ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

VICERRECTORADO DE INVESTIGACIÓN Y VINCULACIÓN CON LA COLECTIVIDAD

MAESTRÍA EN INGENIERÍA DE SOFTWARE
PROMOCIÓN 2009 – 2010

TESIS DE GRADO MAESTRÍA EN INGENIERÍA DE SOFTWARE

TEMA: "SISTEMA DE CONTROL Y SEGURIDADES PARA EL PROYECTO AVES (APLICACIÓN DE TECNOLOGÍAS DE VIRTUALIZACIÓN PARA LA ESPE) EMPLEANDO LA METODOLOGÍA AUP"

AUTORES: Meneses Becerra, Fausto Honorato
Guerra Cruz, Luis Alberto

DIRECTOR: Ing. Fuertes Díaz, Walter Marcelo, Ph.D.

Latacunga, Marzo 2015

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

MAESTRÍA EN INGENIERÍA DE SOFTWARE

CERTIFICADO

ING. WALTER M. FUERTES Ph.D

CERTIFICA

En mi calidad de tutor del trabajo de grado, titulado: SISTEMA DE CONTROL Y SEGURIDADES PARA EL PROYECTO AVES (APLICACIÓN DE TECNOLOGÍAS DE VIRTUALIZACIÓN PARA LA ESPE) EMPLEANDO LA METODOLOGÍA AUP, presentado por el ING. FAUSTO HONORATO MENESES BECERRA y el ING. LUIS ALBERTO GUERRA CRUZ, requisito previo para la obtención del título de MAGISTER en Ingenería del Software, doy fé de que dicho trabajo reúne los requisitos y los méritos suficientes para ser sometido a presentación pública y evaluación por parte del jurado examinador que se designe.

En la ciudad de Latacunga, a los 03 días del mes de marzo de 2015.

_____

Ing. Walter M. Fuertes Ph.D

Director

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

MAESTRÍA EN INGENIERÍA DE SOFTWARE

DECLARACIÓN DE RESPONSABILIDAD

ING. FAUSTO HONORATO MENESES BECERRA
ING. LUIS ALBERTO GUERRA CRUZ

DECLARO QUE

El contenido e información que se encuentra en esta Tesis denominada "SISTEMA DE CONTROL Y SEGURIDADES PARA EL PROYECTO AVES (APLICACIÓN DE TECNOLOGÍAS DE VIRTUALIZACIÓN PARA LA ESPE) EMPLEANDO LA METODOLOGÍA AUP", es responsabilidad exclusiva de los autores y han respetado derechos intelectuales de terceros, conforme a las fuentes que se incorporan en las Referencias Bibliográficas.

En la ciudad de Latacunga, a los 03 días del mes de marzo de 2015.

_____
Ing. Fausto Honorato Meneses Becerra
C.C. 170336445-3

_____
Ing. Luis Alberto Guerra Cruz
C.C. 1705547527

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

MAESTRÍA EN INGENIERÍA DE SOFTWARE

AUTORIZACIÓN

ING. FAUSTO HONORATO MENESES BECERRA
ING. LUIS ALBERTO GUERRA CRUZ

Autorizo a la Universidad de las Fuerzas Armadas ESPE, la publicación, en la biblioteca virtual de la Institución del trabajo de grado denominado “SISTEMA DE CONTROL Y SEGURIDADES PARA EL PROYECTO AVES (APLICACIÓN DE TECNOLOGÍAS DE VIRTUALIZACIÓN PARA LA ESPE) EMPLEANDO LA METODOLOGÍA AUP”, cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

En la ciudad de Latacunga, a los 03 días del mes de marzo de 2015.

_____
Ing. Fausto Honorato Meneses Becerra
C.C. 170336445-3

_____
Ing. Luis Alberto Guerra Cruz
C.C. 1705547527

DEDICATORIA

Esta tesis está dedicada a.

A nuestras familias, a nuestros padres a quienes amamos tanto, a nuestros queridos hermanos y por siempre a Dios; quien, a través de ellos ha sabido sembrar en nosotros el amor al estudio, al trabajo, a la comprensión y la perseverancia para seguir adelante, así como el deseo de superación; a quienes les debemos nuestro ser y los amamos por siempre.

Luis Alberto

&

Fausto Honorato

AGRADECIMIENTO

Agradecemos al señor ingeniero Walter Fuertes Ph.D, director de la presente Tesis, por su guía en la estructura del tema, por todo su empeño, tiempo, dedicación al avance y corrección de esta tesis para poder culminar con esta meta; un agradecimiento especial por su decidida colaboración al Doctor J. E. López de Vergara por su trabajo y esfuerzo por conseguir y enviar información concerniente al tema de investigación; además, nuestro sincero agradecimiento al señor Ing. Hernán Aules por su perseverante colaboración en el diseño de los gráficos estadísticos. Los autores agradecemos al Ing. Crnl. Fidel Castro Director del Dpto. de Ciencias de la Computación, al Ing. Ramiro Delgado Coordinador de Investigación y al Grupo de Investigación de Sistemas Distribuidos del mencionado Departamento de la Escuela Politécnica del Ejército del Ecuador, por la provisión del laboratorio dónde se realizó la experimentación de esta investigación, y a todas las personas que han colaborado de una u otra forma y que han creído en nosotros. Entregamos nuestro aporte para el bien de la ciencia, la tecnología y la sociedad. Gracias.

ÍNDICE GENERAL

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN

Las tecnologías de Virtualización procuran aprovechar al máximo los recursos físicos disponibles en las empresas, convirtiéndose en una estrategia de compartición de hardware. Sin embargo, a la hora de configurar y desplegar un entorno de red virtual (VNE), se presentan problemas de interoperabilidad y escalabilidad. De interoperabilidad, ya que existen en la industria una diversidad de plataformas de virtualización, que provocan que un VNE no pueda ser abstraído en base a las características de la tecnología de virtualización subyacente. De escalabilidad, debido a que los VNE son desplegados en un solo equipo físico, provocando dificultad en el despliegue de VNE complejos. Ante este escenario, la presente investigación se enfoca en el despliegue automático de un VNE en un ambiente distribuido, independiente de la plataforma de virtualización. Para llevarlo a cabo, se han utilizado técnicas de modelado fundamentadas en el Modelo de Información Común (CIM) del Grupo de Trabajo para Gestión Distribuida (DMTF). Para evaluar la viabilidad del modelo, se ha construido un cliente WBem desarrollando una API, el cual permite desplegar VNE automáticamente. Además, este API, facilita la distribución física de un VNE en diferentes servidores con diferentes plataformas, simulando un planificador de balanceo de carga, sin perder el control de ninguna máquina virtual ni del escenario al que pertenecen. Finalmente, para la transferencia de archivos y la administración remota del VNE, se ha incorporado a la API los servicios de Jcraft (API WBem & Jcraft), lo cual facilita la administración del VNE.

Palabras claves:

TECNOLOGÍAS DE VIRTUALIZACIÓN

ENTORNO DE RED VIRTUAL

GRUPO DE TRABAJO PARA GESTIÓN DISTRIBUÍDA

MODELO DE INFORMACIÓN COMÚN

SERVICIOS DE JCRAFT

ABSTRACT

Virtualization technologies seek to maximize the available physical resources in enterprises, becoming a strategy of sharing hardware. However, when configuring and deploying a virtual network environment (VNE), there are problems of interoperability and scalability. Interoperability, as there are in the industry a variety of virtualization platforms, which cause an VNE can not be abstracted based on the characteristics of the underlying virtualization technology. Scalability, because VNE are usually deployed in a single physical machine, causing difficulty in the deployment of VNE complex. Given this scenario, this research focuses on the automatic deployment of VNE in a distributed environment, independent of the virtualization platform. To accomplish this, techniques have been used to model based on the Common Information Model (CIM) Working Group for Distributed Management (DMTF). To evaluate the feasibility of the model has been constructed by developing a WBEM client API, which allows VNE deploy automatically. In addition, this API facilitates the physical distribution of an VNE on different servers with different platforms, simulating a load balancing scheduler, without losing control of any virtual machine or the stage where they belong. Finally, for file transfer and remote management of VNE, has been incorporated into the services Jcraft API (API Wbem & Jcraft), which facilitates the administration of VNE. The results show the solution of the problems mentioned above.

Keywords:

VIRTUALIZATION TECHNOLOGIES

VIRTUAL NETWORK ENVIRONMENT

WORKING GROUP FOR DISTRIBUTED MANAGEMENT

COMMON INFORMATION MODEL

SERVICES JCRAFT

INTRODUCCIÓN

Teniendo en cuenta las posibilidades que ofrece la Virtualización en la administración de la Tecnología de la Información en cuanto a aislamiento, seguridad, portabilidad, optimización de recursos y flexibilización a la inversión, tanto actual como futura se puede asegurar su aporte a la investigación y con ello al desarrollo e innovación de la industria.

Al generar varias máquinas virtuales en una misma máquina física *se produce un problema* de saturación de hardware, por lo que se los debe distribuir en varios VNEs con el fin de que se ejecuten los procesos en varias máquinas virtuales. Ante este escenario, el presente trabajo se enfoca en aportar con soluciones aplicando criterios de computación distribuida para VNEs.

Los principios de computación distribuída para VNEs; se sustentan en el *Modelo Genérico para la Gestión de Entornos Virtuales de Red*, el cual tiene como soporte el Modelo de Información Común *(CIM)* del Grupo de Trabajo para Gestión Distribuida *(DMTF).*

Con el propósito de administrar los VNEs, de manera autónoma, se necesitó que el repositorio CIMOM trabaje centralizadamente en un servidor; y las plataformas de virtualización en otros e interconectados. Además, con el fin de estandarizar nombres y de conseguir una independencia entre la aplicación API-WBem-Jcraft y el repositorio CIMOM, se debe establecer en este último una parametrización.

La administración de los objetos CIM se lo hace a través del repositorio CIMOM el cual esta implementado en un servidor remoto; un segundo servidor administra la plataforma de virtualización XEN, en un tercer servidor administra conjuntamente CIMON y XEN, y finalmente un cuarto servidor administra la plataforma VMWare, gestionando VMs de un VNE.

Para evaluar la viabilidad de este modelo y la obtención de los resultados, se ha construido un cliente WBem el cual permite modelar sistemas, e implementar sus atributos y métodos; es decir, se ha implementado una interfaz de aplicación *(API)* que permite al software desplegar VNEs automáticamente, independientemente de la plataforma de virtualización.

Para alcanzar los propósitos planteados el presente trabajo se ha organizado de la siguiente manera:

El primer capítulo explica, el estado del arte con respecto a la virtualización; seguridades en entornos virtualizados en cuanto a riesgos y problemas. Se establece una arquitectura abstracta en la que se basa el Modelo de Información; con las herramientas de virtualizaciòn Xen, VMware y CIMOM se evaluará el performance de las MVs que conforman los VNE, así como su balanceo de carga; para el desarrollo de la aplicación se referencia la Metodología AUP.

El segundo capítulo determina la Infraestructura de virtualización.

El tercer capítulo determina la plataforma que soporta herramientas de virtualización.

El cuarto capítulo enfoca el diseño y la implementación; la Especificación de requerimientos del software se establece de acuerdo a la norma IEEE 830 (ERS).

En el quinto capítulo se implementa un mecanismo de control y seguridades para facilitar el acceso de los usuarios al proyecto AVES, manteniendo un registro de pistas de auditoría garantizando de esta manera que no se desconfiguren los laboratorios de redes.

El sexto capítulo indica las conclusiones y recomendaciones a los que se llega al finalizar este trabajo, contiene la Bibliografía correspondiente para el desarrollo del proyecto, se adjunta un acrónimo y finalmente se añade un anexo.

CAPÍTULO I

ESTADO DEL ARTE

1.1. INTRODUCCIÓN

El presente capítulo comprende el Estado del Arte, el conocimiento de ella permitirá comprender y luego aplicar correctamente los conceptos en entornos virtualizados en cuanto a enfoques y seguridades; el modelado de información, el balanceo de carga y los fundamentos de la metodología AUP.

En cuanto a los *enfoques a la virtualización*, se analizan las técnicas y plataformas. En las *seguridades en entornos virtualizados*, se especifican los riesgos y problemas de seguridad, se establecen estrategias para abordar los riesgos, y se dan ciertos enfoques de seguridad.

El modelado de información, muestra la arquitectura abstracta en la que se basa el modelo del VNE. En *el balanceo de carga*, se analiza la alta disponibilidad, y la repartición en forma equitativa de peticiones que recibe un determinado servicio.

En los *fundamentos de la Metodología AUP*, se desglosan los lineamientos concernientes a fases, disciplinas, hitos, roles y entregables de acuerdo a la metodología.

1.2. ENFOQUES A LA VIRTUALIZACIÓN

Las empresas y las universidades están incorporando el uso de la virtualización[1] para diversas *aplicaciones* en las que se incluyen: consolidación de servidores, la migración en directo, la ejecución de varios sistemas operativos, entornos de pruebas y validación de software, emulación de servicios de redes, entre los de más importancia.

Las principales *ventajas* de la virtualización son: Ahorro de costos de inversión en HW, simplificación de la gestión dada la administración

centralizada de todas las *MVs*, portabilidad entre servidores físicos, facilidad de técnicas de recuperación de desastres; mejoramiento de los tiempos, integración y calidad de puesta a punto de las aplicaciones.

Algunas *desventajas* son: falta de estandarización; dificultad en el acceso a la red de información del host desde las *MVs* y la introducción de una penalización u *overhead* debido al consumo de recursos reduciendo el rendimiento.

1.2.1. TÉCNICAS DE VIRTUALIZACIÓN

i) *Virtualización Completa,* reproduce el funcionamiento de un ordenador origen en otro destino, sin modificar el SO[2]. Utiliza una MV que media entre el SO invitado y el HW nativo. El rendimiento de ésta virtualización es menor debido a la mediación del hipervisor. La ventaja de ésta técnica es que un SO puede ejecutarse sin modificaciones y las instancias corren al mismo tiempo. Fig 1.1.

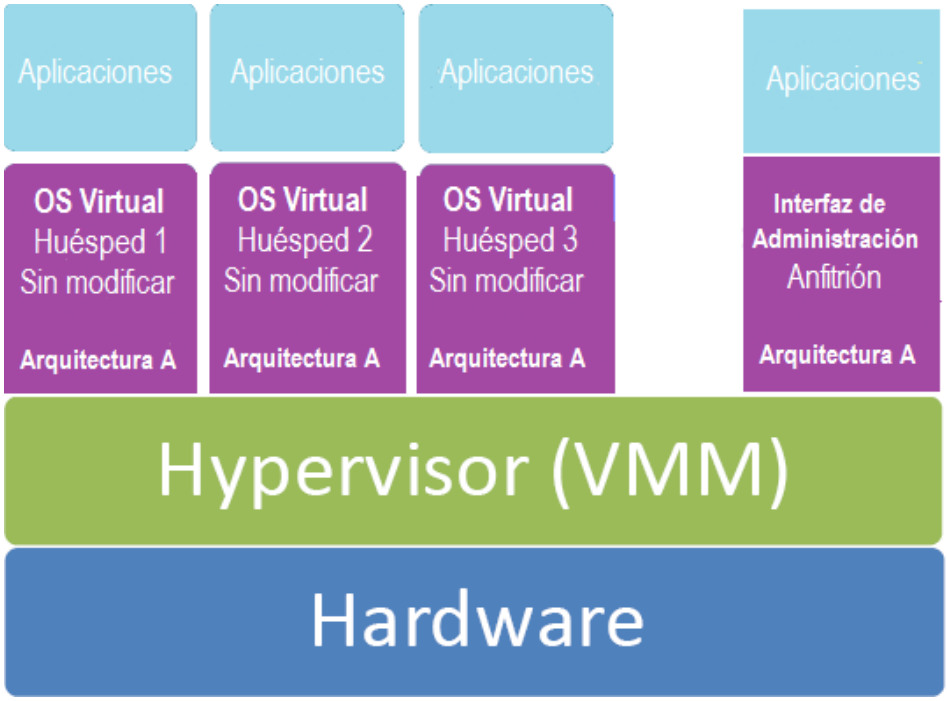

Figura 1.1. La Virtualización Completa utiliza un Hipervisor para compartir el hardware subyacente

ii) *Paravirtualización,* utiliza un *hypervisor*, capa de virtualización intermedia entre el HW y los sistemas operativos hospedados, controla el acceso de éstos a los recursos del computador[2].

El hipervisor modifica los sistemas operativos, lo que es una desventaja. Pero ofrece un rendimiento próximo al de un sistema no virtualizado, Fig 1.2.

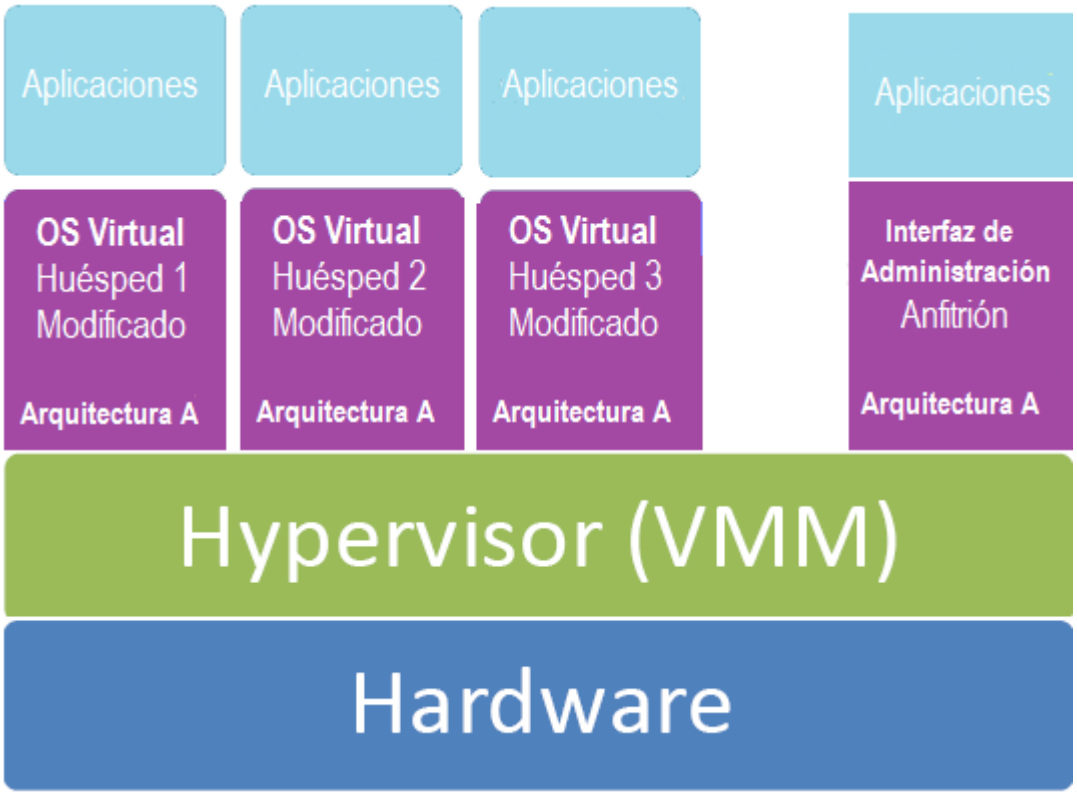

Figura 1.2. La Paravirtualización comparte el proceso con el OS alojado

iii) *Virtualización a nivel del sistema operativo,* agrupan procesos y recursos en contenedores, tienen un kernel común, por lo que un fallo en él, compromete a todas las MVs[2], figura 1.3.

Gestiona tareas, tales como:

- Virtualiza los servidores encima del propio SO.
- Soporta un solo SO y aisla los servidores.
- Requiere cambios en el núcleo del SO.
- La ventaja es un rendimiento igual a la ejecución nativa.

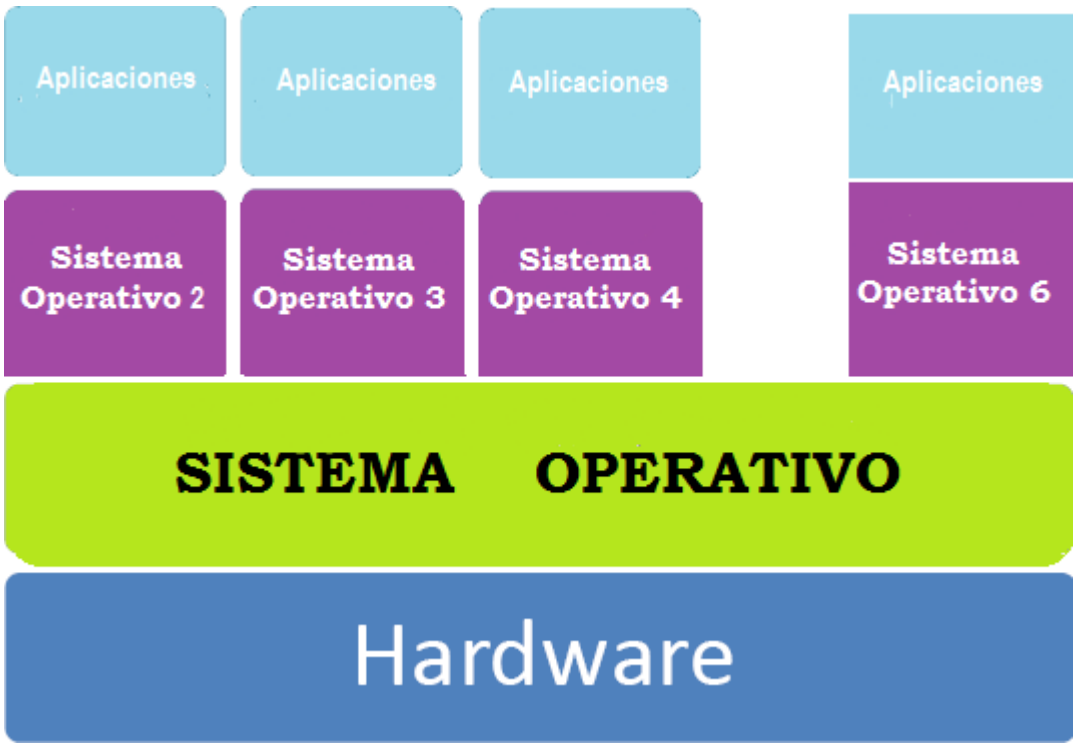

Figura 1.3. La Virtualización a Nivel del Sistema Operativo aisla a los servidores

iv) *Virtualización por hardware*, en el sistema anfitrión se utiliza una MV que emula el HW[2]. La desventaja de ésta técnica es que debido a que cada instrucción al ser simulada por el HW subyacente, resulta ser demasiado lenta, Fig 1.4.

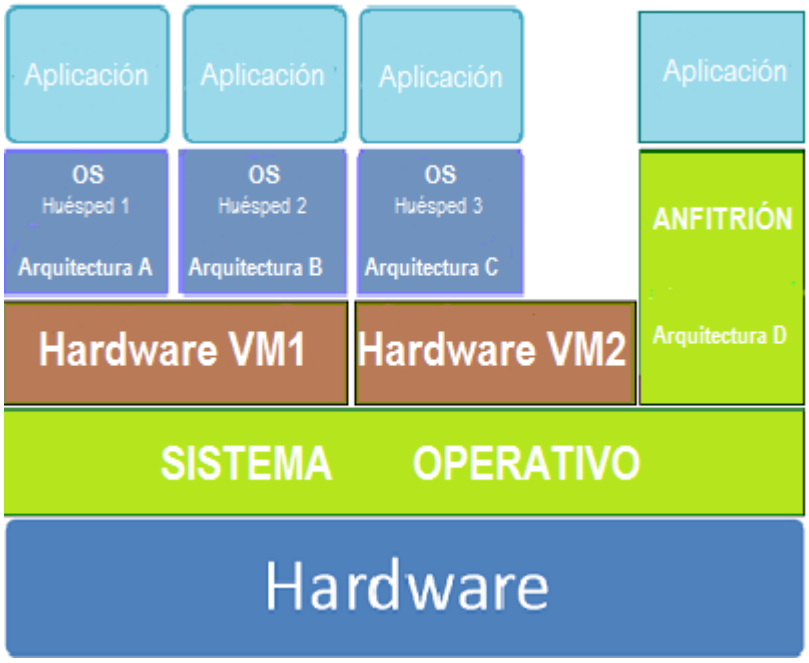

Figura 1.4. La Virtualización por Hardware utiliza una MV para simular el HW

1.2.2. PLATAFORMAS DE VIRTUALIZACIÓN

A continuación se presenta una descripción de ciertas herramientas de virtualización[3][4][5]:

Bochs emula todo el ordenador y periféricos. Permite ejecutar distribuciones Linux en Linux, Microsoft Windows 95/98/NT/2000, aplicaciones en Linux y sistemas operativos BSD (FreeBSD, OpenBSD) sobre Linux, Fig 1.5.

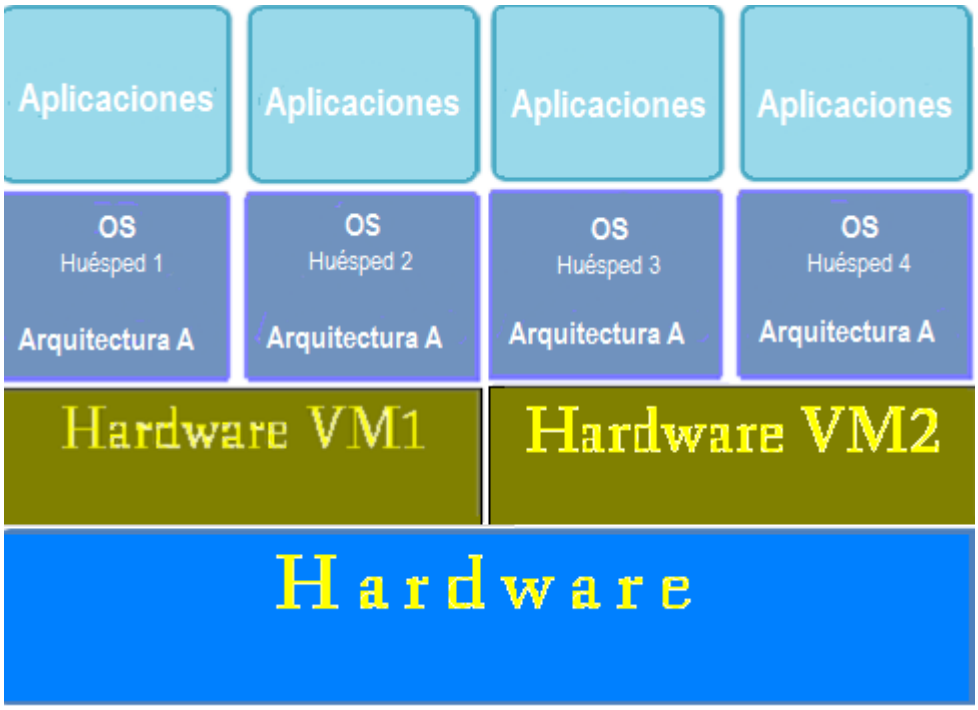

Figura 1.5. Sistema de Emulación mediante un Hardware Subyacente

QEMU es un emulador genérico de procesadores, ejecuta MVs en Linux o Windows; soporta dos modos de operación. *El modo de emulación de sistema completo*; emula todo un ordenador con su procesador y periféricos; además, los OS Windows, Linux sobre Linux, Solaris y FreeBSD. *El segundo modo User Mode Emulation;* sólo puede ser alojado en Linux.

VServer es una solución de virtualización a nivel de SO; virtualiza el núcleo Linux de manera que varios entornos de espacio de usuario denominados *Virtual Private Servers* (VPS) se ejecuten de forma independiente sin tener conocimiento del resto, Fig 1.6.

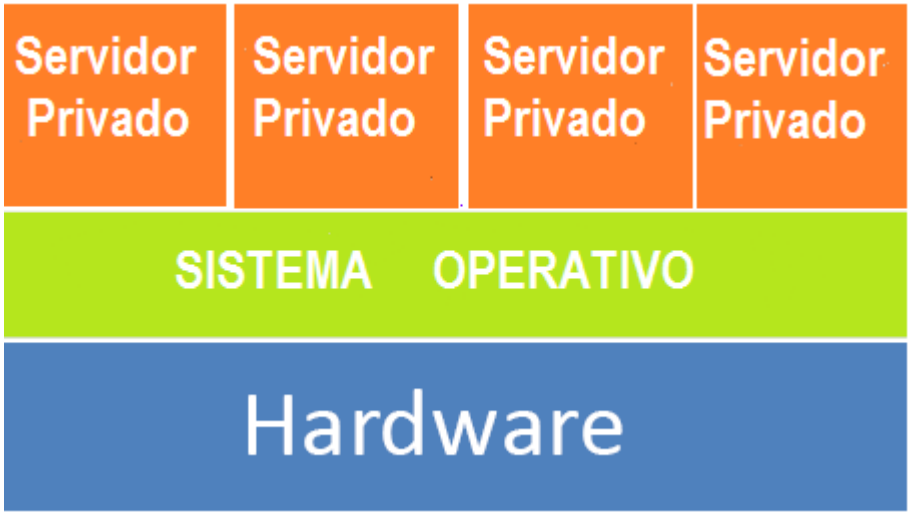

Figura 1.6. La Virtualización a Nivel de Sistema Operativo

UML requiere de un kernel y un sistema de archivos; permite que un OS Linux ejecute otros sistemas operativos Linux en el espacio del usuario. Cada OS Linux alojado existe como un proceso en el sistema operativo Linux anfitrión, Fig 1.7.

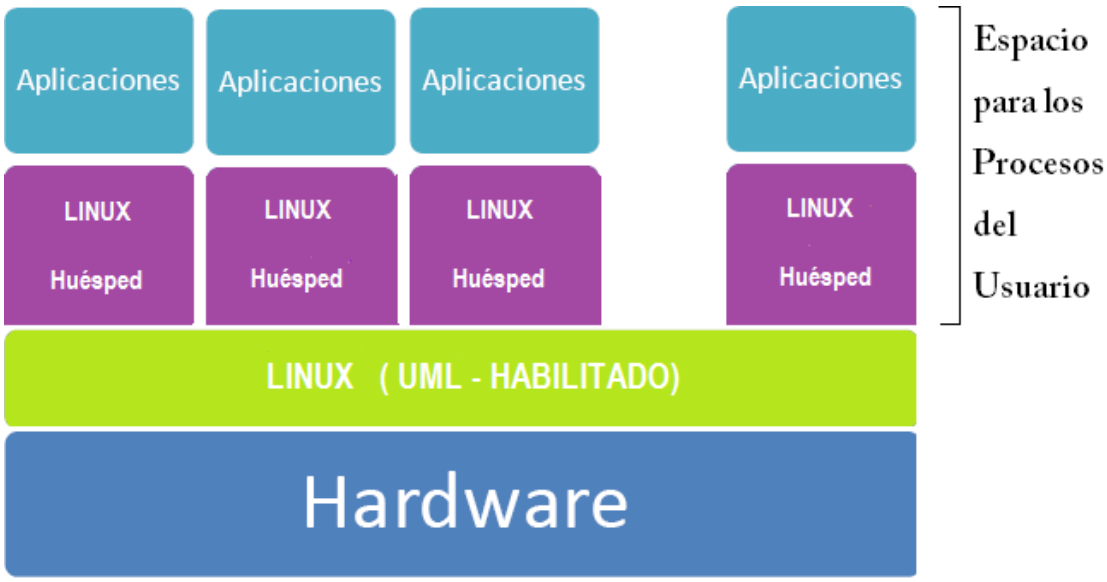

Figura 1.7. Alojamiento de Linux en User Mode Linux

Linux KVM es una herramienta de libre distribución, que emplea la técnica de virtualización completa al convertir al núcleo Linux en un hipervisor utilizando un módulo del núcleo. Este módulo permite a otros sistemas operativos alojados ejecutarse en el espacio de usuario del núcleo Linux anfitrión, Fig 1.8.

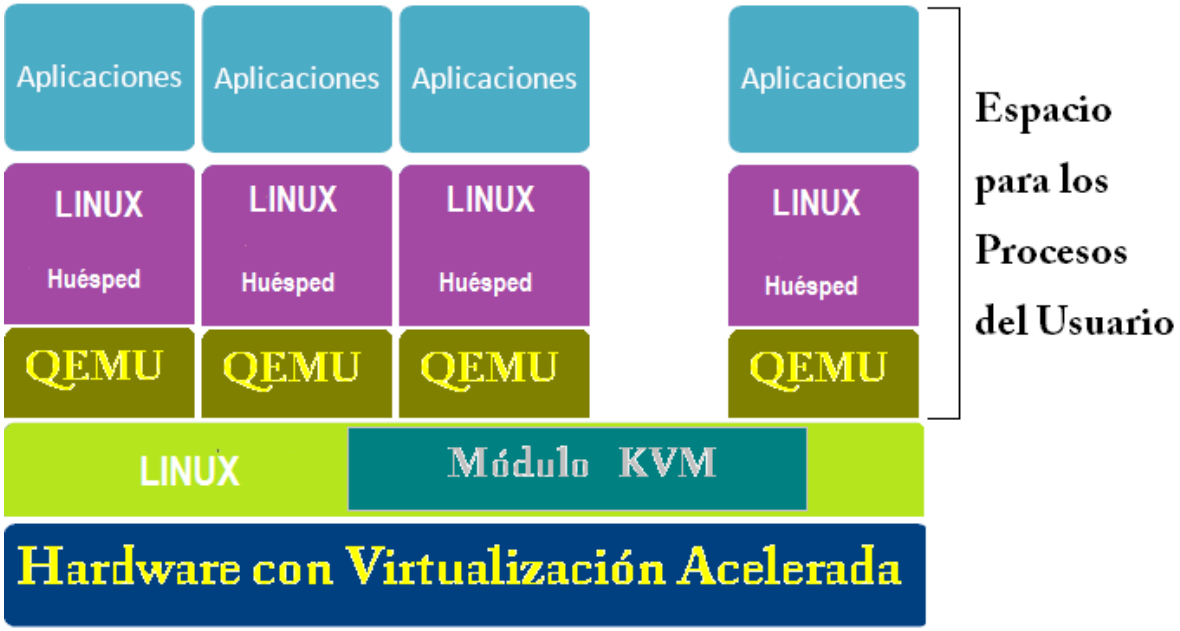

Figura 1.8. Virtualización con Kernel Virtual Machina

Virtualbox está basada en virtualización completa. Se distribuye bajo licencia GNU LGPL (*Lesser* GPL). Fig 1.9.

