tener una cuenta de administrador, sino que también debe tener los derechos necesarios.

- **Seguridad del sistema operativo Windows:** además de las configuraciones de seguridad nombradas anteriormente, WMI también respeta las configuraciones de seguridad del sistema operativo estándar y nunca las sustituye o anula.

❖ **Conexiones entre diferentes sistemas operativos.** Es posible crear conexiones entre la mayoría de las versiones de los sistemas operativos Windows. Sin embargo, si un computador A quiere conectarse a un "namespace" WMI sobre el computador B, éste puede requerir niveles de seguridad más altos. Solamente las cuentas de administrador pueden establecer una conexión a un "namespace" sobre un computador remoto, el cual requiere ciertas configuraciones para los niveles de autenticación y representación para aceptar una conexión remota. Por ejemplo, si un computador A tiene Windows 2000 Server con Service Pack 2 y el computador B tiene Windows XP, el nivel de representación debe ser Packet porque Windows XP no acepta conexiones con un nivel de autenticación más bajo.

WMI tiene configuraciones por defecto para el nivel de representación, autenticación, y los servicios de autenticación. El computador A puede utilizar configuraciones que el computador B no acepta. En este caso es necesario cambiar las configuraciones cuando se realiza la conexión. Sin embargo, para el servicio de autenticación, se recomienda utilizar el método por defecto y dejar que DCOM escoja el servicio apropiado para la máquina remota. La Tabla 3.6 muestra las configuraciones por defecto de representación, autenticación y servicios de autenticación requeridos por el computador B en una conexión remota.

Tabla 3.6 Configuraciones por defecto de representación, autenticación y servicios de autenticación requeridos por el servidor

Sistema operativo del computador B	Nivel de representación	Nivel de Autenticación	Servicio de Autenticación
Windows NT 4.0 SP4 (WMI 1.01)	Identify	Connect	NTLMDomain
Windows NT 4.0 SP 4 (WMI 1.5)	Impersonate	Connect	NTLMDomain
Windows 2000 (WMI 1.5)	Impersonate	Connect	Kerberos
Windows XP Pro	Impersonate	Pkt	Kerberos
Windows Server 2003	Impersonate	Pkt	Kerberos

Conectarse a WMI sobre el computador local tiene un nivel de autenticación por defecto de PktPrivacy.

Las siguientes conexiones entre versiones de sistemas operativos no son soportadas:

- No es posible conectarse a un computador que está corriendo Windows XP Home Edition.
- Un computador Windows NT no puede conectarse a un sistema operativo superior a Windows 2000, como Windows XP o Windows Server 2003.
- No es posible acceder un computador Windows 2003 Server desde un Windows 9x.

Las siguiente conexiones entre sistemas operativos tienen requerimientos especiales:

- Para conectarse a un Windows NT 40. SP 4 con WMI versión 1.01, se debe configurar explícitamente el nivel de representación a Impersonate.
- Los computadores Windows 2000 deben tener instalado Service Pack 2 para poder conectarse a un Windows XP y sistemas operativos superiores.
- Para conectarse desde un computador Windows Server 2003 a computadores corriendo Windows 98 Windows 98, ME, y NT 4.0 SP 3 y anteriores, se deben especificar las credenciales. Si se intenta utilizar el usuario actual por defecto y no se proporcionan un nombre de usuario y un password, entonces la conexión no funciona.

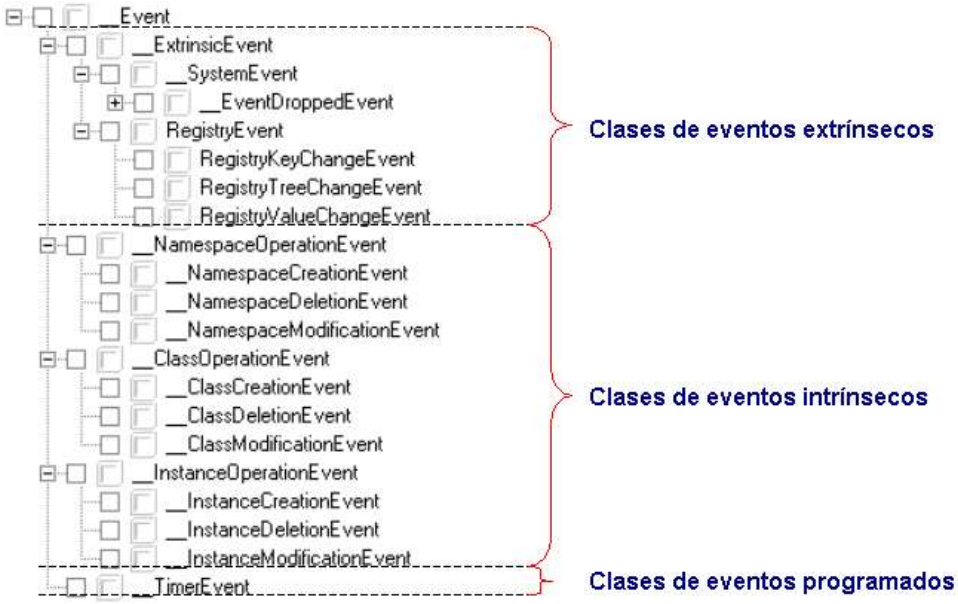
❖ **Manejo de eventos.** En la misma forma en que una clase WMI representa cada tipo de recurso del sistema que puede ser gestionado, hay una clase WMI que representa cada tipo de evento. Es decir, cada vez que ocurre un evento se crea una instancia de la clase de eventos correspondiente.

WMI soporta varios tipos de eventos: eventos programados (timer events), intrínsecos, y extrínsecos. Un evento programado es un evento que ocurre en una fecha y hora específica, o cada cierto intervalo de tiempo. Un evento intrínseco se genera cuando ocurren cambios en los datos almacenados en el repositorio. Estos cambios pueden ser en un “namespace”, clase o instancia. Sin embargo, monitorear cambios a “namespaces” o clases no es de mucha utilidad para los administradores, ya que monitorear un recurso utilizando WMI es monitorear las instancias que corresponden a ese recurso. Los eventos extrínsecos se utilizan para monitorear algo que no está representado por una clase en el repositorio. Por ejemplo, para monitorear si un proceso modifica una entrada del registro, se deben utilizar eventos extrínsecos, porque no hay clase WMI que represente una entrada de registro, entonces no se puede utilizar eventos intrínsecos. Es decir, un evento extrínseco es un evento definido por el desarrollador.

Los eventos se pueden originar de la infraestructura WMI y de los proveedores. La infraestructura WMI puede generar eventos programados o eventos intrínsecos. Un proveedor puede realizar notificaciones de eventos intrínsecos y extrínsecos. Este tipo de proveedor es llamado un proveedor de eventos.

Hay clases para cada uno de los tipos de eventos WMI: clases de eventos intrínsecos, clases de eventos extrínsecos, y clases de eventos programados. Todas ellas se derivan de la clase __Event. Las clases derivadas de __Event deben estar presentes en cada “namespace” que incluya recursos que puedan ser monitoreados a través de WMI. La Figura 3.4 muestra la jerarquía de clases derivadas de __Event:

Figura 3.4 Jerarquía de clases derivadas de la clase __Event



Un evento intrínseco es representado por una instancia de una clase derivada de __InstanceOperationEvent, __NamespaceOperationEvent, o __ClassOperationEvent. Cualquier cambio a las instancias en WMI son representadas por la clase __InstanceOperationEvent y las clases derivadas de ella, que son: __InstanceCreationEvent, __InstanceModificationEvent, y __InstanceDeletionEvent. Esto significa que monitorear recursos, es monitorear instancias de las clases derivadas de __InstanceOperationEvent. A diferencia de los eventos intrínsecos, los eventos extrínsecos deben tener un proveedor de eventos WMI asociado.

Cuando una clase no tiene un proveedor de eventos, se utilizan consultas para monitorear eventos. Con este mecanismo, el repositorio es revisado cada cierto intervalo de tiempo, para ver si se han hecho cambios. Teóricamente, algunos eventos no pueden ser notificados, dependiendo del intervalo de consulta establecido. Por ejemplo, si se crea una suscripción de eventos para chequear cada 30 segundos si Notepad ha sido iniciado, WMI comenzará, en este caso, a registrar el estado de todos los procesos corriendo en el computador. 30 segundos después, WMI realizará el mismo proceso, y comparará los nuevos resultados con los previos. Por esto, si Notepad ha sido iniciado y parado inmediatamente, este evento no será reportado. Cuando la clase tiene un proveedor de eventos, no se utiliza el mecanismo de consultas, sino que el proveedor vigila y notifica los cambios.

Una aplicación de gestión que recibe eventos desde la infraestructura WMI o un proveedor, y realiza una tarea específica de acuerdo al evento generado, se conoce como un consumidor de eventos. Ellos se registran para recibir notificaciones de los eventos en los que estén interesados especificando un filtro utilizando WQL. Él describe las condiciones bajo las cuales el consumidor quiere recibir notificación de eventos.

Para que los consumidores de eventos trabajen independientemente de los proveedores de eventos, el CIMOM actúa como un intermediario, enviando los eventos generados por los proveedores a los consumidores apropiados.

Los consumidores de eventos pueden ser temporales o permanentes. Los consumidores de eventos temporales son aplicaciones que envían requerimientos de eventos y procesan las instancias retornadas. Ellos pueden procesar eventos solamente mientras están corriendo. Un

consumidor de eventos permanente se inicia cada vez que el sistema se reinicia y corre mientras no se elimine la suscripción almacenada en el repositorio.

Para monitorear eventos en WMI se requieren tres pasos:

1. Realizar una conexión a un “namespace” sobre un computador.

2. Establecer qué eventos deben ser notificados a través de una “query” WQL,
Ejemplo:

SELECT * FROM __InstanceModificationEvent WITHIN 5 WHERE TargetInstance ISA Win32_Processor AND TargetInstance.DeviceID='CPU0' AND TargetInstance.LoadPercentage > 90

Las principales características de las “queries” de notificación de eventos son las siguientes:

- Las clases utilizadas deben ser clases de eventos.
- Cuando una clase no tiene un proveedor de eventos, se debe utilizar la palabra clave WITHIN, para indicar que se deben realizar consultas.
- TargetInstance proporciona una forma de especificar las instancias en las que se está interesado. TargetInstance es un objeto creado en respuesta a un evento que tiene las mismas propiedades y valores del objeto que produjo el evento. Además de TargetInstance, tamién existe un objeto llamado PreviousInstance. Como el nombre lo indica, este objeto mantiene las propiedades y valores del objeto antes que el evento ocurra. Esto permite reconocer qué cambio cuando un objeto fue modificado.
- La palabra clave ISA permite chequear si una instancia particular pertenece a cierta clase.

Es posible ser más específico acerca del recurso que se quiere monitorear mejorando la cláusula Where. Se puede incluir restricciones adicionales a los valores de cualquiera de las propiedades de las clases utilizando las palabras clave AND y OR.

3. Establecer la acción que se debe tomar cuando se produzca el evento.

❖ **Seguridad en eventos WMI.** WMI soporta tres tipos de operación: síncrona, semisíncrona y asíncrona, que se definen en Tabla 3.7.

Tabla 3.7 Tipos de operaciones en WMI

Llamada	Descripción
Síncrona	Bloquea la ejecución de la aplicación hasta que el proveedor retorna todos los objetos a WMI. Esto afecta el desempeño de la aplicación, por lo que las operaciones síncronas se utilizan generalmente para métodos que no retornan una gran cantidad de datos.
Asíncrona	No bloquea la ejecución de la aplicación porque tan pronto como se llame el método, la aplicación continua con la siguiente línea de código. Se debe crear un objeto “sink” que reciba los resultados de la llamada, y una subrutina especial o manejador de eventos, que se llama cuando la operación se ha completado. Es decir se crea un hilo cuyo único propósito es recibir los resultados de la operación requerida y notificar a la aplicación de esto. Esto permite que los recursos no sean monopolizados, ya que el proceso principal puede realizar otras actividades mientras espera. La mayoría de los métodos de la API Scripting que retornan un gran número de

	objetos tienen una versión asíncrona y semisíncrona.
Semisíncrona	Una llamada semisíncrona es más segura que una llamada asíncrona y no bloquea la ejecución de la aplicación hasta que se retornen todas las instancias.

Estos tres modos de operación son utilizados por los métodos, los requerimientos y los eventos. De los métodos que proporciona WMI, tres utilizan llamadas síncronas, nueve llamadas asíncronas, y los restantes utilizan llamadas semisíncronas.

La seguridad en conexiones remotas está basada en cifrar la comunicación entre el cliente y el proveedor sobre el computador remoto. El cifrado se configura a través del nivel de autenticación, como se mencionó anteriormente.

Los riegos de seguridad para las llamadas asíncronas son altos porque WMI disminuye el nivel de autenticación hasta que se pueden obtener los resultados. WMI obtiene las configuraciones de seguridad de la llamada cliente e intenta realizar la llamada de respuesta con el mismo nivel de autenticación. Si la llamada falla, WMI baja el nivel de autenticación hasta que se pueda llevar a cabo, si es necesario, a nivel de autenticación None. Esto hace que las llamadas asíncronas presenten serios riesgos de seguridad porque el objeto sink puede recibir resultados de una aplicación diferente de la que hizo el requerimiento inicial.

- **Entregando eventos en forma segura.** Un proveedor de eventos puede controlar quién recibe los eventos de tres formas:

  - ✓ Utilizar control de acceso: el proveedor determina si el consumidor tiene los privilegios para recibir un evento, sino WMI retorna acceso denegado.
  - ✓ Configurar un descriptor de seguridad para el objeto sink.
  - ✓ Configurar un descriptor de seguridad para cada evento.

- **Recibiendo eventos en forma segura.** Existen diferentes métodos para asegurar la entrega de eventos. Ellos son:

  - ✓ Cuotas de suscripción: el mecanismo de consultas puede degradar el desempeño si la cantidad de datos es demasiado grande. Además, las suscripciones de eventos tienen el potencial de llevar a cabo ataques de Negación del Servicio. Por esto, WMI establece lo que se muestra en la Tabla 3.8.

Tabla 3.8 Valores máximos para las cuotas de suscripción

Total/Por usuario	Cuota
TemporarySubscriptionsTotal TemporarySubscriptionsPerUser	10,000 1,000
PermanentSubscriptionsTotal PermanentSubscriptionsPerUser	10,000 1,000
PollingInstructionsTotal PollingInstructionsPerUser	10,000 1,000
PollingMemoryTotal PollingMemoryPerUser	10,000,000 bytes 5,000,000 bytes

Estos valores son globales, no para cada “namespace”.