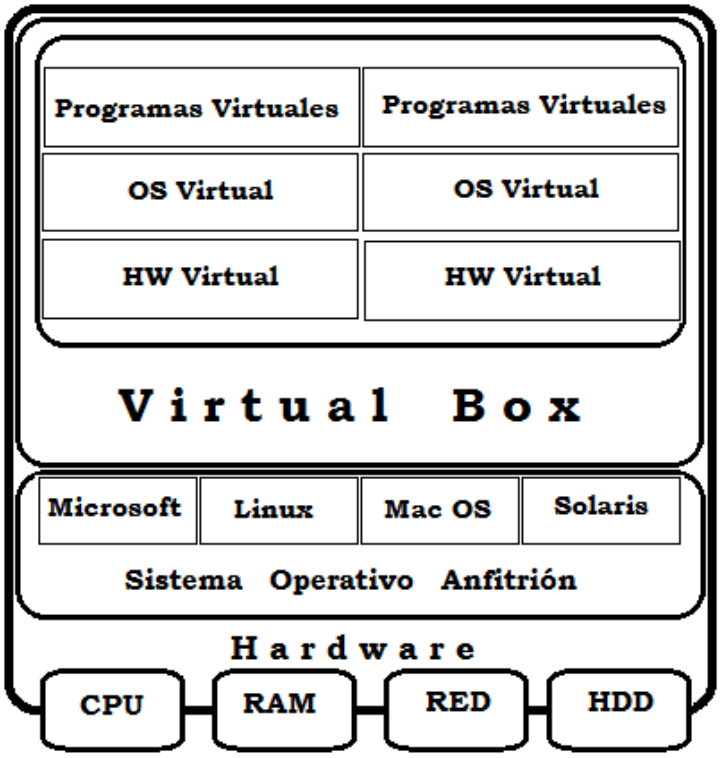

Figura 1.9. Virtualización Completa VirtualBox

Dispone de una interfaz gráfica denominada Virtual Box Manage, permite crear MVs con Windows o Linux y su respectiva configuración de red. Ofrece

un mecanismo de acceso remoto a las MVs mediante RDP (*Remote Desktop Protocol*).

Xen es un entorno de virtualización de código abierto[6][7]. Se distribuye bajo licencia GPL de GNU. Permite ejecutar múltiples instancias de sistemas operativos con todas sus características, pero carece de entorno gráfico. El núcleo de Xen, que administra las MVs, se conoce como *hypervisor.*

Xen precisa modificaciones en el sistema operativo alojado, solo pueden virtualizarse en Xen sistemas operativos parcheados.

Los sistemas operativos soportados por Xen son: Windows, Minix, NetBSD, FreeBSD y OpenSolaris. En la figura 1.10 se muestra la arquitectura de Xen.

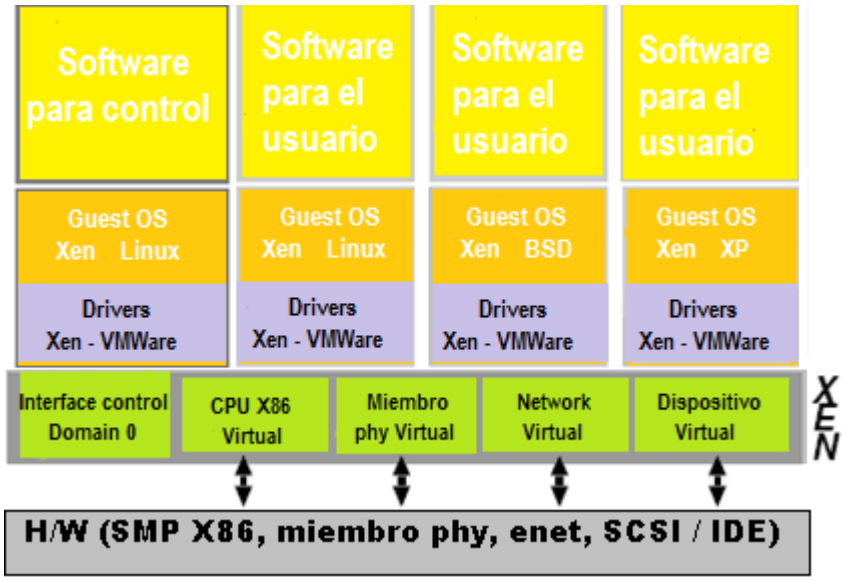

Figura 1.10. Estructura de una MV corriendo el hypervisor Xen

En la figura 1.10 se muestra una MV corriendo el hypervisor Xen, el cual hospeda diferentes sistemas operativos anfitriones e incluye un software que controla la corrida en el Domain0 en un ambiente Xen o Linux.

VMware es un sistema de virtualización completa por software[8][9]; simula un sistema físico (ordenador, HW) con unas características de HW

determinadas, la mayor parte de las instrucciones se ejecutan directamente sobre el HW físico.

Cuando se ejecuta el programa (simulador), proporciona un ambiente de ejecución con los efectos de un ordenador físico, CPU, BIOS, tarjeta gráfica, memoria RAM, tarjeta de red, sistema de sonido, conexión USB, disco duro.

Entre los sistemas operativos alojados y el HW existe un hipervisor funcionando como capa de abstracción. Esta capa de abstracción permite que cualquier SO se ejecute sobre el HW sin ningún conocimiento de otro sistema operativo alojado.

También virtualiza el HW de E/S disponible y ubica drivers para dispositivos de alto rendimiento en el hipervisor. El entorno virtualizado se respalda en un fichero, es decir, que un sistema completo (el sistema operativo alojado, la MV y el hardware virtual) puede migrarse con facilidad y rapidez a una nueva máquina anfitrión para balancear la carga, Fig 1.11.

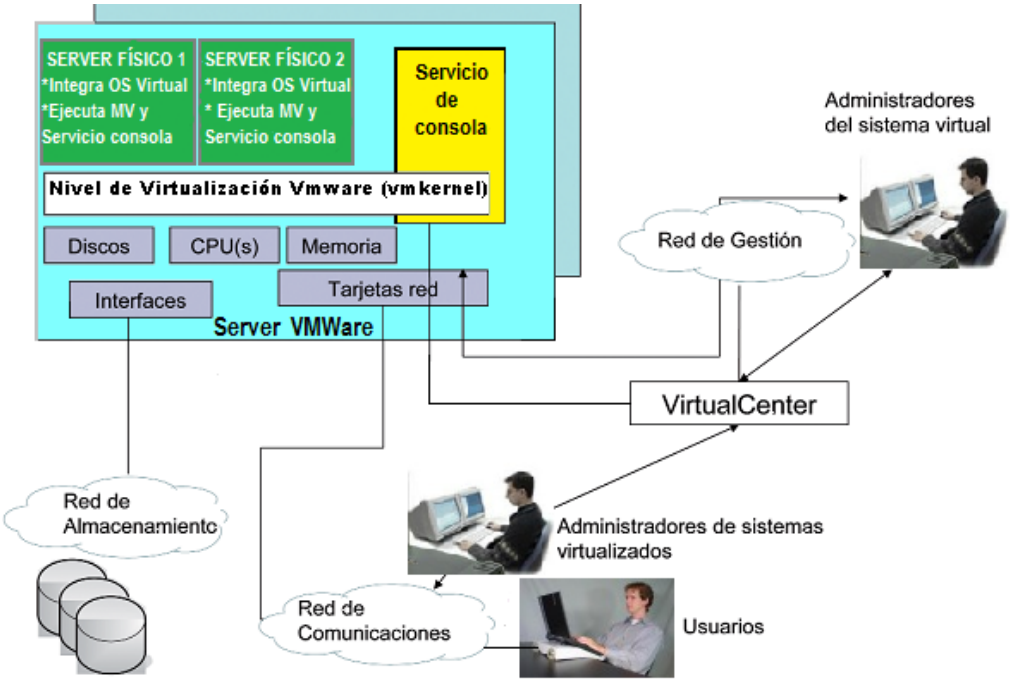

Figura 1.11. Arquitectura de Virtualización VMware

DMTF ha estandarizado un modelo para recursos de red sobre la base de CIM en el framework de la WBEM[10]. WBEM se basa en CIM, el cual es el

Modelo Conceptual de Información orientado a objetos para describir las entidades, su composición, y las relaciones. Se utiliza el *Formato para la Administración de Objetos* (MOF) para definir la descripción de los objetos.

El elemento principal en una arquitectura WBEM es el gestor de objetos CIM (CIMOM), el cual es un gestor de base de datos para instancias de clases CIM. CIM permite implementar el acceso normalizado a los datos y a las operaciones de gestión de los recursos que están representados en el CIMOM.

DMTF es una organización creada para proponer estándares de Gestión de TI. Ha creado CIM, WBEM, DMI y OVF entre otros. En la Fig 1.12 se adjuntan los desarrollos de la DMTF.

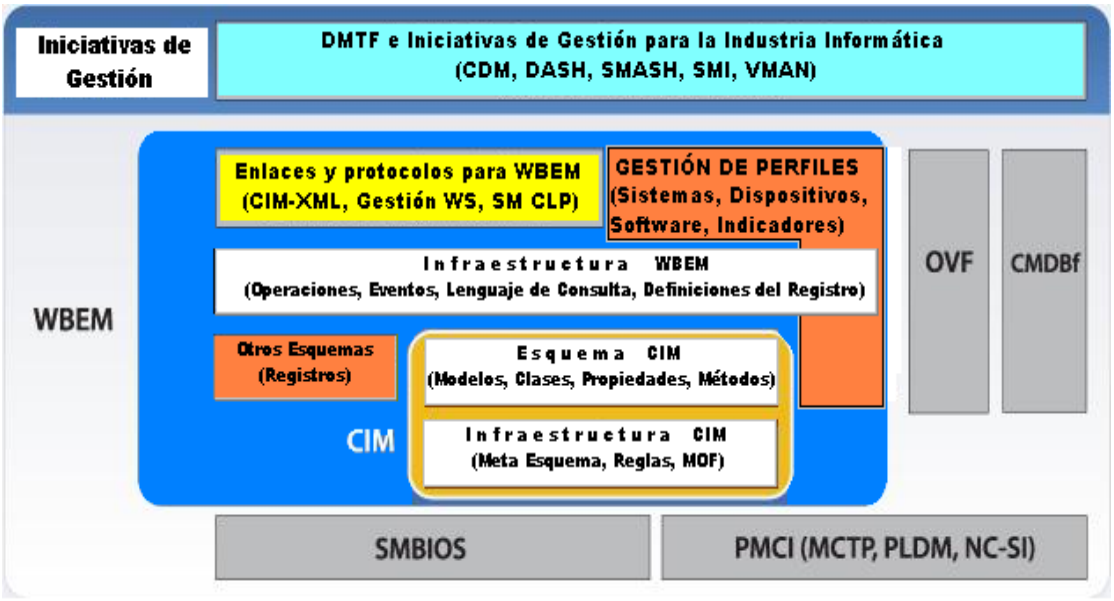

Figura 1.12. Estándares para Infraestructuras de Virtualización Distribuidas propuestas por DMTF

OVF se gestiona como un mecanismo de transporte de plantillas de MVs[11][12]. Un mecanismo basado en certificados que garantiza la integridad de la plantilla.

Una plantilla OVF puede estar compuesta por una o más MVs. Un archivo descriptor (xml) que proporciona visibilidad de la información del S.O.,

licencia y HW de las MVs transportadas en la plantilla. Por ejemplo, se puede desplegar una pila de aplicaciones LAMP (Linux + Apache + MySQL + PHP) desde un único ovf.

Tanto OVF como OVA son las dos presentaciones del estándar. El formato OVF está formado por varios archivos, mientras que el formato OVA es una versión del OVF en formato tar (archivo único) para facilitar la descarga desde la red.

Se considera que a futuro el formato OVF permitirá proporcionar interoperabilidad entre nubes; además, puede que se lo utilice para tener una nube híbrida, una parte privada y otra pública, fluyendo los VNEs de una parte a otra según los criterios corporativos de rentabilidad, seguridad o eficiencia energética.

CIMOM es una base de datos que permite almacenar objetos CIM[13][10]. Los objetos CIM sirven para representar algo que existe, sea físico (HW) o lógico (SW), Fig 1.13. De esta manera se administra información sobre un equipo, discos, dispositivos, periféricos, sistemas operativos, impresoras, procesos, seguridad, servicios, carpetas compartidas, cuentas de usuarios y grupos.

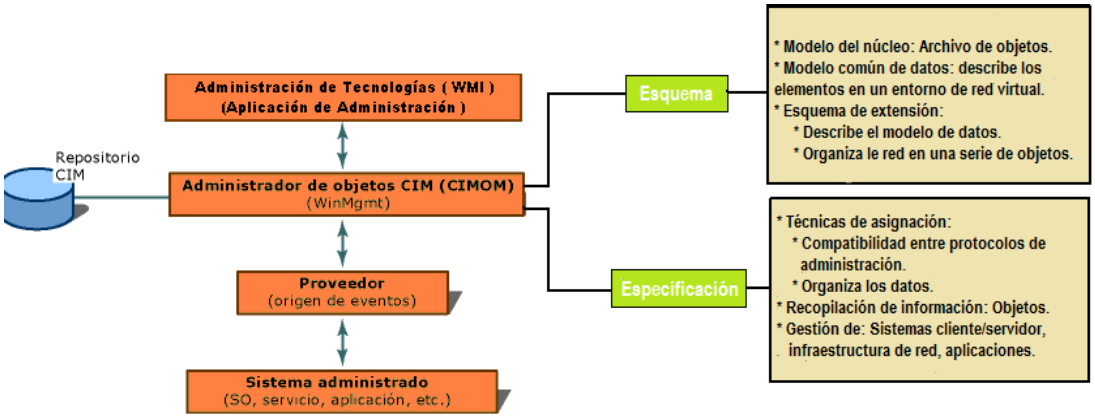

Figura 1.13. Arquitectura del Repositorio del Modelo de Información Común (CIM)

Un lenguaje con significado textual denominado MOF, permitirá la definición de clases e instancias. Las sentencias en lenguaje de MOF son compiladas por el compilador MOF. El compilador MOF interactúa con el CIMOM. Para recuperar los objetos CIM, con el soporte de una API, se gestionarán los estándares WBEM.

A continuación, se muestran detalles importantes del CIM:

- El CIM es un modelo de datos orientado a objetos que se usa para describir la información.
- El CIM describe la manera en la que se organizan los datos y no necesariamente el modelo de transporte que se usa para transferir los datos.
- Las aplicaciones de administración habilitadas para CIM recopilan información de una variedad de dispositivos y objetos de CIM, incluidos sistemas cliente y servidor, dispositivos de infraestructura de red y aplicaciones.
- El estándar CIM proporciona detalles sobre técnicas de asignación que ofrecen una mejor compatibilidad con otros protocolos de administración.
- El modelo de datos de CIM resume y describe todos los elementos en un entorno de red.
- El esquema CIM proporciona las descripciones del modelo de datos real y organiza la red en una serie de objetos administrados, todos ellos interrelacionados y clasificados.
- El esquema CIM se define por medio del archivo de objetos administrados (MOF), el cual proporciona un modelo estandarizado para describir información de administración entre clientes.
- El archivo MOF no está restringido a una implementación particular, permite el intercambio de información de administración entre sistemas y clientes diferentes.

WBEM es un estándar de gestión y de tecnologías de la entidad DMTF independiente de las plataformas y los recursos[14][10].

Es una iniciativa que se caracteriza por su arquitectura distribuida, es un modelo orientado a *Información Común del Objeto*; la integración de una red, sistemas, aplicaciones, servicios; y estándares basados en la gestión.

Incorpora representaciones de elementos de sistemas a través de un CIM, permite relacionar esos elementos y así crear una descripción del sistema, facilita el trabajo recíproco con las interfaces propietarios y define los protocolos basados en Internet con el fin de tener acceso y gestionar esas descripciones en XML.

Un componente del CIM es el repositorio CIMOM, desde donde se recupera la información del objeto. WBEM corre sobre el sistema gestionado CIMOM. Al CIMOM accede una aplicación cliente que monitorea y controla el dispositivo gestionado.

Para atender los requerimientos de la aplicación cliente, el CIMOM usa proveedores, quienes son los que realizan las tareas de gestión facilitando los diferentes recursos. La Figura 1.14 muestra la arquitectura WBEM.

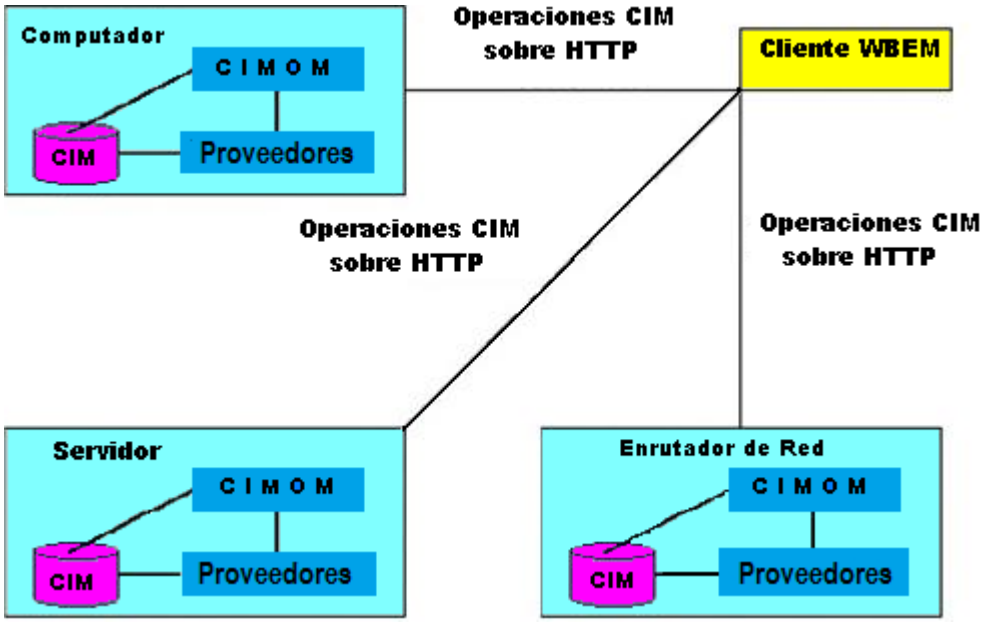

Figura1.14. Arquitectura WBEM

La iniciativa de WBEM presenta las siguientes ventajas:

- *Escalabilidad:* Utiliza un web browser para supervisar y mantener una gama de recursos y dispositivos: routers, hubs, PCs, estaciones de trabajo, aplicaciones distribuidas, DBs, unidades centrales y todo lo implicado.

- *Creciente variedad de aplicaciones y mayor funcionalidad:* Creación y selección de aplicaciones más rápido; así como la agregación de funcionalidades.
- *Costes más bajos para el levantamiento y operación:* Un sistema cliente web habilitado facilitará el acceso a los datos de gestión para las redes de UNIX, del NT de Microsoft Windows, del MVS, Open VMS[14].
- *Arquitectura de unificación:* Permite el acceso a datos desde diversas tecnologías subyacentes tales como: Win32, WMI, Desktop Management Interface (DMI) y el protocolo (SNMP).

JCRAFT es un grupo de herramientas de software que permite implementar transferencia de archivos, ejecución remota, realizar ssh automático, entre otras[6][15]. Tales herramientas funcionan como librerías .jar las cuales se incorporan a las aplicaciones java.

Se describe a continuación cada una de ellas:

- **JSch** permite conectarse a un servidor sshd para facilitar la transferencia de archivos.
- **JZib** sirve para comprimir o descomprimir información que es enviada o receptada por la transferencia de archivos.
- **JCTerm** es un emulador de para terminal ssh2.
- **JOrbis** permite manipular objetos para multimedia.
- **JHttp Tunnel** es la implementación del protocolo GNU httptunnel, el cual crea una coneccion bidireccional virtual de datos.
- **JRexec** facilita la ejecución remota entre el cliente y el servidor.

GlassFish es una plataforma JavaEE5 que permite cargar aplicaciones, soporta versiones de tecnologías: JSP, JSF, Servlets, EJBs, java API para Servicios Web (JAX-WS), Arquitectura Java para enlaces XML (JAXB), Metadatos de Servicios Web para la Plataforma Java 1.0 [16].

JavaEE puede ser utilizada en más de un servidor sin importar el fabricante, no necesita de cambios, la Figura 1.15 muestra la arquitectura GlassFish.

GlassFish brinda soporte en cuanto a: minimización del tiempo en desarrollo de aplicaciones JEE, administración del servidor, Monitoreo de peticiones, mejor servidor de aplicaciones Opensource, gestiona servicios web y el monitoreo de trabajos repetitivos, eficiente velocidad de procesamiento, alta escalabilidad, administración centralizada de clústeres, bajo consumo de memoria, interoperabilidad con .NET 3, utiliza código de Sun Java Application Server.

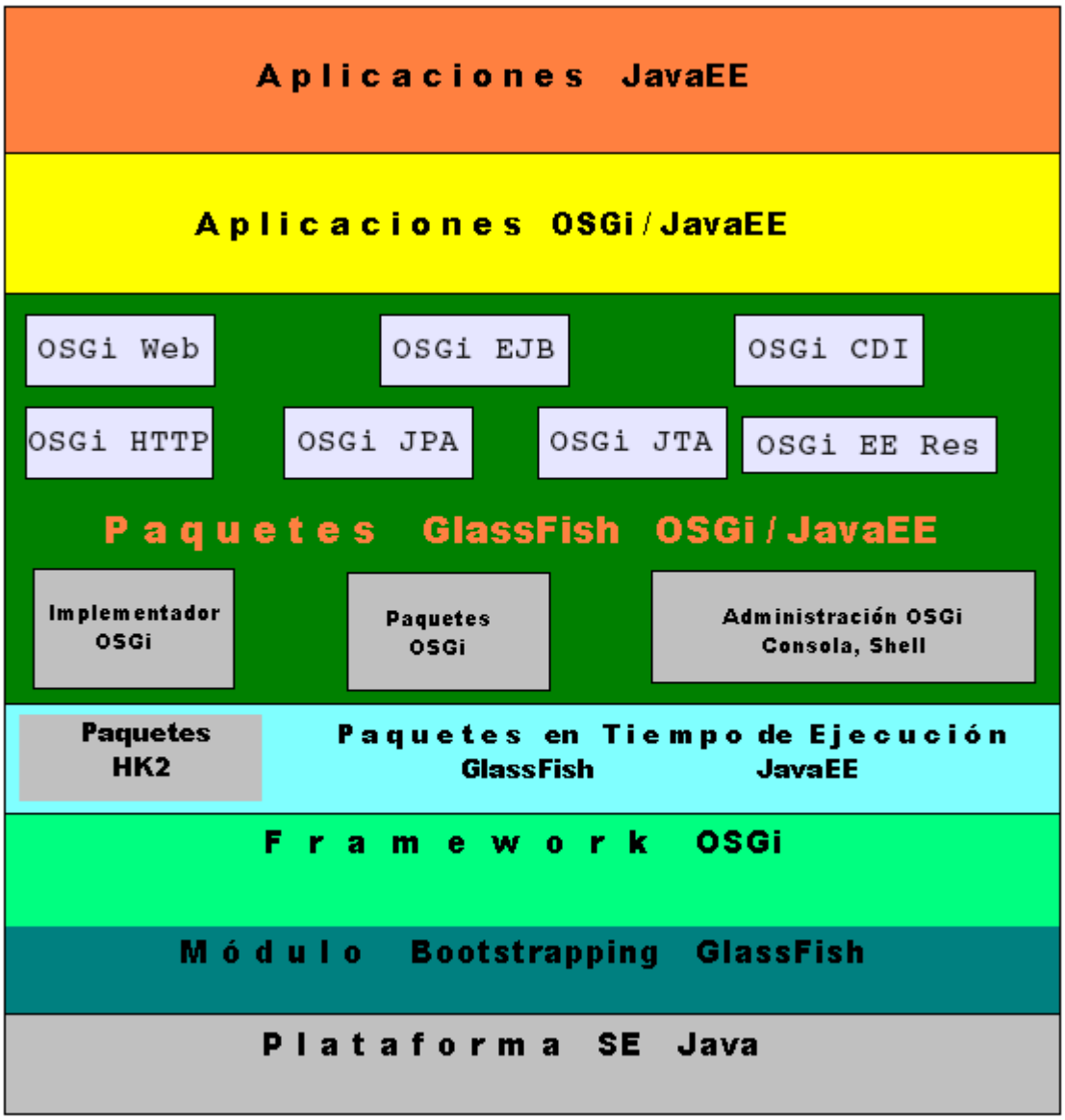

Figura 1.15. Arquitectura GlassFish

1.3. SEGURIDADES EN ENTORNOS VIRTUALIZADOS

1.3.1. RIESGOS Y PROBLEMAS DE SEGURIDAD EN ENTORNOS VIRTUALIZADOS

Se entiende como una amenaza cualquier evento que puede agredir la confidencialidad, la integridad o la disponibilidad de los recursos, provocando un impacto en la organización o al usuario final; a estos fallos de seguridad se los conoce como vulnerabilidades, las mismas que están asociadas a la tecnología, a sus aplicaciones y protocolos[17].

Ataques a la Infraestructura del entorno virtualizado: Existen dos tipos principales de ataques a una infraestructura de virtualización: hyperjacking y virtual machine jumping (VMJ).

- *El hyperjacking,* método que inyecta un hipervisor fraudulento (monitor de máquina virtual [VMM]) en una infraestructura legítima (VMM u sistema operativo) con control sobre todas las interacciones entre el sistema de virtualización atacado y el hardware[18].

- El *VM jumping o guest-hopping,* método de ataque que explota las vulnerabilidades en los hipervisores, por cuanto permite que malware o ataques remotos pongan en peligro las protecciones de VNEs y obtengan acceso a otras máquinas virtuales, hosts e hipervisor.

Ataques a las características del entorno virtualizado: La migración de un VNE, si se hace de manera insegura, puede exponer todos los elementos de un VNE tanto a *ataques de búsqueda pasiva* como *ataques de manipulación activa*.

- *Ataques de bùsqueda pasiva*, método de ataque basado en contraseñas y claves de *búsqueda* de la memoria.

- *Ataques de manipulación activa*, puede suscitarse: al momento de la configuración del sistema de manipulación mientras las MVs migran por toda la red; al hacer una inyección de malware en la memoria de una MV sobre la

marcha; al explotarse la asignación de direcciones de control de acceso a medios (MAC), el enrutamiento local y el tráfico de red.

Cumplimiento y desafíos en la administración: La auditoría del cumplimiento y la administración diaria del sistema son aspectos difíciles cuando se trabaja con sistemas virtualizados. La expansión desmedida de un VNE plantea un desafío para una empresa, en razón de que:

- La infraestructura de un VNE es complejo de proveer e implementar, ya que comprende: sistemas físicos y lógicos, cantidades y tipos de MVs.
- La disposición de MVs es a menudo administrada por distintos grupos dentro de una organización, lo que hace difícil para la función de TI controlar qué aplicaciones, sistema operativo y datos se le instalarán.
- La expansión desmedida de MVs e incluso MVs inactivas hará que sea un reto obtener resultados a partir de las evaluaciones de vulnerabilidad, aplicación de parches/actualizaciones y auditoría.

1.3.2. ESTRATEGIAS PARA ABORDAR LOS RIESGOS EN ENTORNOS VIRTUALIZADOS

- El método eficiente para mitigar la amenaza del hyperjacking es utilizar la confianza en HW y el inicio seguro del hipervisor con el soporte de las tecnologías en el procesador[17].

- Para enfrentar los riesgos asociados a los ataques de VM jumping y la migración de la MV, se debe tener claro que el hipervisor y los anfitriones que admite son SW y que, como tal, se les debe aplicar parches, fortalecer el hipervisor y los huéspedes que soporta[17].

- Los riesgos se reducen mediante aislamiento físico de la red y segmentación de las MVs con los sistemas; lo que permitirá agrupar MVs con aplicaciones de iguales caraterísticas[17].
- El uso de cifrado de transporte garantizará la seguridad en la migración de MVs[17]. Se pueden implementar túneles de red privada virtual (VPN) de un sistema a otro.

1.3.3. ENFOQUES DE SEGURIDAD EN ENTORNOS VIRTUALIZADOS

Incluye la seguridad en los cálculos de costos totales de propiedad:

- En entornos virtuales es importante considerar el ahorro de costos que se consigue basados en reducciones de hardware y consumo energético[17].

- Los servicios de seguridad instalados en cada plataforma física, que permiten controlar los dispositivos virtuales que realizan la monitorización, la detección de intrusos, la validación de rutas, el seguimiento[17].

Hacer de la seguridad una prioridad en la fase del diseño de la virtualización:

- Se optimizará el consumo eléctrico al aislar la información a un único VNE[17].

- Es importante acotar, el peligro de acceso a los datos, al colocar el servidor Web en Internet para que tome la información digital en el mismo equipo físico que el servidor que hace la gestión y el seguimiento de la información[17].

Monitorizar la red invisible:

- Un servidor se conecta a una red virtual, con el fin de monitorear y proteger los datos de intrusos en tránsito[17].

- Otra solución es el aumento de la monitorización a nivel del hipervisor, gestión de rutas virtuales y herramientas en sistemas virtualizados[17].

Controlar el uso de dispositivos de almacenamiento portátil:

- Las empresas pueden comprar imágenes del sistema en un disco duro, llevarlas a un sitio de recuperación, conectarlas a los datos replicados y situarse muy por delante de los tiempos de recuperación habitual[17].

- Sin embargo, las unidades flash USB, las tarjetas digitales y los iPods pueden proporcionar gigabytes de información portátil a cualquier usuario[17].

Mantenerse al día de los avances en seguridad para virtualización:

- Las organizaciones necesitan estar al corriente de lo que ocurre para desplegar las medidas apropiadas.

- Poner en marcha otros mecanismos de control que puedan anticiparse a problemas para los que ahora no hay solución.

La construcción de una topología de red basada en la segregación:

- Para gestionar las comunicaciones entre servidores de manera remota y centralizada, éstos requieren de interfaces de administración, lo cual se logra utilizando una red física o VLAN independiente.

- La red de almacenamiento debería estar separada a nivel físico o lógico.

- Los servidores deberían utilizar sistemas de almacenamiento masivo para almacenar todas las MVs, ficheros de auditoría de eventos, cifrado en la transmisión de datos, e implementar autenticación.

- Los VNEs conviene que utilicen interfaces de red y redes independientes; además, esta red se puede segregar en función de la topología que se necesite construir, e introducir firewalls y switches virtuales entre las diferentes zonas para aumentar la seguridad.

La gestión de usuarios a nivel del Sistema operativo y los que administran las MVs pero sin privilegios sobre el equipo físico:

- Para los dos tipos de usuarios se debe implementar una política de contraseñas robusta que asegure unos mínimos de complejidad.

- A nivel de sistema operativo, se debería deshabilitar la cuenta del usuario root para que no puedan iniciar sesión desde ningún servicio y utilizar cuentas especializadas para las labores de administración.

- En la gestión del sistema de virtualización, prima la segregación de tareas, debiéndose procurar que cada cuenta tenga los permisos necesarios para que el usr pueda realizar tareas que le corresponden.

Las seguridades como estratégica de que la información debe ser integral a las fases de evaluación, diseño e implantación de los VNEs para proteger los activos:

- Se deben elegir soluciones maduras en el mercado, con el fin de certificar la seguridad de sus productos.

- Es indispensable aprovechar las soluciones de virtualización para mejorar la seguridad de los servicios y sistemas, separando aquello que no se podía separar y añadiendo niveles de disponibilidad o mejorando los actuales.

- Consultar y aplicar las recomendaciones de seguridad de los fabricantes de las soluciones obtenidas durante la fase de experimentación y pruebas.

- Es necesario revisar con precisión las definiciones de acceso a las redes de almacenamiento para garantizar que los sistemas virtuales accedan únicamente a los elementos que necesitan.

- El diseño y configuración de mecanismos de gestión garatizará de que los aministradores dispongan del nivel de acceso necesario, y que todas las acciones sobre el VNE estén debidamente auditadas y supervisadas.

- Es necesario redefinir las políticas de seguridad y los controles de acceso de los sistemas virtualizados en atención a su nueva situación; además, se deberán reevaluar los controles que fueron desechados inicialmente.

1.4. MODELADO DE INFORMACIÓN

La fig. 1.16 muestra la arquitectura abstracta en la que se basa el modelo de información. Se compone de redes hosts físicos (sistema de computadoras) interconectados a través de enlaces lógicos (VPN, VLAN, entre otros) y enlaces físicos (por lo general, en el caso de redes de área local).

En particular, dentro de cada host físico es posible crear e implementar una red de topología denominada VNE, utilizando una plataforma de virtualización dado (por ejemplo, Xen o VMware Server). En cada uno de estos VNEs se muestra cómo las MVs están interconectados a través de conexiones virtuales. Cada MV tiene su propia tarjeta virtual de red (NIC) y la dirección IP establecida.

Esta arquitectura permite que uno o más VNEs puedan ser desplegadas en la misma máquina cliente.

Por último, cabe señalar que cada MV tiene una dependencia en el respectivo host dentro del mismo VNE.

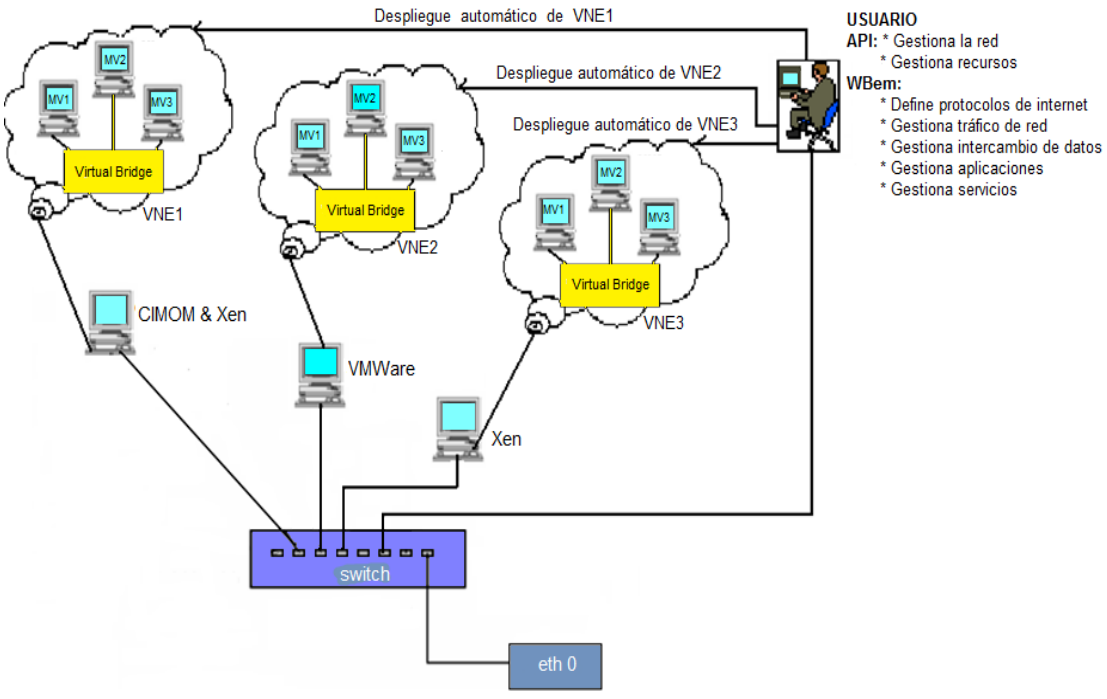

Figura 1.16. Arquitectura abstracta en la que se basa el Modelo de Información Distribuido

1.5. BALANCEO DE CARGA

El balanceo de carga[19] permite compartir el trabajo a realizar entre varios procesos, ordenadores, discos u otros recursos, de tal manera, que se consiga una mayor disponibilidad y rendimiento.

Está íntimamente ligado a los sistemas de multiprocesamiento, o que hacen uso de más de una unidad de procesamiento para realizar labores útiles.

La Tecnología de clústeres virtuales sustenta su actividad en el *balanceo de carga*, cluster que permite que un conjunto de servidores de red compartan la carga de trabajo y de tráfico de sus clientes, aunque aparezcan para estos clientes como un único servidor.

Este tipo de servicio es de gran valor para compañías que trabajan con grandes volúmenes de tráfico y trabajo en sus webs, servidores de correo, y aplicaciones. S*e ha considerado la Alta Disponibilidad y el Balanceo de Carga, Tabla 1.1.*

Tabla 1.1. Descripción de acciones de la alta disponibilidad y balanceo de carga.

CONSIDERACIONES	ACCIONES
ALTA DISPONIBILIDAD	Garantiza la continuidad de un determinado servicio independientemente de ocurre algún tipo de fallo en el sistema. En caso de que uno de sus servidores falle, el sistema de balanceo de car dirigirá el tráfico a las máquinas que estén trabajando.
BALANCEO DE CARGA	Reparte de forma equitativa el conjunto de peticiones que recibe un determinado servicio. Al balancear la carga de trabajo en un conjunto de servidores, se optimiza y mejora el tiempo de acceso y la confiabilidad.

Se realizaron pruebas pertinentes para simular el balanceo de carga, cuyos resultados se indican en la sección 4.5.2.

Su gestión, tiene como soporte un *Algoritmo de balanceo de carga*, el cual proporciona funcionalidad de redundancia activa/pasiva, lográndose mediante su ejecución que se recoloquen las MVs en Hosts en los que los recursos estén disponibles balanceando continuamente la capacidad y asegurando que cada MV tenga acceso a los recursos en tiempo real.

1.6. FUNDAMENTOS DE LA METODOLOGÍA AUP

AUP es una versión simplificada de RUP[20][21], la misma que define un proceso ágil para el desarrollo de aplicaciones de software, gestionando entregables incrementales en el tiempo; aplica técnicas ágiles, tales como:

- Desarrollo basado en pruebas.
- Modelado ágil mediante UML.
- Gestión ágil de cambios.
- Rediseño de la base de datos.
- Mejora la productividad.

Esta metodología consta de cuatro fases, siete disciplinas, cuatro hitos, catorce roles y varios entregables (Fig 1.17); los cuales servirán de soporte para el desarrollo del proyecto " SISTEMA DE CONTROL Y SEGURIDADES PARA LA APLICACIÓN DE TECNOLOGÍAS DE VIRTUALIZACIÓN PARA LA ESPE".

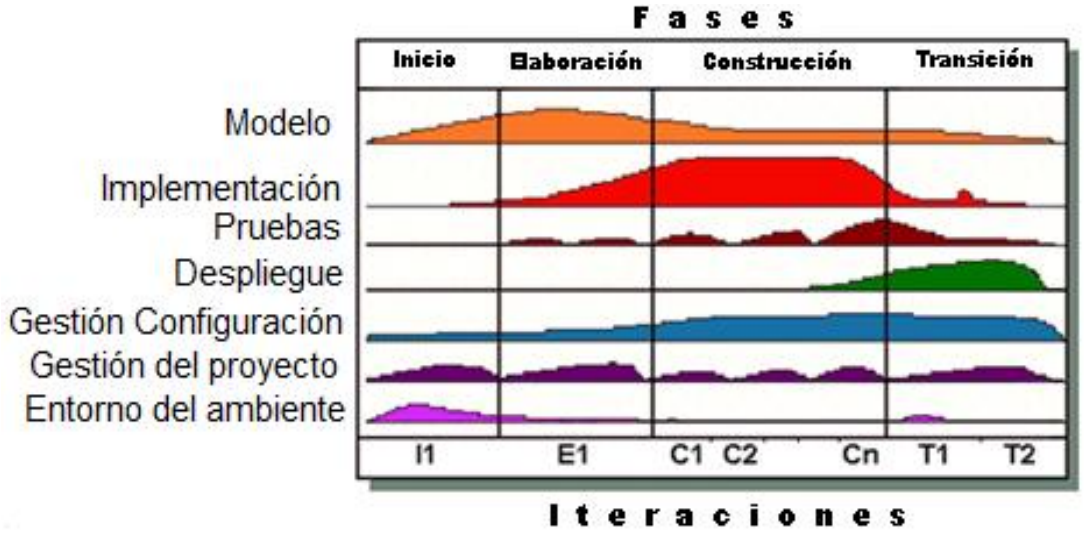

Figura 1.17. Fases y Disciplinas de la Metodología AUP (Fuente: RUP de IBM)

1.6.1. FASES DE LA METODOLOGÍA AUP

Comprende la inicialización, la elaboración, la construcción, y la transición; cuyas acciones se describen en la Tabla 1.2.

Tabla 1.2. Descripción de las fases de la metodología AUP.

FASES	ACCIONES
INICIALIZACIÓN	• Planteamiento del problema. • Propuesta de la solución. • Objetivos: general y específicos. • Justificaciones del proyecto: teórica y práctica. • Alcance previsto. • Levantamiento de requerimientos. • Especificaciones de requerimientos del SWV en base a la norma IEEE 830. • Selección del SWV en base a la norma IEEE 830.
ELABORACIÓN	• Gestiona las rectificaciones de los artefactos generados en la fase de inicialización.
CONSTRUCCIÓN	• Diseño de la arquitectura distribuida. • Recopilación de los requerimientos mediante un prototipo inicial. • Diseño del entorno virtual de red. • Diseño de diagramas de clases. • Diseño de los diagramas de casos de uso. • Diseño de los diagramas de secuencia. • Diseño de diagramas de despliegue. • Diseño del diagrama conceptual de datos. • Diseño del diagrama físico de datos. • Generación de la BDs en MySQL. • Desarrollo de scripts java, mof, sh, cfg, vmx. • Desarrollo de aplicativos de seguridades, pistas de auditoría virtualización.
TRANSICIÓN	• Correcciones de los artefactos aprobados generados en la fase construcción. • Pruebas de caja blanca, de caja negra, de integración y de rendimient • Publicación de papers como un aporte técnico-científico sobre VNEs. • Documentación de la tesis.

1.6.2. DISCIPLINAS DE LA METODOLOGÍA AUP

Comprende la inicialización, implementación, pruebas, despliegue, administración de configuración, administración del proyecto y el entorno; las mismas que se describen en la Tabla 1.3.

Tabla 1.3. Descripción de las disciplinas de la metodología AUP.

DISCIPLINAS	DESCRIPCIÓN
MODELADO	• Entendimiento del problema. • Identifica una solución viable para el proyecto AVES.
IMPLEMENTACIÓN	• Convierte el modelo del proyecto en código ejecutable. • Realiza actividades de pruebas (testing).
PRUEBAS	• Asegura la calidad del software mediante una evaluación objetiva. • Detecta defectos. • Asegura que el sistema funcione según lo previsto mediante una validación. • Verifica que se cumplan los requisitos.
DESPLIEGUE	• Cumplimiento del plan para la entrega del proyecto AVES. • Instalación en el entorno de los usuarios finales.
ADMINISTRACIÓN DE CONFIGURACIÓN	• Administración de los artefactos que componen el proyecto AVES. • Administración de las versiones del proyecto. • Administración sobre los cambios en las versiones del proyecto.
ADMINISTRACIÓN DEL PROYECTO	• Coordinación en la ejecución de las actividades implícitas en el sistema. • Administración de seguridades: gestión de riesgos, asignación de personal y seguimiento.
ENTORNO	• Garantiza que el proceso sea adecuado. • Orientación de normas y directrices. • Disponibilidad de Hw y Sw para el equipo de desarrollo.

1.6.3 HITOS DE LA METODOLOGÍA AUP

Comprende las fases de inicialización, elaboración, construcción y transición de proyecto; las mismas que se describen en la Tabla 1.4.

Tabla 1.4. Hitos de la metodología AUP.

HITOS	GESTIÓN
INICIALIZACIÓN	• Objetivos del ciclo de vida del proyecto.
ELABORACIÓN	• Arquitectura del ciclo de vida del proyecto.
CONSTRUCCIÓN	• Capacidad operacional inicial del proyecto.
TRANSICIÓN	• Puesta en marcha del proyecto.

1.6.4. ROLES DE LA METODOLOGÍA AUP

Para considerar los Roles son necesarios los participantes en el presente proyecto, los mismos que se describen en la Tabla 1.5.

Tabla 1.5. Roles de la metodología AUP.

ROLES	PERSONAL
INVOLUCRADOS	Ing. Walter Fuertes Ph.D. Ing. Luis Guerra e Ing. Fausto Meneses
ADMINISTRADORES DEL PROYECTO	Ing. Luis Guerra e Ing. Fausto Meneses.
ESPECIALISTAS HERRAMIENTAS DE DESARROLLO	Ing. Luis Guerra e Ing. Fausto Meneses.
ADMINISTRADORES DE CONFIGURACIÓN	Ing. Luis Guerra e Ing. Fausto Meneses.
INGENIEROS DE PROCESOS	Ing. Luis Guerra e Ing. Fausto Meneses.
MODELADORES ÁGILES	Ing. Luis Guerra e Ing. Fausto Meneses.
ADMINISTRADORES DBs ÁGILES	Ing. Luis Guerra e Ing. Fausto Meneses.
DESARROLLADORES	Ing. Luis Guerra e Ing. Fausto Meneses.
IMPLANTADORES	Ing. Luis Guerra e Ing. Fausto Meneses.
EXAMINADORES/AUDITORES	Ing. Walter Fuertes Ph.D e Ing. Edison Lasca MsC.
DOCUMENTADORES TÉCNICOS	Ing. Luis Guerra e Ing. Fausto Meneses.
ADMINISTRADORES DE PRUEBAS	Ing. Luis Guerra e Ing. Fausto Meneses.
EJECUTORES DE PRUEBAS	Ing. Luis Guerra e Ing. Fausto Meneses.

1.6.5. ENTREGABLES DE ACUERDO A LA METODOLOGÍA AUP

De acuerdo a la Metodología AUP se desarrollarán los artefactos que se indican en cada una de las siguientes fases, se explica en la Tabla 1.6.

Tabla 1.6. Entregables de acuerdo a la metodología AUP.

FASES	ARTEFACTOS
INICIALIZACIÓN	Levantamiento de requerimientos. Especificación de requerimientos del Sw aplicando la Norma IEEE 830[22].
ELABORACIÓN	Rectificaciones a los artefactos generados en la fase de inicialización.
CONSTRUCCIÓN	Diseño de la arquitectura distribuida. Recopilación de los requerimientos mediante un prototipo inicial. Diseño del entorno virtual de red. Diseño de diagramas de clases. Diseño de los diagramas de casos de uso. Diseño de diagramas de secuencia. Diseño de diagramas de despliegue. Diseño del diagrama conceptual de datos. Diseño del diagrama físico de datos. Generación de la base de datos en MySQL. Desarrollo de scripts java, MOF, sh, cfg, vmx. Desarrollo de aplicativos de seguridades, pistas de auditoría y virtualización.
TRANSICIÓN	Correcciones a los Artefactos de la Fase de Construcción Aprobados. Pruebas unitarias, caja blanca, caja negra y de integración del Proyecto AVES. Paper publicado como un aporte técnico – científico en la que se considerarán: El estudio de bases teóricas investigadas sobre la Seguridad de datos e Información en Aplicaciones de Tecnologías de Virtualización. El soporte de las Plataformas VMWare y Xen con Ubuntu (LINUX), NetBeans, Glass fish, MySQL. La recopilación de información de casos exitosos en seguridad de datos e información en aplicaciones virtuales generados por las mejores prácticas. Tesis de Grado.

1.7 CONCLUSIÓN

Los conceptos descritos en el Estado del Arte; permitirán junto con el soporte de lineamientos que facilita la metodología AUP, el establecimiento de estrategias para abordar los riesgos que conlleva una arquitectura abstracta en la que se basa el modelo distribuido de un entorno virtual de red, considerando la alta disponibilidad y la repartición en forma equitativa de peticiones que recibe un determinado servicio.

CAPÍTULO II

DETERMINACIÓN DE LA INFRAESTRUCTURA DE VIRTUALIZACIÓN Y ENFOQUES DE MODELADO DE INFORMACIÓN

2.1 . INTRODUCCIÓN

El presente capítulo comprende la infraestructura de virtualización y enfoques de modelado de información, el conocimiento de éstos elementos permitirá integrarlos y aplicarlos correctamente en plataformas de virtualización; la misma que tiene como soporte, los enfoques de modelado para infraestructuras de virtualización distribuidas, las herramientas y criterios que formalizan la evaluación de los métodos de modelización.

El uso de *las plataformas de virtualización* estará gestionada por los *enfoques de modelado de virtualización distribuida* DMTF-CIM y VMWare-CIM. *Las herramientas* que permiten implementar plataformas de virtualización tienen como soporte plataformas Libvirt-CIM, VMWare-CIM y Xen-CIM

Con el soporte de la Norma IEEE 830 se determinan *criterios que permiten formalizar la evaluación de los métodos de modelización* para la Gestión de la red y puntos de información de modelado.

2.2 . PLATAFORMAS QUE SOPORTAN EL PROYECTO AVES

El uso de una plataforma específica estará gestionada por las tareas y el rendimiento del VNE; en razón de que allí correrá una cantidad de MVs, cargas de trabajo, soluciones de servicios de TI compartidos, servicios de aplicaciones[23].

Dicha plataforma deberá disponer de:

- Agregación incluida de recursos de hardware de granjas de servidores físicos, almacenamiento y redes.

- Herramientas de migración en línea, eliminación de tiempos de baja de servicios y disrupciones a los usuarios finales cuando las MVs sean movidas desde un servidor a otro, o de un almacenamiento a otro.

- Asignación de recursos los cuales permitan equilibrar automáticamente las cargas de trabajo.

- Gestión de energía que automáticamente mantenga encendidos o apagados los servidores físicos donde corren las MVs.

- Servicios de disponibilidad tanto para caídas programadas como no programadas concernientes al HW (servidor, componente, almacenamiento, redes y SW) a los efectos de mejorar la disponibilidad de las aplicaciones.

- Las soluciones provean una administración de la virtualización completa para todo el ciclo de vida del VNE.

- Las soluciones integren una amplia gama de sistemas operativos para poder ejecutar todas las aplicaciones y las que se dispondrán a futuro.

- Las soluciones posean una base instalada de invitados (guest) en entornos productivos y de desarrollo con una cantidad de MVs.

En cuanto a los procesos que integran, las soluciones deberán soportar la virtualización de las cargas de trabajo; tales como:

- Servicios Web, Servicios de Directorio, Servidores de Archivos, servidores de Bases de Datos de E/S, Bases de Datos, Sistemas de Mensajería.

- Las soluciones deberán estar preparadas para poder gestionar aplicaciones de misión crítica en VNEs sin degradación de performance y alta disponibilidad.

- Analizar la escalabilidad de las soluciones a medida que se ejecutan más cargas de trabajo virtuales sobre un VNE.

2.2.1. ENFOQUES DE MODELADO PARA INFRAESTRUCTURAS DE VIRTUALIZACIÓN DISTRIBUIDAS

Tabla 2.1. Modelados para Infraestructuras de Virtualización Distribuidas.

MODELADO	GESTIÓN
Virtualización DMTF-CIM[24]	• Creación de perfiles. • Diseño de especificaciones. • Sistema de virtualización perfil[25], gestiona el modelado de la relación entre un host físico y los recursos virtuales. • Sistema de virtualización perfil[25], gestiona las tareas de virtualización como la creación o modificación de los sistemas virtuales y sus configuraciones respectivas, y el sistema virtual. • Sistema de virtualización perfil[26], define un modelo de objetos destinado a realizar la inspección y el funcionamiento de un sistema virtual (MV), sus componentes y ciclo de vida.
Virtualización VMWare-CIM	• Especifica un esquema de extensión de la CIM. • Especifica el modelo de la estructura lógica de las plataformas: VMWare, MVs y recursos asignados[27].

Las clases de este esquema se han organizado en tres grandes grupos:

- El modelo Dominio Virtual de VMWare.
- El modelo de Subsistemas de Almacenamiento Virtual de VMware, y
- El modelo de Servidor Múltiple de VMware.

2.2.2. HERRAMIENTAS QUE PERMITEN IMPLEMENTAR LAS PLATAFORMAS DE VIRTUALIZACIÓN

Tabla 2.2. Herramientas que permiten Implementar Plataformas de Virtualización

HERRAMIENTAS	ACTIVIDADES
Libvirt-CIM[28]	• Gestiona Xen hypervisor, emulador QEMU, KVM, contenedores Linux OpenVZ. • Proporciona el acceso a máquinas remotas a través de conexiones autenticadas y cifradas. • Gestiona plataformas heterogéneas de virtualización utilizando una serie de archivos XML basado en estándares CIM.
VMWare-CIM[28]	• Explora las MVs y examina el almacenamiento físico asignado a una MV[16]. • Explora los VMWare de forma remota con el soporte del CIMOM.
Xen-CIM[28]	• Con el soporte de código abierto gestiona los perfiles de DMTF CIM. • Xen-CIM[29] utiliza Libvirt-CIM[30] el cual es un API estándar para la gestión de diferentes tecnologías de virtualización.

2.2.3. CRITERIOS QUE PERMITEN FORMALIZAR LA EVALUACIÓN DE LOS MÉTODOS DE MODELIZACIÓN Y LAS PLATAFORMAS

Para formalizar la evaluación de los diferentes métodos de modelización y de las implementaciones analizadas, en la Tabla 2.1 se ha considerado ciertos criterios para la ***Gestión de la red y puntos de información de modelado***. Se hace una breve descripción en cuanto al Enfoque de Modelización.

Tabla 2.3. Criterios fundamentales sobre el Enfoque de Modelización.

CRITERIOS	APLICACIÓN	GESTIÓN
GESTIÓN CICLO DE VIDA	• Gestiona el ciclo vida de MVs.	• Crear, destruir y configuración de las funciones de la MV.
ASIGNACIÓN DE RECURSOS	• Establece el modelo de asignación de recursos	• Para grupos de recursos, asignaciones datos.
MONITOREO RED VIRTUAL	• Tiene una interfaz de gestión de virtualización.	• Para monitorizar el tráfico o el rendimiento de la red.
SEGUIMIENTO VM	• Tiene la capacidad para monitorear disponibilidad.	• De: RAM, ciclos de CPU, espacio en HD.
DISPOSITIVOS ALMACENAMIENTO, HERRAMIENTAS ADMINISTRACIÓN	• Incluye funciones.	• Para administrar recursos con estándares abiertos.
GESTIÓN ACCESO REMOTO	• Permite accesar de forma remota.	• A equipos virtuales.
INTEROPERABILIDAD	• Tiene la capacidad de dar soporte.	• A diferentes plataformas de virtualización.
SEGURIDAD	• Evalúa la capacidad para trabajar con el cifrado	• De seguridad o autenticación.
IMPLEMENTACIÓN DE LA AUTOMATIZACIÓN DE VNE	• Incluye el despliegue automático.	• De todo un VNE.

Al hacer un análisis de la tabla 2.3, se denota que en el enfoque de modelización; el ciclo de vida se centra en la creación, modificación y configuración de las funciones de las MVs en concordancia con el monitoreo de su disponibilidad y del rendimiento de la red.

La administración de sus recursos se basa en estándares abiertos, teniendo la capacidad de dar soporte a diferentes plataformasde virtualización a través de un acceso remoto.

La seguridad o autenticación se centra en su gestión de trabajar en forma de cifrado; mientras que la implementación de la automatización del entorno de red virtual distribuida se manifiesta mediante un despliegue automático.

La Tabla 4.3 (Selección de software para virtualización) muestra los resultados obtenidos sobre la evaluación de los métodos de modelización e implementaciones, gestionados en base a los criterios de *Gestión de red y puntos de información de modelado* integrándola con la Norma IEEE 830 ERSV con el fin de obtener la funcionalidad automática de VNE, así como garantizar: la consolidación, integridad, confidencialidad y la disponibilidad de la información.

2.3 CONCLUSIÓN

El conocimiento de los elementos analizados en la infraestructura de virtualización y enfoques de modelado de virtualización DMTF-CIM, VMWare-CIM; permitirán junto con las herramientas implementar plataformas de virtualización sobre Libvirt-CIM, VMWare-CIM y Xen-CIM; y con el soporte de la Norma IEEE 830 se determinarán establecer *criterios que ayuden a formalizar la evaluación de los métodos de modelización* para la Gestión de la red y puntos de información de modelado.

CAPÍTULO III

DETERMINACIÓN DE LA PLATAFORMA QUE SOPORTE HERRAMIENTAS DE VIRTUALIZACIÓN

3.1 . INTRODUCCIÓN

El presente capítulo comprende la plataforma que soporte herramientas de virtualización[23][39], su conocimiento permitirá la gestión y ejecución óptima de aplicaciones en los servidores virtualizados; para lo cual es necesario hacer un análisis de la especificación de la virtualización desde ciertas perspectivas, tales como: procesador, memoria, I/O, partición en caliente, migración aplicativa, gestión de sistemas, cargas de trabajo, características de los servidores, hardware y software de los servidores.

El procesador, asocia la ejecución de las aplicaciones a la gestión de los servidores, mediante un diseño que permita agrupar recursos y su uso a través de VNEs y SO. Para alcanzar una virtualización eficiente, *la memoria* deberá ser asignada dinámicamente de una MV a otra.

Con el fin de compartir recursos con todos los usuarios es indispensable Virtualizar *recursos I/O*. La *partición en caliente* evita la interrupción aplicativa para el mantenimiento del sistema, integración, y gestión de carga de trabajo.

La capacidad de virtualización de *migración aplicativa* permitirá ejecutar una gran variedad de plataformas de aplicaciones en plataformas Linux.

Es óptimo que la gestión de sistemas se lo haga con un Gestor de Virtualización Integrada en razón de que facilita la consolidación de cargas de trabajo.

En base a los análisis de las especificaciones anteriores, se establecen las características en cuanto a hardware y software que deben tener los servidores virtuales (VS).

3.2. ESPECIFICACIÓN DE LA VIRTUALIZACIÓN DESDE LA PERSPECTIVA DEL PROCESADOR

La optimización para ejecutar aplicaciones asociados al manejo de servidores físicos múltiples, se obtendrá mediante un diseño que permita agrupar recursos y su uso a través de VNEs y sistemas operativos múltiples, con el soporte de:

- Una sóla MV tendrá la capacidad de actuar como entorno operativo completamente separado, usando recursos del sistema compartido.
- Los Recursos compartidos, se ajustarán automáticamente el procesador compartido, recursos de memoria o almacenamiento a través de sistemas operativos múltiples, obteniendo capacidad de MVs inactivas para manejar demandas de altos recursos de otras cargas de trabajo.
- La gestión de recursos compartidos se obtendrá la potencia y la flexibilidad de requisitos informáticos múltiples del sistema en una sola máquina.
- La partición de discos se obtendrán múltiples MVs por núcleo del procesador, y dependiendo del modelo, deberán funcionar gran cantidad MVs en un solo servidor; cada uno con su procesador, memoria, y recursos I/O.
- La consolidación de sistemas se podrá contribuir a la reducción de costos operativos, mejorar la disponibilidad, facilitar la gestión y mejorar los niveles de servicio, permitiendo que las empresas implementen las aplicaciones con rapidez.
- Los grupos múltiples de procesadores compartidos se obtendrá el equilibrio automático no disruptivo de potencia de procesamiento entre MVs asignadas a grupos compartidos, dando como resultado un aumento de rendimiento.
- La capacidad compartida dedicada se optimizará los ciclos en reserva CPU de MVs dedicadas del procesador a un Pool Compartido del Procesador.

3.3. ESPECIFICACIÓN DE LA VIRTUALIZACIÓN DESDE LA PERSPECTIVA DE LA MEMORIA

Para los propósitos de una virtualización eficiente se deberá disponer de una tecnología que permita:

- Reasignar memoria inteligente y dinámicamente de una MV a otra para aumentar la utilización, flexibilidad y rendimiento.
- Dar las facilidades para compartir un pool de memoria física entre MVs en un servidor, ayudando a aumentar la utilización de memoria y a reducir los costos del sistema.
- Asignar dinámicamente memoria entre las MVs según se necesite, para optimizar el uso de la memoria física global en el pool.

3.4. ESPECIFICACIÓN DE LA VIRTUALIZACIÓN DESDE LA PERSPECTIVA DE I/O

La disponibilidad de un Servidor Virtual I/O (VIOS) concerniente a una MV nos deberá permitir:

- Virtualizar recursos I/O con el fin de compartir recursos con todos los usuarios.
- Que la disposición de un adaptador físico asignado al VIOS pueda ser compartido por una o más MVs.
- Que VIOS deberá estar diseñado para reducir costos eliminando la necesidad de adaptadores dedicados de red, adaptadores de disco, unidades de disco y drivers en cada cliente MV.
- La inclusión de pools de almacenamiento compartido lo que facilitará que los subsistemas de almacenamiento se combinen en un pool común de almacenamiento virtualizado.

3.5. ESPECIFICACIÓN DE LA VIRTUALIZACIÓN DESDE LA PERSPECTIVA DE LA PARTICIÓN EN CALIENTE

La gestión de una partición en caliente deberá:

- Soportar una VM en ejecución de uno de sus servidores a otro sin período de inactividad de aplicación.
- Evitar la interrupción aplicativa para el mantenimiento planificado del sistema, aprovisionamiento, y gestión de carga de trabajo.
- Usarse para transferir fácilmente entornos de funcionamiento a nuevos servidores temporal o permanentemente.

3.6. ESPECIFICACIÓN DE LA VIRTUALIZACIÓN DESDE LA PERSPECTIVA DE UNA MIGRACIÓN APLICATIVA

La capacidad única de virtualización de interplataforma deberá permitir:

- Ejecutar una gran variedad de plataformas de aplicaciones en plataformas Linux.
- Soportar la consolidación de diversas aplicaciones y de esta manera aprovechar el rendimiento avanzado, la dimensionalidad, y las características RAS.
- Traducir instrucciones para un sistema basado en un adecuado procesador y mediante caching de las instrucciones traducidas para optimizar el rendimiento.

3.7. ESPECIFICACIÓN DE LA VIRTUALIZACIÓN DESDE LA PERSPECTIVA DE LA GESTIÓN DE SISTEMAS

Es conveniente que las características de virtualización se gestionen a través de una Consola de Gestión de Hardware(HMC) o de un Gestor de Virtualización Integrada(IVM) desde su nivel de inicio, en razón de que:

- El aporte de un IVM facilita la consolidación de cargas de trabajo con su interfaz basada en Web
- Con IVM, se gestionará un solo sistema, incluyendo la creación de MVs, almacenamiento virtualizado y conexiones de red virtualizadas.
- IVM permitirá la virtualización en servidores múltiples y heterogéneos.
- IVM simplifica la creación y la gestión de dispositivos virtuales estándar y pools de sistemas o combinaciones de MVs en servidores múltiples que pueden ser gestionados como una sola entidad.

3.8. ESPECIFICACIÓN DE LA VIRTUALIZACIÓN DESDE LA PERSPECTIVA DE LAS CARGAS DE TRABAJO

Los diversos grupos de software deberán estar optimizados para un entorno que permita:

- El soporte de la consolidación de cargas de trabajo desde bases de datos y servidores de aplicaciones hasta infraestructura web.
- Trabajar conjuntamente para proveer una infraestructura de aplicación cruzada de sistema virtualizado que pueda reducir los costos operativos y de energía que se necesitan para crear, ejecutar y gestionar aplicaciones de empresa y entornos SOA.
- El aumento de la flexibilidad y agilidad con el fin de asegurar la integridad de los procesos, mejorar el servicio y el rendimiento aplicativo, y gestionar mejor las aplicaciones.

3.9. ESPECIFICACIÓN DE LAS CARATERÍSTICAS DE LOS SERVIDORES QUE SOPORTAN PLATAFORMAS DE VIRTUALIZACIÓN

La profundidad y amplitud de la experticia de los investigadores, deberá estar sustentada en:

- La familiaridad y experiencia de participación activa con los servidores de tecnología de punta.
- Los conocimientos profundos de las tecnologías emergentes, ediciones de software y del hardware.
- El trabajo con equipos de desarrollo y laboratorios de investigación.

En la Tabla 3.1 presentamos las características y beneficios generales que deben disponer los diversos servidores que soportan plataformas de virtualización.

Tabla 3.1. Características de Servers que soportan Plataformas de Virtualización.

CARACTERÍSTICAS DE LOS SERVIDORES	BENEFICIOS PARA SU USO EN PLATAFORMAS DE VIRTUALIZACIÓN
HYPERVISOR	• Soporta entornos operativos múltiples en un solo sistema.
MICRO PARTICIÓN	• Como mínimo deberá soportar 10 MVs por núcleo del procesador.
DINÁMICA DE PARTICIÓN LÓGICA	• El procesador, memoria y recursos I/O deberán moverse dinámicamente al ser gestionados entre MVs. • Las MVs deberán usar recursos de procesador dedicados o compartidos. • Los recursos del procesador deberán moverse automáticamente entre MVs en base a las demandas de carga de trabajo.
POOLS DE PROCESADOR COMPARTIDO	• Los recursos del procesador para un grupo de MVs deberán recortarse con el fin de reducirse los costos de licencia de software.
POOLS DE ALMACENAMIENTO COMPARTIDO	• Los recursos de almacenamiento para los servidores deberán estar centralizados en pools con el fin de optimizar la utilización de recursos.
GESTOR INTEGRADO DE VIRTUALIZACIÓN	• Simplifica la creación de MV y la gestión para los servidores.
MOVILIDAD DE LA PARTICIÓN VIVA	• Live AIX y Linux MVs permiten moverse entre servidores, eliminando el período de inactividad planificado.
COMPARTICIÓN ACTIVA DE MEMORIA	• Facilita que la memoria fluya inteligentemente de una MV a otra para mayor utilización de memoria.
HERRAMIENTA DE PLANIFICACIÓN DEL SISTEMA	• Facilita la planificación y la instalación de servidores.

3.10. ESPECIFICACIÓN DE LOS SERVIDORES DESDE LA PERSPECTIVA DE HW Y SW PARA EL PROYECTO AVES

Todas las pruebas se llevaron a cabo en tres host físicos con las siguientes características, Tabla 3.2.

Tabla 3.2. Características de HW y SW de Hosts que facilitan la Topología de Prueba.

HOSTS	10.1.18.5	10.1.18.60	10.1.18.71
CARACTERÍSTICAS			
HARDWARE			
N^{0}. PROCESADORES	4	2	2
TIPO	Intel Pentium (R)	Intel (R)	Intel ® Core ™
VELOCIDAD	3.2 GHz	2.13 GHz	2.40 GHz
MEMORIA RAM	1 GB	2 GB	3.82 GB
MEMORIA HD	108 GB	88 GB	130 GB
LONGITUD DE PALABRA	64 bits	32 bits	64 bits
SOFTWARE			
SISTEMA OPERATIVO/VERSIÓN	Linux Ubuntu/8.1	Linux Ubuntu/9	Linux Ubuntu/8.1
PLATAFORMA VIRTUALIZACIÓN/VERSIÓN	Xen/3.3	VMWare/3.0.0	Xen/3.3
REPOSITORIO CIMOM	CIM Workshop Copyright 2002 Sun Microsystems, Inc		

En esta topología de prueba, un solo puente virtual se utiliza para conectar todos los VNEs a la interfaz de red física hipervisor.