- ✓ Los consumidores temporales no pueden controlar quien les proporciona eventos, es decir, no es posible configurar una suscripción temporal para controlar cuales proveedores de eventos entregan eventos a la suscripción. Por lo tanto se recomienda que se utilize comunicación semisíncrona o síncrona en lugar de asíncrona.

- ✓ WMI puede identificar cual proveedor de eventos se asocia con una clase consumidora específica, por lo tanto los consumidores permanentes pueden controlar quién les envía eventos. Esto se logra configurando el descriptor de seguridad en la propiedad EventAccess de la instancia de la clase __EventFilter, a la identidad del proveedor de eventos. WMI compara la identidad del proveedor de eventos con el descriptor de seguridad para determinar si el proveedor tienen los derechos de acceso correspondientes. Además, las aplicaciones crean instancias de la clase consumidora como parte de la suscripción. Esta clase define qué información contiene el evento, lo que el cliente puede hacer con el evento, y como es entregado el evento.

  Los consumidores permanentes tienen los siguientes requerimientos de seguridad adicionales:

  - Una suscripción permanente no trabaja cuando el consumidor, el filtro y la asociación no han sido creadas por el mismo usuario, es decir las instancias de la clase consumidor, de la clase __EventFilter, y de __FilterTo ConsumerBinding deben tener el mismo SID (definido en la propiedad CreatorSID).
  - La cuenta que crea la suscripción debe ser o una cuenta de dominio con privilegios de administrador local o una cuenta del grupo de administradores local.

Después de comparar las principales implementaciones WBEM se puede concluir que las de fuente abierta cumplen con los estándares WBEM y son más portables e interoperables. Sin embargo, la mayoría de ellas están aún en proceso de desarrollo. WMI por su parte, no cumple totalmente con los estándares WBEM y está diseñada para gestionar equipos Windows únicamente, pero es la implementación más completa y estable.

En el siguiente capítulo se hace un estudio de opciones para desarrollar este proyecto, teniendo en cuenta aspectos como la interoperabilidad, portabilidad, escalabilidad y funcionalidad de las implementaciones WBEM estudiadas.

4. ESTUDIO DE OPCIONES

En este capítulo se presentan las diferentes opciones que se estudiaron, para lograr la gestión de los equipos de usuario con sistema operativo Windows, de una manera escalable, portable e interoperable.

Como se vio en el capítulo anterior, C++ y Java son los lenguajes de programación utilizados por las diferentes implementaciones WBEM. C++ tiene algunas ventajas y desventajas con respecto a Java, las cuales se muestran a continuación.

Ventajas:

- Al compilar un programa en C++, se genera código nativo para cada máquina, por tanto C++ es más rápido que Java.
- C++ es una extensión de C, por tanto muchos programadores encuentran muy sencilla la transición a este lenguaje orientado a objetos.
- C++ permite el control de memoria y tiene una gran capacidad de programación a bajo nivel.

Desventajas:

- C++ no es multiplataforma. Para lograr que las aplicaciones se ejecuten en varios sistemas operativos, se requiere de cierto esfuerzo.
- C++ no presenta una arquitectura estándar de desarrollo orientado a Internet. Java es más que un lenguaje, es toda una plataforma que está apoyada por muchas empresas, lo que le brinda un alto grado de calidad.
- Debido a que C++ es una extensión de C, se sacrifican bastantes paradigmas de la POO, mientras que Java corrige esos problemas.
- C++ no presenta un kit de desarrollo tan rico como el de Java. Java soporta el desarrollo rápido de aplicaciones y muchas de las tareas de un programador están resueltas en su kit de desarrollo. Aunque hay muchas librerías en la red para C++, no son estándar del lenguaje y algunas cuestan.
- C++ es más complicado de aprender que Java. Java también es complicado cuando se quiere realizar un buen programa, pero obliga mucho más a seguir una metodología. C++, por ser en parte C, es demasiado libre en ocasiones.

Por lo anteriormente comentado, se puede concluir que Java es el lenguaje más adecuado para realizar este proyecto, entonces, de las implementaciones WBEM nombradas en el capítulo anterior, Pegasus y OpenWBEM fueron descartadas porque utilizan C++ como lenguaje de programación. Además, Pegasus está aún en proceso de desarrollo y OpenWBEM no es soportado por equipos Windows (un aspecto esencial para la gestión de estaciones de usuario, teniendo en cuenta que en la mayoría de empresas la mayor parte de sus equipos son Windows).

La mejor opción por utilizar Java como lenguaje de implementación y programación, por ser soportada por todos los sistemas operativos y ser la más completa, interoperable y clara en la documentación, es WBEM Services de Sun Microsystems. Aunque sólo proporciona proveedores específicos para Solaris, es posible desarrollar proveedores propios. Además, puede ser considerara como una implementación WBEM de referencia y es utilizada por otras como SNIA y OpenWBEM.

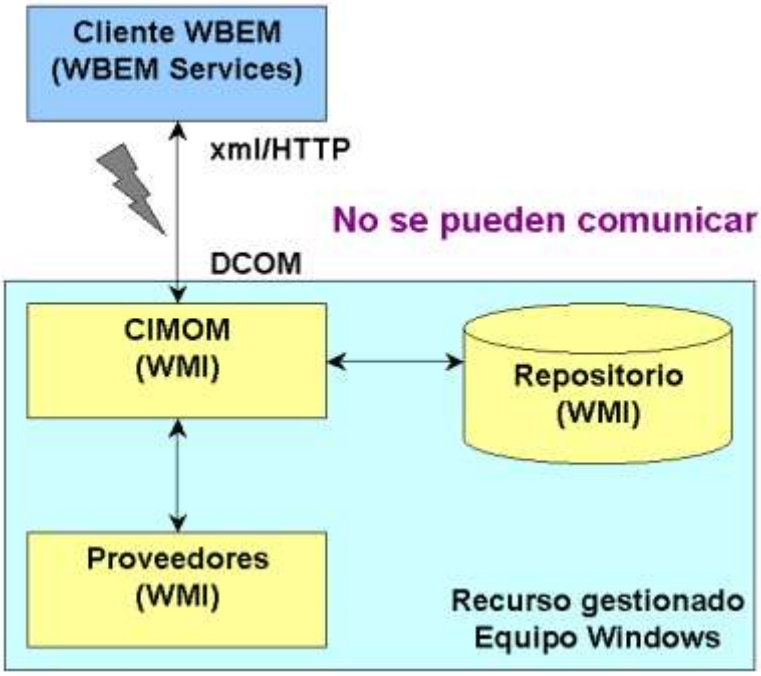
WMI viene incluido en el sistema operativo, excepto en Windows 95 y 98, para los cuales se puede descargar sin ningún costo del sitio web de Microsoft. Utiliza CIM como modelo de información, pero utiliza DCOM en lugar de operaciones CIM sobre HTTP. Sin embargo, la mayoría de las operaciones son soportadas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

Teniendo en cuenta lo anterior, se establecen las siguientes opciones:

4.1 PRIMERA OPCIÓN

Utilizar la API cliente de WBEM Services, con el fin de realizar la aplicación en Java, y el CIMOM de WMI que ya viene instalado en los equipos Windows (excepto en Windows 95 y 98), como se muestra en la Figura 4.1. Sin embargo, 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 (WBEM Services), que sí soporta estos estándares.

Figura 4.1 Primera opción



Prodexnet[7], 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 de WMI. Esta solución consiste de un "wrapper"[8] WBEM/WMI y un proveedor "proxy"[9], el cual se debe instalar solamente en el caso en el que la aplicación esté sobre un sistema no Windows. Esto demuestra que una solución de este tipo es bastante compleja.

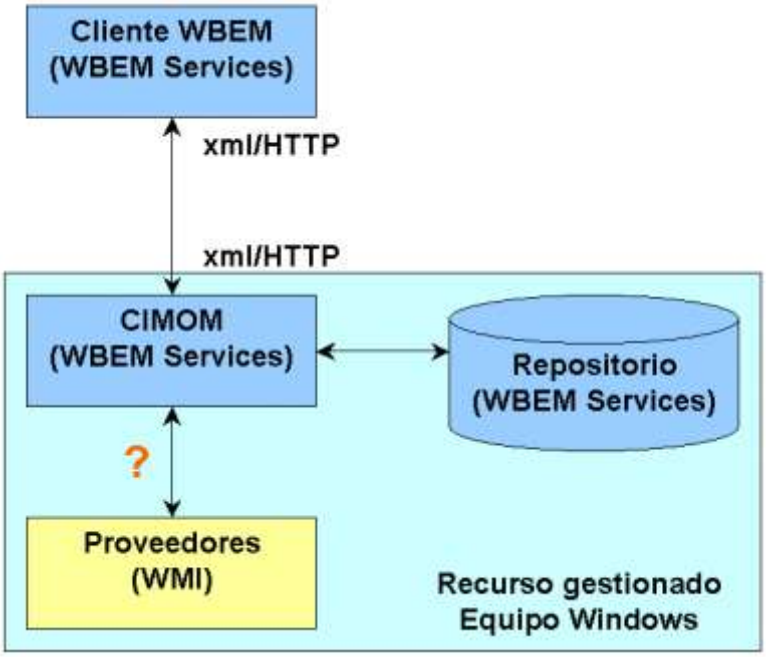
4.2 SEGUNDA OPCIÓN

Utilizar el CIMOM y la API cliente de Sun, como se muestra en la Figura 4.2, para evitar los problemas de comunicación de la opción anterior, sin embargo esto tiene varios inconvenientes. Uno de ellos es que con esta solución se debe instalar el CIMOM de Sun en cada una de las máquinas que se van a gestionar, algo que no favorece la escalabilidad al no ser tan viable sobre una red de gran tamaño. El otro inconveniente 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 estudiaron varias opciones, teniendo en cuenta que los proveedores de Windows consisten de una Librería de Enlace Dinámico, DLL (Dynamic Link Library) y de un archivo MOF. Los proveedores de

[7] Prodexnet - www.prodexnet.com.
[8] Wrapper, "proxy". Se prefiere su uso en Inglés.
[9] Proxy, representante. Se prefiere su uso en Inglés.

WMI son proveedores nativos. Un proveedor nativo es un programa escrito específicamente para un dispositivo gestionado particular.

Figura 4.2 Segunda opción



Lo primero que se debe tener en cuenta, es que un proveedor nuevo o modificado se debe registrar con el CIMOM, para comunicarle información acerca de los datos y operaciones que el proveedor soporta, y para notificarle la localización del proveedor. El CIMOM utiliza esta información para cargar e inicializar el proveedor, y para determinar el proveedor apropiado para el requerimiento de un cliente particular. Para lograr esto se debe compilar el archivo MOF del proveedor, que define las clases que soporta el proveedor. Entonces, de los proveedores de WMI se escogió el proveedor Win32, que es el proveedor básico de Windows, y se compiló su archivo MOF.

Ya que también se tiene la DLL del proveedor Win32, se pensó en utilizar la API proveedor de WBEM Services, para crear un proveedor en Java que utilizara la DLL. Para esto se buscaron diferentes interfaces nativas Java.

Tanto Netscape como Microsoft crearon sus propias interfaces nativas Java, para facilitar la interacción de Java con sus productos. Netscape creó la Interfaz de Runtime Java, JRI (Java Runtime Interface) para interfazar Java con el núcleo basado en C del Navegador de Netscape. Micrsoft creó la Interfaz Nativa Natural, RNI (Raw Native Interface) para permitir a aplicaciones Java interactuar con aplicaciones y librerías Windows nativas. Pero las soluciones de Netscape y Microsoft no funcionan si el código Java se corre en máquinas virtuales diferentes a las que cada uno tiene.

La Interfaz Nativa Java, JNI (Java Native Interface) es la interfaz de programación estándar actual de Sun. JNI permite a código Java que corre dentro de una máquina virtual Java, operar con aplicaciones y librerías escritas en otros lenguajes, tales como C, C++ y ensamblador. JNI ofrece un conjunto de funciones interfaz estándar. Utilizando estas funciones interfaz, se pueden llamar funciones JNI desde el código del método nativo para hacer cosas tales como acceder y manipular objetos Java, liberar objetos Java, crear nuevos objetos, llamar métodos Java, y así sucesivamente. Además se puede embeber una máquina virtual, JVM (Java Virtual Machine) en una aplicación nativa, cargando la JVM desde una librería nativa.

Cómo lo que se necesita es utilizar una DLL desde Java, entonces, se estudió el proceso para escribir métodos nativos para programas Java cuyos pasos se listan a continuación:

1. Se crea una clase Java que declara el método nativo.
2. Se compila la clase Java que declara el método nativo.
3. Se genera el archivo de encabezamiento para el método nativo utilizando *javah*. Una vez se ha generado el archivo de encabezamiento se tiene la declaración formal del método nativo.
4. Se escribe la implementación del método nativo en el lenguaje de programación de elección, tal como C o C++.
5. Se compilan los archivos de encabezamiento y de implementación en una librería compartida. En terminología Windows 95/NT, una librería compartida se llama una DLL.
6. Se corre el programa Java.

Si se tuviera el código de la DLL, en este caso, de la DLL de un proveedor de WMI, específicamente del proveedor Win32, en C o C++, se podría utilizar JNI como se comentó anteriormente. Desafortunadamente no se tiene este código, entonces es necesario hacer una DLL "wrapper" en C++ adicional.

El proceso es el siguiente. Primero se define el método nativo en una clase Java. Una vez se hace esto, se corre la utilidad *javah* sobre el archivo de la clase Java compilada, para crear el archivo de encabezamiento en C. A continuación se crea una DLL "wrapper" en C++ que incluye el archivo de encabezamiento generado, importa la DLL que se tiene originalmente y referencia la DLL en la implementación del método nativo, definido en el archivo de encabezamiento. En seguida se crea una segunda DLL, la cual se copia a WINNT/System32 y se registra utilizando regsvr32. En ese momento se pueden llamar las funciones de la DLL original desde Java. Para poder hacer esto es necesario conocer los métodos de la DLL, en este caso, de la DLL de un proveedor de WMI, entonces, esto no se pudo utilizar.

También se pensó utilizar, *Runtime.getRuntime().load("archivo.dll")*. Pero de esta forma no se puede llamar ninguna función de la DLL. Si la función no se declara como nativa, Java espera encontrarla en el código Java.

Entonces el SDK de Microsoft podía ser una buena opción, este SDK ofrece varias herramientas:

JavaTLB. Es un componente de Visual J++ 5.0. JavaTLB es una aplicación de línea de comandos que genera archivos de clase para un objeto COM. A diferencia del Java Type Library Wizard, JavaTLB no genera un archivo resumen con la información de la librería de tipo Java. En Visual J++ 6.0, la herramienta de línea de comandos JavaTLB es reemplazada por la herramienta JActiveX.