- *En el caso de Xen*, la creación de redes virtuales se ha configurado por defecto para utilizar la red-puente; permite la ejecución de MVs sobre distintos sistemas operativos simultáneamente en un mismo servidor.

- *En el caso de VMware Server*, se configura como puente de red con el fin de consolidar y flexibilizar los servidores.

En ambos casos, se trata de un tipo de conexión de red entre una MV y la red física externa en el host.

En una topología de puente de red, una MV se comporta como un equipo adicional de host físico en la misma red Ethernet física.

Ponderaciones de las Topologías de Virtualización de Prueba:

Las dos plataformas basan su topología de virtualización en que una MV se comporta como un equipo adicional de host físico en la misma red Ethernet física[24].

Plataforma de Virtualización Xen:

- La topología red-puente facilita la creación de redes virtuales.
- Esta topología trata de un tipo de conexión de red entre una MV y la red física externa en el host.
- Este tipo de gestión red puente permite una configuración por defecto.
- Permite la ejecución de MVs sobre distintos sistemas operativos simultáneamente en un mismo servidor.

Plataforma de Virtualización VMware Server:

- Se configura como puente de red, conexión de red entre una MVl y la red física externa en el host.
- Esta configuración facilita la consolidación y flexibilidad de los servidores.

3.11. CONCLUSIÓN

El conocimiento de la plataforma que soporte herramientas de virtualización junto con las especificaciones de procesador, memoria, I/O, partición en caliente, migración aplicativa, gestión de sistemas, cargas de trabajo, características de servidores, Hw y Sw de los servidores; permitirán establecer un diseño que agrupe recursos y su uso a través de VNEs y SO con el fin de obtener una gestión y ejecución óptima de aplicaciones en los servidores virtuales.

CAPÍTULO IV

DISEÑO E IMPLEMENTACIÓN

4.1. INTRODUCCIÓN

El presente capítulo comprende el diseño e implementación con el soporte de la metodología AUP[21], su desarrollo se basa en el cumplimiento de los requerimientos funcionales; con los resultados obtenidos se contribuirá al conocimiento y al mejoramiento continuo en Aplicaciones de Tecnologías de Virtualización, de manera que puedan adaptarse a cualquier tipo de universidad, lo cual será posible mediante la difusión y transferencia de tecnología. La metodología AUP permite la gestión de la fase de inicialización, elaboración, construcción y la fase de transición.

La fase de inicialización, describe el caso de estudio y la especificación de requerimientos con el fin de determinar el SV de acuerdo a la norma IEEE 830. *La fase de elaboración*, facilita la gestión de las rectificaciones a los artefactos generados en la fase de inicialización.

La fase de construcción, permite el diseño de la arquitectura distribuida, la recopilación de requerimientos mediante un prototipo, el diseño del entorno virtual de red; diagramas de: clases, casos de uso, secuencia, despliegue, conceptual de datos, físico de datos; la generación de la BD en MySQL, el desarrollo de scripts java (MOF, sh, cfg, vmx) y el desarrollo de aplicativos de seguridades, pistas de auditoria y virtualización.

La fase de transición, gestiona las correcciones a los artefactos de la fase de construcción aprobados; facilita las pruebas de caja blanca, de caja negra, de integridad y de rendimiento del proyecto AVES. Se referencian los papers publicados como aporte técnico-científico editados a nivel nacional e internacional. La tesis de grado basa su perfil de investigación en un “Sistema de control de seguridades para el proyecto AVES (Aplicación de tecnologías de virtualización para la ESPE), empleando la metodología AUP”.

4.2. FASE DE INICILIZACIÓN

4.2.1. DESCRIPCION DEL CASO DE ESTUDIO

Planteamiento del Problema:

El presente proyecto se enmarca en una línea de investigación cuyos temas predominantes son el modelado de infraestructuras virtualizadas[31] y los mecanismos de seguridades[32] para el despliegue de servicios en éstos entornos. Además, en el presente trabajo se ha direccionado la aplicación de tecnologías de virtualización[3] como un medio de experimentación, investigación y puesta en producción. En éste contexto surgen los siguientes cuestionamientos:

- ¿Qué Plataformas de Virtualización soportaría el Proyecto AVES?.
- ¿Qué características de los servidores de virtualización se necesitarían para que soporten plataformas en el caso de virtualizar aplicaciones o servidores?
- ¿Cuáles serían las tecnologías estandarizadas que a través del modelado de información de infraestructuras virtualizadas permitirían integrar el CIMOM[4] al proyecto que se presenta?.
- ¿Cuáles serían los mecanismos de seguridad más idóneos para infraestructuras virtualizadas?.
- ¿Cuáles son las herramientas de software existentes para evaluar los tiempos de respuesta al acceso en un entorno virtualizado?.

Para construir soluciones a éstos cuestionamientos y en virtud de que la ESPE requiere de la implantación de un Sistema de Control y Seguridades para poner en ejecución servicios y aplicaciones en plataformas de virtualización, ésta Tesis se orienta a diseñar e implementar soluciones para facilitar el acceso; mantener un registro de pistas de auditoría de los usuarios del Proyecto AVES; garantizar que no se desconfiguren los laboratorios generales, entre otros.

Propuesta de la solución:

En los últimos años se ha suscitado un crecimiento exponencial del Internet[33], por lo que las universidades deben estar preparadas para satisfacer la demanda de los educandos que buscan disponer de servicios

remotos con acceso seguro utilizando plataformas de virtualización independiente de la distancia; en tal virtud, el presente proyecto aborda en concreto el problema de la inseguridad en el acceso a los servicios y a las aplicaciones virtualizadas, con el fin de reducir costos de inversión en hardware y software[29].

Las prácticas que realizan los estudiantes en los laboratorios pueden afectar la configuración física de la red. Para resolver los cuestionamientos planteados y el problema declarado, el proyecto tiene como finalidad:

- Evaluar y definir las Plataformas de Virtualización[30], las mismas que soportarán las necesidades del Proyecto AVES.
- Definir las características de los servidores que se necesitarían para que soporten plataformas determinadas en un entorno virtualizado.
- Diseñar un modelo para la aplicación de mecanismos de seguridad.
- Integrar al CIMOM los mecanismos de seguridad que serán implementados.
- Implementar al sistema controles y seguridades para facilitar el acceso de los usuarios al Proyecto AVES; mantener un registro de pistas de auditoría[34].
- Evaluar los tiempos de respuesta de acceso a una máquina virtual en función del número de usuarios.

OBJETIVOS

Objetivo general:

Desarrollar un Sistema de Control y Seguridades Genérico para la Aplicación de Tecnologías de Virtualización para la ESPE (AVES) sobre ambiente Linux, el cual permita garantizar la integridad, confidencialidad y disponibilidad de la información, mediante la Metodología AUP.

Objetivos específicos:

- Analizar el Estado del Arte: Plataformas de Virtualización; Seguridades en redes; Modelado de información[35][36]; Evaluación de tiempos de acceso a máquinas virtuales; Aplicación de la Metodología AUP.

- Modelar la solución generando una extensión de CIM, luego aplicar la Metodología AUP[20] para desarrollar el sistema utilizando plataformas de virtualización, ambientes de desarrollo como NetBeans, GlassFish y MySQL.
- Aplicar las Buenas Prácticas sobre las seguridades en las redes de datos e información en Aplicaciones de Tecnologías de Virtualización.
- Diseñar e implementar topologías de prueba sobre VMware y Xen con el fin de desplegar VNEs a fin de evaluar las seguridades del Proyecto AVES.
- Evaluar y validar las prestaciones y los resultados experimentales a través de pruebas y procesamiento estadístico.

JUSTIFICACIÓN DEL PROYECTO

Justificación teórica:

Es de especial interés realizar éste proyecto, porque permite analizar una de las diez tecnologías de mayor importancia en la industria según Gartner Inc.[37] (Tecnología Green y dentro de esta la virtualización puesto que hay una reducción importante en costos de espacio, energía, repuestos, mantenimiento y administración de los servidores); en segundo lugar porque permite contribuir a la ciencia con soluciones a situaciones reales.

Justificación práctica:

De este proyecto se beneficiarán los directivos, docentes y estudiantes de la ESPE al implementar un mecanismo de seguridad e integrar las soluciones al Proyecto AVES.

Las Plataformas de Virtualización son tecnologías que permiten reproducir redes reales en escenarios virtuales[38][39][40]; además, permiten ejecutar y probar múltiples ambientes de validación de software y brindan facilidades para realizar el dimensionado de prestación de servicios de redes, lo que posibilita ventajas de virtualización, tales como: ahorro de costos de inversión, simplificación de su gestión dada por su administración centralizada de todas las MVs, portabilidad entre servidores físicos, facilidad de técnicas de recuperación de desastres.

Mediante la filosofía en Aplicaciones de Tecnologías de Virtualización, los beneficios que se pretenden alcanzar son:

- La experimentación e investigación, utilizando VNEs, con las mismas funcionalidades de una red real, generará ahorro de gastos en equipos y dispositivos de red para ambientes virtuales, reducirá costos de mantenimiento de HW y disminuirá costos de energía eléctrica.
- Tecnológicamente el presente estudio de Plataformas y Tecnologías Virtuales servirá como punto de partida para otras pruebas relacionadas con redes IP.
- Con los resultados obtenidos se contribuirá al conocimiento y al mejoramiento continuo en Aplicaciones de Tecnologías de Virtualización, de manera que puedan adaptarse a cualquier tipo de universidad, lo cual será posible mediante la difusión y transferencia de tecnología.

Alcance previsto:

- El proyecto pretende cubrir los artefactos recomendados por la Metodología AUP hasta la fase de pruebas.
- El proyecto no cubre la fase de implantación, mantenimiento y cambios.
- A fin de dar cumplimiento a las pruebas del modelo se implementará una red local compuesta por una máquina que hará las veces de servidor de CIMOM y Xen, otra como servidor de Xen para simular balanceo de carga y un tercer servidor con VMware. La tercera trabajará como estación cliente.

LEVANTAMIENTO DE REQUERIMIENTOS

Levantamiento de los Requerimientos del Sistema de Control y Seguridades para el Proyecto AVES (Aplicación de Tecnologías de Virtualización para la ESPE)

- Determinar las Plataformas de Virtualización[8] que darán soporte a las necesidades del proyecto AVES.
- Establecer las características de los Servidores de Virtualización que darán soporte a las Plataformas de Virtualización; en el caso de virtualizar escritorios, aplicaciones o servidores.

- Establecer los Mecanismos de Seguridad más idóneos para Infraestructuras Virtualizadas.
- Diseñar un Modelo para la Aplicación de Mecanismos de Seguridad.
- Determinar las Tecnologías estandarizadas que a través del Modelado de Información de Infraestructuras Virtualizadas permitan integrar el CIMON.
- Integrar al CIMON los Mecanismos de Seguridad más eficientes.
- De las Herramientas de Software existentes; seleccionar las herramientas que permitan evaluar los tiempos de respuesta al momento del acceso a un VNE en función del número de usuarios; así como, el balanceo de carga.
- Implementar al sistema controles y seguridades que permitan facilitar el acceso de los usuarios al Proyecto AVES sin que desconfiguren las máquinas de los laboratorios; manteniendo un registro de pistas de auditoría[9].

4.2.2. ESPECIFICACIÓN DE REQUERIMIENTOS PARA DETERMINAR EL SOFTWARE DE ACUERDO A LA NORMA IEEE 830 (ERS)

La Especificación de los Requerimientos del Software que se va a utilizar, estará estructurado en base a los criterios planteados en el Plan de Tesis del Proyecto “Sistema de Control y Seguridades para el proyecto AVES empleando la Metodología AUP”.

Definiciones: En el **Anexo A** se describe cada uno de los elementos.

Referencias

Se Hace uso de las siguientes referencias para obtener la información requerida para establecer cada requisito que debe cumplir el software de virtualizaciòn:

- IEEE-STD-830-1998 [22]: Especificaciones de los Requisitos del Software.
- Proyecto de Titulación: Centra la investigación de las Tecnologías de Virtualización y su empleo en el Manejo de Control y Seguridades manteniendo una pista de auditoria, lo cual estará sujeto a la: evaluación y definición de las plataformas de virtualización utilizando IEEE 830 para selección de software; definición de las características de los servidores que soporten entornos virtuales; modelización de mecanismos de seguridad en el

CIMOM; evaluación de los tiempos de respuesta de acceso a una MV y a la compartición de trabajo a realizar entre varios procesos, ordenadores, discos u otros recursos de las MVs en los VNEs (balanceo de carga).

Funciones del Producto de Virtualización

El software de virtualización[17] deberá optimizar una infraestructura existente mediante una infraestructura virtual que cuente con las funciones desplegadas en la siguiente tabla 4.1.

Tabla 4.1. Funciones principales del software de virtualización.

OPTIMIZACIÓN	CONTINUIDAD	COMPATIBILIDAD	RESTRICCIONES	INFRAESTRUCT
Consolidación de servidores	Alta Disponibilidad	Independencia de HW	Soporta tecnology Fibre Channel	Instala máquinas SO Linux
Ahorros costos de propiedad	Migración de MVs en caliente	SOs Guest soportados	Soporta tecnology iSCSI	Config máquinas SO Linux
Rápido retorno la inversión	Manejo backups	Almacenamiento centralizado SAN	Soporta tecnology NAS	SWVirtual con procesador INTEL
Facilidad gestión	Balanceo Asignación MVs	Almacenamiento centralizado NAS	Compatible con HW arquitect X86	SWVirtual con procesador AMD

Requisitos Específicos

Esta sección de la especificación de requisitos de software (ERS) contiene todos los requerimientos que han sido considerados tomando en cuenta criterios de estabilidad, disponibilidad y confiabilidad, en el **Anexo B** se describe cada uno de los Requisitos.

Selección del Software de Virtualización utilizando la Norma IEEE 830

Establecimiento de Valorización a los Requerimientos

Una vez establecidos los Requerimientos para la selección del Software de virtualización se procede a asignar un puntaje para cada requerimiento, tabla 4.2.

Tabla 4.2. Valorización y descripción de los requerimientos del SWV.

REQUERIMIENTO	TIPO	PUNTAJE	DESCRIPCIÓN
REQ 01	ADMINISTRACIÓN	0	No posee ninguna interfaz administr.
		1	Posee una interfaz de administració.
		2	Posee dos interfaces administración
REQ 02	COMPATIBILIDAD DE HW	1	Soporta pequeña lista compatibilidad
		2	Soporta mediana lista compatibilidad
		3	Soporta extensa lista compatibilidad
REQ 03	CAPA VIRTUALIZACIÓN	0	Funciona sobre un SO base
		1	Se instala directamente en el HW
REQ 04	INSTALACIÓN SW DE VIRTUALIZACIÓN	1	Instalación mediante discos locales
		2	Instalac discos locales y a través SAN
REQ 05	CONSOLIDACIÓN DE SERVIDORES	0	No realiza consolidación de servers
		1	Realiza consolidación de servers
REQ 06	SOPORTE DE VARIOS SO INVITADOS	1	Soporte 0 – 10 sistemas operativos
		2	Soporte 11 – 20 sistemas operativos
		3	Soporte 21 – 30 sistemas operativos
REQ 07	SOPORTE MULTIPROCESADOR	1	Soporta 2 procesadores p/c MV
		2	Soporta 4 o más procesadores p/c MV
REQ 08	MIGRACIÓN EN VIVO DE MÁQUINAS VIRTUALES	0	No realiza migración de MVs
		1	Realiza migración en vivo de MVs
REQ 09	ALTA DISPONIBILIDAD DE MV ENTRE HOSTS DE UN CLUSTER	0	No soporta alta disponibilidad
		1	Soporta caída simultánea de 1 host
		2	Soporta caída simultánea de 2 hosts
		3	Soporta caída simultánea de 4 hosts
REQ 10	BALANCEO DE CARGA DE HOSTS EN UN CLUSTER	0	No permite balanceo de carga
		1	Permite balanceo de carga
REQ 11	BALANCEO DEL TRÁFICO DE RED	0	No realiza balanceo de tráfico de red
		1	Tiene 1 método de tráfico de red
		2	Tiene 2 métodos de tráfico de red
		3	Tiene 3 métodos de tráfico de red
REQ 12	ASIGNACIÓN DINÁMICA DE RECURSOS PARA MVs	0	No realiza asignación dinámica recur
		1	Realiza asignación dinámica recursos
REQ 13	CLONACIÓN DE MVs	0	No realiza clonación de MVs
		1	Realiza clonación de MVs apagadas
		2	Realiza clonación de MVs encendidas
REQ 14	CREACIÓN DE MVs A PARTIR DE TEMPLATES	0	No permite crear MVs a partir templat
		1	Permite crear MVs a partir templates
REQ 15	SOPORTE DE VLANs	0	No soporta VLANs
		1	Soporta VLANs
REQ 16	GESTIÓN DE BACKUPS	1	Permite respaldar toda la MV
		2	Permite respaldo toda la MV y archivo
REQ 17	ALMACENAMIENTO COMPARTIDO	0	No soporta almacenamiento compart
		1	Soporta almacenamiento compartido
REQ 18	CONTROL DE ACCESO A INFRAESTRUCTURA VIRT	0	No permite control de acceso
		1	Permite control de acceso
REQ 19	MONITOREO DE RECURSOS	1	No realiza monitoreo de recursos
		2	Realiza monitoreo de recursos

Calificación para cada Solución de Virtualización

En la Tabla 4.3 utilizando los requerimientos analizados en base a la Norma IEEE 830 y mediante la calificación asignada a cada requerimiento, se procede a la calificación respectiva para cada solución de virtualización.

Tabla 4.3. Selección de Software para Virtualización

REQUERIMIENTO	Citrix XenServer	Microsoft Windows Server 2008 Hiper-V	Parallels Server	Sun xVM Server	VMWare ESX Server
REQ1	2	2	2	2	2
REQ2	2	2	1	1	3
REQ3	1	0	1	1	1
REQ4	2	2	1	1	2
REQ5	1	1	1	1	1
REQ6	2	2	1	1	3
REQ7	2	2	2	1	2
REQ8	1	1	0	0	1
REQ9	2	1	0	1	3
REQ10	1	1	0	0	1
REQ11	1	1	0	0	3
REQ12	0	0	0	0	1
REQ13	0	1	0	0	2
REQ14	1	0	0	1	1
REQ15	1	1	0	0	1
REQ16	1	1	1	1	2
REQ17	1	1	0	0	1
REQ18	0	1	0	0	1
REQ19	1	1	0	1	1
TOTAL	**22**	**21**	**10**	**12**	**32**

Una vez realizada la calificación de cada requerimiento se concluye que el software que tiene mayor puntaje es *VMWare ESX Server*, la madurez y la estabilidad de ésta herramienta de virtualización hace que sea la más óptima para que sea utilizada en la Metodología y consecuentemente en la Implementación de la Infraestructura Virtual.

4.3. FASE DE ELABORACIÓN

4.3.1 RECTIFICACIONES A LOS ARTEFACTOS GENERADOS EN LA FASE DE INICIALIZACIÓN

Disciplina Modelado 1: Modelo Genérico para la Gestión del VNE

El modelo se centra en la distribución de los procesos en varios servidores, para administrar los objetos CIM se lo hace a través del

repositorio CIMOM, se implementa en un servidor; para administrar la plataforma XEN, en un segundo servidor y para administrar la plataforma VMWare, en un tercer servidor.

Para evaluar la viabilidad de éste modelo al generar varias MVs en una misma máquina física se produce un problema de saturación de HW, motivo por el cual el crecimiento debe repartirse en varios VNEs los mismos que distribuyan los procesos en varias MVs, para lo cual es indispensable que diversos servidores soporten Plataformas de Virtualización: Xen, VMWare, Cimom (figura 4.1).

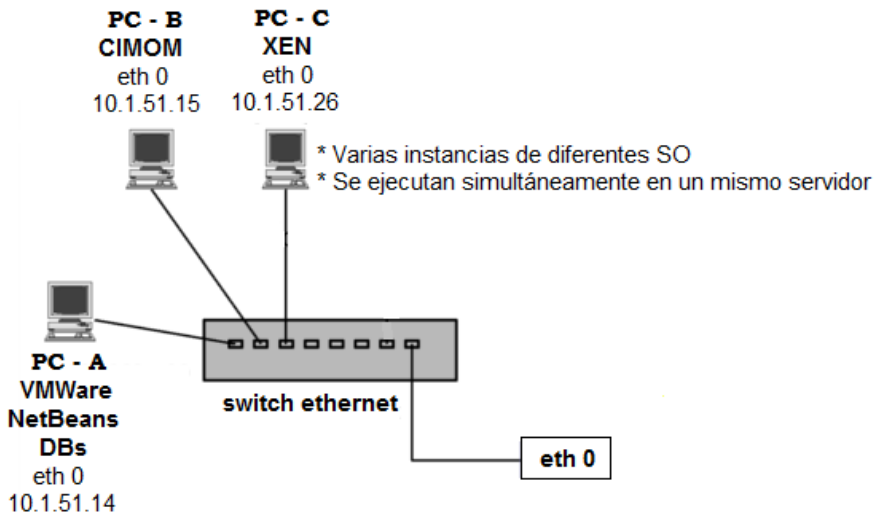

Figura 4.1. Servidores que soportan diversas Plataformas de Virtualización, Xen, VMWare, Cimom.

Las contribuciones principales del presente modelo son:

- Obtener una independencia de las plataformas: Xen, VMware, CIMOM.
- La administración independiente de los objetos del repositorio CIMOM; y las plataformas de virtualización, dando lugar a que estos funcionen en servidores que pueden estar ubicados en cualquier lugar.
- Facilitar la administración de las direcciones IP de las MVs, a través del repositorio CIMOM.

Disciplina Modelado 2: Modelo Genérico para la Gestión del VNE

El presente modelo se centra en la distribución de los procesos en varios servidores, así se muestra en la figura 4.2, que para administrar los objetos CIM a través del repositorio CIMOM, se lo implementa en un servidor; para administrar la plataforma XEN, en un segundo servidor y finalmente para administrar la plataforma VMWare, en un tercer servidor.

Para evaluar la viabilidad de este modelo, se ha construido un cliente WBem para la implementación de la API, que permite desplegar VNEs automáticamente, independientemente de la plataforma de virtualización utilizada. Los resultados del ensayo demostraron la eficacia de esta aplicación en la estación cliente.

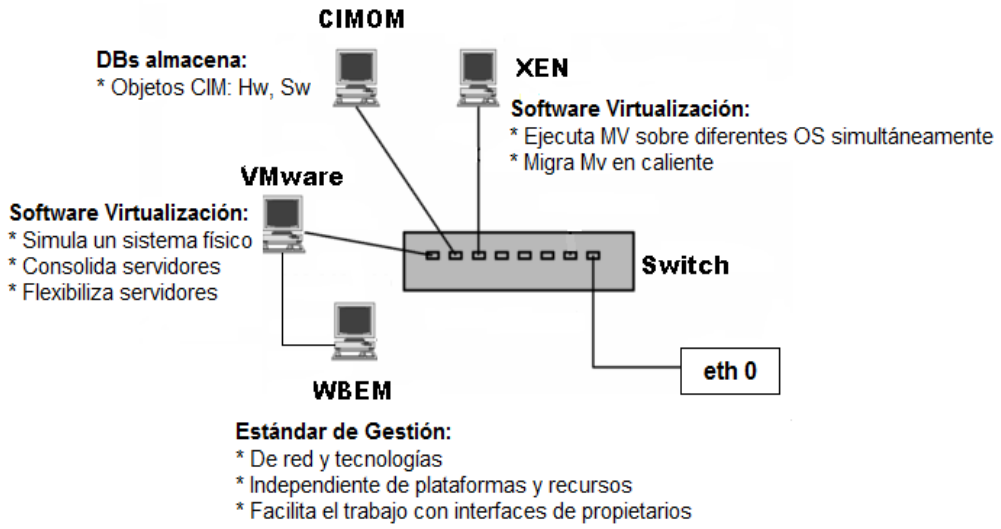

Figura 4.2. Gestión de los Servidores que soportan diversas Plataformas de Virtualización, Xen, VMWare, Cimom y cliente WBem.

Las contribuciones principales del presente modelo son:

- Obtener una independencia en la administración del repositorio CIMOM; y las plataformas de virtualización, dando lugar a que estos funcionen en servidores que pueden estar ubicados en cualquier lugar.
- Facilitar la administración de las direcciones IP de las máquinas virtuales, a través del repositorio CIMOM.
- Extender el margen de las operaciones a más servidores, mismos que pueden soportar otras plataformas de virtualización u otros repositorios CIMOM.

- Con la inclusión de un cliente WBem *(Web-Based Enterprise Management)* será posible modelar sistemas e implementar sus atributos y métodos.

Disciplina Modelado 3: Modelo Genérico para la Gestión del VNE

El presente Modelo de Información (figura 4.3) muestra la arquitectura distribuida para la gestión de entornos virtuales de red. Se compone de redes hosts físicos (sistema de computadoras) interconectados a través de enlaces lógicos (VPN, VLAN, entre otros) y enlaces físicos (redes LAN).

En particular, dentro de cada host físico es posible crear e implementar una red de topología denominada VNE, utilizando una plataforma de virtualización.

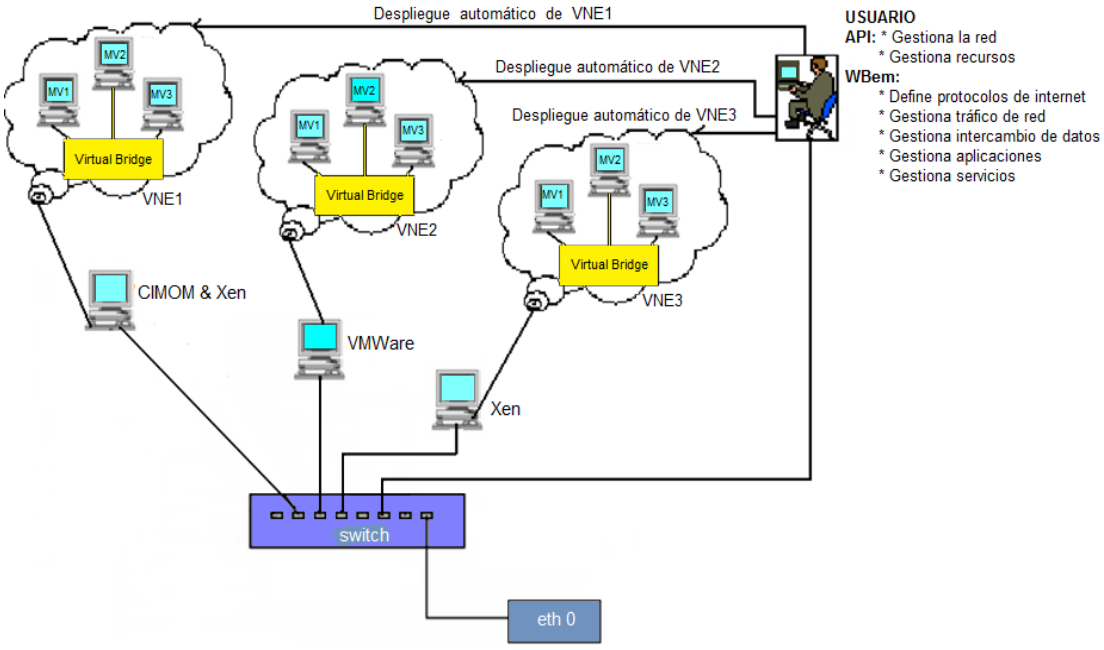

Figura 4.3. Arquitectura Distribuida del Modelo de Información para la Gestión de Entornos Virtuales de Red.

Las contribuciones principales del presente modelo son:

- Crear e implementar una red de topología denominada VNE dentro de cada host físico, utilizando una plataforma de virtualización, por ejemplo: Xen o VMware Server.

- Gestionar en cada uno de los VNEs la interconexión entre MVs a través de enlaces virtuales; garantizando de esta manera que cada MV tenga una dependencia en el respectivo host dentro del mismo VNE.
- Garantizar que una o más VNEs puedan ser desplegadas en la misma máquina cliente WBem.
- Obtener una independencia en la administración del repositorio CIMOM; y las plataformas de virtualización, dando lugar a que estos funcionen en servidores que pueden estar ubicados en cualquier lugar.
- Facilitar la administración de las direcciones IP de las MVs, a través del repositorio CIMOM.
- Extender el margen de las operaciones, sin perder la calidad a más servidores, los mismos que pueden soportar otras plataformas de virtualización u otros repositorios CIMOM.
- Parametrización de las direcciones IP; usuario y contraseña del host de virtualización, indicador de arranque de una plataforma, entre otros. Todos estos controlados desde el repositorio CIMOM.
- Para evaluar la viabilidad de este modelo y la obtención de los resultados pertinentes, se ha construido un cliente WBem el cual permite modelar sistemas y la implementación de sus atributos y métodos; es decir, se ha implementado una Interfaz de Aplicación *(API).*
- La API permitirá al SW, desplegar VNEs automáticamente, independientemente de la plataforma de virtualización; cuyos resultados producidos en la estación cliente WBem permitirá optimizar el despliegue y la administración de los VNEs en entornos distribuidos; *contribuyendo* a que los VNEs distribuyan los procesos en varias MVs.
- Por último, como trabajo pendiente, se integrará este modelo distribuido a través de un sistema de seguridades (control del acceso y pistas de auditoria al entorno de red virtual).

4.4. FASE DE CONSTRUCCIÓN

4.4.1. DISEÑO DE LA ARQUITECTURA DISTRIBUIDA

Con el fin de cumplir con los requisitos de diseño y con la arquitectura Distribuida, el esquema muestra la solución para obtener un modelo distribuido que permita gestionar el despliegue automático de VNE independientemente de la plataforma de virtualización subyacente, figura 4.4.

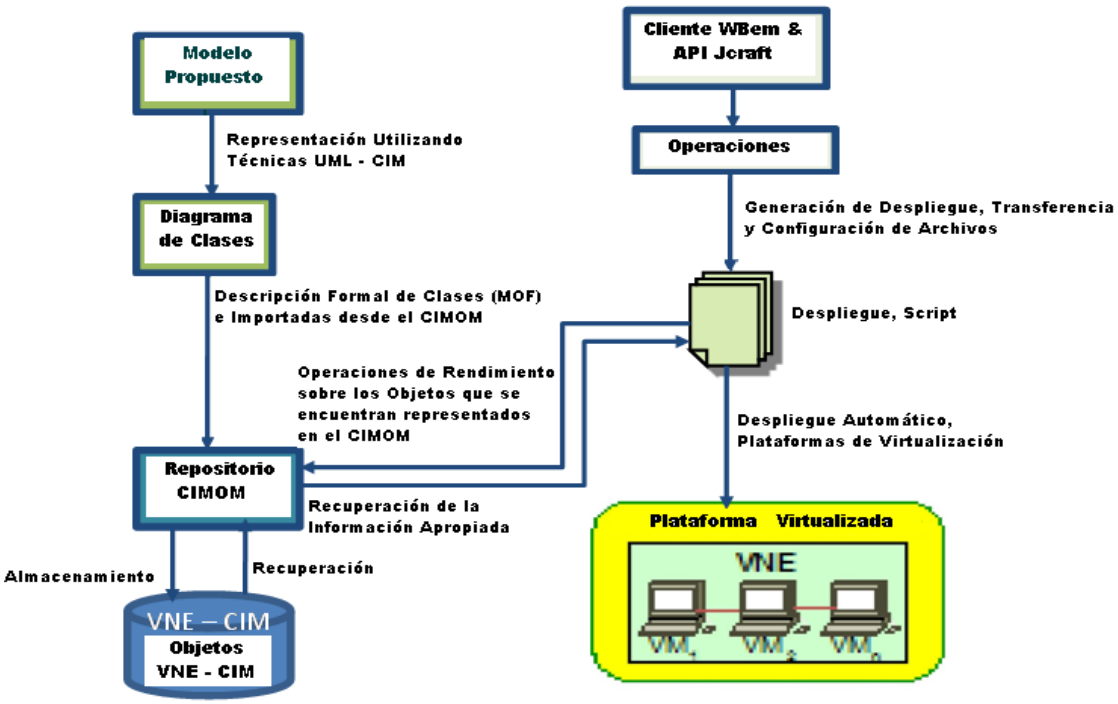

Figura 4.4. Modelo Distribuido que permite Gestionar el Despliegue Automático de VNE Independientemente de la Plataforma de Virtualización.

4.4.2. RECOPILACIÓN DE LOS REQUERIMIENTOS MEDIANTE UN PROTOTIPO INICIAL

Se suscitan varios problemas relacionados con el diseño y la gestión de VNEs, sin considerar los que se indican en el *Modelo Genérico para la Gestión de Entornos Virtuales de Red* [32], figura 4.5.

El diseño de VNEs es un proceso complejo, ya que al generar varias MVs en una misma máquina física se produce un problema de saturación de HW,

motivo por el cual el crecimiento debe repartirse en varios VNEs los mismos que distribuyan los procesos en varias MVs.

Ante este escenario, el presente trabajo se enfoca en aportar con nuevas soluciones aplicando criterios de computación distribuida para VNEs; para lo cual, se basa en el *Modelo Genérico para la Gestión de Entornos Virtuales de Red* [32], el mismo que se sustenta en los preceptos del Modelo de Información Común *(CIM)* del Grupo de Trabajo para Gestión Distribuida *(DMTF).*

Además, la creación de varios escenarios virtuales desplegados en diversos dominios de administración incrementa su complejidad, razón por la cual se requiere de un método de procesamiento de Computación Distribuida la cual proporcione una plataforma de ejecución segura, haciendo posible la coordinación de todos los recursos físicos y virtuales en topologías distribuidas.