JactiveX. Lee una librería de tipo y genera archivos fuente Java que contienen directivas especiales de compilador. Los archivos fuente generados son parte de un paquete que se importa en los archivos fuente Java cliente COM. Visual J++6.0 proporciona esta herramienta.

JavaReg. Lee un archivo de clase Java, genera una librería de tipo correpondiente y un Identificador Único Global, GUID (Globally Unique IDentifier), y registra en el Registro de Windows la clase. Javareg se puede utilizar para registrar servidores remotos. Esta herramienta está incluida en todas las versiones de Visual J++.

J/Direct. Está incluido en Visual J++ y proporciona fácil acceso a las DLL Microsoft Windows. Al utilizar J/Direct, se pueden hacer llamadas directas a DLLs estándar del sistema Win32 y DLLs de terceras partes. J/Direct hace llamadas directas a DLLs sin requerir un "wrapper" intermedio. Además, J/Direct proporciona conversión transparente y automática de muchos tipos de datos utilizados comúnmente, eliminando la mayoría de conversión manual de tipos. Las interfaces nativas anteriores, incluyendo RNI y JNI, requieren que los desarrolladores escriban DLLs "wrapper" para el código nativo existente con el fin de enlazar el entorno Java y el código nativo subyacente.

Si las DLLs representan un componente COM/ActiveX, JActiveX crea clases “wrapper” que mapean los objetos COM a objetos Java. Si la DLL no está basada en COM, se utiliza J/Direct para llamarla directamente y para convertir los tipos de datos entre Java y el lenguaje nativo de la DLL (tal como C o C++).

El inconveniente de estas herramientas, es que solamente trabajan con la máquina virtual de Microsoft, que solo corre sobre plataformas Windows, y entonces, la aplicación no sería portable ni interoperable.

Además, en un acuerdo establecido en enero de 2001, para resolver la disputa sobre la distribución de Microsoft, de su implementación de Java, Sun y Microsoft acordaron limitar la duración del uso, por parte de Microsoft, del código fuente y los conjuntos de pruebas de compatibilidad de Sun, para soportar la Máquina Virtual Java de Microsoft, MSJVM (Microsoft Java Virtual Machine). Debido a que algunos desarrolladores y empresas han expresado interés acerca de la posibilidad de eliminar dependencias de la MSJVM en el período de tiempo originalmente proporcionado, Sun y Microsoft acordaron extender la licencia de Microsoft para la utilización del código fuente y del conjunto de pruebas de compatibilidad de Sun. Esta extensión permite a Microsoft soportar la MSJVM hasta Septiembre 30 de 2004.

En preparación para el final de la licencia de Microsoft, para la utilización del código fuente Java y los conjuntos de compatibilidad de Sun, Microsoft no ha estado incluyendo la MSJVM en sus productos desde que se estableció el acuerdo. La MSJVM no será incluida en ningún producto futuro de Microsoft. Debido a esto, aunque la MSJVM permitiría una fácil integración de Java y COM, se descartó la posibilidad de utilizarla.

Entonces se pensó utilizar un puente Java-COM, como por ejemplo xFunction, para poder llamar una DLL, específicamente la DLL del proveedor Win32, desde el programa Java, el proveedor realizado con la API proveedor de WBEM Services. Pero al utilizar el CIMOM de Sun esta solución no es óptima si se tienen que gestionar muchos equipos Windows.

4.3 TERCERA OPCIÓN

Finalmente, se estudió la posibilidad de utilizar el “framework”[10] .NET porque proporciona soporte para WMI y además porque .NET se pueden comunicar con COM.

Desde el punto de vista de un desarrollador, .NET es un entorno integral para desarrollar y desplegar aplicaciones basadas en Web y Servicios Web. Los desarrolladores generalmente crean aplicaciones .NET utilizando el Entorno de Desarrollo Integrado, IDE (integrated development environment) de Microsoft, Visual Studio.NET (vendedores tales como Borland han anunciado IDEs que compiten con Visual Studio.NET, pero no están en el mercado todavía) y los corren utilizando el “framework” .NET. El “framework” .NET será incorporado en todos los sistemas operativos Windows de Microsoft.

La versión de WMI que se lanzó con Windows 2000 expuso las interfaces COM clásicas a las aplicaciones cliente. Por lo tanto, se puede trabajar con datos y eventos de gestión WMI a través de la capa de interoperabilidad .NET COM. Pero existe un modo mucho mejor para trabajar con WMI desde .NET. La “namespace” *System.Management*, definida como parte del “framework” .NET, proporciona una API intuituva, fácil de utilizar, para trabajar con datos y eventos WMI. Muchas de los aspectos de las APIs WMI son tenidos en cuenta por System.Management, y sus definiciones de clases siguen el paradigma de diseño estándar del “framework” .NET.

Sin embargo, si se decidiera trabajar con la “namespace” *System.Management*, se debería instalar el “framework” .NET en cada uno de los equipos que se van a gestionar, que no muy deseable en una red de gran tamaño. Entonces se estudió la comunicación entre .NET y COM.
En primer lugar, es necesario entender la diferencia entre código gestionado y no gestionado. El código que es ejecutado y gestionado por el “framework” .NET se llama “código gestionado”. Este

[10] Framework, infraestructura. Se prefiere su uso en Inglés.

código debe proporcionar la información necesaria para que el Runtime de Lenguaje Común, CLR (Common Language Runtime), proporcione sus servicios, los cuales incluyen gestión de memoria, integración cross-lenguage, seguridad para acceso al código y control automático del tiempo de vida de los objetos. Todo el código basado en el Lenguaje Intermedio de Microsoft se ejecuta como código gestionado. En contraste, los desarrolladores también pueden escribir "código no gestionado", el cual es código que es ejecutado directamente por el sistema operativo, fuera del Runtime de Lenguaje Común del "framework" Microsoft .NET. El código no gestionado debe proporcionar su propia gestión de memoria, chequeo de tipo, soporte de seguridad, a diferencia del código gestionado, que recibe estos servicios desde el Runtime de Lenguaje Común. El código no gestionado se debe ejecutar fuera del "framework" .NET. Con estos conceptos se puede entender la interoperabilidad entre .NET y COM.

El entorno de código gestionado proporcionado por el CLR tiene muchas características que hacen el desarrollo de software agradable. Sin embargo, el sistema operativo en el cual el CLR se ejecuta también ofrece características propias, y no sería muy adecuado si las aplicaciones gestionadas no pudieran utilizar los servicios del sistema operativo nativo o de otro código no gestionado.

Para soportar este requerimiento, el CLR ofrece una "namespace", *System.Runtime.InteropServices*. Esta "namespace" contiene un conjunto de tipos que permiten a las aplicaciones gestionadas acceder código no gestionado. Específicamente, se soportan tres escenarios de interoperabilidad:

- Código gestionado llamando funciones de DLLs no gestionadas.
- Código gestionado instanciando y llamando métodos de interfaces de objetos COM
- Código no gestionado instanciando y llamando métodos en objetos .NET.

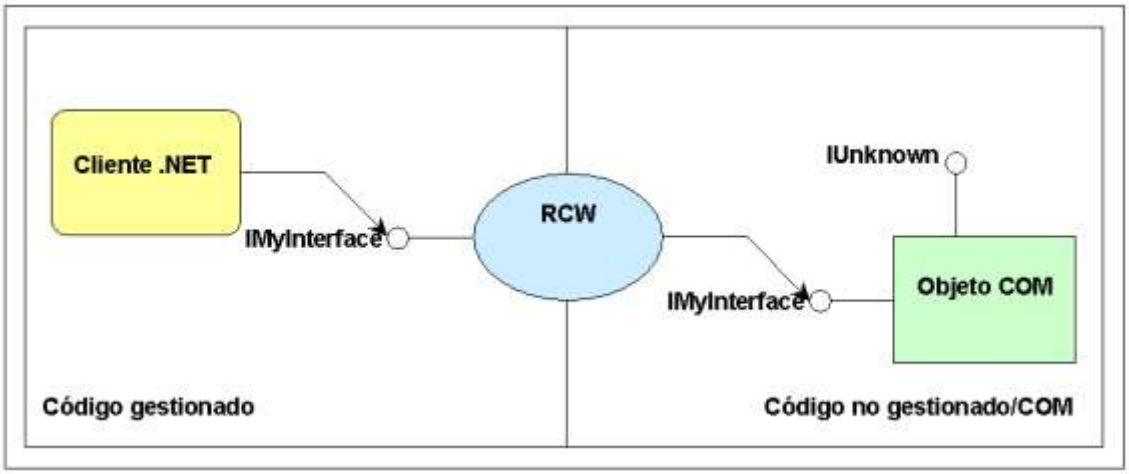
Interoperabilidad de .NET con funciones de DLLs no gestionadas. Para llamar una función de una DLL no gestionada, se debe informar al CLR, el nombre de la función que se quiere llamar, el nombre de la DLL que contiene la función y cómo convertir los parámetros de la función (por ejemplo, tipos de datos de los parámetros y cuales parámetros son parámetros de entrada, salida o entrada/salida).

Interoperabilidad de .NET con COM. Para trabajar con un objeto COM, se debe crear un "wrapper" .NET, para que los clientes .NET crean que están llamando código .NET. El CLR maneja los detalles subyacentes transparentemente. Una utilidad analiza la librería de tipo COM y produce una DLL gestionada, que describe los métodos para "wrappear"[11] la interfaz del objeto COM. Una aplicación gestionada puede entonces crear instancias de los objetos y utilizarlos como si fueran de un tipo gestionado nativo. Para cada objeto COM, el CLR genera un Wrapper Llamable por el Runtime, RCW (Runtime-Callable Wrapper). Este "wrapper" actúa como un "proxy" entre el código gestionado y no gestionado, manejando todas las tareas administrativas tales como "marshalling"[12], gestión del tiempo de vida, etc.

En otras palabras, el "framework" "wrappea" objetos COM, que son accedidos desde un entorno gestionado en lo que se llama un RCW. Desde el punto de vista de un solicitante .NET, un RCW hace que un objeto COM se vea como cualquier otro objeto basado en .NET. Un RCW, en esencia, actúa como un "proxy" al objeto COM, como se muestra en la Figura 4.3.

[11] Wrappear, hacer un "proxy". Se prefiere su uso en Inglés.
[12] Marshalling, transformación. Se prefiere su uso en Inglés.

Figura 4.3 Interoperabilidad de .NET con COM



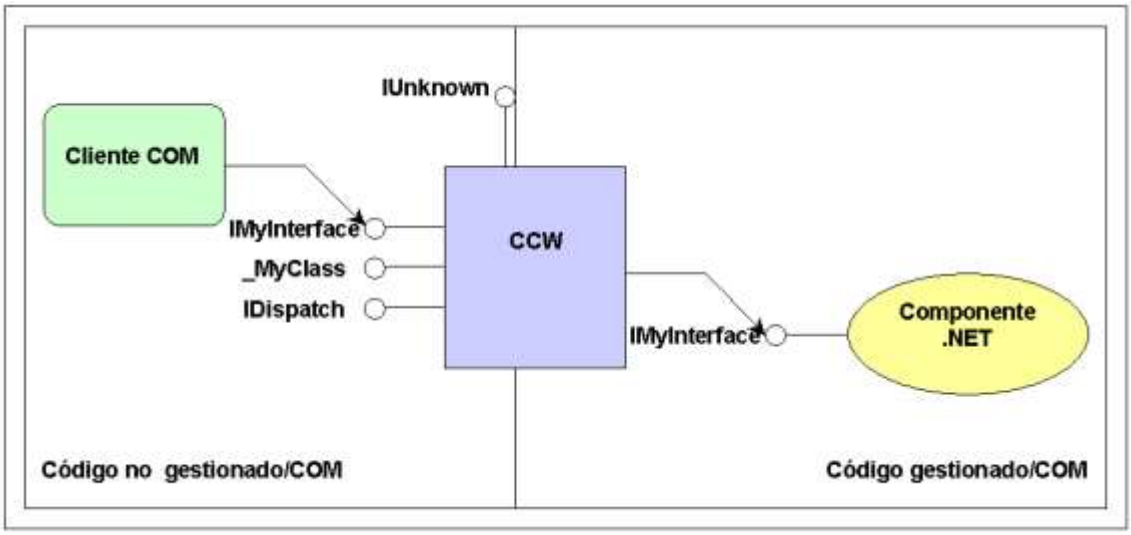
Interoperabilidad de código no gestionado con .NET. Para instanciar y utilizar un objeto .NET gestionado, desde una aplicación no gestionada, se debe adicionar la definición de la clase .NET al registro del sistema, ya que así es como el Runtime COM espera localizar objetos COM. El SDK de .NET proporciona una utilidad para registrar un "assembly"[13] (*regasm*), que genera un GUID para el objeto .NET y actualiza el registro. Un "assembly" es el bloque de construcción primario de una aplicación del "framework" .NET, es aproximadamente el equivalente de un módulo COM.

Si el código no gestionado necesita una librería de tipo, se puede generar una. Entonces se pueden crear instancias del objeto .NET desde una aplicación no gestionada y utilizarla como si fuera un objeto COM. Por cada tipo de objeto .NET gestionado, el CLR genera un Wrapper Llamable por COM, CCW (COM-Callable Wrapper). Este "wrapper" actúa como un "proxy" entre el código no gestionado y gestionado, manejando todas las tareas administrativas tales como "marshalling" y gestión del tiempo de vida.

En otras palabras, el "framework" .NET "wrappea" objetos basados en .NET en algo llamado un CCW, para hacerlos accesibles a clientes COM. El CCW expone las interfaces públicas implementadas en un objeto .NET dado, como se muestra en la Figura 4.4.

[13] Assembly, unidad básica de .NET. Se prefiere su uso en Inglés.

Figura 4.4 Interoperabilidad de código no gestionado con .NET



Entonces, para poder gestionar equipos Windows desde .NET, se tendrían que generar los RCWs de WMI y utilizarlos en la aplicación cliente, que se debería desarrollar en cualquiera de los lenguajes .NET. Con esta solución, la aplicación no sería portable ni interoperable, debido a que .NET sigue siendo muy cerrado y la comunicación se limitaría a un entorno Windows.

Hasta este punto, existía un conflicto entre el deseo de utilizar Java, que permite obtener aplicaciones escalables, portables e interoperables, y reducir enormemente los costos de desarrollo y despliegue, y la necesidad de utilizar las API de WMI, que además de ser las más adecuadas y potentes para gestionar estaciones de usuario Windows, son las únicas que se puede comunicar con el CIMOM de WMI, ya instalado en los equipos Windows de los que constan casi en su totalidad, la mayoría de las empresas en la actualidad, además el CIMOM de WMI se comunica fácilmente con los proveedores WMI. Entonces para solucionar este problema se decidió utilizar un puente Java-COM, que permitiera manejar la API WMI desde Java. Existen herramientas gratis, como Jawin, Jace, JACOB, entre otras, y otras comerciales, como R-JAX, jacoZoom, J-Integra, entre otras.

A continuación se presenta una una comparación de los aspectos más importantes de los principales puentes Java-COM, teniendo en cuenta la información disponible de cada uno de ellos. En la comparación de los puentes Java-COM se consideraron los siguientes aspectos: tipos de elementos que comunican, modo por el que es posible la comunicación, plataformas en las que corren, si son gratis o comerciales. En el Anexo A se encuentra una comparación más amplia y una descripción más detallada de los diferentes puentes Java-COM.

4.4 PUENTES JAVA-COM

4.4.1 Jawin

- Permite que programas Java utilicen objetos COM o DLLs Win32 sin tener que escribir código JNI.
- Corre sobre cualquier plataforma Windows.
- De fuente abierta, gratis.
- Comunicación:

- El generador de código lee la librería de tipo del componente COM y genera los "stubs"[14] Java necesarios.
- Utiliza JNI para lograr la interoperabilidad.
- No permite comunicaciones remotas.

❖ Otras características:

- Un "stub" transparente se genera a partir de la información de tipo y es diferente para cada objeto COM o DLL. Los "stubs" compartidos manejan declaraciones de métodos comunes al llamar métodos nativos con declaraciones JNI correlacionadas. El "stub" genérico maneja los métodos que tienen declaraciones menos comunes y por lo tanto no son "stubs" compartidos. Tanto el "stub" genérico como los "stubs" compartidos utilizan funciones de ayuda (llamadas funciones intrínsecas) para convertir tipos de datos particulares.
- Maneja excepciones e hilos.

4.4.2 Jace

❖ Permite manipular objetos Java como si fueran objetos C++ e incluir código C++ en Java.

❖ Jace ha sido construido para Windows, Solaris, HP-UX y Linux.

❖ De fuente abierta, gratis.

❖ Comunicación:

- Las herramientas Java de Jace generan clases "proxy" C++ y clases "peer"[15] C++ de las clases Java.
- La librería de "runtime"[16] de Jace utiliza JNI para lograr la interoperabilidad.
- No permite comunicaciones remotas.

❖ Otras características:

- Las clases "proxy" generadas permiten a los desarrolladores instanciar y manipular objetos Java en tiempo de ejecución, como si fueran clases C++ nativas. Las clases "peer" generadas proporcionan un método fácil para que los desarrolladores implementen métodos nativos declarados en sus clases Java, es decir implementen cualquier función nativa Java en C++.
- Realiza gestión de hilos, gestión de excepciones y conversión automática de tipos, entre otras.

4.4.3 R-JAX

❖ Permite que programas Java se comuniquen con objetos COM y DLLs localizados en servidores locales y remotos.

❖ Su fabricante es Stryon.

❖ Tiene una versión de evaluación válida por 30 días.

❖ Comunicación:

- El generador de Beans lee la librería de tipo del componente COM y genera JavaBeans.

---

[14] Stub, representante local del extremo remoto. Se prefiere su uso en Inglés.
[15] Peer, par. Se prefiere su uso en Inglés.
[16] Runtime, tiempo de ejecución. Se prefiere su uso en Inglés.

- Los Beans se utilizan en una aplicación Java que corre en cualquier plataforma.
- El servidor R-JAX se instala en el equipo Windows donde están los componentes COM y DLLs.
- Tiene tres posibles adaptadores de comunicación remota: RMI, TCP o HTTP.

❖ Otras características:

- **Bean Cliente:** punto de conexión para aplicaciones Java que necesitan conectarse a objetos COM/DLL nativos. El Bean se puede incorporar en cualquier aplicación Java.
- **Generador de Beans:** genera JavaBeans para un componente COM al leer su librería de tipo (archivos .olb, .ocx, .exe, .tlb y dll). Los JavaBeans generados son realmente "proxies", que incluyen todo el código de conectividad para lograr la comunicación con el Servidor R-JAX.
- **Adaptador de comunicación:** sirve como un conducto para RMI, TCP, o HTTP entre el programa Java y el componente COM/DLL remoto. Los desarrolladores pueden conmutar entre diferentes adaptadores de comunicación. Se proporciona una interfaz de acceso directo para los programas Java y COM que residen en la misma máquina Windows del servidor R-JAX.
- **Servidor R-JAX:** corre en el servidor Windows donde residen COM/DLLs. Es una combinación de binarios Java y nativos. El código nativo se utiliza para la comunicación con DLLs y componentes COM disponibles. La porción Java es responsable de todas las otras tareas incluyendo comunicaciones y gestión del servidor.
- **Gestor del servidor:** hace posible administrar remotamente el servidor R-JAX a través de un applet Java.

4.4.4 Java2COM

❖ Permite que las aplicaciones Java utilicen objetos COM y hace posible exponer objetos Java como si fueran objetos COM.

❖ Soporta todas las máquinas virtuales de Java basadas en Win32.

❖ Su fabricante es Neva Object.

❖ Tiene una versión de evaluación válida por 30 días.

❖ Comunicación:

- Java2COM Automation Wizard lee la librería de tipo de los componentes COM y genera "wrappers" Java.
- La aplicación Java IBuilder genera skeletons Java para los componentes COM.
- Las comunicaciones Java-COM y COM-Java utilizan una interfaz Java-a-nativo-a-Java llamada Coroutine para lograr la interoperabilidad.
- No permite comunicaciones remotas.

❖ Otras características:

- La jerarquía de clases Java2COM está modelada según la jerarquía de interfaces COM. Por lo tanto, un desarrollador que conozca los principios COM puede extender fácilmente estas clases para adicionar soporte para las interfaces COM que actualmente no están incluidas en Java2COM.
- Ya que Java2COM fue diseñado según COM, tener un conocimiento de este modelo permite entender fácilmente Java2COM.
- La librería de clases Coroutine para Java proporciona el mecanismo para que Java llame funciones externas exportadas desde librerías de enlace dinámico de Windows. Coroutine también permite construir dinámicamente "wrappers" alrededor de métodos Java, así que funciones externas pueden llamar métodos Java como si fueran funciones nativas.

4.4.5 JunC++ion

- Permite utilizar objetos Java en C++ y C++ en Java.
- Corre sobre Windows, Solaris y Linux .
- Su fabricante es Codemesh.
- Tiene una versión de evaluación con un período de prueba de 30 a 60 días.
- Comunicación:
  - El generador de código genera clases “proxy” C++ para las clases Java.
  - Las clases “proxy” C++ utilizan la librería de “runtime” que utiliza JNI para lograr la interoperabilidad.
  - No permite comunicaciones remotas.
- Otras características:
- **Java-C++.** JunC++ion permite a clases Java ser utilizadas en C++ como si fueran otras clases C++:
  - ✓ El Generador de Código de JunC++ion genera clases “proxy” C++ para clases Java.
  - ✓ Un programador utiliza las clases “proxy” generadas en un programa C++.
  - ✓ En tiempo de ejecución, la librería de “runtime” de JunC++ion traslada las llamadas a “proxy” C++ a sus llamadas Java correspondientes.
  - **C++-Java.** Con JunC++ion, los programadores pueden implementar métodos Java nativos en C++, utilizando las clases “proxy” C++ generadas, que corresponden a las clases Java, y su propio código C++.
    - ✓ Un programador escribe una clase Java con métodos nativos.
    - ✓ El Generador de Código genera clases “proxy” C++ de todas las clases Java que el programador utilizá, incluyendo aquellas con métodos nativos.
    - ✓ Un programador implementa los métodos nativos en C++, sin utilizar JNI.
    - ✓ En tiempo de ejecución, la librería de “runtime” de JunC++ion maneja todas las operaciones para transferir información entre los componentes Java y C++, así como también maneja excepciones y chequeo de errores.

4.4.6 xFunction

- Permite hacer llamadas desde Java, a funciones externas, implementadas en cualquier lenguaje y que estén en DLLs o librerías compartidas, o realizar llamadas a métodos Java desde funciones externas.
- Corre sobre Windows y Linux.
- Su fabricante es Excelsior.
- Tiene una versión de evaluación válida por 30 días.
- Tiene documentación poco clara.
- Comunicación:
  - Utiliza JNI para lograr la interoperabilidad.

- No permite comunicaciones remotas.

❖ Otras características:

- Se extienden y se instancian clases xFunction para crear objetos Java convencionales, que representan funciones externas, estructuras de datos, punteros, entre otras.
- Todas las conversiones de datos necesarias y las llamadas a funciones externas son hechas transparentemente por la librería de xFunction.

4.4.7 JCOM

❖ Permite que programas Java utilicen objetos COM.

❖ Corre sobre Windows95/98/2000 y NT.

❖ Tiene licencia LGPL (Lesser General Public License)

❖ No tiene documentación acerca del funcionamiento de la herramienta pero tiene algunos ejemplos.

4.4.8 JACOB

❖ Permite llamar componentes COM/ActiveX y librerías Win32 desde Java.

❖ Corre sobre cualquier plataforma Windows.

❖ Tiene una licencia que es una ligera modificación de la licencia BSD proporcionada por el opensource.org.

❖ Comunicación:

- Para llamar un componente COM/DLL se tiene que instanciar el componente COM. Cuando se tiene el objeto instanciado, se puede llamar a un método para invocar un método COM simple pasando los parámetros que el método necesita.
- También se pueden utilizar “wrappers”. Estos son clases Java que exponen un modelo de objetos similar al de los componentes COM. Se pueden generar “wrappers” para componentes COM utilizando Jacobgen, pero éste todavía no está completo.
- Utiliza JNI para lograr la interoperabilidad.
- No permite comunicaciones remotas.

❖ Otras características:

- La meta fue hacer a Jacob tan compatible como fuera posible, con la API de Microsoft que implementa COM en la JVM de Microsoft, así que se implementaron algunas interfaces COM utilizando JNI.
- No tiene documentación, pero ya que las clases Jacob son casi compatibles con las clases COM para Java de Microsoft se puede revisar su documentación.
- Jacobgen es la abreviatura para JACOB Generator. Es un generador de código para JACOB que genera automáticamente clases “wrapper” para componentes COM/ActiveX, y que por tanto permite a estos componentes ser accedidos como si fueran componentes Java puros.
- Jacobgen está licenciado bajo Licencia Pública GNU.
- No se pueden desplegar componentes COM visuales ya que JACOB no proporciona soporte para tener controles ActiveX gráficos dentro de componentes Java.

4.4.9 JWindows

- Permite que un programa Java acceda al entorno de programación de Microsoft Windows debido a que es básicamente un "wrapper" de la librería MFC (Microsoft Foundation Classes) escrita por Microsoft.

- JWindows está dirigido principalmente a Windows98/95 pero trabaja bien bajo Windows NT/ME/2000. JWindows no correrá en Windows CE. JWindows trabaja con cualquier compilador y máquina virtual.

- Tiene licencia GNU lesser public.

- Comunicación:

  - Utiliza JNI para lograr la interoperabilidad.
  - No permite comunicaciones remotas.

- Otras características:

  - Una librería de clases que realiza esto, ya está disponible en Microsoft. Se llama SDK de Microsoft para Java y se proporciona con MS J++ (Microsoft J++). Sin embargo esa librería requiere el uso de tanto el compilador Java de Microsoft como la máquina virtual Java de Microsoft, y por lo tanto no es para nada portable. Además, debido a un acuerdo entre Sun y Microsft, a partir de septiembre de 2004, Microsoft no está autorizado para continuar soportando su JVM.
  - No soporta controles ActiveX.
  - JWindows actualmente sólo soporta un subconjunto del conjunto de características Windows, pero suficiente para permitir la creación de las aplicaciones Windows más básicas.
  - Si hay alguna función o una clase que falte se puede adicionar fácilmente al código fuente y recompilar.
  - No se esperan actualizaciones en el futuro.

4.4.10 jacoZoom

- Permite utilizar objetos COM desde Java.

- Corre sobre cualquier plataforma Windows.

- Su fabricante es InfoZoom.

- Tiene una versión de evaluación por 30 días.

- Comunicación:

  - La utilidad de línea de comandos produce clases "wrapper" Java que corresponden directamente a objetos e interfaces COM.
  - Utiliza JNI para lograr la interoperabilidad.
  - No permite comunicaciones remotas.

- Otras características:

  - Soporta controles ActiveX.
  - Se puede utilizar para acceder al Directorio Activo.
  - Incluye un paquete que permite a los programadores Java acceder el Registro de Windows.
  - Soporta el manejo de eventos COM por clientes Java.

4.4.11 Bridge between Java and Windows

- Permite que programas Windows y programas Java se comuniquen.

- El paquete es gratis (para uso comercial y no comercial) pero el código fuente no se proporciona al público en este momento. El código que se proporciona son versiones simplificadas de los componentes clave en ese paquete. Se puede utilizar y modificar el código en cualquier forma que se desee si se reconoce en el trabajo al autor.

- Comunicación:

  - Existe un control ActiveX y una clase Java llamados XYMessenger, que tienen métodos casi idénticos y pueden enviar y recibir mensajes uno al otro.
  - El control ActiveX debe estar en un programa que utiliza MFC y corre en Windows, y la clase Java en una aplicación que corre sobre cualquier plataforma.
  - La comunicación se realiza por medio de sockets.
  - Permite comunicaciones remotas.

- Otras características:

  - La idea básica es insertar el control XYMessenger.ocx en el programa que utiliza MFC, para que pueda actuar tanto como cliente como servidor para otros programas que también estén utilizando XYMessenger. Para los programas Java solamente se necesita derivar una clase de XYMessenger para lograr el mismo objetivo. Una llamada a un método es todo lo que se necesita para lograr la comunicación. El primer programa que llama al método estará actuando como el nodo raíz en el árbol llamado árbol de comunicación. El nodo raíz necesita utilizar un puerto local para que otros programas se conecten. Otros programas se conectarán a este nodo raíz utilizando la dirección IP y el número del puerto (del nodo raíz). Estos programas serán los clientes del nodo raíz, sin embargo, cada uno de ellos también puede ser un servidor, así que más y más programas se pueden conectar.
  - El control ActiveX XYMessenger.ocx está implementado con VC ++5.0 utilizando la clase CSocket de MFC. La clase Java XYMessenger está implementada con el JDK1.2.2 utilizando las clases Socket y ServerSocket de java.net.