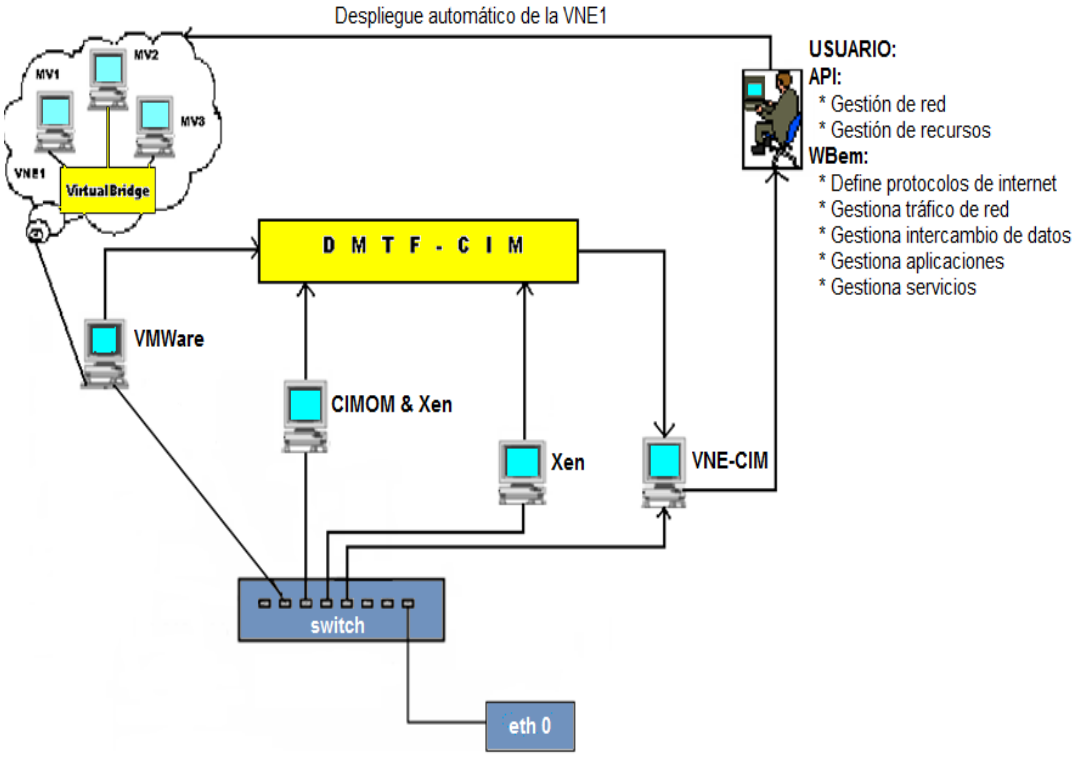

Figura 4.5. Modelo Genérico para la Gestión de Entornos Virtuales de Red

Ante esta situación, con el fin de poder administrar los VNEs, de una manera autónoma, es indispensable que el repositorio CIMOM trabaje centralizadamente en un servidor; y las plataformas de virtualización en otros, los cuales deben estar interconectados, dando lugar a que estos funcionen en servidores que pueden estar ubicados en cualquier lugar.

También, *se requiere resolver el problema* de la baja escalabilidad de un sistema mono-anfitrión, para lo cual se deberán asignar otros servidores que dispongan de la misma plataforma de virtualización.

Otro aspecto a considerar es que, conforme el VNE vaya haciéndose compleja se deberá disponer que varios repositorios CIMOM trabajen de forma distribuida.

Además, con el fin de estandarizar nombres y de conseguir una independencia entre la aplicación API-WBem y el repositorio CIMOM, se deberá establecer en este último una parametrización.

Finalmente, la Distribución física de un VNE se la deberá gestionar en diferentes host anfitriones, simulando un planificador de balanceo de carga, sin perder el control de ninguna MV.

En el marco de esta investigación, un VNE se define como un conjunto de dispositivos virtuales (sistemas finales, los routers y switches) conectado colectivamente en una topología desplegados en serie de uno o múltiples ambientes, que emula un sistema equivalente en el que el entorno se percibe como si fuera real [32].

Con el fin de ofrecer soluciones a estos problemas; este estudio trata de definir un modelo distribuido que caracterice VNEs IP. Por lo tanto, la cuestión a resolver es el diseño y la construcción de un modelo mínimo distribuido para describir VNEs a desplegarse en cualquier plataforma de virtualización subyacente. Para habilitar la interoperabilidad de la gestión de VNEs, se sustentará el modelo en base al conjunto de especificaciones que se derivan de los modelos normalizados aceptados como DMTF-CIM y otros.

4.4.3. DISEÑO DEL ENTORNO DE RED VIRTUAL

En la figura 4.6 se muestra la arquitectura abstracta en la que se basa el VNE[37]. Se compone de redes hosts físicos (sistema de computadoras) interconectados a través de enlaces lógicos (VPN, VLAN, entre otros) y enlaces físicos (por lo general, en el caso de redes de área local).

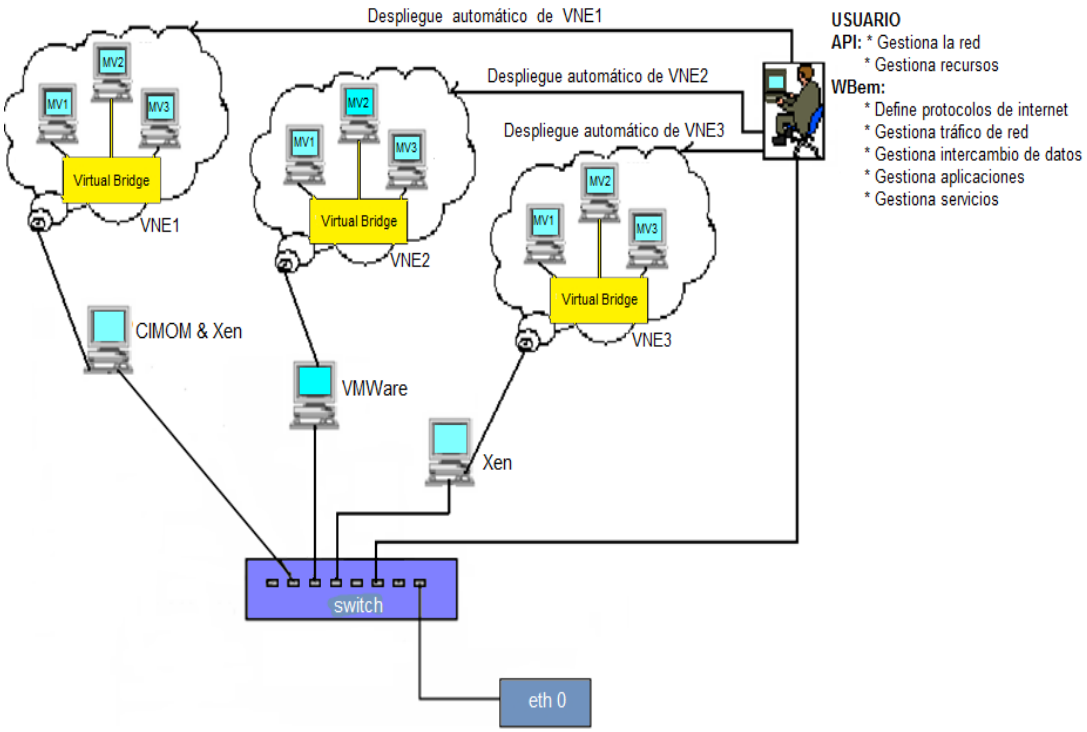

Figura 4.6. Arquitectura Abstracta en la que se basa el Entorno de Red Virtual

En particular, dentro de cada host físico es posible crear e implementar una red de topología denominada VNE, utilizando una plataforma de virtualización dado (por ejemplo, Xen o VMware Server). En cada uno de estos VNEs se muestra cómo las MVs están interconectados a través de conexiones virtuales. Cada MV tiene su propia tarjeta virtual de red (NIC) y la dirección IP establecida. Adicionalmente, esta arquitectura permite que una o más VNEs puedan ser desplegadas en la misma máquina cliente. Por último, cabe señalar que cada MV tiene una dependencia en el respectivo host dentro del mismo VNE.

4.4.4. DISEÑO DE DIAGRAMAS DE CLASES

Diseño del Modelo Distribuido VNE basado en CIM

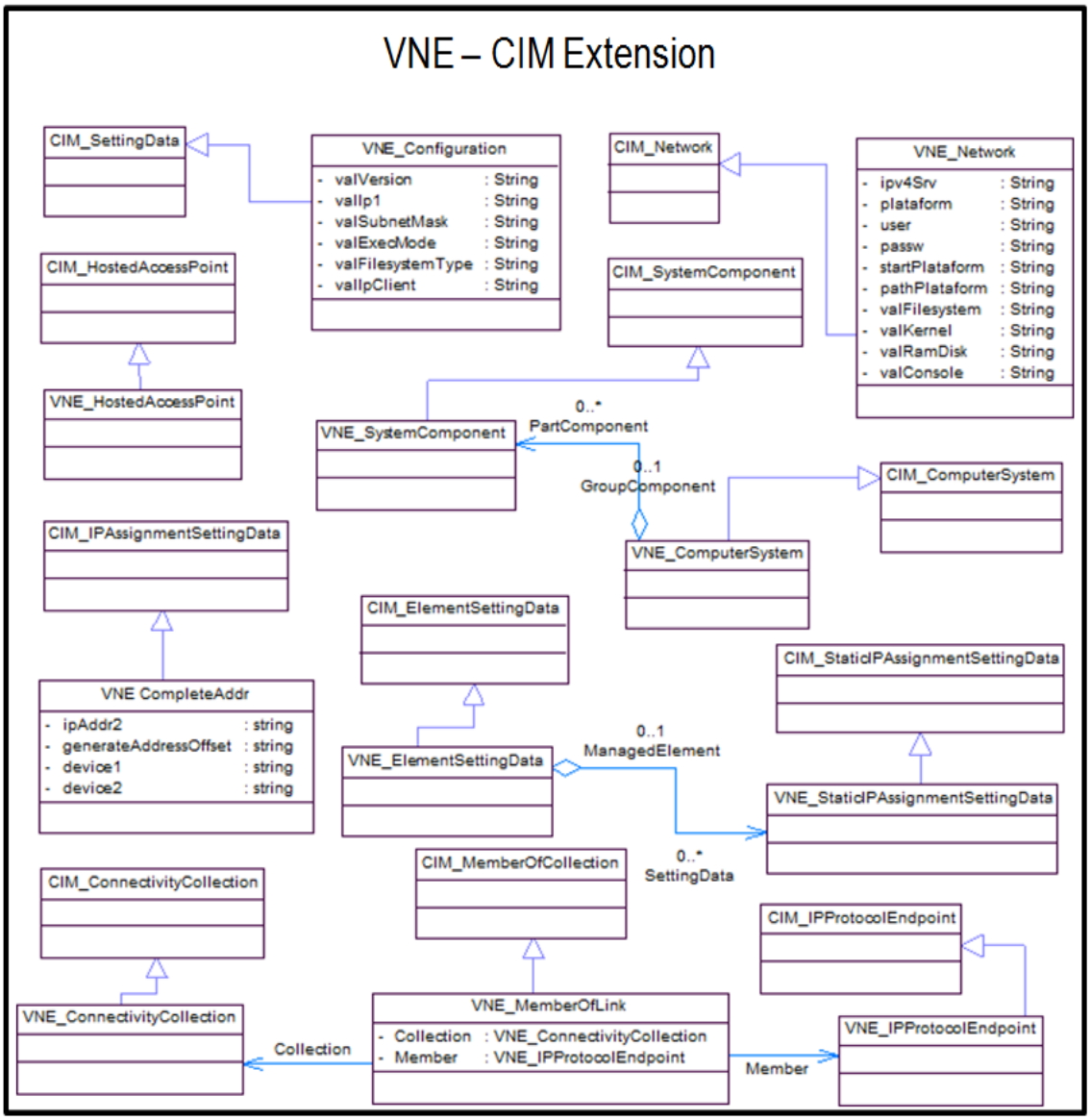

Figura 4.7. Modelo Distribuido VNE basado en CIM

El modelo se representa mediante técnicas de UML [41] - CIM que dio lugar a un diagrama de clases UML.

De acuerdo con este marco, una descripción formal de las nuevas clases VNE-CIM esquema de extensión (heredado de la CIM) se ha expresado en MOF.

A continuación, compila en un CIMOM para hacer una validación sintáctica y almacenarlos en el repositorio CIMOM. Esto se puede hacer en el mismo paso, ya que la mayoría de las implementaciones CIMOM pueden incluir también un compilador MOF para la lectura de los archivos MOF y la adición de las clases en el repositorio CIMOM, como se muestra en la figura 4.7.

Diseño del Modelo Cliente API WBem & Jcraft

La Aplicación Cliente API WBem, permite una interfaz para acceder y administrar clases y objetos, se gestionan operaciones en los objetos que se almacenan en el CIMOM & Jcraft, se genera e implementa de forma automática un VNE para diferentes plataformas de virtualización.

Una vez que se importa la VNE como un conjunto de instancias de MOF de las clases VNE-CIM, el administrador de la API permite ejecutar una operación de VNE (de clases de Java) para acceder al CIMOM, la generación y transferencia de los archivos de script a los respectivos servidores de virtualización con el fin de producir la operación deseada (por ejemplo, implementar la VNE). Por lo tanto, lo principal es recuperar las instancias de confirmación de la extensión del modelo almacenados en el CIMOM.

Para ello, se ha integrado una aplicación de código abierto llamado WBEM Services [8], para interactuar con el servidor CIMOM. WBEM Services es amplia y estable para la gestión de la aplicación CIM API.

Para la transferencia de archivos con el fin de levantar las máquinas virtuales y de configuración, además de la ejecución remota, se incorporan a la aplicación, los servicios de Jcraft [6]. El diagrama de clases UML se presenta en la figura 4.8.

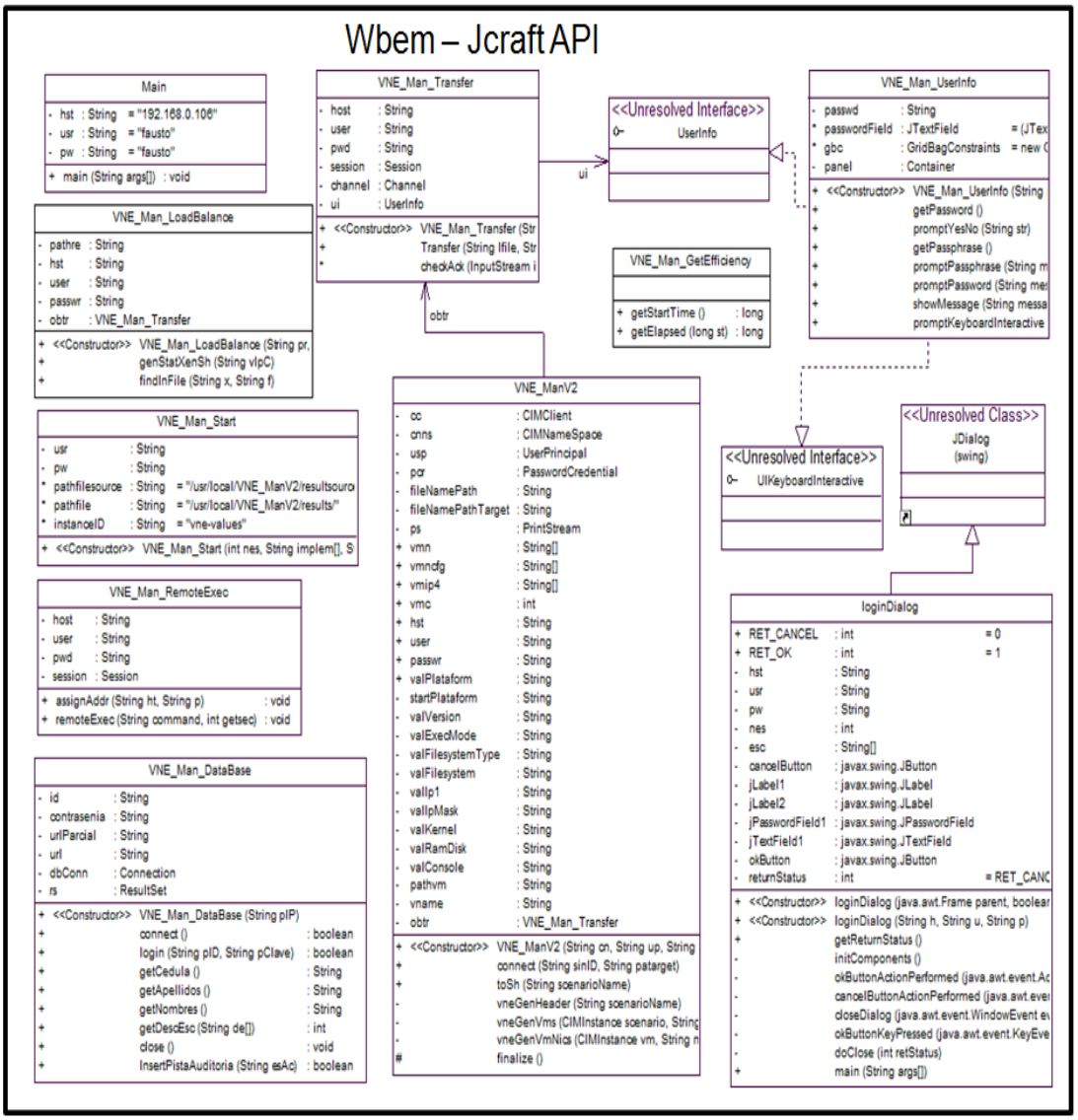

Figura 4.8. Diagrama de Clases API Wbem - Jcraft

4.4.5. DISEÑO DE LOS DIAGRAMAS DE CASOS DE USO

En la elaboración del diagrama de casos de uso, se ha considerado dos grandes conjuntos: el de los usuarios que accederán a los entornos virtualizados y el de las arquitecturas virtualizadas (escenarios).

En la figura 4.9 se puede apreciar que un usuario puede tener acceso a uno o varios escenarios.

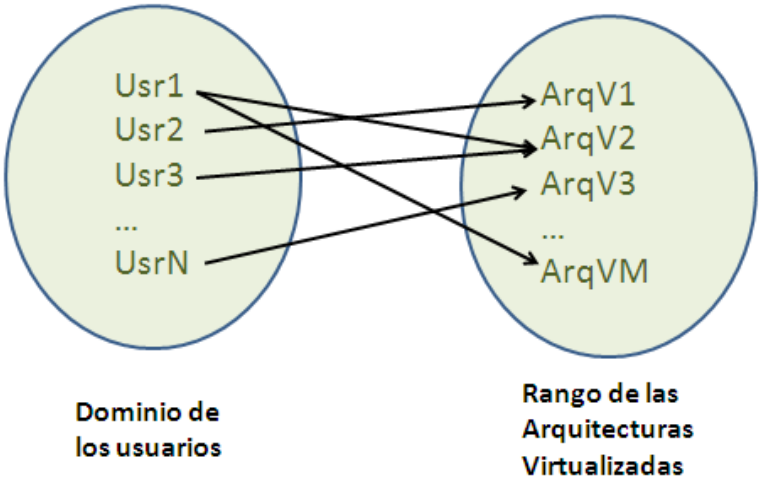

Figura 4.9. Diagrama de Casos de Uso

4.4.6. DISEÑO DE DIAGRAMAS DE SECUENCIA

La Fig. 4.10 representa el diagrama de secuencia de la API WBem - Jcraft, es decir, la interacción entre el usuario y el programa principal, con el objeto de validar el modelo. El funcionamiento general es el siguiente:

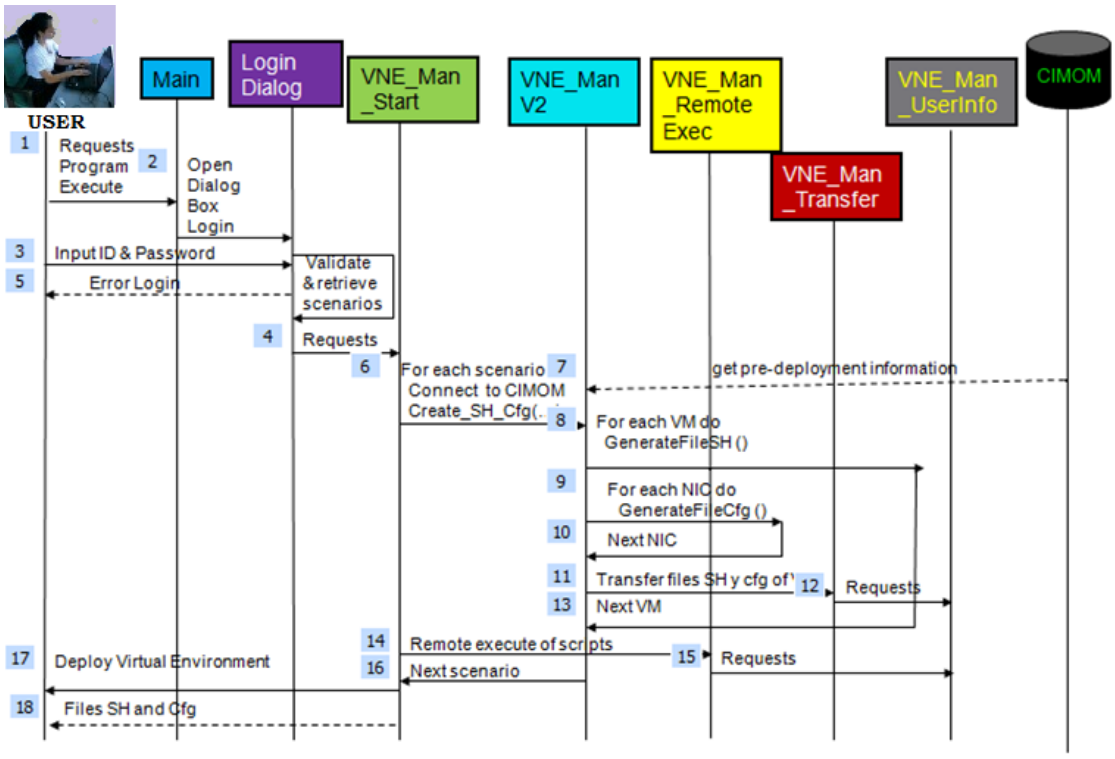

Figura 4.10. Diagrama de Secuencia de la API Wbem - Jcraft

En primer lugar, se solicita una operación de gestión (por ejemplo, el despliegue) (1). Este programa crea un objeto Main de clase Java para abrir

un cuadro de diálogo para ingresar al sistema (2). El usuario registra el ID y la contraseña (3).

El objeto LoginDialog valida los datos ingresados; si se encuentra autentificado, recupera de la BDs el escenario que se encuentra autorizado acceder (4).

Si el usuario se encuentra autorizado para simular el balanceo de carga (5) (ver figura 4.11), realiza el proceso respectivo y continua para levantar una MV en Xen (6). Activa un objeto de la clase VNE_Man_Start (7).

Mediante un ciclo para barrer los escenarios recuperados se conecta al repositorio CIMOM, y obtiene la información para crear los archivos SH y de configuración (8, 9, 10, 11, 15 y 18). Luego, transfiere los archivos SH y de configuración desde la aplicación API, hacia el respectivo servidor de Virtualización (13). Ejecuta remotamente los scripts para levantar el VNE (16). Despliega el entorno virtual en la estación cliente (19).

Gestión del programa principal:

- Crea todos los scripts de implementación y archivos de configuración.
- Lleva a cabo la transferencia de estos archivos, la ejecución remota y el despliegue automático de cada VNE.

Esto ha sido probado para Xen y las plataformas de VMware Server. Se ha utilizado tres host físicos.

La Fig. 4.11 representa el diagrama de secuencia del balanceo de carga, es decir, la interacción entre el usuario y el programa principal, con el objetivo de optimizar la asignación de MVs.

El funcionamiento es el siguiente:

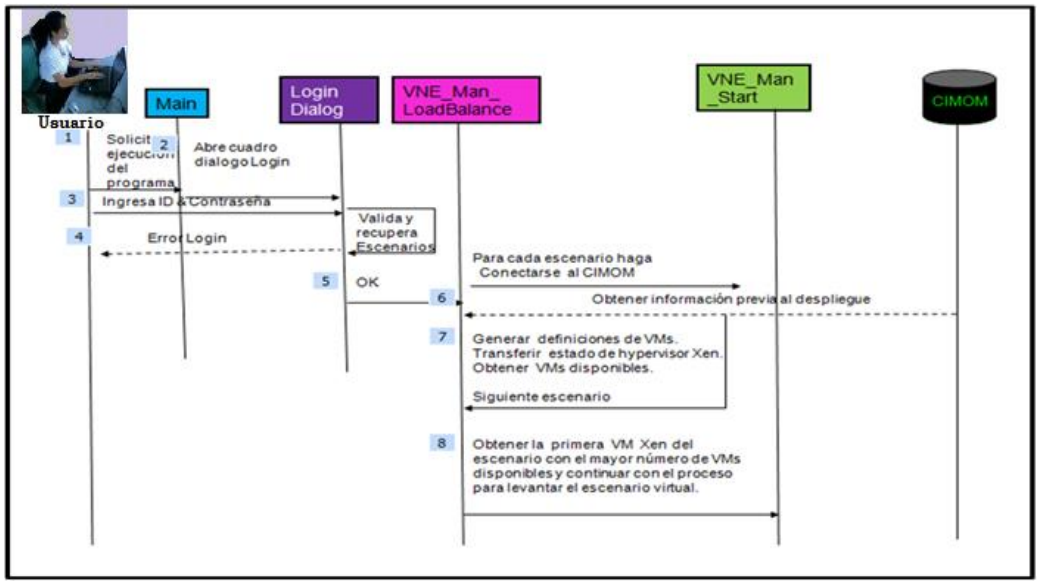

Figura 4.11. Diagrama de Secuencia de Balanceo de Carga

En primer lugar, se solicita una operación de gestión (por ejemplo, el despliegue) (1). Este programa crea un objeto Main clase Java para abrir un cuadro de diálogo para ingresar al sistema (2). El usuario registra el ID y la contraseña (3).

El objeto LoginDialog valida los datos ingresados, si no se encuentra autentificado, reporta el error de login (4); si se encuentra autentificado, recupera de la base de datos la condición de que no se encuentra autorizado acceder a balanceo de carga (5).

Si el usuario se encuentra autorizado para simular el balanceo de carga (6), a través de un ciclo para barrer los escenarios que tienen la mayor cantidad de MVs en Xen, se conecta al CIMOM y accede a las características de las MVs de los respectivos servidores; transfiere el estado de cada hipervisor Xen desde el servidor hacia el cliente API; verifica que la MV respectiva no se encuentre en el hipervisor a fin de ponerla en la lista de MVs disponibles (7).

Concluido el ciclo, asigna el escenario de una MV del servidor que dispone la mayor cantidad de MVs y continua con el proceso para levantar el VNE correspondiente (8).

4.4.7. DISEÑO DE DIAGRAMAS DE DESPLIEGUE.

Con el objeto de diseñar el hardware a utilizarse, se ha desarrollado el diagrama de despliegue de la figura 4.12.

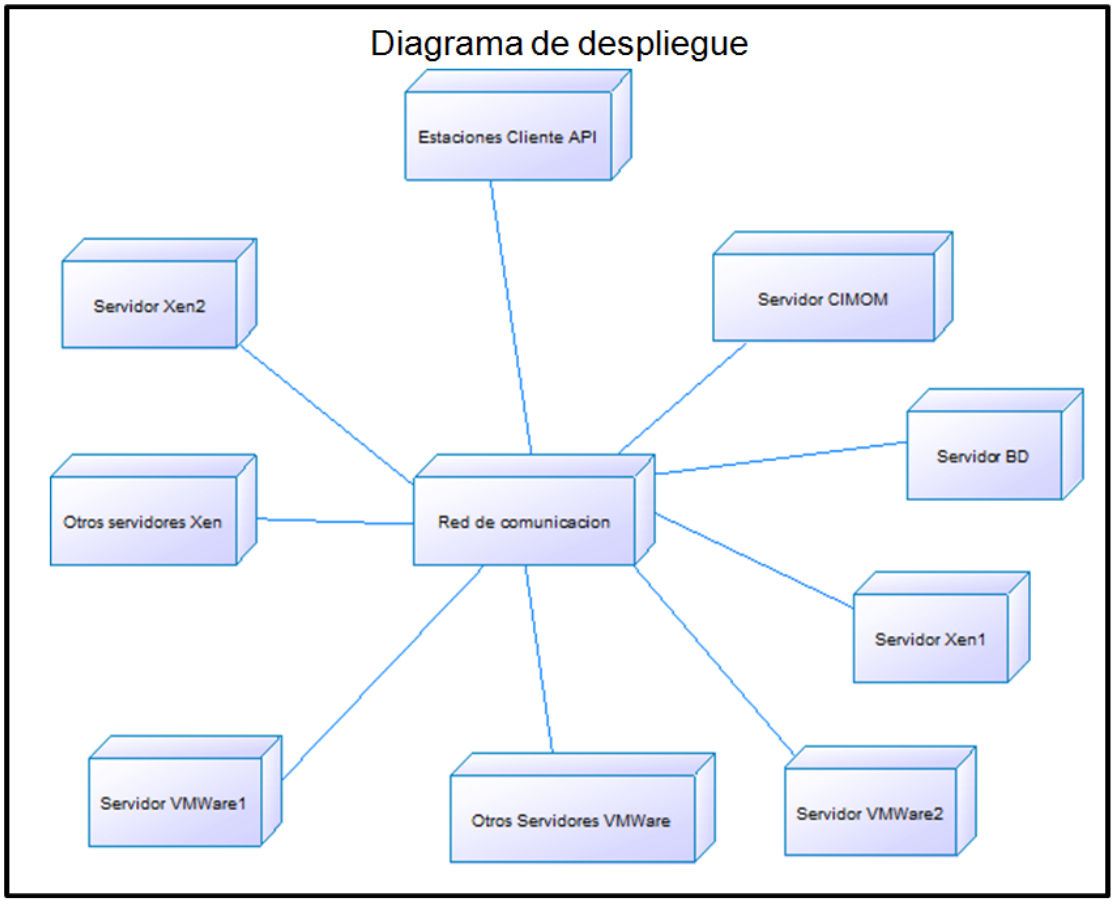

Figura 4.12. Diagrama de Despliegue

Como puede apreciarse en el presente diseño, el modelo distribuido soporta una configuración para n servidores de virtualización, un repositorio CIMOM y un servidor para la base de datos.

4.4.8. DISEÑO DEL DIAGRAMA CONCEPTUAL DE DATOS

La figura 4.13 muestra el modelo conceptual de datos, compuesto por las siguientes entidades:

Persona: permite almacenar los datos de los empleados: administradores de servidores, bases de datos, redes, operadores y usuarios, entre otros.

Usuario: facilita el registro de la identificación y contraseña de los usuarios, con el objeto de que permitan la autentificación al momento de ingresar al sistema.

Escenario: Almacena todos los escenarios que accesan al repositorio CIMOM.

Perfil: Permite registrar los grupos de usuarios de los entornos virtuales.

Pistas Auditoría: Registra las novedades presentadas por un usuario en una sesión.

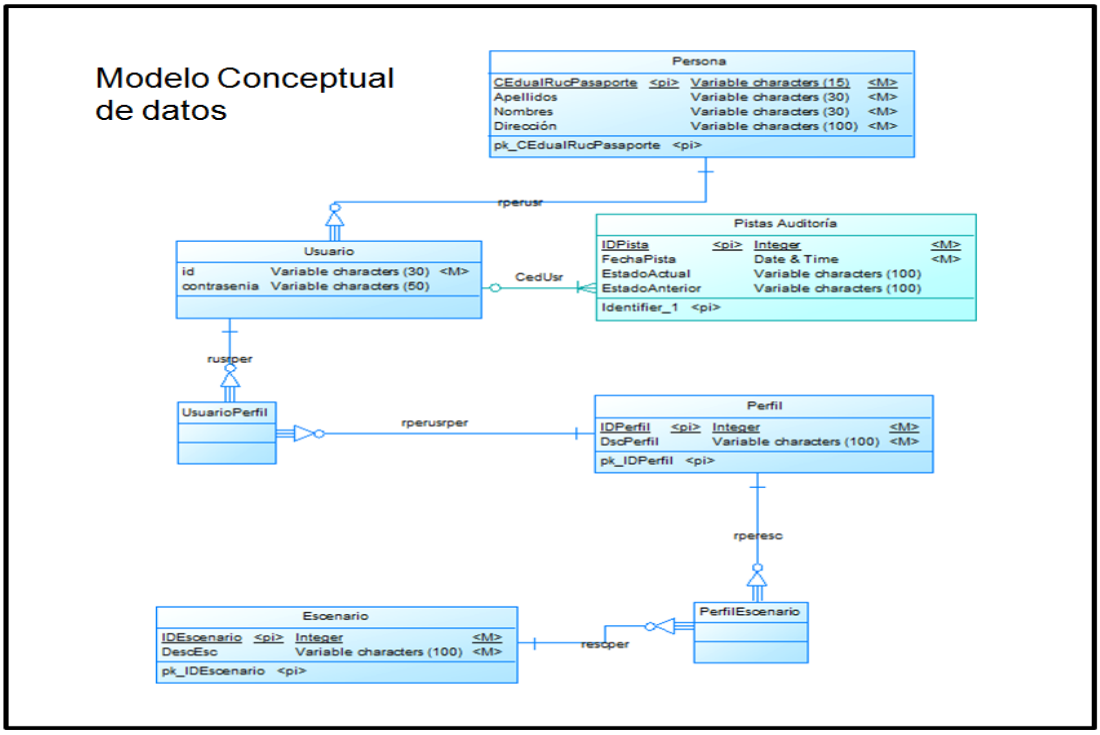

Figura 4.13. Diagrama Conceptual de Datos

Las entidades Usuario Perfil y Perfil Escenario permiten romper la relación muchos a muchos entre las que relacionan.

4.4.9. DISEÑO DEL DIAGRAMA FÍSICO DE DATOS

La herramienta CASE utilizada (Power Designer), permite generar automáticamente el diagrama físico de datos a partir del correspondiente diagrama conceptual, la figura 4.14 muestra el diagrama físico de datos.

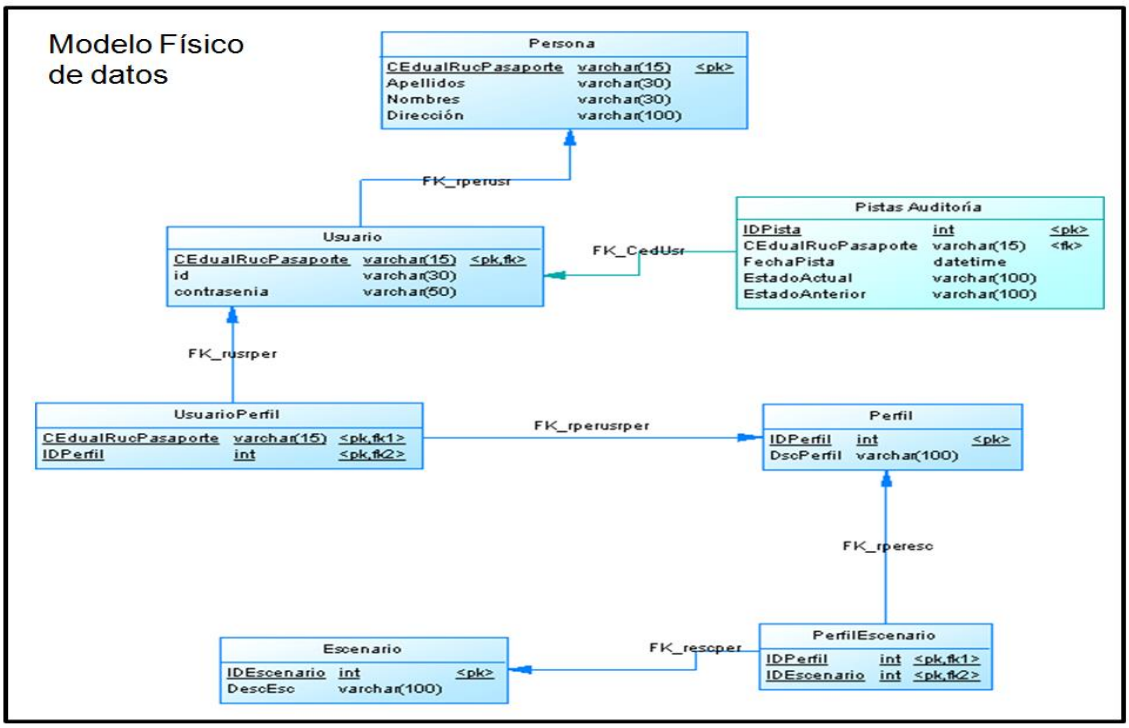

Figura 4.14. Diagrama Físico de Datos

4.4.10. GENERACIÓN DE LA BASE DE DATOS EN MySQL

En virtud de no disponer al momento de una herramienta CASE en el ambiente Ubuntu Linux, se tuvo que crear la base de datos, tablas y relaciones (claves principales y foráneas e índices) utilizando sentencias SQL, los datos se registraron empleando sentencias insert.

Para poder exportar la base de datos de un servidor a otro, se empleó el comando: mysqldump -u root -p dbseguraves > dbseav150511.sql

Mismo que produce el archivo dbseav150511.sql, cuyo contenido se muestra en el ***Anexo C***.

4.4.11. DESARROLLO DE Scripts Java, MOF, sh, cfg, vmx.

Scripts Java: Los scripts Java constituyen la parte medular del aplicativo cliente API, fue desarrollado en Netbeans 6.5 y ha sido estructurado en base a jerarquías de clases. Para mantener coherencia con los diagramas de secuencia expuestos anteriormente, se describen a continuación en el mismo orden, ***Anexo D***:

Scripts MOF: Los scripts MOF, son códigos fuente que permiten definir objetos para almacenarlos en el repositorio CIMOM tanto en estructuras como en datos (instancias). Las sentencias en lenguaje de MOF son compiladas por el compilador MOFCOMP. El compilador MOF es el responsable para interactuar con el CIMOM, en el ***Anexo E*** se describe cada una de ellas.

Scripts sh: Los scripts sh son archivos ejecutables bajo el ambiente Linux, para el desarrollo de la tesis, se los ha utilizado para generar, levantar y bajar el repositorio CIMOM, levantar MVs en Xen o VMWare, en el ***Anexo F*** se describe cada uno de ellos.

Scripts cfg: Los scripts cfg son archivos ejecutables bajo el ambiente Linux, que permiten levantar MVs en Xen, en el ***Anexo G*** se describe cada uno de ellos.

Scripts vmx: Los scripts vmx son archivos ejecutables bajo el ambiente Linux, que permiten levantar MVs en VMWare, en el ***Anexo H*** se describe cada uno de ellos.

4.4.12. DESARROLLO DE APLICATIVOS DE SEGURIDADES, PISTAS DE AUDITORIA Y VIRTUALIZACIÓN

Para desarrollar el aplicativo Wbem – Jcraft – API, se ha considerado el diseño de clases que muestra la figura 4.8, la aplicación ha sido desarrollada completamente en java y ha usado las libreriás wbem.jar para acceder a los objetos CIM, jsch-0.1.42.jar para integrar los procesos de transferencia de archivos y ejecución remota y mysql-connector-java-5.1.7-bin.jar para

acceder a la base de datos. Como resultado del desarrollo, se han generado los scripts que se describen en la sección 4.4.11.

4.5. FASE DE TRANSICIÓN

4.5.1. CORRECCIONES A LOS ARTEFACTOS DE LA FASE DE CONSTRUCCIÓN APROBADOS

Para dejar constancia de las correcciones a los artefactos en la fase de construcción, se han realizado las respectivas revisiones y se han suscrito formularios, con el Doctor Walter Fuertes, Director de Tesis, mismos que se detallan a continuación:

INSTITUCIÓN: ESCUELA POLITÉCNICA DEL EJÉRCITO – EXTENSIÓN LATACUNGA

POSTGRADO: PROYECTO DE GRADO

TEMA: SISTEMA DE CONTROL Y SEGURIDADES PARA EL PROYECTO AVES (APLICACIÓN DE TECNOLOGÍAS DE VIRTUALIZACIÓN PARA LA ESPE) EMPLEANDO LA METODOLOGÍA AUP

FASE DE INICIO.

NOVEDADES EN LA REVISION DEL PROTOTIPO. **FECHA:** 18/Febrero/2011 (10:30 a.m)

NVD-001	Soporte de XEN en otros Servidores
Versión	1.0
Autores	Ing. Fausto Meneses & Ing. Luis Guerra
Requisitos Asociados	Servidores de Virtualización. Plataformas de Virtualización: VMWare, Xen. Repositorio: CIMOM. Mecanismos de Seguridad. Tiempos de respuesta.
Descripción	Permitir el soporte de XEN en otros servidores a fin de balancear la carga de trabajo de las Máquinas Virtuales
Datos Específicos	Username. Password. Protocolos de red.
Alternativas	a) Mantener un control en el acceso a las máquinas virtuales del servidor host, cuando el nivel de saturación sea elevado. b) Considerar otro(s) servidor(es) que soporte(n) XEN y permita(n) satisfacer el requerimiento solicitado.
Solución	Considerar otro(s) servidor(es) que soporte(n) XEN y permita(n) satisfacer requerimiento solicitado.
Importancia	Vital
Estado	En investigación.
Estabilidad	Alta
Comentario	Se adopta la alternativa de solución (b), por ser la más eficiente.

INSTITUCIÓN: ESCUELA POLITÉCNICA DEL EJÉRCITO – SEDE LATACUNGA

POSTGRADO: PROYECTO DE GRADO

TEMA: SISTEMA DE CONTROL Y SEGURIDADES PARA EL PROYECTO AVES (APLICACIÓN DE TECNOLOGÍAS DE VIRTUALIZACIÓN PARA LA ESPE) EMPLEANDO LA METODOLOGÍA AUP

FASE DE INICIO.

NOVEDADES EN LA REVISION DEL PROYECTO. **FECHA:** 18/Febrero/2011 (10:30 a.m)

Firmas de Responsabilidad:

MAESTRANTE
Fausto Honorato Meneses Becerra

MAESTRANTE
Luis Alberto Guerra Cruz

DIRECTOR DE TESIS
Ing. Walter Fuertes, Ph.D.

INSTITUCIÓN: ESCUELA POLITÉCNICA DEL EJÉRCITO – EXTENSIÓN LATACUNGA

POSTGRADO: PROYECTO DE GRADO

TEMA: SISTEMA DE CONTROL Y SEGURIDADES PARA EL PROYECTO AVES (APLICACIÓN DE TECNOLOGÍAS DE VIRTUALIZACIÓN PARA LA ESPE) EMPLEANDO LA METODOLOGÍA AUP

FASE DE ELABORACIÓN Y CONSTRUCCIÓN.

NOVEDADES EN LA REVISION DEL PROYECTO. **FECHA: 27**/Mayo/2011 (14:30 p.m)

NVD-002	Soporte de XEN en otros Servidores e Incorporación del proceso de Balanceo de Carga
Versión	2.0
Autores	Ing. Fausto Meneses & Ing. Luis Guerra
Requisitos Asociados	Servidores de Virtualización. Plataformas de Virtualización: VMWare, Xen. Repositorio: CIMOM. Mecanismos de Seguridad. Tiempos de respuesta. Elaboración de un algoritmo que permita el balanceo de carga entre las máquinas virtuales.
Descripción	Permitir el Balanceo de trabajo de las máquinas virtuales, en el conjunto de servidores que soporten la plataforma de virtualización XEN.
Datos Específicos	Username. Password. Protocolos de red. Tiempos de respuesta: ejecución, despliegue y transferencia
Alternativas	c) Considerar otro(s) servidor(es) que soporte(n) XEN y permita(n) satisfacer el requerimiento solicitado. d) Mantener un control en el acceso a las máquinas virtuales del servidor host, cuando el nivel de saturación sea elevado. e) Alta disponibilidad al garantizar la continuidad de un determinado servicio independientemente de si ocurre algún tipo de fallo en el sistema.
Solución	Considerar otro(s) servidor(es) que soporte(n) XEN y permita(n) satisfacer el requerimiento del Balanceo de Carga con el fin de mejorar los tiempos de acceso y la confiabilidad de tráfico en la red distribuida.
Importancia	Vital
Estado	En Elaboración y Construcción.
Estabilidad	Alta
Comentario	Se adopta la alternativa de solución (a), (b) y (c) por ser las más eficientes.

INSTITUCIÓN: ESCUELA POLITÉCNICA DEL EJÉRCITO – SEDE LATACUNGA

POSTGRADO: PROYECTO DE GRADO

TEMA: SISTEMA DE CONTROL Y SEGURIDADES PARA EL PROYECTO AVES (APLICACIÓN DE TECNOLOGÍAS DE VIRTUALIZACIÓN PARA LA ESPE) EMPLEANDO LA METODOLOGÍA AUP

FASE DE ELABORACIÓN Y CONSTRUCCIÓN.

NOVEDADES EN LA REVISION DEL PROYECTO. **FECHA:** 27/Mayo/2011 (14:30 p.m)

Firmas de Responsabilidad:

MAESTRANTE
Fausto Honorato Meneses Becerra

MAESTRANTE
Luis Alberto Guerra Cruz

DIRECTOR DE TESIS
Ing. Walter Fuertes, Ph.D.

INSTITUCIÓN: ESCUELA POLITÉCNICA DEL EJÉRCITO – EXTENSIÓN LATACUNGA

POSTGRADO: PROYECTO DE GRADO

TEMA: SISTEMA DE CONTROL Y SEGURIDADES PARA EL PROYECTO AVES (APLICACIÓN DE TECNOLOGÍAS DE VIRTUALIZACIÓN PARA LA ESPE) EMPLEANDO LA METODOLOGÍA AUP

FASE DE TRANSICIÓN.

NOVEDADES EN LA REVISION DEL PROYECTO. **FECHA: 2**9/Agosto/2011 (11:30 a.m)

NVD-003	Soporte de XEN en otros Servidores e Incorporación del proceso de Balanceo de Carga y medición de los tiempos de respuesta en el momento de ejecución y de despliegue; así como, el análisis de la gráfica estadística correspondiente consumo de CPU y de memoria durante el despliegue de los escenarios virtuales.
Versión	3.0
Autores	Ing. Fausto Meneses & Ing. Luis Guerra
Requisitos Asociados	Servidores de Virtualización. Plataformas de Virtualización: VMWare, Xen. Repositorio: CIMOM. Mecanismos de Seguridad. Tiempos de respuesta: Ejecución, despliegue y transferencia. Rendimiento: CPU y memoria
Descripción	Permitir el Balanceo de Carga del trabajo de las máquinas virtuales que se encuentran ejecutándose en los servidores con plataformas de virtualización XEN, CIMOM y VMWARE.
Datos Específicos	Username. Password. Protocolos de red. Tiempos de respuesta: ejecución, despliegue y transferencia.
Alternativas	f) Considerar otro(s) servidor(es) que soporte(n) XEN y permita(n) satisfacer el requerimiento solicitado. g) Mantener un control en el acceso a las máquinas virtuales del servidor host, cuando el nivel de saturación sea elevado. h) Alta disponibilidad al garantizar la continuidad de un determinado servicio independientemente de si ocurre algún tipo de fallo en el sistema. i) Balanceo de carga al repartir de forma equitativa el conjunto global de peticiones que recibe un determinado servicio.
Solución	Balancear la carga entre las máquinas virtuales que se encuentran ejecutándose en los servidor(es) que soporte(n) XEN.
Importancia	Vital
Estado	En Transición.
Estabilidad	Alta
Comentario	Se adopta la alternativa de solución (a), (b), (c) y (d) por ser las más eficientes.

INSTITUCIÓN: ESCUELA POLITÉCNICA DEL EJÉRCITO – SEDE LATACUNGA

POSTGRADO: PROYECTO DE GRADO

TEMA: SISTEMA DE CONTROL Y SEGURIDADES PARA EL PROYECTO AVES (APLICACIÓN DE TECNOLOGÍAS DE VIRTUALIZACIÓN PARA LA ESPE) EMPLEANDO LA METODOLOGÍA AUP

FASE DE TRANSICIÓN.

NOVEDADES EN LA REVISION DEL PROYECTO. **FECHA: 2**9/Agosto/2011 (11:30 a.m)

Firmas de Responsabilidad:

MAESTRANTE
Fausto Honorato Meneses Becerra

MAESTRANTE
Luis Alberto Guerra Cruz

DIRECTOR DE TESIS
Ing. Walter Fuertes, Ph.D.

4.5.2. PRUEBAS DE CAJA BLANCA, DE CAJA NEGRA, DE INTEGRACIÓN Y DE RENDIMIENTO DEL PROYECTO AVES

Pruebas de caja blanca:

Se realizaron las pruebas de caja blanca, verificando que el código fuente de cada una de las clases[42], y que cumplan con los estándares de calidad ISO IEC 29119 [43].

Pruebas de caja negra:

Se realizaron las pruebas de caja negra, verificando que las entradas satisfagan los requerimientos para cada una de las clases, y que cumplan con los estándares de calidad ISO IEC 29119[43].

Pruebas de integración:

Se realizaron las pruebas de integración, verificando que el código fuente asi como las entradas satisfagan los requerimientos para cada una de las clases funcionando integradamente, y que cumplan con los estándares de calidad ISO IEC 29119[43].

Pruebas de rendimiento:

Se realizaron las pruebas de rendimiento[44], habiéndose observado que los tiempos de respuesta, consumo de CPU y de memoria, son bastante aceptables.

Para desarrollar las pruebas, se utilizaron los equipos que se describen en la tabla 3.2

Pruebas de rendimiento en el entorno distribuido virtualizado:

Se realizaron experimentos del modelo distribuido[44], la evaluación se describe en la siguiente sección.

4.5.3. EVALUACION DE RESULTADOS

Se evaluaron los tiempos de acceso a las MVs, lo cual se gestiona en la Tabla 4.4.

En la tabla 4.4, se puede observar, que las plataformas de virtualización Xen y VMware toman muy en cuenta las especificaciones del procesador y la memoria.

Tabla 4.4. Mediciones de los tiempos de acceso a las MVs del VNE.

MEDICIONES PARCIALES DE LOS TIEMPOS DE ACCESO						
PLATAFORMAS	**USUARIO1**		**USUARIO2**		**USUARIO3**	
	%CPU	**%MEMORY**	**%CPU**	**%MEMORY**	**%CPU**	**%MEMORY**
XEN	0.0	0.2	0.0	0.2	0.0	0.2
	0.0	0.1	0.0	0.1	0.0	0.1
	0.0	0.1	0.0	0.2	0.0	0.2
	0.0	0.1	0.0	0.1	0.0	0.1
	0.0	0.1	0.0	0.2	0.0	0.2
	0.0	0.0	0.0	0.1	0.0	0.1
	0.0	0.0	0.0	0.1	0.0	0.1
	0.0	0.0	0.0	0.1	0.0	0.1
	0.0	0.0	0.0	0.1	0.0	0.1
TOTAL	**0.0**	**0.6**	**0.0**	**1.2**	**0.0**	**1.2**
VMWARE	0.0	0.0	0.2	2.3	0.4	2.3
	0.0	0.0	0.0	0.1	0.1	0.1
	1.1	2.3	0.5	2.3	0.4	2.3
	0.0	1.0	0.0	0.1	0.0	0.1
	0.0	0.0	0.5	2.3	0.4	2.3
	0.1	1.1	0.0	1.0	0.0	1.0
	0.0	0.0	0.0	0.0	0.0	1.1
	10.1	6.6	0.0	0.0	4.5	5.4
	0.0	0.0	0.0	1.1	4.0	7.3
	0.0	0.1	4.3	7.3	3.8	6.9
	0.0	0.0	0.0	0.6	0.0	0.2
	0.0	0.0	4.6	6.8	0.0	0.1
	0.0	0.0	0.1	0.0	0.0	0.2
	0.0	0.0	0.0	0.1	0.0	0.1
TOTAL	**11.3**	**11.1**	**10.2**	**24.0**	**13.6**	**29.4**

Basados en el diseño y el soporte de las plataformas de virtualización Xen, VMWare y CIMOM se evalua el rendimiento de cada una de las MVs que conforman los VNEs distribuidos comparadas con las máquinas anfitrionas; además, es posible medir la velocidad del procesador, la memoria o el rendimiento del sistema operativo.

Tabla 4.5. Evaluación de los Tiempos de Acceso a las MVs.

USUARIO	HOST	MÁQUINA VIRTUAL	TIEMPO EJECUCIÓN API (ms)	TIEMPO DESPLIEGUE VNE (ms)	CONSUMO CPU	CONSUMO MEMORIA
user 1	10.1.18.71	vm3	16248	5091	0%	0,60%
	10.1.18.60	vmw3	27	40049	11,30%	11,10%
user 2	10.1.18.71	vm1	16277	5091	0%	1,20%
		vm2	17330	5091		
		vm3	17300	5091		
	10.1.18.60	vmw1	23	40049	10,20%	24,00%
		vmw2	49	40049		
		vmw3	163	40049		
user 3	10.1.18.71	vm1	16352	5091	0%	1,20%
		vm2	16361	5091		
		vm3	16373	5091		
	10.1.18.59	vm1	16196	5091	0%	0,10%
		vm2	16247	5091		
	10.1.18.60	vmw1	27	40049	13,60%	29,40%
		vmw2	34	40049		
		vmw3	35	40049		

La Tabla 4.5, resultados obtenidos al ejecutar el cliente API WBem & JCraft:

- *El tiempo de ejecución* es el tiempo que se tarda en encontrar la información en el CIMOM, generar el script de implementación, transferir los archivos a los respectivos hosts y ejecutar los scripts para arrancar los VNEs; mientras que,
- *El tiempo de despliegue* es el tiempo necesario para ejecutar la secuencia de comandos para desplegar un VNE.
- *El tiempo de ejecución* en VMware (vmw1, vmw2, vmw3) es alrededor del 0.2 % en milisegundos en comparación del tiempo de ejecución con MVs Xen (vm1, vm2, vm3); en contraste,
- *El tiempo de despliegue* con VMware es 8 veces mayor comparado con el tiempo de despliegue de un VNE en Xen.
- El escenario Xen se levanta remotamente en servidores distintos, mientras que VMWare en el mismo host.
- Las MVs con VMWare consumen 10 veces más la CPU y la memoria que las MVs con Xen.

Con las mediciones obtenidas con el soporte de las plataformas de virtualizaciòn Xen, VMWare y CIMOM se evalúa y grafica los tiempos de ejecución y tiempos de despliegue de cada una de las MVs que conforman los VNEs distribuidos comparadas con las máquinas anfitrionas, de acuerdo a lo que se observa en la fig. 4.15. Además, esta aplicación permite desplegar los VNEs en varios hosts de una manera automática a través de perfiles de usuario.

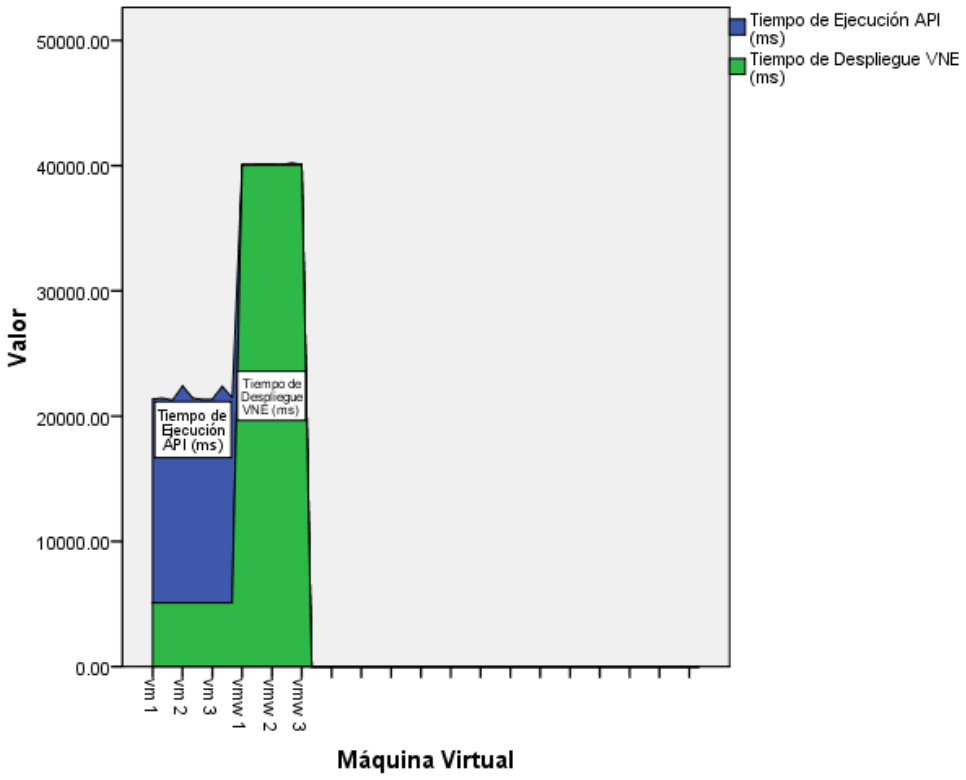

Figura 4.15. Tiempo de ejecución y de despliegue de los escenarios virtuales

De la figura 4.15, se observa que:

- El entorno virtual tiene la capacidad de abastecerse a sí mismo para aprovechar los recursos disponibles.
- Los tiempos de respuesta son más rápidos, mientras que,
- Los tiempos improductivos pueden llegar a ser casi nulos
- Los tiempos de respuesta rápidos mejoran la agilidad y el rendimiento.

Las plataformas de virtualización Xen, VMWare y CIMOM nos permiten evaluar y graficar los consumos de CPU y de memoria durante el despliegue del escenario virtual, de acuerdo a lo que se observa en la fig. 4.16.

Cabe destacar que nuestra implementación es personalizable, bastará con modificar los valores en el repositorio CIMOM para que por ejemplo se reasignen las direcciones IP a las MVs, o no se desplieguen escenarios.

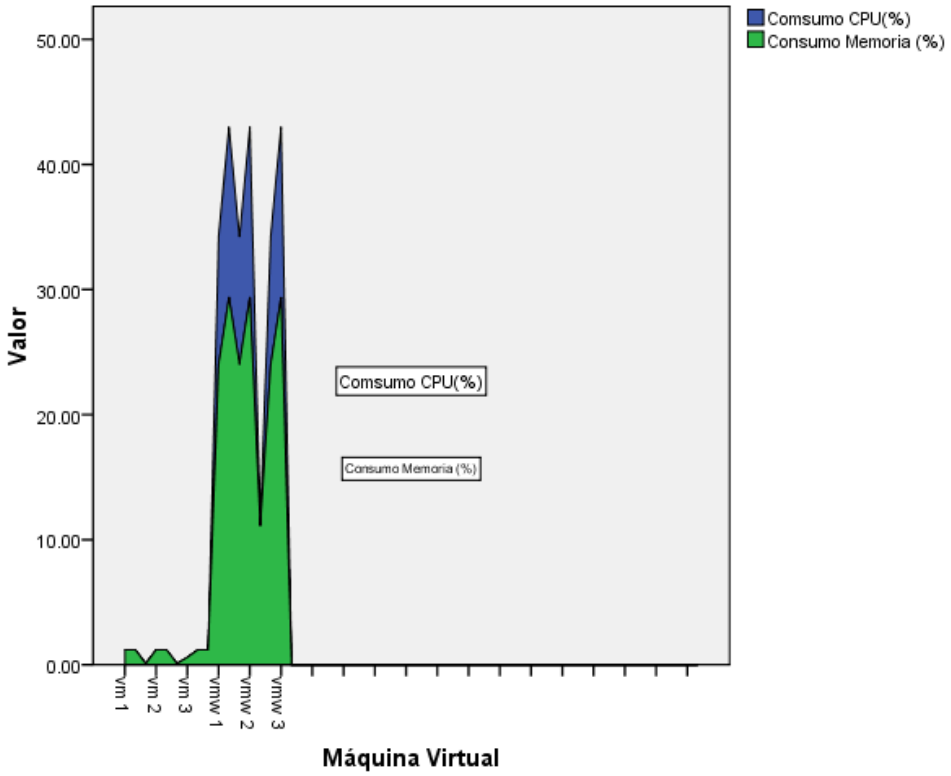

Figura 4.16. Consumo de CPU y de memoria durante el despliegue de los escenarios virtuales.

La implementación de varios VNEs garantiza las buenas prácticas de alta disponibilidad, la redundancia y la conexión de apoyo, ya que se pueden trasladar las cargas de trabajo a donde éstas sean más eficientes. Por lo tanto, la virtualización no sólo se centra en la efectividad (hacer lo correcto) sino también la eficiencia (hacer las cosas de una manera más rápida, más barata y más fiable).

Evaluación de resultados de Rendimiento Simulando Balanceo de Carga:

Se desarrollaron experimentos cuyos resultados se indican en la Tabla 4.6 y Tabla 4.7. En la Tabla 4.6 se ha incluido el *tiempo de transferencia*, que es el tiempo que tarda el hipervisor en asignar los recursos necesarios para levantar un VNE (la información del estado del Hipervisor Xen entre el servidor y el cliente API), en este caso son Xens[45].

Tabla 4.6. Resultados de las pruebas para balanceo de carga entre dos servidores**.**

USUARIO	HOST ASIGNADO	MÁQUINA VIRTUAL ASIGNADA	TIEMPO TRANSFERENCIA ESTADO HIPERVISOR XEN (ms)	TIEMPO EJECUCIÓN API (ms)	TIEMPO DE DESPLIEGUE VNE (ms)	CONSUMO CPU	CONSUMO MEMORIA
uxen1	10.1.18.71	vm1	2789	16287	5091	0%	0.00%
uxen2	10.1.18.59	vm1	1438	16287	5091	0.50%	0.00%
uxen3	10.1.18.71	vm2	1602	16321	5091	0%	0.00%
uxen4	10.1.18.59	vm2	1499	21207	5091	1.30%	0.00%
uxen5	10.1.18.71	vm3	3050	16364	5091	0.00%	0.00%
uxen6	10.1.18.59	vm3	1482	20240	5091	0.60%	0.00%

La Tabla 4.7 muestra los resultados al integrar una MV con VMWare; se demuestra que el *tiempo de transferencia* es cero, mientras que al usar Xen en un equipo remoto se demora un promedio de 1700 ms en el tiempo de ejecución.

Tabla 4.7. Resultados de las pruebas combinando balanceo de carga y escenarios distribuidos en Xen y VMWare**.**

USUARIO	HOST ASIGNADO	MÁQUINA VIRTUAL ASIGNADA	TIEMPO TRANSFERENCIA XEN (ms)	TIEMPO EJECUCIÓN API (ms)	TIEMPO DE DESPLIEGUE VNE (ms)	CONSUMO CPU	CONSUMO MEMORIA
uxen1	10.1.18.71	vm3		16468	5091	0.20%	0.00%
	10.1.18.60	vmw3		19	40049	13%	6.80%
uxen1	10.1.18.59	vm1	1442	16292	5091	0.85%	0.00%
uxen2	10.1.18.71	vm1	2390	16360	5091	0%	0.00%
uxen3	10.1.18.59	Vm2	1512	18025	5091	1.24%	0.00%
uxen4	10.1.18.71	vm2	1540	16494	5091	0%	0.00%
uxen5	10.1.18.59	vm3	1620	20120	5091	0.80%	0.00%

En este punto conviene resaltar que cuando se produce un pico de demanda de los recursos de las MVs, las peticiones totales de recursos pueden exceder los recursos disponibles en un host. Es en este momento que mediante un algoritmo de balanceo de carga que proporciona funcionalidad de redundancia activa – pasiva, se logre que automáticamente se recoloquen las MVs en hosts en los que los recursos están disponibles balanceando continuamente la capacidad y asegurando que cada MV tenga acceso a los recursos apropiados en tiempo real.

El balanceo de carga se mantiene gracias al algoritmo que divide de la manera más equitativa el despliegue de las máquinas virtuales, para evitar la saturación del equipo centralizado en el que se distribuye el VNE.

Las Figuras 4.17 y 4.18 muestran a continuación los resultados obtenidos durante las pruebas realizadas.

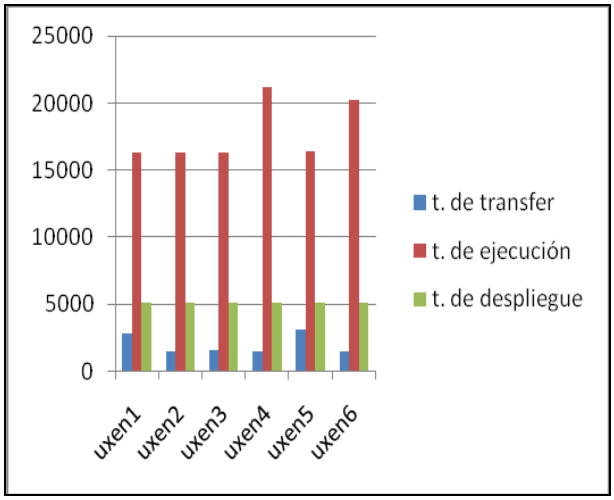

Figura 4.17. Tiempo de transferencia ejecución y despliegue de los VNEs

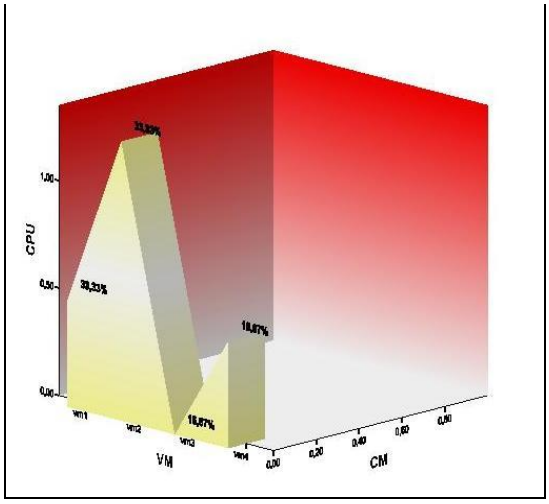

Figura 4.18. Consumo de CPU durante el despliegue

Además, los porcentajes de consumo de CPU, de memoria y la gestión de tráfico IP han demostrado que el balanceo dinámico de cargas de trabajo entre máquinas virtuales es un requisito indispensable, no solo para conseguir la mejor utilización del hardware, sino también la virtualización de las aplicaciones y la optimización del uso de los recursos.

4.5.4. PAPERS PUBLICADOS COMO UN APORTE TÉCNICO - CIENTÍFICO

En los papers se consideraron:

- El estudio de bases teóricas investigadas sobre la Seguridad de datos, e
- Información en Aplicaciones de Tecnologías de Virtualización;
- El soporte de las Plataformas VMWare y Xen con Ubuntu (LINUX), NetBeans, MySQL;
- La recopilación de información de casos exitosos en Seguridad de datos, e
- Información en Aplicaciones virtuales generados por las Mejores Prácticas sobre Plataformas Virtualizadas.