4.4.12 The JavaBeans Bridge for ActiveX

- Proporciona a los usuarios de contenedores OLE (Object Linking and Embedding)/COM/ActiveX legados la habilidad para incorporar y utilizar componentes JavaBeans portables en el mismo modo en que ellos previamente incorporarían y utilizarían componentes OLE/COM/ActiveX específicos a la plataforma.

- Corre sobre cualquier plataforma.

- Su fabricante es Sun Microsystems

- Gratis.

- Comunicación:

  - El Packager utiliza el archivo jar que contiene el(los) componente(s) JavaBean(s). El Packager crea una librería de tipo OLE y un registro Win32 para el componente. Esto permite a los contenedores OLE/COM/ActiveX analizar y presentar correctamente un componente JavaBean.
  - El puente JavaBeans para ActiveX es parte del producto Java Plug-in. Java Plug-in está disponible para entornos Java2, así como también para el entorno JDK1.1.

- No permite comunicaciones remotas.

❖ Otras características:

- Los beans pueden lanzar eventos que pueden ser capturados por contenedores OLE/COM/ActiveX.
- Los beans pueden actuar como servidores para la invocación de métodos de OLE/COM/ActiveX. Esto significa que programas Windows pueden invocar métodos en Beans.

4.4.13 Bridge2Java

❖ Permite a un objeto ActiveX ser tratado como un objeto Java.

❖ Corre sobre cualquier plataforma.
❖ Su fabricante es IBM AlphaWorks.

❖ Tiene una versión de evaluación válida por 90 días.

❖ Comunicación:

- Se crean "proxies" Java a partir de la librería de tipo del control ActiveX por medio de una herramienta de generación de "proxies".
- Utiliza JNI para lograr la interoperabilidad.
- No permite comunicaciones remotas.

4.4.14 JNI++

❖ Permite acceder clases C++ desde Java y viceversa.

❖ Corre sobre Windows2000/NT y Linux.

❖ Tiene licencia LGPL.

❖ Comunicación:

- Tiene dos utilidades de generación de código, la primera de estas utilidades se utiliza para generar clases "proxy" C++ dada una clase o interfaz Java. La segunda utilidad se utiliza para simplificar la creación y exposición de código nativo a Java. Su entrada es una interfaz Java, y su salida incluye una clase "proxy" Java que implementa la interfaz así como también una clase "peer" C++ y código de mapeo.
- Tiene una librería núcleo nativa o C++ que proporciona una interfaz para JVM simplificada, además de clases "wrapper" para los tipos de datos JNI.
- Utiliza JNI para lograr la comunicación.
- No permite comunicaciones remotas.

❖ Otra características:

- El código que mapea las llamadas a métodos y los accesos a campos desde las clases "proxy" C++ generadas a sus clases "peer" Java, y desde las clases "proxy" Java generadas a sus clases "peer" C++ generadas es generado. Entonces es necesario llenar la implementación para las clases "peer" C++. Las clases "proxy" C++ generadas se pueden utilizar como si estuvieran implementadas en C++, aunque las llamadas a los métodos son realmente delegadas a la clase "peer" Java correspondientes en el otro lado de JNI.

4.4.15. The Win32-Java hybrid, WJH

- ❖ Permite la comunicación entre programas Windows y programas Java.
- ❖ Corre sobre cualquier plataforma.
- ❖ Gratis, de fuente abierta.
- ❖ Comunicación:
  - No permite comunicaciones remotas.
- ❖ Otras características:
  - Una aplicación WJH está compuesta por un ejecutable Win32 y al menos un archivo de clase Java. El ejecutable Win32, entre otras cosas, carga la DLL de la JVM y crea una instancia de la máquina virtual. El archivo de la clase Java de inicio y su método main son identificados. Esta información y sus argumentos de línea de comandos asociados se pasan a la JVM. La JVM entonces carga este archivo de clase e interpreta las instrucciones localizadas en el archivo. Para ayudar a ejecutar estas instrucciones, la JVM puede llamar la funcionalidad de la API Win32 o las DLLs Win32 que soporta la JVM.
  - La idea detrás de WJH es escribir una pequeña cantidad de código en C o C++ (el lanzador de la JVM Win32) y escribir la mayor parte del código (la porción de aplicación) en Java.

4.4.16. J-Integra

- ❖ Permite que objetos Java se comuniquen con objetos COM y viceversa.
- ❖ Corre sobre cualquier plataforma.
- ❖ Su fabricante es Intrinsyc.
- ❖ Tiene una versión de evaluación válida por 60 días.
- ❖ Comunicación:
  - Ofrece herramientas tales como: com2java, java2com, regtlb, regjvm, setdllhost entre otras, que permiten lograr la comunicación entre los mundos Java y COM.
  - A nivel más bajo, J-Integra utiliza las clases de “networking”[17] de Java totalmente estándar.
  - Tiene dos modos de comunicación: modo DCOM que permite comunicaciones locales y remotas, y modo nativo, que sólo permite comunicaciones locales.
- ❖ Otras características:

[17] Networking, trabajo en red. Se prefiere su uso en Inglés.

- J-Integra incorpora una implementación Java pura del protocolo DCOM de Microsoft, lo que significa que se puede utilizar en cualquier plataforma y con cualquier JVM.
- J-Integra internamente utiliza DCOM protocolo para proporcionar acceso a componentes COM tanto locales como remotos desde un entorno Java puro. Es decir, el "runtime" Java puro de J-Integra se comunica con componentes COM utilizando DCOM que está sobre RPCs, que a su vez están sobre TCP/IP. Por lo tanto, a más bajo nivel, J-Integra utiliza las clases de "networking" de Java totalmente estándar.
- DCOM es una parte integral de Windows NT y Windows 98, y está disponible en forma gratuita para Windows 95 en el sitio web de Microsoft. Esto significa que no es necesario instalar software especial en la máquina que tiene el componente COM.
- J-Integra permite el manejo de eventos, excepciones e hilos, así como también recolección de basura en Java y COM.

4.4.17. WebLogic. Intrinsyc Software, anunció el 25 de septiembre de 2003, su tercer contrato de mantenimiento y soporte anual con BEA Systems. El contrato es una extensión de los servicios de soporte adquiridos por BEA en julio del 2001, cuando licenció el toolkit de desarrollo de J-Integra de Intrinsyc para poder incluirlo en el Servidor WebLogic de BEA.

4.5 OTROS PUENTES

4.5.1 iNET

❖ Se puede escribir una aplicación en .NET y utilizar iNET de Stryon, para convertirla a Java, entonces ella se puede ejecutar transparentemente en un entorno habilitado con Java, incluyendo sistemas operativos Unix y Linux y los principales servidores de aplicaciones tales como IBM WebSphere, BEA WebLogic, SunONE, Oracle 9i, and Jboss. Esto es posible debido a que iNET de Stryon ha recreado todo el "framework" de desarrollo .NET, en Java.

4.5.2 iHUB

❖ iHUB de Stryon, permite a objetos remotos, en "frameworks" diferentes, interactuar con otros como si se estuvieran comunicando con objetos que son locales y de su propia especie. El término objeto aquí se aplica a aplicaciones, Servicios Web, servlets, EJB's, componentes COM y MTS (Microsoft Transaction Server), ActiveX, JavaBeans, etc. El soporte para más "frameworks" y tecnologías se adicionará en un futuro.

❖ IHUB incluye los siguientes puentes:

- **Puente tecnología Java a .NET (J2N):** permite a entidades Java integrarse, incorporar y utilizar aplicaciones, objetos y servicios .NET transparentemente. Él comprende un generador de "proxies", APIs cliente (APIs remotas para la tecnología Java), canales y una máquina de comunicación.
- **Puente .NET a tecnología Java (N2J):** la contraparte de J2N que permite a entidades .NET acceder los binarios de Java como si fueran componentes .NET locales. Los objetos de tecnología Java pueden ser desde clases simples Java a JavaBeans, Servlets a incluso EJBs. El puente N2J incluye el generador de "proxies", el servidor iHUB y un wizard de despliegue.
- **Puente tecnología Java a Windows (J2W):** proporciona a los objetos Java acceso completo a binarios Windows tales como DLLs, objetos COM/DCOM/MTS o COM+ y componentes ActiveX.
- **Puente Windows a tecnología Java (W2J):** la contraparte de J2W que abre el mundo de la tecnología Java a todas las aplicaciones Windows.
- **Puente para Servicios Web:** proporciona un conversor WSDL2Java (Web Services Description Language to Java) el cual permite a los usuarios generar archivos Java a partir de sus archivos de Servicios Web .NET (WSDL). El usuario puede acceder Servicios Web .NET

en su código sin la necesidad de un conversor de tercera parte. Las librerías iHUB hacen toda la conversión requerida. Contrariamente, el generador WAR (Web ARchive) permite a los usuarios generar archivos WAR y WSDL a partir de cualquier objeto de tecnología Java incluyendo JavaBeans. Los usuarios pueden entonces publicar y consumir cualquier objeto Java como un Servicio Web.

4.5.3 JuggerNET

- JuggetNET, es la solución de integración Java-.NET de Codemesh. Con JuggerNET, Java y .NET pueden trabajar juntos al nivel más fino de integración: clase por clase, método por método, campo por campo. JuggerNET genera clases "proxy" .NET a partir de clases Java. Estas clases "proxy" se utilizan para lograr que cualquier programa .NET trabaje con Java.

- Las clases "proxy" .NET utilizan clases utilidad de la librería de "runtime" de JuggerNET, la cual a su vez utiliza código JNI para comunicarse con una JVM.

4.5.4 Ja.NET

- Ja.Net de Intrinsyc, es un puente entre el mundo de Java y el mundo de Microsoft.NET.

- Ja.NET impulsa NET Remoting, el nuevo protocolo de objetos distribuido de Microsoft .NET Remoting combina lo mejor de las tecnologías distribuidas más nuevas, Servicios Web, con lo mejor de las tecnologías distribuidas anteriores tales como DCOM, CORBA y RMI. .NET Remoting se utiliza dentro de .NET para permitir que componentes CLR (Common Language Runtime) en diferentes dominios de aplicación se comuniquen con otros. CLR soporta muchos lenguajes tales como: C#, Visual Basic.NET, ASP.NET y C++. Al utilizar Ja.NET, los componentes Java parecen ser componentes CLR, y los componentes CLR parecen ser componentes Java.

- Una de las características más interesantes de .NET Remoting es que el protocolo de transporte y el formato de datos son configurables y extensibles. Actualmente .NET y Ja.NET soportan los protocolos de transporte HTTP y TCP/IP y los formatos de datos SOAP y binario.

4.6 RESULTADO DE LA COMPARACIÓN ENTRE PUENTES JAVA-COM

Teniendo en cuenta, que la API Scripting WMI está implementada en una DLL, e incluye una librería de tipo que se encuentra en un archivo .tlb, los puentes que se podrían utilizar son: Jawin, R-JAX, Java2COM, xFuncion, Jacob, jacoZoom, Bridge between Java and Windows, y J-Integra. Con base en el estudio presentado anteriormente, el puente escogido fue J-Integra, por sus mejores características con respecto a los otros puentes.

Los otros fueron descartados principalmente por las siguientes razones:

- Jawin, debido a que no soporta comunicaciones remotas, no permitiría la gestión de equipos remotos.

- R-JAX, porque sería necesario instalar el Servidor R-JAX en cada uno de los equipos que se van a gestionar, algo muy dispendioso en una red con un gran número de equipos.

- Java2COM, porque además de no soportar comunicaciones remotas, está basado en COM que lo hace un poco complicado de utilizar.

- xFunction, ya que además de no permitir comunicaciones remotas, tiene una documentación poco clara.

- Jacob, debido a que no tiene documentación propia y no soporta comunicaciones remotas.

- JWindows, debido a que no permite comunicaciones remotas.

- jacoZoom, porque con él no se pueden comunicar objetos remotamente.

- Puente entre Java y Windows, debido a que sería necesario instalar el programa que contiene el control ActiveX XYMessenger en cada uno de los equipos Windows a gestionar, algo que no es muy óptimo en una red de gran tamaño.

Por estas razones y porque J-Integra permite comunicaciones remotas a través de DCOM, y debido a que con J-Integra no es necesario instalar ningún software especial en los equipos Windows, y tiene una documentación muy clara y muy completa, J-Integra es la mejor elección para realizar la gestión de estaciones de trabajo de usuario utilizando WMI y Java.

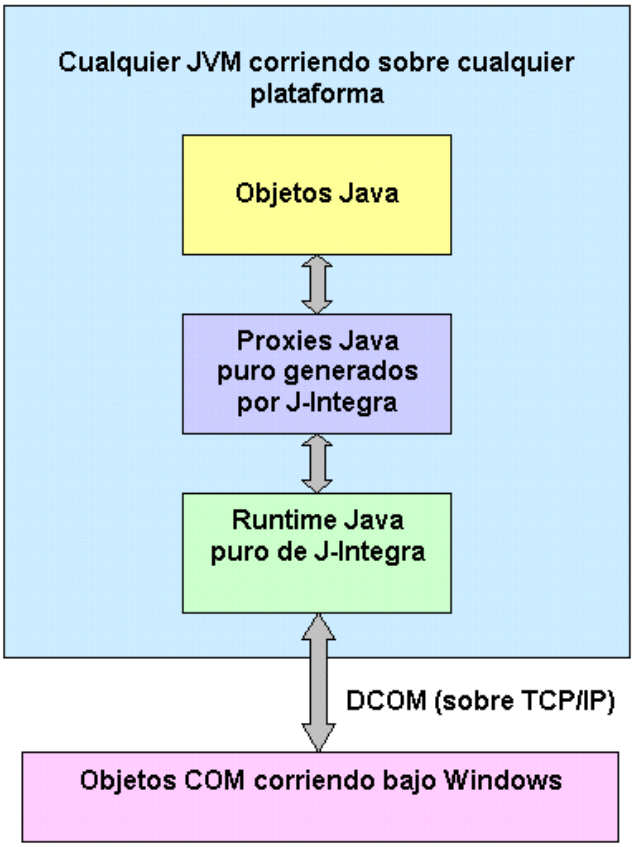
A continuación se hace una descripción de J-Integra.

4.7 J-INTEGRA

J-Integra es puente Java-COM que se puede utilizar para acceder componentes COM como si fueran objetos Java y objetos Java como si fueran componentes COM.

J-Integra incorpora una implementación Java pura del protocolo DCOM de Microsoft, lo que significa que se puede utilizar en cualquier plataforma y con cualquier JVM. J-Integra internamente utiliza este protocolo para proporcionar acceso a componentes COM tanto locales como remotos desde un entorno Java puro. Es decir, el "runtime" Java puro de J-Integra se comunica con componentes COM utilizando DCOM que está sobre RPCs, que a su vez están sobre TCP/IP. Por lo tanto, a más bajo nivel, J-Integra utiliza las clases de "networking" de Java totalmente estándar. La arquitectura de J-Integra se muestra a continuación. La arquitectura de J-Integra se muestra en Figura 4.5.