Se desarrollaron y publicaron dos artículos científicos, el primero de carácter nacional, que se describe a continuación:

F. Meneses, W. Fuertes, L. Guerra, J. E. López de Vergara y H. Aules, “Modelo Distribuido para la Gestión de Entornos Virtuales de Red”, Publicado en las memorias del VI Congreso de Ciencia y Tecnología ESPE 2011, realizado en Sangolquí, Ecuador del 8 al 10 de junio de 2011. ISBN 1390-4663 [46].

Y el segundo de carácter internacional, que se describe a continuación:

Fausto Meneses Becerra, Walter Marcelo Fuertes Díaz, Luis Alberto Guerra Cruz, “Modelo Distribuido para la Gestión de Entornos Virtuales de Red Simulando Balanceo de Carga”, Publicado en las memorias del XVI Congreso Internacional de Contaduría, Administración e Informática Universidad Nacional Autónoma de Mexico (UNAM), realizado en la Ciudad de México, México del 5 al 7 de octubre de 2011. UNAM ISBN 978-607-02-2548-2 [47].

4.5.5. TESIS DE GRADO

Presente documento que basa su perfil de investigación en un "Sistema de control de seguridades para el proyecto AVES (Aplicación de Tecnologías de Virtualización para la ESPE), empleando la metodología AUP".

En el primer capítulo se analizaron: el estado del arte sobre virtualización con respecto a: enfoques, técnicas, plataformas; seguridades en entornos

virtualizados en base a riesgos y problemas, planteamientos de estrategias para abordar los riesgos; mediante una arquitectura basamos el Modelo de Información; el soporte que brindan las plataformas de virtualizaciòn Xen, VMware y CIMOM; el balanceo de carga; y la Metodología AUP.

En el segundo capítulo se determinaron la Infraestructura de Virtualización y los enfoques del Modelado de Información. En el tercer capítulo se determinaron las características de la plataforma que soporten herramientas de virtualización.

El cuarto capítulo comprende el diseño e implementación, con el soporte de la Metodología AUP se generó: el Modelamiento del Sistema y el desarrollo de las diversas fases del proyecto; la Norma IEEE 830(ERS) nos ayudó en la Especificación de los Requerimientos del Software de virtualización.

El quinto capítulo comprende la implementación de un mecanismo de control y seguridades para facilitar el acceso de los usuarios al proyecto AVES, manteniendo un registro de pistas de auditoría. En el sexto capítulo enfocamos ciertas conclusiones y recomendaciones que consideramos importantes, e incluímos la Bibliografía utilizada durante el desarrollo, un acrónimo y el anexo.

4.6. CONCLUSIÓN

La Metodología AUP se ha acoplado perfectamente en cada una de sus fases, junto al cumplimiento de los requerirmientos funcionales del software de virtualización de acuerdo a la norma IEEE 830; lo que permitirá un diseño de la recopilación de requerimientos mediante un prototipo, los diagramas correspondientes; la generación de la BD en MySQL, el desarrollo de scripts java y el desarrollo de aplicativos de seguridades, pistas de auditoria e implementación de una arquitectura del entorno virtual de red distribuida validado mediante pruebas de caja blanca, caja negra, integridad y rendimiento; así, como el balanceo de carga de acuerdo a las normas y estándares que se ha incorporado al modelo del VNE.

CAPÍTULO V

IMPLEMENTACIÓN DE UN MECANISMO DE CONTROL Y SEGURIDADES PARA FACILITAR EL ACCESO DE LOS USUARIOS AL PROYECTO AVES

5.1. INTRODUCCIÓN

El presente capítulo comprende la implementación de un mecanismo de control y seguridades para facilitar el acceso de los usuarios al proyecto AVES, su implementación permitirá garantizar la integridad, confidencialidad y disponibilidad de la información del sistema.

La clave para alcanzar un exitoso laboratorio de VNE, se centra en el cumplimiento correcto de todos los controles normales de seguridad; tales como: gestionando un registro de pistas de auditoría [48], garantizando que no se desconfiguren los laboratorios virtuales, considerando actividades previas a la puesta en marcha del proyecto y la ejecución exitosa del aplicativo.

5.2. MANTENER UN REGISTRO DE PISTAS DE AUDITORÍA

- El Registro de pistas de auditoría es un archivo o base de datos.
- En la base de datos, el sistema lleva la cuenta de las operaciones realizadas en cada uno de los entornos de red virtual gestionada por los usuarios.

5.3. GARANTIZAR QUE NO SE DESCONFIGUREN LOS LABORATORIOS DE REDES DE LA ESPE

Más allá de la virtualización aplicada, todos los controles normales de seguridad de información deben cumplirse correctamente, por lo que, la clave para alcanzar un exitoso laboratorio de VNE, se deberá concentrar en:

- Aplicar los mismos controles TI que se utilizan en la infraestructura física, pero considerando las particularidades del VNE, que en esencia actúan

como mecanismos que mantienen el balance entre brindar un servicio de calidad y la gestión de los riesgos.

- Tener claro, la flexibilidad que brinda la virtualización al generar un alto valor para quienes la utilizan, considerar las nuevas dificultades; analizar los nuevos conceptos, captar las nuevas funcionalidades e innovar los procesos.

- Posicionar en el lugar adecuado del VNE, los controles de acceso, antivirus, las restricciones contra servicios innecesarios e inseguros y demás factores.

- Cumplir con normas de seguridad que incluyen mantener una red segura, utilizar firewall de protección de datos, no usar los parámetros de seguridad que los proveedores dan por defecto, cifrar las transmisiones en redes, aplicar un programa de gestión de vulnerabilidades, utilizar antivirus efectivos, utilizar aplicaciones seguras, implementar medidas de control de acceso, asignar identificaciones únicas a cada usuario, supervisar las redes, y mantener una Política de Seguridad de la Información.

5.4. ACTIVIDADES PREVIAS A LA PUESTA EN MARCHA DEL PROYECTO

Con el objeto de ejemplificar esta y la siguiente fase, se ha considerado un grupo de dos servidores, interconectados entre sí y funcionando bajo el SO Ubuntu Linux.

En el primer servidor se encuentra instalado la plataforma de virtualización Xen con escenarios que permiten simular el balanceo de carga.

En el segundo servidor se tiene la plataforma de virtualizacion VMWare, el repositorio CIMOM, el aplicativo Wbem - JCraft - API, funcionando bajo Java Netbeans, la base de datos en MySql.

A continuación presentamos las actividades previas a la puesta en marcha:

a) Configuración de las direcciones IP: 192.168.0.101 para el servidor 1 y 192.168.0.102 para el servidor 2.

b) Inicialización del repositorio CIMOM y asignación de algunos valores: Ingresado en la ruta /usr/local/VNE_ManV2/scripts/ ejecutar el comando sh StartCimom.sh.

El repositorio responde presentando la pantalla del login; a continuación se deben llenar los campos de la indicada pantalla:

url: http://192.168.0.102:5988//root/cimv2184

id: fausto

password: *******

A lo que el sistema responde presentando la pantalla de la figura 5.1.

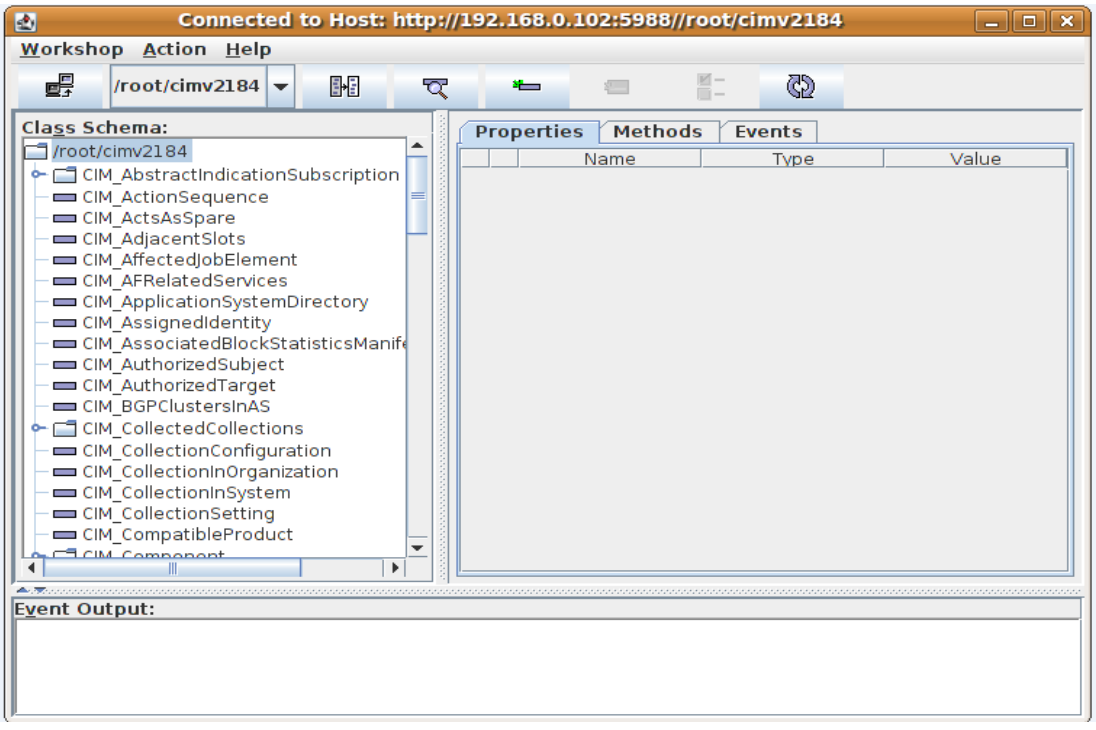

Figura 5.1. Jerarquia de clases del repositorio CIMOM

En la ventana de la opción Action/Find Class… registrar VNE_Configuration. CIMOM responde con la pantalla de la figura 5.2.

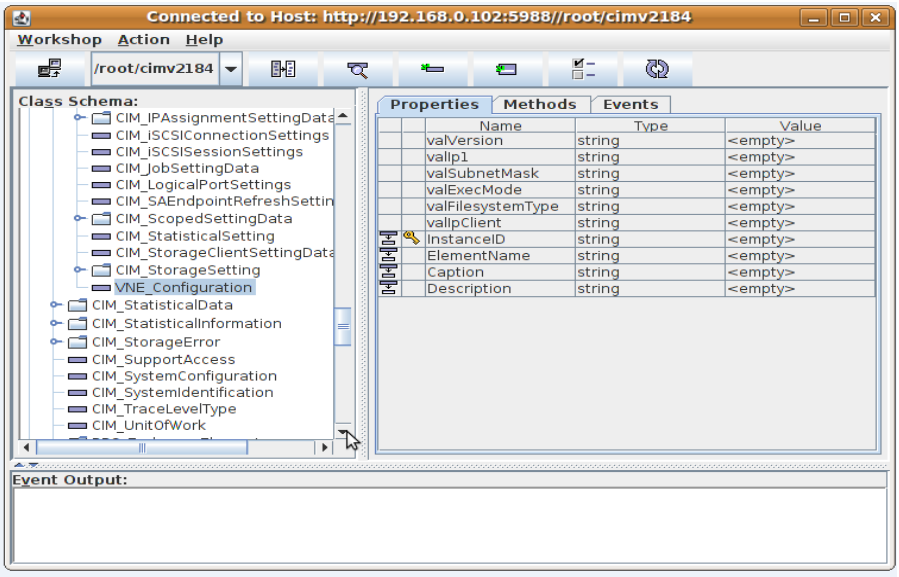

Figura 5.2. Clase VNE_Configuration

Al seleccionar la opción Action/Show Instances… CIMOM responde con la pantalla de la figura 5.3.

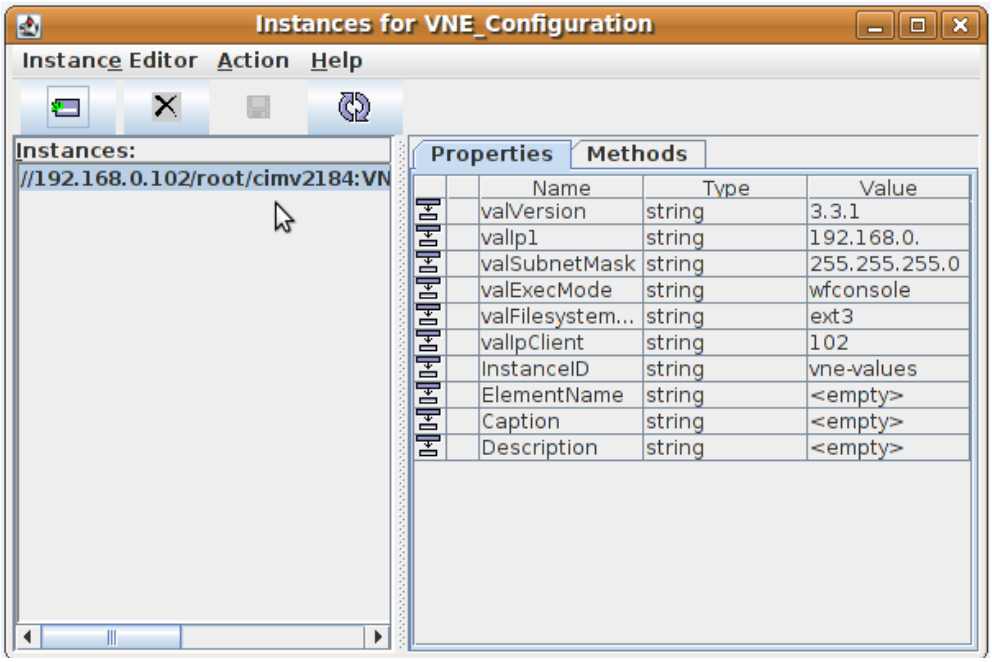

Figura 5.3. Instancia de la Clase VNE_Configuration

Pese a que la mayor parte de los atributos fueron generados al compilar el CIMOM, hay algunos que necesitan modificarse, tal es el caso de valIp1 que debe tener como valor la IP de la subred y valIpClient que debe tener el último octeto de la IP de la aplicación cliente. En la ventana de la opción

Action/Find Class… registrar VNE_Network;. CIMOM coresponde con la pantalla de la figura 5.4.

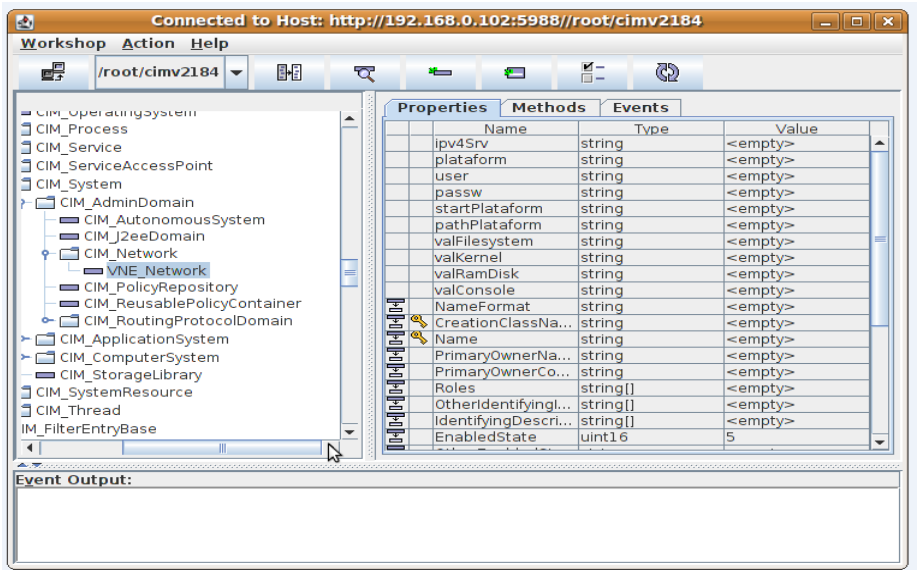

Figura 5.4. Clase VNE_Network

Al seleccionar la opción Action/Show Instances… CIMOM responde con la pantalla de la figura 5.5.

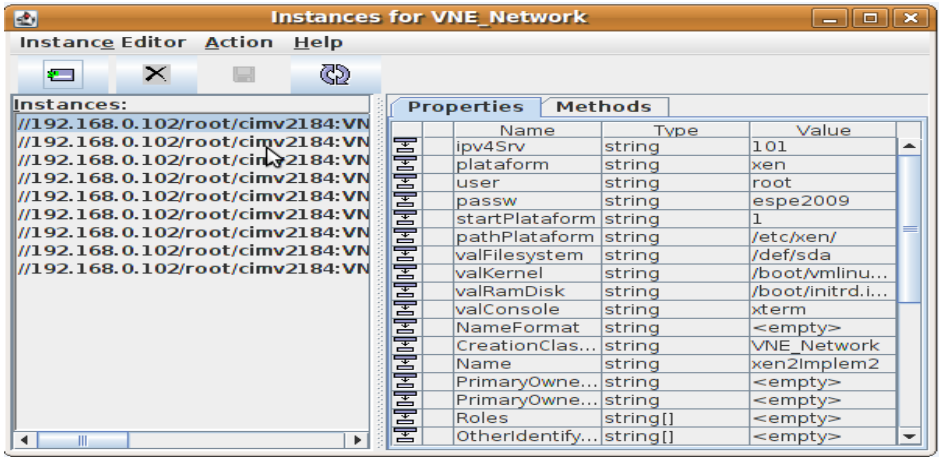

Figura 5.5. Instancias de la Clase VNE_Network

De igual manera de lo que ocurrió con la clase VNE_Configuration, se debe modificar para cada instancia el valor del atributo ipv4Srv, mismo que corresponde al último octeto de la IP de cada escenario.

c) Dentro del entorno Netbeans y el aplicativo, acceder a la clase Main y asignar en la variable hst, el valor de la IP del servidor que tiene el CIMOM.

5.5. PUESTA EN MARCHA DEL PROYECTO

Al ejecutar el aplicativo, se presenta la pantalla de la figura 5.6, en la que se registra el usuario que se indica.

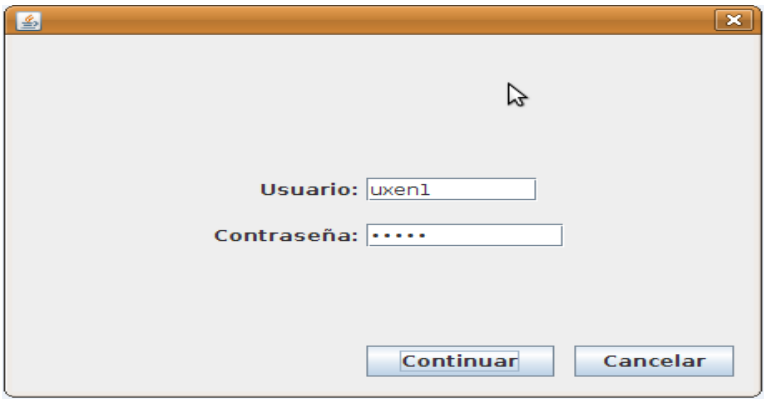

Figura 5.6. Login para un usuario de balanceo de carga.

Luego de unos instantes, el aplicativo presenta la pantalla de la figura 5.7

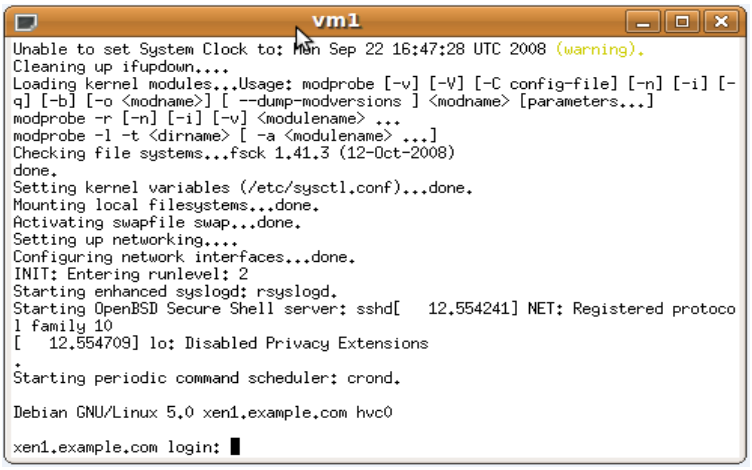

Figura 5.7. Máquina virtual 1 del escenario xenImplem3

Al registrar el usuario que se indica.en la figura 5.8

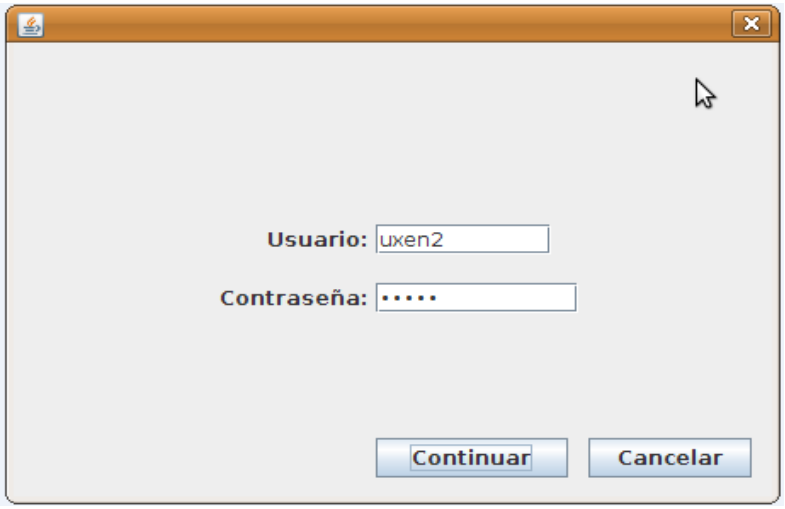

Figura 5.8. Login para un usuario de balanceo de carga.

Luego de unos instantes, el aplicativo presenta la pantalla de la figura 5.9

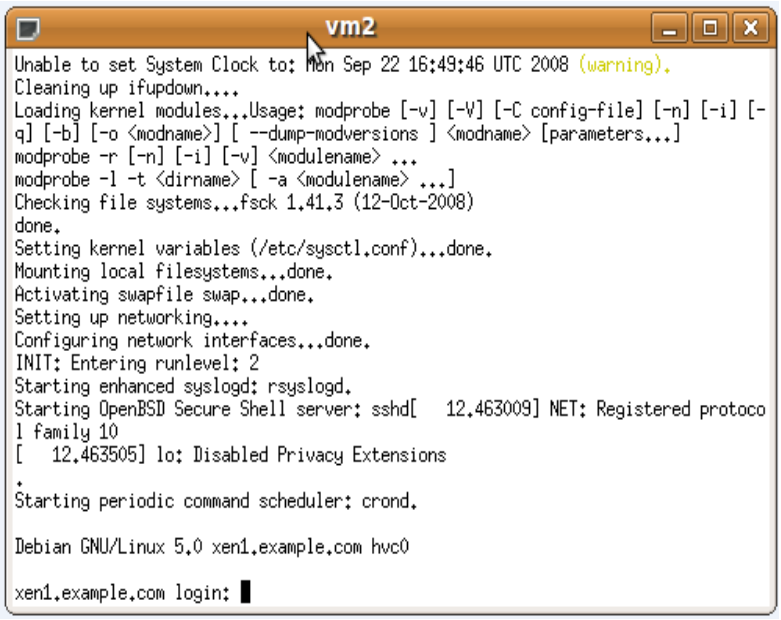

Figura 5.9. Máquina virtual 2 del escenario xenImplem3

Al registrar el usuario que se indica.en la figura 5.10

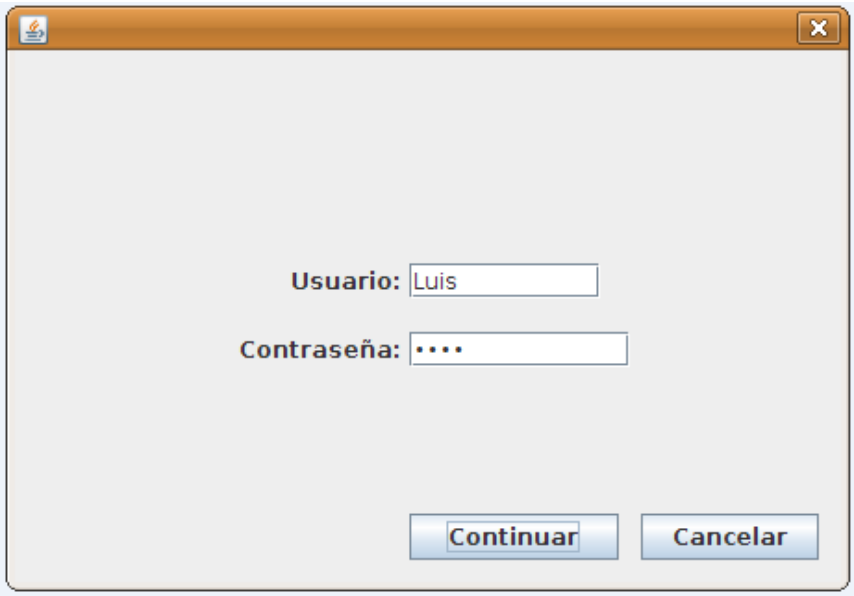

Figura 5.10. Login para un usuario del entorno distribuido virtualizado.

Luego de unos instantes, el aplicativo presenta las pantallas de la figura 5.11 y 5.12

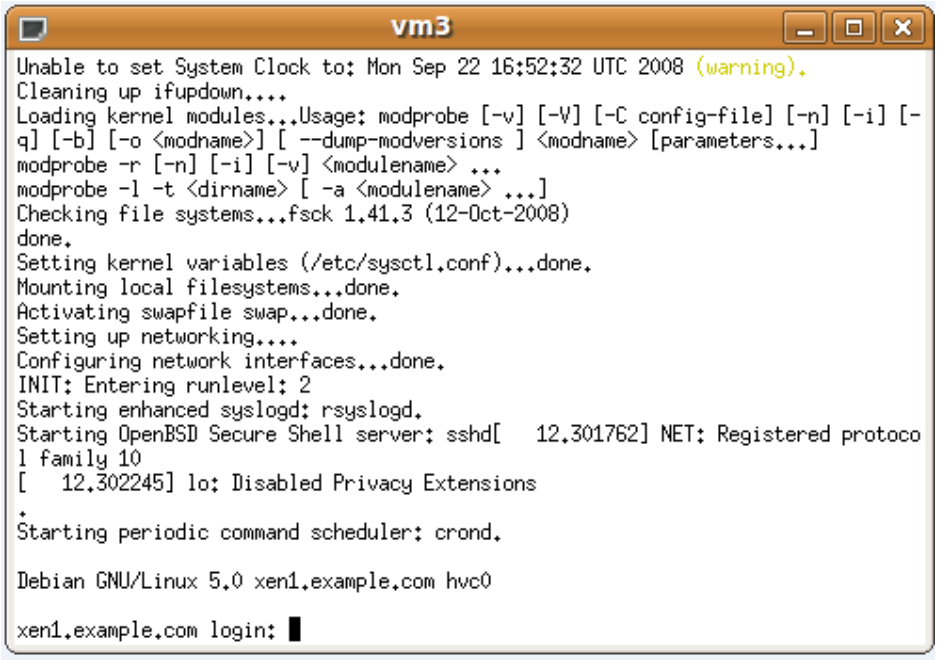

Figura 5.11. Máquina virtual 3 del escenario xenImplem3

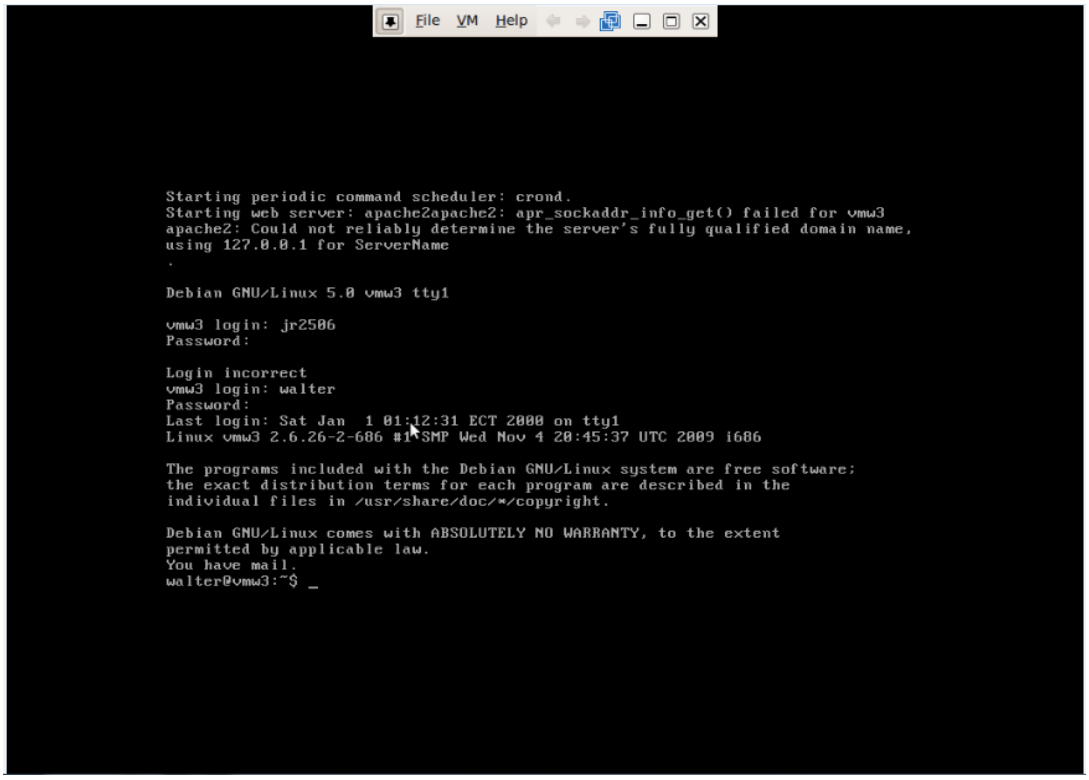

Figura 5.12. Máquina virtual 3 del escenario vmwareImplem3

En el escenario vmwareImplem3, se deben gestionar e ingresar los siguientes datos en forma correcta:

Gestión con errores:

vmw3 login:jr2506

Password:

Ejecución incorrecta:

Login incorrect

Gestión exitosa:

vmw3 login:walter

Password:

Ejecución exitosa:

Last login:Sat Jan 101:12:31 ECT 2000 on ttgl

Linux vmw3 2.6.26-2-686 #1 SMP Wed Nov 1 28:15:37 UTC 2009 1686

Como resultado del ingreso de los usuarios y levantamiento de escenarios virtuales, se crean registros en la tabla de pistas de auditoría, pantallas 5.13, 5.14:

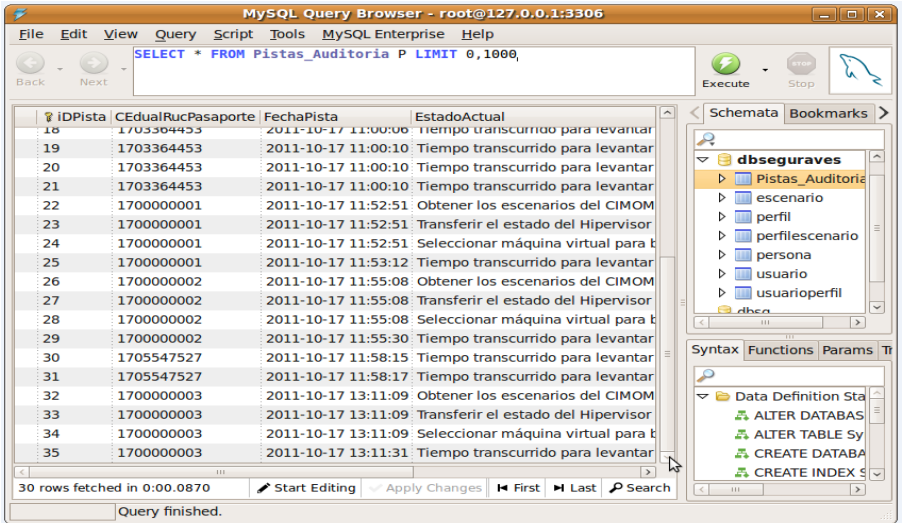

Figura 5.13. Ventana 1 de pistas de auditoria.

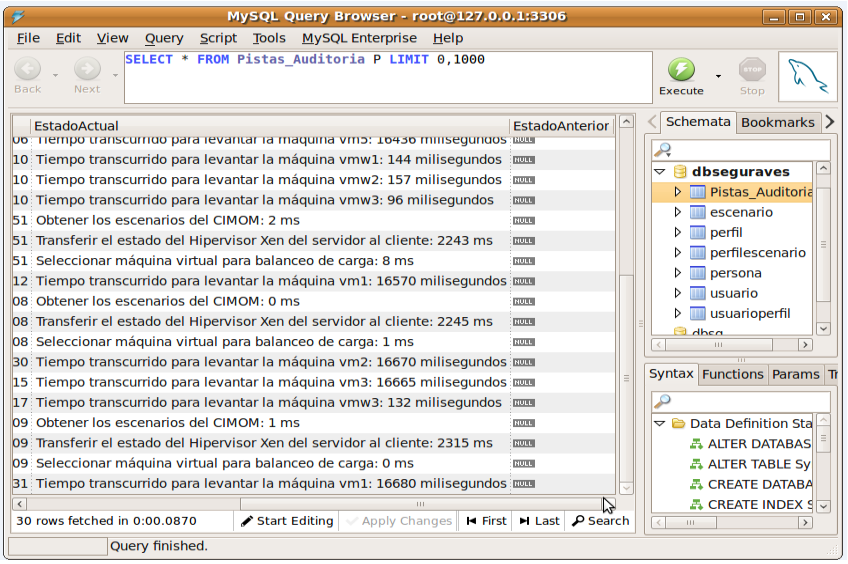

Figura 5.14. Ventana 2 de pistas de auditoria.