Figura 4.5 Arquitectura de J-Integra



DCOM es una parte integral de Windows NT y Windows 98, y está disponible en forma gratuita para Windows 95 en el sitio web de Microsoft. Esto significa que no es necesario instalar software especial en la máquina que tiene el componente COM.

4.7.1 Acceso

COM-Java. J-Integra soporta acceso Late Bound (IDispatch) y Early Bound (vtbl). El acceso Late Bound significa que no hay información disponible en tiempo de compilación acerca del objeto que se está accediendo, todo se evalúa dinámicamente en tiempo de ejecución. Esto significa que sólo hasta que se corre el programa se sabe si los métodos y propiedades que se están accediendo realmente existen. El acceso Early Bound, por el contrario, proporciona información acerca del objeto que se está accediendo mientras se está construyendo el programa.

El acceso Late Bound es más lento que el acceso Early Bound, y ya que no hay información disponible acerca de los objetos que se están accediendo, es más propenso a errores de usuario.
Desde una perspectiva COM, los objetos COM son instanciados ya sea utilizando una cadena legible por el usuario llamada “moniker”[18], o utilizando un identificador único legible por la máquina, llamado Class ID.

J-Integra mapea monikers y Class IDs a clases Java que corren en una JVM específica, identificada por un JVM ID. Un JVM ID es simplemente un nombre corto.

[18] Moniker, cadena que permite instanciar un objeto . Se prefiere su uso en Inglés.

1. Cuando una JVM se inicia, se registra el hecho de que está corriendo utilizando un método con el JVM ID.
2. Cuando la JVM se corre, se define una Propiedad Java que le dice al "runtime" de J-Integra por qué puerto TCP/IP se escuchan los requerimientos DCOM.
3. En cada cliente Windows que necesita acceder la JVM, se registra el nombre y la localización de la JVM utilizando la herramienta regjvmcmd o regjvm de J-Integra.

Al hacer esto, se pueden crear instancias de cualquier clase Java pública cargable por la JVM, que tenga un constructor por defecto. Se podrían entonces, invocar todos los métodos públicos y acceder todas las propiedades públicas (incluyendo los estáticos) en el nuevo objeto.

- **Early Binding (Static Access).** La herramienta java2com de J-Integra analiza clases Java, y produce:

  - Un archivo IDL (Interface Definition Language) COM, y
  - Código "marshalling" DCOM Java puro utilizado por el "runtime" de J-Integra para facilitar el acceso a objetos Java desde COM utilizando acceso vtable (Early Binding).

  Por cada interfaz Java pública que java2com descubre, crea una definición de interfaz COM correspondiente. Por cada clase Java pública que java2com descubre, crea una definición de interfaz COM correspondiente. Además si la clase Java tiene un constructor público por defecto, entonces java2com genera una clase COM.

  En seguida se compila el archivo IDL generado utilizando la herramienta MIDL de Microsoft. Ésta viene con Visual C++ y se puede descargar desde el sitio web de Microsoft. Esto producirá una librería de tipo que se debe registrar con la herramienta regtlb. Esta herramienta registra una librería de tipo en un cliente Windows COM que desea acceder objetos Java utilizando el mecanismo Early Binding de COM.

  Para registrar una librería de tipo se especifica el nombre del archivo de la librería de tipo a ser registrada y el JVM ID de la JVM en la que se encuentran las clases COM descritas en la librería de tipo.

  Si la librería de tipo se genera a partir de un archivo IDL, que a su vez es generado por la herramienta java2com de J-Integra, entonces el comando regtlb puede determinar automáticamente el nombre de la clase Java que corresponde a cada clase COM en la librería de tipo. Para desregistrar una librería de tipo se especifica el nombre del archivo de la librería de tipo. También es posible desregistrar todas las librerías de tipo registradas por regtlb. Esta herramienta no sirve para registrar librerías de tipo estándar.

- **Late Binding (Dymanic Access).** Si se utiliza el mecanismo Late Binding no se utilizan las herramientas java2com ni regtlb.

Java-COM. Por cada clase COM que la herramienta com2java encuentra en un librería de tipo, genera una clase Java que se puede utilizar para acceder la clase COM. Estas clases generadas tienen varios constructores:

- El constructor por defecto, que crea una instancia de la clase COM en el host local, sin autenticación.

- Un segundo constructor, que crea una instancia de la clase COM en un host específico, sin autenticación.

- Un tercer constructor, que crea una instancia de la clase COM en el host local, con autenticación específica.

- Un cuarto constructor, que crea una instancia de la clase COM en un host específicado, con autenticación específica.

- Un constructor final, que se puede utilizar para wrappear una referencia a un objeto retornado, la cual referencia una instancia de una clase COM. En otras palabras, se puede utilizar para acceder una referencia a una clase COM que fue retornada por otra clase COM (por una llamada a un método, o a través de una propiedad o evento). Si un método o propiedad retorna una referencia a una clase COM, entonces la herramienta com2java genera automáticamente una declaración del método, que retorna la clase Java apropiada, si la clase COM retornada está definida en la misma librería de tipo.

  Por cada interfaz definida en la librería de tipo, la herramienta com2java genera dos archivos Java: una interfaz Java y una clase Java, y por cada enumeración, genera una interfaz Java que contiene definiciones constantes de cada elemento en la enumeración.

4.7.2 Herramientas

CheckConfig. Imprime la información acerca de la plataforma y su configuración. Se puede utilizar la salida de esta herramienta para hacer chequeos de que el sistema Windows está configurado correctamente para correr DCOM.

com2java. La herramienta com2java de J-Integra lee información de una librería de tipo y genera archivos Java que se pueden utilizar para acceder las clases e interfaces COM definidas en esa librería de tipo.

La herramienta com2java procesa tres clases de constructores en una librería de tipo: enumeraciones, interfaces COM (incluyendo interfaces fuente -eventos), clases COM. Por cada enumeración, le herramienta genera una interfaz Java que contiene definiciones constantes para cada elemento en la enumeración, por cada clase COM una clase Java correspondiente, y por cada interfaz una clase Java y una interfaz Java.

java2com. La herramienta java2com de J-Integra analiza clases y obtiene:

- Un archivo IDL COM, y
- Código “marshalling” DCOM Java puro utilizando por el “runtime” de J-Integra para facilitar el acceso a objetos Java desde COM utilizando acceso vtable (Early Binding).

Herramienta de licencia. J-Integra tiene un nuevo mecanismo de licenciamiento. La licencia es un archivo XML, se llama *jintegra.xml*. Este archivo de licencia no es legible por el usuario y su manipulación puede causar que el producto deje de funcionar. Debido a que los productos se pueden instalar y actualizar, una herramienta de gestión de licencia se lanza con el “runtime”, para asistir al realizar estas tareas. Este archivo no viene con el “runtime”, sino que es enviado desde Intrinsyc.

regjvm y regjvmcmd. Estas herramientas registran la localización de una JVM en un cliente Windows que necesita crear instancias de clases Java en la JVM. La herramienta regjvm permite crear y gestionar todas las referencias a JVMs en una máquina. regjvmcmd es la versión de línea de comandos y regjvm es una versión gráfica de la misma herramienta.
Una JVM se puede acceder desde clientes COM utilizando uno de tres modos diferentes: modo DCOM, modo nativo (out of process) y modo nativo in process.

Setdllhost. Las DLLs necesitan ser cargadas en un proceso corriendo ya que no pueden correr por ellas mismas. La infraestructura DCOM en Windows tiene una facilidad para manejar esto y se llama surrogacy. Básicamente se configura el registro para indicar el nombre de un programa sustituto (surrogate) que cargará la DLL bajo demanda. Microsoft ha proporcionado tal programa sustituto que

puede tener casi todos los objetos COM que han sido implementados como DLLs, este programa se llama dllhost.exe.

Setdllhost.exe es un programa proporcionado por J-Integra para establecer que el sustituto por defecto para un servidor en proceso (DLL) especificado es dllhost.exe. La razón por la que esto es necesario es porque un cliente COM no puede comunicarse remotamente con un servidor en proceso, debe estar en un ejecutable. DLLHOST hace esto al crear varias entradas en el registro.
El uso de sustitutos para acceder DLLs no es una especificación de J-Integra, sino una especificación COM. Setdllhost simplemente sigue la especificación COM para establecer apropiadamente el sustituto por defecto. Se sugiere utilizar esta herramienta a menos que se conozca qué entradas de registro son necesarias, y cómo trabajar con el registro.

No se tiene que correr setdllhost si se corre en modo nativo. En este caso, la DLL corre en proceso y no necesita un sustituto.

4.7.3 J-Integra y WMI. WMI ofrece dos APIs que utilizan DCOM. Es posible acceder ambas APIs utilizando J-Integra. Sin embargo, solamente la Interfaz Scripting es accesible utilizando modo DCOM. Esto es, solamente la Interfaz Scripting es accesible desde una máquina no Windows.

Acceder la Interfaz Scripting WMI, a través de J-Integra, desde una máquina no Windows, requiere establecer un ejecutable sustituto (dllhost.exe) para la DLL que contiene los objetos COM WMI (wbemdisp.dll). El sustituto se puede establecer utilizando la herramienta setdllhost de J-Integra. Una vez establecido, DCOM se debe configurar para permitir acceso a este sustituto y WMI se debe configurar para permitir acceso. El sustituto debe ser establecido solamente en una máquina Windows. Ningún software relacionado con J-Integra se requiere en ninguna de las máquinas Windows durante la operación del sistema (el sustituto dllhost.exe es un componente estándar de DCOM).

No es posible acceder la interfaz No Scripting utilizando modo DCOM. Cuando se accede WMI, se necesita obtener el objeto IWbemServices. Esto se hace a través del objeto IWbemLocator. Una vez el objeto ha sido obtenido, se accede Servicios WMI en máquinas remotas a través del mecanismo DCOM normal. El objeto IWbemLocator siempre es un servidor COM en proceso (además, la interfaz IWbemLocator se declara como local en el archivo IDL que viene con el SDK WMI). Otros objetos WMI tienen la misma limitación.
Esto significa que solamente se puede acceder el objeto IWbemLocator si WMI está instalado en la máquina que tiene la aplicación cliente. Por lo tanto, para acceder la Interfaz No Scripting se necesita correr J-Integra en Modo Nativo. En este modo, J-Integra utiliza librerías nativas para instanciar los objetos WMI.

En un futuro, cuando el “framework” .NET esté instalado en la mayoría de equipos Windows, se podría utilizar Ja.NET en lugar de J-Integra, por esto a continuación se detalla esta herramienta.

4.8 Ja.NET

Ja.Net es un puente entre el mundo de Java y el mundo de Microsoft.NET. Ja.Net es un puente bi-direccional.

Ja.NET utiliza .NET Remoting, el nuevo protocolo de objetos distribuido de Microsoft. .NET Remoting combina lo mejor de las tecnologías distribuidas más recientes, Servicios Web, con lo mejor de las tecnologías distribuidas anteriores tales como DCOM, CORBA y RMI. .NET Remoting se utiliza dentro de .NET para permitir que componentes CLR (Common Language Runtime), en diferentes dominios de aplicación, se comuniquen con otros. Es importante recordar que CLR soporta muchos lenguajes, como por ejemplo C#, Visual Basic.NET, ASP.NET y C++. Al utilizar Ja.NET, los componentes Java parecen ser componentes CLR y los componentes CLR parecen ser componentes Java.

Una de las características más interesante de .NET Remoting es que el protocolo de transporte y el formato de datos son configurables y extensibles. Actualmente .NET y Ja.NET soportan los protocolos de transporte tanto HTTP como TCP/IP, y ya sea SOAP o un formato de datos binario.

Existen cuatro escenarios de despliegue:

- Java accede .NET sobre TCP/IP.
- Java accede .NET sobre HTTP.
- .NET accede Java sobre http.
- .NET accede Java sobre TCP/IP.

4.8.1 Componentes. Ja.NET tiene de cinco componentes diferentes.

El "runtime" de Ja.NET (janet.jar). El núcleo del "runtime" está almacenado en janet.jar. Es el único componente de Ja.NET que se requiere en tiempo de ejecución. Requiere un archivo de configuración (por defecto janet.xml, que se puede sobreescribir) que es generado durante el despliegue por Janetor. El archivo de configuración se requiere en tiempo de ejecución.

Janetor. Janetor se utiliza para crear un archivo de configuración para el "runtime" de Ja.NET. Este archivo:

- Tiene la localización, el tipo y el formato del canal de objetos remotos.
- Tiene el nombre del equipo, el puerto y el tipo de objetos locales.
- Sirve para instalar una licencia.
- Permite Habilitar/deshabilitar logging.
- Permite exportar a un archivo WAR (Web Applicarion aRchive), para el despliegue del "runtime" de Ja.NET en un servidor de aplicaciones basado en J2EE (Java 2 Platform Enterprise Edition).

GenJava (genjava.jar). GenJava es una herramienta de desarrollo utilizada para generar "proxies" Java a partir de clases .NET. Trabaja en colaboración con la herramienta GenService. GenJava se utiliza cuando se acceden componentes .NET desde Java.

GenNet (gennet.jar). GenNet es una herramienta de desarrollo utilizada para generar "proxies" .NET a partir de clases Java. Trabaja en colaboración con la herramienta GenService. GenNet se utiliza al acceder componentes Java desde .NET.

GenService. GenService es una herramienta de desarrollo que actúa como parte de la generación de "proxies". Es el único componente no Java de Ja.NET y es desplegable solamente en plataformas Windows. Al generar "proxies" él actúa como el puente entre las herramientas basadas en Java, GenJava y GenNet, y la plataforma .NET. GenService nunca es invocado por el usuario final directamente.

4.8.2 .NET Remoting. Microsoft.NET Remoting proporciona un "framework" que permite a objetos interactuar con otros a través de dominios de aplicación. El "framework" proporciona un número de servicios, incluyendo activación y soporte del tiempo de vida, así como también canales de comunicación responsables de transportar mensajes a y desde aplicaciones remotas. Los Formatos se utililzan para codificar y decodificar los mensajes antes de transportarlos por el canal. Las aplicaciones puede utilizar codificación binaria donde el desempeño es crítico, o codificación XML donde la interoperabilidad con otros "frameworks" remotos es esencial. Toda la codificación XML utiliza el protocolo SOAP al transportar mensajes desde un dominio de aplicación a otro. Remoting fue diseñado con seguridad.

Gestionar el tiempo de vida de objetos remotos sin el soporte del "framework" subyacente es a menudo difícil. .NET Remoting proporciona un número de modelos de activación para escoger. Estos modelos son de una de dos categorías:

- Objetos activados por cliente.
- Objetos activados por servidor.

Un objeto activado por cliente es un objeto cuya creación y destrucción son controladas por la aplicación cliente. Los objetos activados por cliente están bajo el control de un gestor de tiempo de vida basado en lease que garantiza que el objeto es colectado como basura cuando su lease expira. El tiempo de vida de un objeto activado por servidor es gestionado por el servidor remoto. En el caso de objetos activados por servidor, los desarrolladores tienen la posibilidad de seleccionar ya sea un modelo “single call” o “singleton”. Los objetos single call sólo pueden trabajar con una solicitud entrante, mientras que los objetos Singleton sirven a varios clientes. El tiempo vida de los singletons también es controlado por tiempo de vida basado en lease.

Después del análisis realizado en este capítulo, se puede concluir que WMI y J-Integra son la mejor elección, ya que permiten obtener una aplicación escalable, portable e interoperable, que son los principales objetivos de este proyecto. En el siguiente capítulo se presentan los aspectos tenidos en cuenta para la implementación, la arquitectura de la aplicación y las futuras extensiones de la misma.

5. DESARROLLO DE LA APLICACIÓN

En este capítulo se presentan los aspectos tenidos en cuenta en el diseño e implementación de la aplicación. Además se muestra su arquitectura, sus escenarios de aplicación y sus posibles extensiones.

5.1 METODOLOGÍA

La metodología para el desarrollo de una aplicación consta de un lenguaje y un proceso para modelar. El lenguaje de modelado (la notación gráfica de que se valen los métodos para expresar los diseños) que se utilizó en este proyecto fue UML (Unified Modeling Language), y el proceso (la orientación sobre los pasos a seguir para hacer el diseño) fue RUP (Rational Unified Process). En el Anexo B se encuentra el modelado de la aplicación.

5.2 REQUERIMIENTOS

Para el desarrollo de este proyecto era necesario obtener las expectativas de las Administraciones de Red con respecto a la gestión de estaciones de trabajo de usuario. Para lograr esto se tomó como base las necesidades de la Red de Datos de la Universidad del Cauca.

En las reuniones realizadas con el ingeniero encargado de la atención a los usuarios de la Red de Datos y con el jefe de la División de Sistemas de la Universidad, se detectaron los siguientes problemas:

- Desconfiguración de los equipos e instalación de software no autorizado por la Administración, debido a la falta de control de los derechos de los usuarios sobre las estaciones de trabajo.
- Falta de información sobre el software instalado y corriendo en los equipos.
- Violaciones de seguridad.
- Antivirus no actualizados.
- Mal desempeño de los computadores debido a la utilización inadecuada del espacio del disco duro o la instalación de software no adecuado a las características del equipo.
- Daños en las estaciones de trabajo por la falta de notificaciones a tiempo sobre eventos o errores.
- Dificultad para obtener las características más importantes de las estaciones de trabajo de usuario (procesador, memoria, BIOS, tarjeta de red, etc.) debido a la gran inversión de tiempo y esfuerzo para conseguir la información necesaria.
- Desconfiguración de direcciones IP.

Algunos de estos problemas no afectan directamente al usuario y no son reportados, pero van en contra de las políticas de la administración y empiezan a crecer con el tiempo, hasta que se vuelven inmanejables. Por otra parte, los problemas que afectan el trabajo del usuario generan insatisfacción y exigen una atención inmediata. Sin embargo, en algunas ocasiones esto no es posible, ya que no hay personal disponible y generalmente es necesario desplazarse hasta el sitio del problema.

5.3 DISEÑO

La aplicación se diseñó utilizando el paradigma Modelo-Vista-Controlador, MVC (ModeL-View-Controller) . En este modelo de diseño, la lógica o procesamiento se divide en tres partes:

- **Modelo (Beans):** corresponde a la lógica y a los datos.
- **Vista (Servlets o JSPs):** corresponde a la presentación.
- **Control (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.

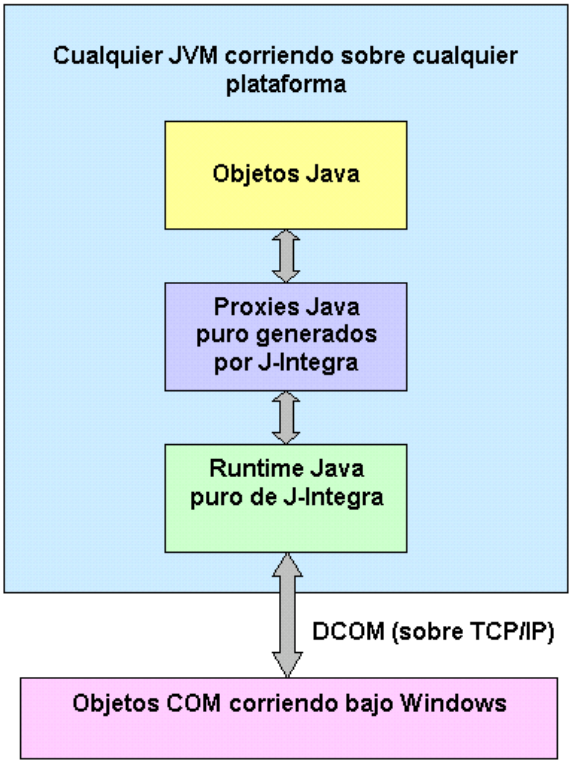
Además, el diseño de la aplicación se hizo de tal forma que cada clase CIM utilizada correspondiera a un Bean. Cada Bean tiene entonces, todas las propiedades y métodos de la clase CIM correspondiente. Además, el modelo MVC permite adicionar funcionalidad (Beans), de una forma rápida y sin mayores complicaciones, lo que permite obtener una aplicación totalmente flexible.

5.4 ARQUITECTURA

Para desarrollar esta aplicación utilizando WBEM y Java, la mejor opción es utilizar WMI y J-Integra, como se demostró anteriormente.

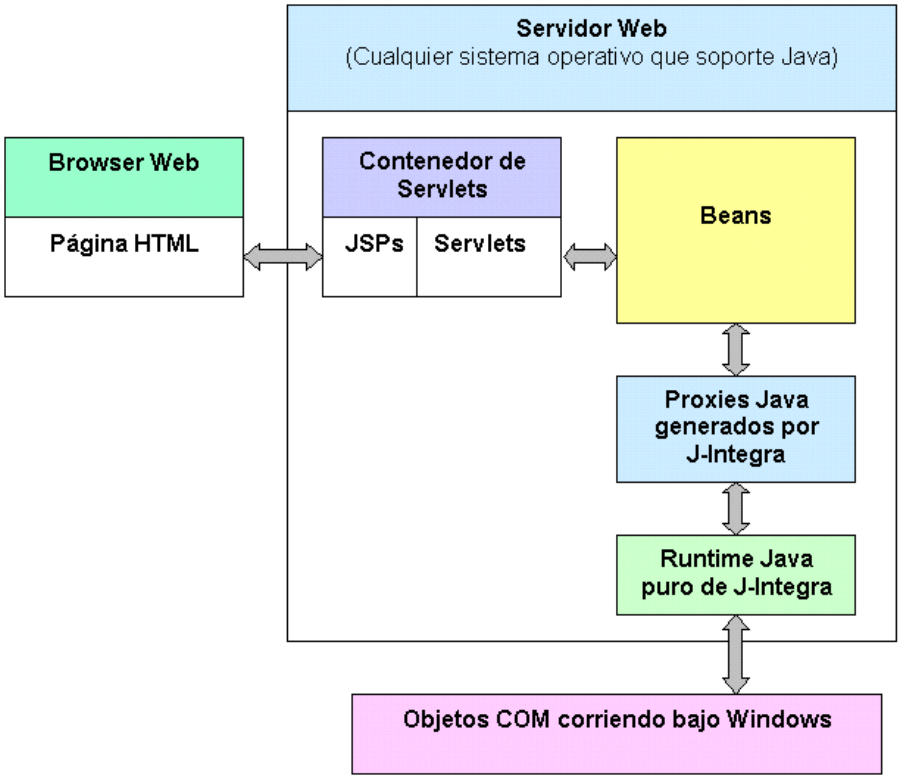
La Figura 5.1 muestra la arquitectura de J-Integra.

Figura 5.1 Arquitectura de J-Integra



Teniendo en cuenta la arquitectura de J-Integra y el paradigma MVC, la arquitectura de la aplicación es como se muestra en la Figura 5.2.

Figura 5.2 Arquitectura utilizando el paradigma MVC



5.5 IMPLEMENTACIÓN

La selección de las clases y los proveedores WMI a utilizar en la aplicación, se hizo tomando como base las necesidades de gestión nombradas anteriormente. 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 SNMP, Proveedor de Registro del Sistema y Proveedor de Registro de Eventos.

WMI ofrece dos APIs: la API Scripting y la API COM. Es posible acceder ambas APIs usando J-Integra, pero debido a que J-Integra se debe usar en modo DCOM para lograr la gestión de equipos remotos, es necesario utilizar la API Scripting, que es la única que el modo DCOM soporta.

Los proxies generados por la herramienta com2java de J-Integra, a partir de la librería de tipo de WMI, junto con el runtime Java de Jintegra (jintegra.jar), se deben configurar en el JDK seleccionado para manejar la API Scripting desde Java y utilizar las clases WMI.

Teniendo en cuenta los requerimientos de la Administración de la Red de Datos de la Universidad del Cauca y toda potencialidad de WMI y J-Integra, la aplicación permite:

- Descubrir los puertos de switches o hubs gestionables a través de SNMP, y proporcionar las direcciones MAC e IP de los dispositivos conectados a esos puertos.

- Obtener información del sistema operativo, BIOS, memoria, motherboard, puertos, teclado, tarjeta de red, procesos, servicios, archivos, cuentas, particiones, usuarios, grupos, protocolos, etc.

- Configurar direcciones IP, DNSs, puertas de enlace y proxies.

- 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 e-mail cuando ocurren eventos.

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

En el Anexo C se encuentra la guía de instalación y configuración necesaria para el correcto funcionamiento de la aplicación.

5.6 ESCENARIOS DE APLICACIÓN

Esta aplicación fue desarrollada para solucionar los problemas de gestión a nivel de usuario que se presentan en una red empresarial. Además, debido a su diseño flexible y escalable se puede adicionar funcionalidad para satisfacer las necesidades no solo a nivel de usuario, sino también a nivel de aplicaciones, bases de datos, dispositivos, red, políticas, entre otros. Actualmente la aplicación está siendo utilizada en la Red de Datos de la Universidad del Cauca.

5.7 EXTENSIONES

En la siguiente fase de esta aplicación se pueden utilizar EJBs, y se puede extender a otros usuarios, como los de dispositivos móviles, a través de Servicios Web. A continuación se describen estas posibles extensiones.

Arquitectura utilizando Servicios Web. Un Servicio Web es una interfaz que describe una colección de operaciones que pueden ser accedidas por red a través de mensajes XML estandarizados. Un Servicio Web tiene una descripción de servicio, que se hace usando WSDL (Web Services Description Language) y cubre todos los detalles necesarios para interactuar con el servicio, incluyendo formatos de mensajes (que detallan las operaciones), protocolos de transporte y localización. La interfaz oculta los detalles de implementación del servicio, permitiéndole ser usado independientemente de la plataforma hardware o software en la cual está implementado, y del lenguaje de programación en el que está escrito.

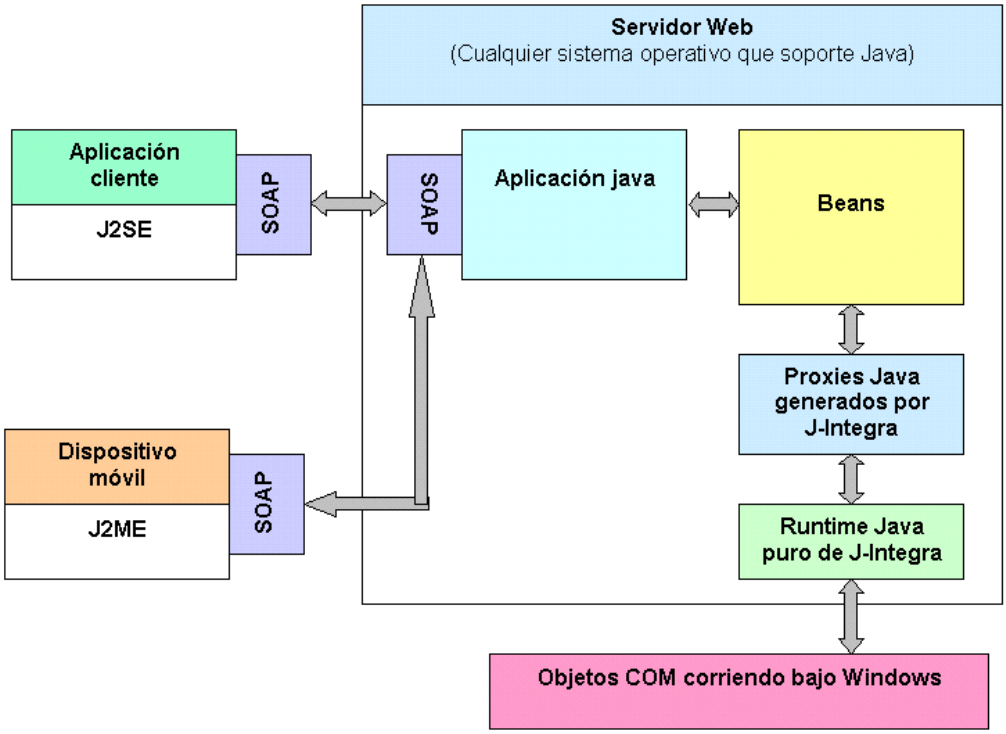
La arquitectura de Servicios Web está basada en las interacciones entre tres roles: proveedor de servicio, registro de servicio y solicitante de servicio. Las interacciones involucran las operaciones de publicar, buscar y conectar. En conjunto, estos roles y operaciones actúan sobre el módulo de software del Servicio Web y su descripción. En un escenario típico, un proveedor de servicio tiene una implementación de un Servicio Web. El proveedor de servicio define una descripción del servicio para el Servicio Web y lo publica al solicitante del servicio o al registro de servicio, también llamado nodo UDDI (Universal Description Discovery and Integration). El solicitante del servicio usa la operación de buscar para conseguir la descripción del servicio localmente, o desde el registro del servicio (nodo

UDDI), y usa la descripción del servicio para conectarse con el proveedor de servicio e invocar o interactuar con la implementación del Servicio Web.

Los mensajes XML que soportan las operaciones de publicar, encontrar y conectar, se transportan sobre SOAP (Simple Object Access Protocol), que a su vez se puede transportar sobre un protocolo como HTTP.

Para poder utilizar Servicios Web dentro de este proyecto, se puede usar un toolkit de Servicios Web cualquiera para crear un Servicio Web a partir de la aplicación Java que accede objetos COM a través de J-Integra. Además se debe desarrollar una aplicación cliente Java del Servicio Web, que use SOAP para comunicarse con el Servicio Web, como se muestra en la Figura 5.3.

Figura 5.3 Arquitectura utilizando Servicios Web



Arquitectura utilizando EJB (Enterprise Java Beans). Los EJB son parte de la especificación J2EE. Los EJBs se encuentran en contenedores de EJBs y proporcionan servicios remotos a clientes por toda la red.

Los cuatro componentes básicos de un EJB son:

- **Interfaz Home:** define los métodos necesarios para la creación-activación de un EJB, tales como create, passivate, activate, entre otros.

- **Interfaz Remote:** declara los métodos que contienen la lógica de negocios de los EJBs. Los métodos declarados en esta interfaz deben estar implementados en la clase Bean.

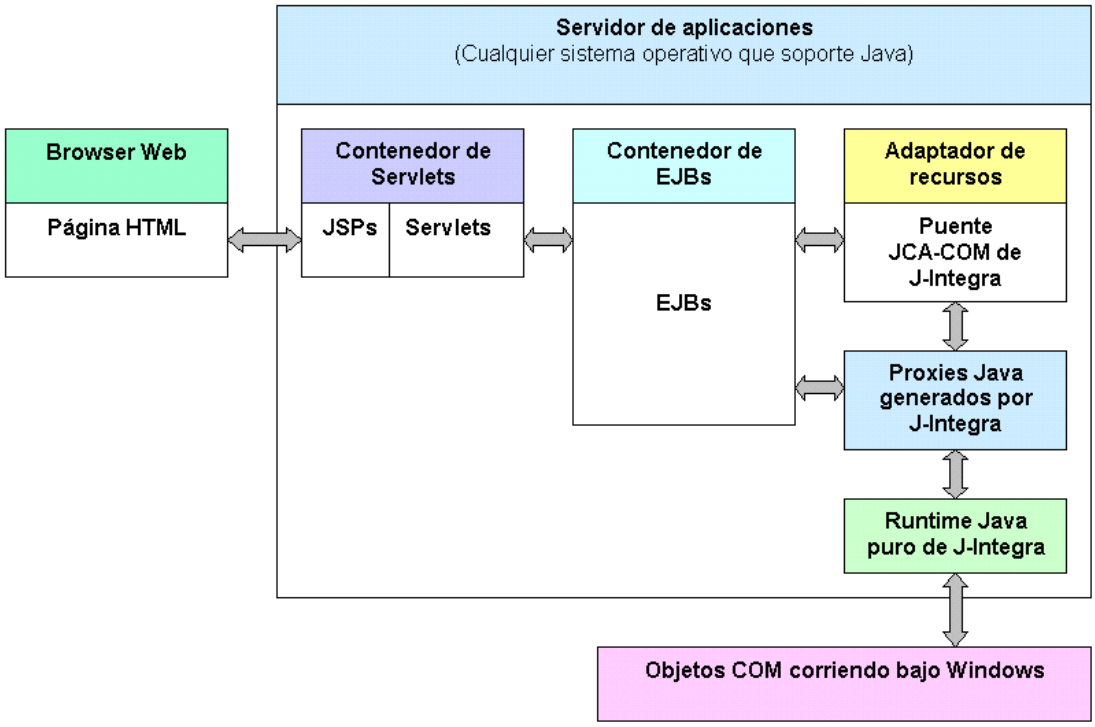
- **Clase Bean:** proporciona la lógica del Bean. Debe implementar los métodos declarados en la interfaz Remote y Home.

- **Descriptor de despliegue:** es un archivo en XML que describe un EJB al contenedor de EJBs.

Intrinsyc, el desarrollador de J-Integra, proporciona una Arquitectura de Conexión genérica a J2EE (plug in JCA-COM) que permite a EJBs acceder componentes COM existentes, a través de un adaptador de recursos simple, desplegado en los servidores de aplicaciones que cumplen con J2EE. Con el plugin JCA-COM, se pueden acceder componentes COM desde EJBs, como si estos componentes estuvieran implementados en Java.

En términos del paradigma MVC, los EJBs se pueden usar para representar la porción Modelo de una aplicación. La Figura 5.4 muestra la arquitectura de una aplicación utilizando EJBs.

5.4 Arquitectura de la aplicación utilizando EJB



Además, la plataforma J2EE proporciona las APIs XML y las herramientas necesarias para diseñar, desarrollar, probar y desplegar rápidamente clientes y Servicios Web que interoperen completamente con otros clientes y Servicios Web (que también pueden estar corriendo en plataformas que no están basadas en Java).

Por otro lado, se puede utilizar la API Cliente de Sun dentro de la aplicación, para poder gestionar equipos Solaris, Linux, entre otros. Esto es posible gracias a que esta API, a diferencia de WMI, soporta los estándares definidos por el DMTF para WBEM, y a que existen implementaciones WBEM para otros sistemas operativos, que incluyen CIMOM y proveedores, que también soportan estos estándares, como WBEM Services para Solaris, y Pegasus para Linux. Por tanto, es posible gestionar un equipo con cualquier sistema operativo, mediante una aplicación desarrollada con la API Cliente de Sun, que se comunica con cualquier CIMOM que cumpla con los estándares del DMTF.

En este capítulo se mostraron los requerimientos que se tuvieron en cuenta para el desarrollo de la aplicación, así como también los aspectos básicos que se consideraron en el diseño e implementación de la aplicación. En el siguiente capítulo se muestran las conclusiones a las que se llegaron y las recomendaciones que se pueden tener en cuenta en futuros trabajos.

6. CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

- La gran cantidad de fabricantes de hardware y software, la falta de estandarización, e incluso, la variedad de interpretaciones para un mismo estándar, han hecho surgir uno de los mayores problemas, la falta de interoperabilidad. Ésto, junto con las múltiples arquitecturas, servicios y protocolos de gestión, y el crecimiento de la capacidad, complejidad y heterogeneidad de las redes, han hecho de la gestión algo difícil de realizar. Esto hace evidente que Java, una plataforma que permite desarrollar aplicaciones portables, escalables e interoperables, y WBEM, un estándar de gestión, que se basa en el modelo de información CIM y estándares de Internet, sean vistos actualmente como las tecnologías con mayores posibilidades de convertirse en un futuro en las más utilizadas para la gestión de redes.

- La mayor ventaja de WBEM con respecto a los estándares de gestión existentes es el modelo de información CIM. La especificación CIM proporciona los mecanismos de conversión a otros estándares de gestión, y el esquema CIM describe cada parte de un sistema gestionado utilizando un solo modelo de información. Esto permite representar datos de diversas fuentes en un formato común, y por lo tanto gestionar cualquier recurso independientemente de su instrumentación, logrando así la integración entre infraestructuras de gestión y la reducción de los costos que implican la gestión de redes grandes y heterogéneas.

- Un punto en contra que tiene WBEM, que no se puede considerar una desventaja porque no se debe a la infraestructura por si misma, es que sus implementaciones no soportan completamente todos los aspectos del estándar WBEM, e incluso utilizan diferentes mecanismos para la codificación y el transporte. Esto trae como consecuencia la falta de interoperabilidad entre implementaciones WBEM. Sin embargo, es importante tener en cuenta que las implementaciones son relativamente nuevas y algunas de ellas aún están en proceso de desarrollo.

- Java permite el desarrollo de aplicaciones potentes en una forma sencilla, caracterizadas por su portabilidad y escalabilidad, dos aspectos muy importantes en el desarrollo de software, que no se pueden conseguir con ningún otro lenguaje de programación. Por su parte, el sistema operativo Windows y todas las aplicaciones desarrolladas por Microsoft, además de ser las más difundidas, son muy potentes y funcionales, sin embargo, solo pueden ser utilizadas en un entorno Windows. Para poder superar los problemas de comunicación entre componentes COM de Microsoft y objetos Java, existen puentes Java-COM que permiten integrar estos dos entornos y sacar el mayor provecho de cada uno de ellos.

- Los estándares juegan un papel fundamental en la gestión. Sin ellos no sería posible la interoperabilidad de entornos heterogéneos. Sin embargo, crear estándares no es suficiente, su adopción es lo que finalmente determina su éxito y continuidad. Estas soluciones sólo serán utilizadas si hay implementaciones soportándolas, que permitan a los usuarios adoptarlas fácil y rápidamente. Por esto, los estándares en cualquier área, e independiente de su complejidad, deben ser lo suficientemente claros y concretos, para que exista una interpretación única de ellos y puedan ser implementados en una forma fácil y uniforme. Esto, además de permitir que sean adoptados rápidamente, garantiza interoperabilidad entre plataformas de diferentes vendedores.

- Cualquier infraestructura de gestión debe ser portable, interoperable, eficiente, robusta, flexible y escalable. Sin embargo, aunque una infraestructura de gestión posea todas estas características, su éxito depende en gran medida de su capacidad para integrar infraestructuras existentes, ya que los usuarios no están dispuestos a perder la inversión de dinero y conocimiento en los sistemas de gestión ya adquiridos en sus empresas.

- El ancho de banda y el número de dispositivos de red están incrementando constantemente, diferentes redes y sistemas están trabajando juntos, y el número de servicios ofrecidos a los

usuarios es cada vez mayor. Debido a esto, el modelo centralizado no es lo suficientemente bueno para redes grandes y complejas. Por esto, es necesario descentralizar la gestión a través de gestores de medio nivel, para crear una arquitectura distribuida donde los gestores del nivel más alto delegan tareas a los de medio nivel, lo cual significa menos consumo de ancho de banda, y distribución de almacenamiento y utilización de recursos.

- Actualmente existe una tendencia a que las aplicaciones estén basadas en Web, debido, entre otros aspectos a su accesibilidad, conectividad, capacidad de comunicación y fácil mantenimiento. Además, las aplicaciones Web son más sencillas en cuanto a su diseño e implementación.

- No existe una infraestructura de gestión que pueda ser considerada como la mejor. Cada una tiene fortalezas, debilidades y campo de aplicación. Por esto, es necesario estudiarlas y compararlas para utilizar en cada caso la más adecuada, como es el caso de WBEM para entornos empresariales.

- Lograr la interoperabilidad entre infraestructuras de gestión involucra conversiones entre modelos de información y comunicación. Un modelo de información se puede mapear directamente (cada objeto en un modelo es mapeado a uno o más objetos en el otro modelo), o en una forma abstracta (el contenido y las relaciones en un modelo son mapeadas a otro). Un modelo de comunicación se puede mapear a otro a través de una traducción del protocolo (mapeando PDUs de un protocolo a otro), o a través de una emulación del servicio (mapeando servicios ofrecidos por un protocolo a otro).

- La tendencia general hacia la cual evolucionan las infraestructuras de gestión es:

  - Infraestructuras basadas en servicios que proporcionan una alta flexibilidad al permitir adicionar y remover servicios de acuerdo a las necesidades y capacidades del dispositivo.

  - Infraestructuras constituidas por componentes altamente especializados, que cooperan entre si, ofreciendo y utilizando servicios de otros componentes en forma independiente de la localización, sistema operativo o lenguaje de implementación.

  - Modelos de Información claros, amplios y extensibles, gracias a la utilización del modelo orientado a objetos.

  - Utilización de esquemas de comunicación ya existentes, basados en Internet, no propietarios, sino estandarizados y de amplia utilización, con el fin de obtener accesibilidad, interoperabilidad y amplia adopción.

  - Infraestructuras que proporcionen los medios para integrar infraestructuras existentes en una forma transparente.

  - APIs que permitan el desarrollo rápido de aplicaciones y que oculten al desarrollador los aspectos relacionados con la comunicación.

6.2 RECOMENDACIONES

- Extender la aplicación para gestionar otros sistemas operativos utilizando la API Cliente de Sun.
- Adicionar funcionalidad a la aplicación a través del uso de otros proveedores de WMI.
- Explorar a nivel práctico las múltiples capacidades de comunicación que brindan los puentes Java-COM.
- Implementar la siguiente fase de la aplicación utilizando la plataforma de desarrollo J2EE.
- Extender los servicios que brinda la aplicación, a los usuarios de dispositivos móviles, utilizando Servicios Web.

REFERENCIAS

HALL, Marty. Core Servlets and JavaServer Pages. Prentice Hall. 617p.

WILCOX, Mark, FERRIS, Matthew, AYERS, Danny y BOGOVICH Mike. Professional Java Data. Wrox. 2001. 1275 p

Frank Scheffler (Septiembre de 2003), Extending the Common Manageability Programming Interface for remote operations

http://www-124.ibm.com/sblim/index.html

http://www-124.ibm.com/sblim/npidoc.html

http://www.wbemsource.org/

http://wbemservices.sourceforge.net/

http://www.openpegasus.org

http://openwbem.sourceforge.net/

http://www.opengroup.org/snia-cimom

http://www.snia.org/smi/home

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/wmi_start_page.asp

http://www.prodexnet.com/nsm/wbem_products.htm

http://staff.develop.com/halloway/JavaWin32.html

http://www.onjava.com/lpt/a/1026

http://cern.ch/service-spi/external/JACE/1.1b3/rh73_gcc32/jace/release/docs/index.html

http://www.stryon.com/products.asp?s=2

http://www.nevaobject.com/_docs/_java2com/java2com.htm

http://www.codemesh.com/en/JunctionCurrentRelease.html

http://www.excelsior-usa.com/xfunction.html

http://sourceforge.net/projects/jcom/

http://danadler.com/jacob/

http://www.rolemaker.dk/JWindows/index.htm

http://www.infozoom.de/IE/jacozoom.html

http://www.codeproject.com/threads/bridgewinjava.asp

http://java.sun.com/products/javabeans/software/

http://www.alphaworks.ibm.com/tech/bridge2java

http://j-integra.intrinsyc.com/j-integra/doc/

http://j-integra.intrinsyc.com/pressreleases/8_06_2002.asp

http://www.stryon.com/products.asp?s=3

http://www.codemesh.com/en/JuggerNETCurrentRelease.html

http://www.intrinsyc.com/support/ja.net/doc/

http://www-3.ibm.com/software/solutions/webservices/pdf/WSCA.pdf

http://j2ee.masslight.com/Chapter6.html

http://www.faqs.org/rfcs/rfc2617.html

http://www.rsasecurity.com/rsalabs/faq/5-1-2.html