5.6. CONCLUSIÓN

Se ha probado que el modelo trabaja en forma óptima y eficiente si se configuran correctamente las direcciones IP de los servidores de virtualización y del cliente Wbem – Jcraft, además de la asignación de algunos valores en el repositorio CIMOM.

CAPÍTULO VI

CONCLUSIONES

El presente proyecto simboliza la finalización de un ciclo y el inicio de otro, lo cual va acompañado de cambios que alteran la estructura del orden del conocimiento, es por esto la importancia de extraer conclusiones, no solamente al finalizar un proyecto y comprobar que las metas que se perseguían han sido cumplidas, sino en cualquier empresa que se emprenda, para que estas sirvan como base de apoyo para iniciar un nuevo ciclo.

A continuación se enfocan las conclusiones que se consideran más importantes:

- La virtualización no supone un riesgo, en razón de que, tanto los equipos virtualizados como la arquitectura de red virtual son totalmente transparentes; sin embargo, es importante tener en cuenta que, la seguridad de los sistemas virtualizados recae en *la configuración de los propios sistemas operativos*, a los que se les deben aplicar las mismas políticas de seguridad que se aplicaría a un sistema real.

- La virtualización de una Infraestructura de servidores permite la consolidación de los mismos; la compatibilidad con aplicaciones y SO; pruebas en entornos aislados y seguros; gestión y control centralizado de recursos; procedimientos de *backup*; migraciones en vivo de MVs entre distintos servidores físicos; fácil recuperación en caso de caídas; alta disponibilidad y facilidad para la puesta en marcha de servicios de Cloud Computing.

- Las especificaciones proporcionadas por DMTF-CIM, permitió analizar y evaluar los diferentes Métodos de Modelización de Virtualización de Infraestructura y Aplicaciones, aunque no cubre el modelado y despliegue de VNEs en su conjunto.

- El tema principal de nuestro trabajo fue diseñar un esquema de extensión de la CIM para caracterizar un modelo genérico para VNEs, incluido el despliegue de VNEs en su conjunto en una plataforma de virtualización; el resultado del modelo muestra que, se ha logrado un modelo general que es independiente de plataformas de virtualización.

- Para validar el modelo general del VNE, se ha construido un cliente CIM de Java el cual recupera los objetos de la modelización del repositorio CIMOM de los VNEs; los resultados de las pruebas evaluada con Xen y VMware Server han demostrado la eficacia de esta aplicación.

- En consecuencia, la ventaja de este trabajo es que reduce la complejidad en la construcción y despliegue de VNEs utilizando diferentes plataformas de virtualización.

- El cliente WBem implementado sobre la base de este modelo, parcialmente toma el control de los recursos disponibles y permite el despliegue del VNE en diferentes hosts físicos, cada uno basado en una plataforma de virtualización diferente y de manera automática a través de perfiles de usuario.

- El valor de CIM deriva su orientación a objetos: abstracción y clasificación, para reducir la complejidad del dominio del problema; herencia de objetos para heredar métodos y propiedades; y la capacidad para describir dependencias, asociaciones y relaciones, consideramos que el uso de CIM en nuestra investigación fue una buena decisión.

- La realización de las *pruebas de rendimiento del VNE*, ha permitido analizar, las plataformas de virtualización Xen, VMware y CIMOM tomando en cuenta las especificaciones del procesador y del chipset, los índices de rendimiento facilitan la medición de las diferencias entre procesadores, ya que cada SW de virtualización saca partido de toda la información accesible a los VNEs.

- Al ejecutar el cliente API WBem & JCraft medimos el *tiempo de ejecución,* es decir, el tiempo que se demora en encontrar la información en el CIMOM, la generación del script de implementación, la transferencia de los archivos a

los respectivos hosts y la ejecución de los scripts para arrancar los VNEs y el *tiempo de despliegue,* es decir, el tiempo necesario para ejecutar la secuencia de comandos para desplegar un VNE.

- Dado que *el entorno virtual tiene la capacidad de abastecerse a sí mismo para aprovechar los recursos disponibles*, la presente aplicación permitió desplegar los VNEs en varios hosts de una manera automática a través de perfiles de usuario, concluyéndose que los tiempos de respuesta son más rápidos y los tiempos improductivos pueden llegar a ser casi nulos.

- La realización de las *pruebas de rendimiento simulando balanceo de carga del VNE*, permitió medir el *tiempo de transferencia*, es decir, el tiempo que tarda el hipervisor en asignar los recursos necesarios para levantar un VNE (la información del estado del Hipervisor Xen entre el servidor y el cliente API).

- Al incluir una MV con VMware, se observó que el *tiempo de transferencia* es cero, mientras que cuando se usa Xen en un equipo remoto se demora un promedio de 1700 ms en el tiempo de ejecución.

- Cuando se produce un pico de demanda de los recursos de las MVs, las peticiones de recursos pueden exceder los recursos disponibles en un host; es en este momento que mediante un algoritmo de balanceo de carga, automáticamente se recolocan las MVs en hosts en los que los recursos estén disponibles balanceando continuamente la capacidad y asegurando que cada MV tenga acceso a los recursos apropiados en tiempo real.

- El balanceo de carga se mantiene gracias al algoritmo que divide de la manera más equitativa el despliegue de las MVs, para evitar la saturación del equipo centralizado en el que se distribuye el VNE; además, los porcentajes de consumo de CPU, de memoria y la gestión de tráfico IP han demostrado que el balanceo dinámico de cargas de trabajo entre MVs es un requisito indispensable, no solo para conseguir la mejor utilización del HW, sino también la virtualización de las aplicaciones y la optimización del uso de los recursos.

RECOMENDACIONES

El presente proyecto tiene una gran perspectiva, es una área multidisciplinaria extensa y que posiblemente algunas de estas recomendaciones sean de utilidad para mejores prácticas futuras o para señalar la dirección de un nuevo modelado de infraestructuras virtualizadas y los mecanismos de seguridades para el despliegue de servicios en éstos entornos:

- Con el propósito de facilitar la gestión de datos y recursos entre diferentes sistemas computacionales, se recomienda el uso de la DMTF, en razón de que éste grupo de investigadores han definido un conjunto de tecnologías y protocolos para modelar y acceder a los recursos de sistemas computacionales y redes de una manera estándar.

- Para obtener un adecuado Modelo Conceptual de Información que permiten describir las entidades a ser gestionadas, su composición y sus relaciones, se recomienda trabajar con el soporte que brinda WBem, el cual comprende un conjunto de tecnologías que permiten escribir programas e interfaces para la gestión de un VNE; además, utiliza el formato de objetos MOF para definir y describir los objetos a ser gestionados.

- Siendo el principal elemento de una arquitectura de WBem el CIMOM y en razón de que los clientes CIM implementan el acceso normalizado a los datos y a las operaciones de gestión de los recursos que están representados en el CIMOM, se recomienda el uso de WBem para tal gestión.

- Con el propósito de usar una metodología que permita representar y modelar la información con la que se trabaja en las fases de análisis y diseño, se recomienda el soporte de CIM, en razón de que tiene una relación formal con el Lenguaje Unificado de Modelado (UML) el cual se ha convertido en un estándar para tales fines.

- Con el fin de obtener un Modelo de Sistemas Virtualizados que permitan la integración de redes, servicios, recursos y el despliegue automático de

VNEs, se recomienda el uso de enfoques de modelado, tales como: Perfil de Virtualización DMTF-CIM, define cómo utilizar CIM para la representación de los sistemas virtuales y sus recursos; Formato Abierto de Virtualización (OVF), estándar abierto para la distribución de aplicaciones virtuales; VMware-CIM, proporciona un modelo de objetos para las MVs y dispositivos de almacenamiento; Libvirt-CIM, conjunto de bibliotecas para la gestión de MVs utilizando el modelo de virtualización de la DMTF; y Xen-CIM, para la definición de modelos de sistema y recursos virtualizados.

- En éste tipo de trabajos, se recomienda la realización de las *pruebas de rendimiento del VNE*, con el soporte de las plataformas de virtualización Xen, VMware y CIMOM.
- Se recomienda la realización de las *pruebas de rendimiento simulando balanceo de carga del VNE*, lo que involucra la gestión del *tiempo de transferencia*, es decir, la gestión del tiempo que tarda el hipervisor en asignar los recursos necesarios para levantar un VNE.
- Cuando se produce un pico de demanda de los recursos de las MVs, es decir, las peticiones totales de recursos excedan a los recursos disponibles en un host; se recomienda, hacer uso de un algoritmo de balanceo de carga que proporcione funcionalidad de redundancia activa–pasiva, con el fin de dividir de manera equitativa el despliegue de las MVs, para evitar la saturación del equipo centralizado en el que se distribuye el VNE.

- Las metodologías han sido creadas para asistir a los equipos de desarrollo de sistemas, a profesionales de seguridad informática y auditores, evaluación de riesgos y establecimiento de controles internos en cualquier componente de los sistemas de información automatizados; razones éstas, se recomienda que las pistas de los perfiles de los objetos del VNE que se encuentran en un repositorio CIMOM deben estar sujetos a un pleno control.

BIBLIOGRAFÍA

ASESORIAS TECNICAS

- Ing. Walter Fuertes PHD

  Director de Postgrados

  Escuela Politécnica del Ejército

- Ing. Edison Lascano MSc

  Catedrático de la Escuela Politécnica del Ejército

  Departamento de Ciencias de la Computación

REFERENCIAS BIBLIOGRÁFICAS

[1] Carmen Pastor, Plataformas Virtuales, 9 Abril 2008, [Online]:www.trendmicro.com/go/virtualization

[2] "Ubuntu y su virtualización", Febrero 21 de 2008. [online]:http://www.luanorma.blogspot.com/2008/02/ubuntu-kvm-su-virtualizacin.html

[3] [Linux], "Juntando varias tecnologías de virtualización", Febrero 24 de 2011, [online]:http://www.phenobarbital.wordpress.com/2011/02/24/linux-juntando-varias-las-tecnologias-de-virtualizacion-i.htm

[4] Fermín Galán, Raúl Muñoz, Ricardo Martínez, "Uso de técnicas de virtualización para la experimentación en redes ópticas basadas en GMPLS y DWDM", Publicado en las memorias del XVI Jornadas Telecom I+D, Centro de Tecnología de Telecomunicaciones de Catalunya, realizado en Madrid, España del 23 al 25 de noviembre de 2006, fermin.galan@cttc.es [online]:http://www.cttc.es/resources/doc/071116-227-galan-telecom1206-19167.pdf

[5] P. Barham, B. Dragovic, k. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. "Virtualization platforms", [online]:http://dnolivieri.net/ACSO/AllDocs/03-Grupo3-Virtualizacion.pdfhttp://dnolivieri.net/ACSO/AllDocs/03-Grupo3-Virtualizacion.pdf

[6] Nuria Cordón, La Virtualización XEN ayuda a reducir los costos en las organizaciones, Septiembre 2007, Computerworld©2010 IDG COMMUNICATIONS, S. A.U

[7] P. Barham, B. Dragovic, k. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. "Xen and the art of virtualization". In Proc. Of the 19th Symp. On Operating Systems Principles, pps: 164-177, Oct.2003

[8] Marta Correas/Natividad de Mateo, Seguridad para la Plataforma VMware, 11 Abril 2008, [Online]:http://es.trendmicro.com

[9] Luis Temes, VMware vSphere4 “La Virtualización llevada al siguiente nivel”, 2 Julio 2009, [Online]:www.iberdeco.com

[10] Caridad Anías Calerón, “Estándares del DMTF”, Departamento de Telemática Cujae, cacha@tesla.cujae.edu.ec.
[online]:http://www.bibliociencias.cu/gsdl/collect/eventos/index/assoc/HASHc32f.dir/doc.pdf

[11] José María González, "¿Qué es el Open Virtualization Format?", JmG Virtual Consulting, S.L. Líderes y expertos en Soluciones de Virtualización de Sistemas, Mayo 27 de 2011.
[online]:http://www.josemariagonzalez.es/tag/ovf

[12] DMTF, “Open Virtualization Format Specification”, Copyrigth@2009 Distributed Management Task Force,Inc.(DMTF).All rights reserved, Documento número DSP0243, Febrero 22 de 2009, version 1.0.0.
[online]:http://xml.coverpages.org/DMTF-OVF-v10-DSP0243.pdf

[13] Anónimo, “Windows Management Instrumentation WMI”, [Online]:http://bieec.epn.edu.ec:8180/dspace/bitstream/123456789/886/5/T10391CAP2.pdf

[14] Luisa Carolina Nieto H, “Modelo de Gestión de Redes de Datos a través de WEB WEBEM” (WEB ENTERPRISE MANAGEMENT), [Online]:http://www.scribd.com/doc/20618787/WBEM-Luisa-Nieto-2007

[15] JCraft, “Code the Craft”, Copyright 1998-2011, JCraft, Inc. All rights reserved,
[online]:http://www.jcraft.com/jsch/

[16] Eduardo Pelegri–Llopart, Glassfish, 19 Enero 2009,
[Online]:http://www.blogs.sun.com/theaquarium

[17] Juan Antonio Martinez Gil, Francisco Jose Mora, “Entorno de red virtual para la realización de prácticas realistas de administración de sistemas operativos y redes de computadores”, Departamento de Tecnología Informática y Computación, Universidad de Alicante.
[online]:http://www.dtic.ua.es/grupoM/recursos/articulos/JENUI-05-B.pdf

[18] David Fernández, Linux y XEN como Tecnologías Alternativa, 3 Diciembre 2009, [Online]:http://mkm-pi.com/

[19] Javier Andrés Alonso, “Balanceo de carga con LVS(I)”, redes Privadas Virtuales, España, Marzo 30 de 2009,
[online]:http://redes-privadas-virtuales.blogspot.com/2009/03/balanceo-de-carga-con-lvs-i.html

[20] Edgar Oviedo, Metodologías ágiles de desarrollo de software con UML, Marzo 2010

[21] Implementación Adempiere en Ubuntu, "Metodología AUP (Agile Unified Process) para la implementación del ERP Adempiere-Parte2". [online]:http://www.ubuntu-adempiere.blogspot.com/2011/09/metodologia-aup-agile-unified-process_22.html

[22] Anónimo, "IEEE-STD-830-1998: Especificaciones de los requisitos del software",
[online]: http://www.ctr.unican.es/asignaturas/is1/IEEE830_esp.pdf

[23] Hilda Gómez, VMware acerca el cloud computing a las Pymes con la Plataforma vSphere4, Abril 21 de 2009, Dealer©2010 IDG COMMUNICATIONS, S. A. U

[24] S. Navaridas Vallejo, E. Jiménez macías, J. Blanco Fernández, M. Pérez de la Parte, Entorno de alta disponibilidad de MTA`s: Modelo Mixto de Sistema Virtualizado y Sistema Real, Universidad de la Rioja, ETSII, 26004 Logroño, La Rioja, España, Departamento de Ingeniería Eléctrica, Departamento de Ingeniería mecánica,
[Online]:http://www.cea-ifac.es/actividades/jornadas/XXIX/pdf/229.pdf

[25] Thomas Leichtenstern, "Herramientas de virtualización",
[Online]:http://www.linux-magazine.es/issue/34/021-026_EscritorioVirtualLM34.crop.pdf

[26] The Open Group, "SNIA CIMOM", 24 Noviembre 2003,
[Online]:http://www.opengroup.org/snia- cimom/

[27] W. Fuertes, J. E. López de Vergara, F. Meneses, F. Galán, A Generic Model for the Management of Virtual Network Environments. In Proceedings of 12th IEEE/IFIP Network Operations and Management Symposium (NOMS 2010), Osaka, Japan, 19-23 April 2010

[28] Anonimo, "5 herramientas de virtualización FLOSS".
[online]:http://www.somoslibres.org/modules.php?name=News&file=article&sid=4207

[29] W. Fuertes, J. E. López de Vergara, F. Meneses, "Educational Platform using Virtualization Technologies: Teaching-Learning Applications and Research Uses Cases". Accepted for its publication in II ACE Seminar: Knowledge Construction in Online Collaborative Communities, Albuquerque, NM – USA, October 2009

[30] Patricia Montanelli, "El ABC de la virtualización", 11 Octubre 2007,
[Online]:http://www.cxo-community.com/articulos/blogs/blogs-el-abc-de/2348-el-abc-de-la-virtualizacion.html

[31] Techweek, “La flexibilidad de una Infraestructura de Virtualización”, 27 Marzo 2009, [Online]:http://www.techweek.es/virtualizacion/soluciones negocio/1005267005901/panda- telefonicaflexibilidad-infraestructura virtualizada.1.html

[32] Knowledge Center, “Cinco consejos para mejorar la seguridad de un sistema virtualizado”, 17 Junio 2009, [Online]:http://www.eweekeurope.es/knowledge-center/knowledge center-green-it/cinco-consejos-para-mejorar-la-seguridad-en-un-sistema virtualizado-1010

[33] Informador.com.mx, “basura electrónica, riesgo ecológico”, 17 Julio 2010, Guadalajara, Jalisco, [Online]:http://www.informador.com.mx/tecnologia/2010/180681/6/basura-electronica-riesgo-ecologico.htm

[34] InforWorld, “Cinco preguntas y respuestas sobre virtualización”, 2 Octubre 2009, [Online]:http://iworld.com.mx/iw_news_read.asp?iwid=7086

[35] Borrmart, S.A, “Un complejo hogar para la información”, [Online]:http://www.borrmart.es/articulo_redseguridad.php?id=2222

[36] IEAISA, “Soluciones virtuales con VMware, Citrix y MS VirutalServer”, [Online]:http://www.ieaisa.es/sistemas.htm

[37] Carl Claunch, “Gartner the future of IT”, [Online]:http://www.gartner.com/it/summits/per02l/keynotes.jsp

[38] W. Fuertes and J. E. López de Vergara, An emulation of VoD services using virtual network environments. In Proceedings of GI/ITG Workshop on Overlay and Network Virtualization NVWS'09, Kassel-Germany, March-200

[39] Fermín Galán, David Fernández, Walter Fuertes, Miguel Gómez, Jorge López de Vergara, Scenario-based Virtual Network Infrastructure Management in Research and Educational Testbeds with VNUML: Application Cases and Current Challenges, Annals of Telecommunications, Special issue on Virtualization

[40] Walter M. Fuertes, Jorge E. López de Vergara, A quantitative comparison of virtual network environments based on performance measurements, Proceedings of the 14th HP Software University Association Workshop, Garching, Munich, Germany, 8-11 July 2007

[41] W, Fuertes, J. E. López de Vergara, Evaluación de Plataformas de Virtualización para Experimentación de Servicios Multimedia en Redes IP, aprobado para su publicación en la Revista Técnica de Ciencia y Tecnología de la Escuela Politécnica del Ejército, Quito, Ecuad

[42] Jorge Rodríguez. Pruebas Unitarias. Marzo 2006, [online]:http://blog.continuum.cl/wp-content/uploads/2008/08/pruebas-unitarias.pdf

[43] Javier Tuya, Nuevo estándar de pruebas de software ISO IEC 29119, Universidad de Oviedo, Jornadas de Innovación y Calidad del Software, Alcalá de Henares 4/09/2009, [online]: http://www.ati.es/IMG/pdf/Tuya.pdf

[44] Arbós, L.M. Amorós, D. González, A. Oller, J. Alcober, "Servicios telemáticos sobre nubes privadas en plataformas virtualizadas y distribuidas", Departamento de Ingeniería Telemática, Universidad Politécnica de Catalunya

[45] Alex Restrepo, Alejandro Montoya, Helmuth Trefftz, "Asignación dinámica de servidores en ambientes virtuales usando octress para la notificación dinámica del espacio", Universidad Medellín, Colombia, Septiembre 14 de 2009, [online]:http://www2.unalmed.edu.co/~pruebasminas/index.php?option=com_docman&task=doc_view&gid=1251&tmpl=component&format=raw&Itemid=285

[46] F. Meneses, W. Fuertes, L. Guerra, J. E. López de Vergara y H. Aules, "Modelo Distribuido para la Gestión de Entornos Virtuales de Red", Publicado en las memorias del VI Congreso de Ciencia y Tecnología ESPE 2011, realizado en Sangolquí, Ecuador del 8 al 10 de junio de 2011. ISBN 1390-4663

[47] Fausto Meneses Becerra, Walter Marcelo Fuertes Díaz, Luis Alberto Guerra Cruz, "Modelo Distribuido para la Gestión de Entornos Virtuales de Red Simulando Balanceo de Carga", Publicado en las memorias del XVI Congreso Internacional de Contaduría, Administración e Informática Universidad Nacional Autónoma de Mexico (UNAM), realizado en la Ciudad de México, México del 5 al 7 de octubre de 2011. UNAM ISBN 978-607-02-2548-2

[48] Eniac, "Software para auditoría interna, Auditoría de sistemas y calidad de datos", [online]:http://www.eniac.com/productos/lumigent.htm

ACRÓNIMOS

A

API: Application Program Interface - Interfaz de Aplicación.

AUP: Agile Unified Process - Proceso Unificado Ágil.

C

CIM: Common Information Model - Modelo de Información Común.

CIMOM: CIM Object Manager - Modelo de Información Común del Gestor de Objetos.

D

DMTF: Distributed Management Task Force - Grupo de Trabajo para Gestión Distribuida.

H

HA: High Availability of Infrastructure - Alta disponibilidad de Infraestructura.

HMC: Hardware Manager Console - Consola de Gestión de Hardware.

I

IVM: Integrated Virtualization Manager - Gestor de Virtualización Integrada.

J

JavaEE - Java Enterprise Edition - Edición Empresarial de Java.

L

Linux KVM: Kernel Virtual Machine Linux - Màquina virtual Kernel Linux.

M

MAC: Media Access Control - Control de Acceso a Medios.

MLE: Measured Startup Environment - Entorno de Inicio Medido

MOF: Managed Object Format – Formato para la Administración del Objeto.

MV: Virtual Machine - Máquina Virtual.

O

OMG: Object Management Group - Grupo de Gestión de Objetos.

OVA: Open Virtual Appliance – Aplicación Virtual Abierta.

OVF: Open Virtualization Format – Formato de Virtualización Abierta.

P

PS: Passive Sniffing – Búsqueda Pasiva.

R

ROI: Fast Return on Investment – Rápido Retorno de la Inversión.

RUP: Rational Unified Process – Proceso Unificado Lógico.

S

SO: Operating System - Sistema Operativo.

T

TCO: Cost Savings for Property - Ahorros de Costos de Propiedad.

TPM: Trusted Platform Module - Módulo de Plataforma de Confianza.

TXT: Trusted Execution Technology - Tecnología de Ejecución de Confianza.

U

UML: Unified Modeling Language - Lenguaje Unificado de Modelado.

UMLX: User Mode Linux - Modo de usuario Linux.

V

VMJ: Virtual Machine Jumping– Salto de Máquina Virtual.

VMM: Virtual Machine Monitor - Monitor de Máquina Virtual.

VNE: Virtual Network Environment - Entorno de red virtual.

VPN: Virtual Private Network - Red privada virtual.

VPS: Virtual Private Servers - Servidor privado virtual.

W

WBem: Web-Based Enterprise Management – Administración de la empresa con el soporte de la Web.

ANEXOS

ANEXO A. ESPECIFICACIÓN DE REQUERIMIENTOS PARA DETERMINAR EL SOFTWARE DE ACUERDO A LA NORMA IEEE 830 (ERS)

Definiciones

Infraestructura Virtual

Está construida a través de una capa abstracta entre los servidores (discos, memoria, tarjetas de red, entre otros elementos de hardware) y programas que están funcionando en éstas máquinas. La Virtualización de la Infraestructura ordena las operaciones de las TIC permitiendo a las empresas usar y gestionar de forma más óptima los recursos de hardware. Los usuarios ven los recursos como suyos y los administradores pueden gestionar los recursos a nivel de toda la universidad.

Máquina Virtual (MV)

Una Máquina Virtual es un software que emula a un computador y puede ejecutar programas como si fuese una PC real e forma aislada y eficiente[Popek 1972].

Entorno Virtual de Red (VNE)

Comprende un conjunto de equipos virtuales: sistemas finales como elementos de red, enrutadores, conmutadores conectados entre sí en una determinada topología desplegada sobre uno o múltiples equipos físicos, que emulan un sistema equivalente y cuyo entorno deberá ser pecibido como si fuera real[18].

Alta disponibilidad de Infraestructura

Es un producto de diseño del sistema y su implementación asociada asegura un cierto grado absoluto de continuidad operacional durante un periodo de medición.

Si se produce un fallo de hardware en alguna de las máquinas del clúster, el software de alta disponibilidad es capaz de arrancar automáticamente los servicios en cualquier máquina del clúster (failover); y cuando la máquina

que ha fallado se recupera, los servicios son nuevamente migrados a la máquina original (failback). Esta capacidad automática de servicios nos garantiza la alta disponibilidad de los servicios ofrecidos por el clúster, minimizando de esta manera la percepción del fallo por parte de los usuarios.

Balanceo de Carga entre Servidores

Es un sistema para distribuir el trabajo entre varios servidores, de tal manera, que se consiga una mayor disponibilidad y rendimiento. En caso de que uno de sus servidores falle, el sistema de balanceo de carga dirigirá el tráfico a las máquinas que estén respondiendo.

Continuidad del Negocio

Permite reducir el tiempo de inactividad mediante soluciones de alta disponibilidad y recuperación ante desastres de entornos físicos y virtuales; esto aporta una función de protección y recuperación completa e integrada para información, aplicaciones y servidores empresariales, sean físicos o virtuales.

Costo Total de la Propiedad (Total Cost of Ownership - TCO)

Es un método de cálculo diseñado para ayudar a los usuarios y a los gestores empresariales a determinar los costos directos e indirectos, así como los beneficios relacionados con la compra de equipos o programas informáticos.

Especificación de Requisitos de Software

Es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye Casos de Uso (Requisitos Funcionales). Los requisitos no funcionales son aquellos que imponen restricciones en el diseño o la implementación.

Retorno de la Inversión (Return of Investment - ROI)

Es el beneficio que se obtiene por cada unidad monetaria invertida en tecnología durante un periodo de tiempo. Se utiliza para analizar la viabilidad de un proyecto y medir su éxito.

ANEXO B. *Requisitos Específicos*

Interfaces Externas

Interfaces de Usuario

REQ01: Administración

El software de virtualización poseerá una consola de administración centralizada donde el usuario administrador tendrá una visión global de la Infraestructura Virtual: host, máquinas virtuales, red y storage para poder gestionarla de acuerdo a la necesidad y políticas de la empresa. La administración deberá ser forma gráfica y mediante línea de comandos.

Interfaces de Hardware

REQ02: Compatibilidad de Hardware

El software de virtualización deberá disponer de una amplia lista de disponibilidad de servidores, storage arrays y dispositivos de I/O que soporten su implementación.

REQ03: Capa de Virtualización

El software de virtualización no deberá correr sobre un sistema operativo base, sino deberá emplear una arquitectura con hipervisor.

REQ04: Instalación de Software de Virtualización

El software de virtualización debe permitir ser instalado en discos locales (SCSI, SATA) y en una red de storage basada en discos (Fiber Channel y iSCSI).

Funciones

REQ05: Consolidación de Servidores

El software de virtualización deberá estar orientado a la optimización de la Infraestructura Virtual mediante la consolidación y el soporte de servidores.

REQ06: Soporte de Varios Sistemas Operativos Invitados

El software de virtualización deberá permitir tener múltiples sistemas operativos corriendo simultáneamente en un servidor.

REQ07: Soporte Multiprocesador

El software de virtualización deberá permitir asignar más de dos procesadores a las máquinas virtuales.

REQ08: Migración en Vivo de Máquinas Virtuales

El software de virtualización deberá permitir la migración en vivo de máquinas virtuales a otros servidores físicos, para realizar tareas de mantenimiento de los servidores sin interrumpir las aplicaciones que están corriendo en los mismos.

REQ09: Alta Disponibilidad de MV entre Hosts de un Clúster

El software de virtualización deberá permitir alta disponibilidad en caso de alguna falla de un host del clúster, haciendo que las máquinas virtuales afectadas sean reiniciadas automáticamente en otro host del clúster.

REQ10: Balanceo de Carga de Hosts en un Clúster

El software de virtualización deberá permitir la distribución automática o manual de las máquinas virtuales a través de clúster de hosts, permitiendo el balanceo de carga de los hosts.

REQ11: Balanceo del Tráfico de Red (NIC Team)

El software de virtualización deberá permitir la distribución automática del tráfico de red a través de varias interfaces de red mediante el uso de diferentes métodos de balanceo de carga en un NIC Team:

- Balanceo de tráfico a través de ID de puerto.
- Balanceo de tráfico a través de MAC de origen.
- Balanceo de tráfico a través de IP origen – destino.

REQ12: Asignación Dinámica de Recursos para Máquinas Virtuales

El software de virtualización deberá asignar los recursos de hosts de manera inteligente entre las máquinas virtuales de acuerdo con reglas predefinidas y prioridades que reflejen las necesidades de las respectivas máquinas virtuales.

REQ13: Clonación de Máquinas Virtuales

El software de virtualización deberá permitir la clonación de máquinas virtuales, las cuales pueden encontrarse prendidas o apagadas.

REQ14: Creación de Máquinas Virtuales a partir de Templates

El software de virtualización deberá permitir la creación de templates de máquinas virtuales, para que a partir del mismo se pueda crear una máquina de forma segura, rápida y sencilla.

REQ15: Soporte de VLANs

El software de virtualización deberá soportar el uso de VLANs, en razón de que su aplicación es frecuentemente utilizada en la administración y seguridad de las redes.

REQ16: Manejo de Backups

El software de virtualización deberá permitir la realización de Backups trasladando el tráfico de backup a una red diferente a la red de producción, eliminando el tráfico de backup para no afectar el rendimiento de las máquinas virtuales.

REQ17: Almacenamiento Compartido

El software de virtualización deberá permitir el uso de almacenamiento compartido (SAN y NAS), el cual permite tener opciones de recuperación ante desastres, alta disponibilidad y movimiento de máquinas virtuales entre servidores.

REQ18: Control de Acceso a la Infraestructura Virtual

El software de virtualización deberá permitir la asignación de diferentes permisos a diferentes usuarios que acceden a la Infraestructura Virtual, para la realización de taras exclusivas de administración.

REQ19: Monitoreo de Recursos

El software de virtualización deberá permitir el monitoreo del performance de las máquinas virtuales a través de gráficos estadísticos en vivo y el uso de mensajes de alarmas enviados a través de mail y snmp.

ANEXO C. Generación de la Base de Datos en MySQL

Para poder exportar la base de datos de un servidor a otro, se empleó el comando: mysqldump -u root -p dbseguraves > dbseav150511.sql

Mismo que produce el archivo dbseav150511.sql, cuyo contenido a continuación:

```
-- MySQL dump 10.11
--
-- Host: localhost    Database: dbseguraves
-- ------------------------------------------------------
-- Server version    5.0.75-0ubuntu10.5


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO'
*/;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `Pistas_Auditoria`
--

DROP TABLE IF EXISTS `Pistas_Auditoria`;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `Pistas_Auditoria` (
  `iDPista` int(11) NOT NULL auto_increment,
  `CEdualRucPasaporte` varchar(15) NOT NULL,
  `FechaPista` datetime default NULL,
  `EstadoActual` varchar(100) default NULL,
  `EstadoAnterior` varchar(100) default NULL,
  PRIMARY KEY  (`iDPista`),
  KEY `FK_rperpau` (`CEdualRucPasaporte`),
  CONSTRAINT `Pistas_Auditoria_ibfk_1` FOREIGN KEY (`CEdualRucPasaporte`) REFERENCES
`persona` (`CEdualRucPasaporte`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
SET character_set_client = @saved_cs_client;

--
-- Dumping data for table `Pistas_Auditoria`
--

LOCK TABLES `Pistas_Auditoria` WRITE;
/*!40000 ALTER TABLE `Pistas_Auditoria` DISABLE KEYS */;
/*!40000 ALTER TABLE `Pistas_Auditoria` ENABLE KEYS */;
UNLOCK TABLES;


--
-- Table structure for table `escenario`
--

DROP TABLE IF EXISTS `escenario`;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `escenario` (
```

```
  `iDEscenario` int(11) NOT NULL auto_increment,
  `descEsc` varchar(100) NOT NULL,
  PRIMARY KEY  (`iDEscenario`)
) ENGINE=InnoDB AUTO_INCREMENT=31 DEFAULT CHARSET=utf8;
SET character_set_client = @saved_cs_client;


--
-- Dumping data for table `escenario`
--

LOCK TABLES `escenario` WRITE;
/*!40000 ALTER TABLE `escenario` DISABLE KEYS */;
INSERT INTO `escenario` VALUES
(1,'xenImplem3'),(2,'xenImplem11'),(3,'xenImplem12'),(4,'xenImplem13'),(5,'xenImplem212'),(6,'xenImp
lem213'),(7,'xenImplem223'),(9,'vmwareImplem3'),(10,'vmwareImplem11'),(11,'vmwareImplem12'),(12,'
vmwareImplem13'),(13,'vmwareImplem212'),(14,'vmwareImplem213'),(15,'vmwareImplem223'),(16,'xe
n2Implem2'),(30,'xenAssign');
/*!40000 ALTER TABLE `escenario` ENABLE KEYS */;
UNLOCK TABLES;


--
-- Table structure for table `perfil`
--

DROP TABLE IF EXISTS `perfil`;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `perfil` (
  `iDPerfil` int(11) NOT NULL auto_increment,
  `dscPerfil` varchar(100) default NULL,
  PRIMARY KEY  (`iDPerfil`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8;
SET character_set_client = @saved_cs_client;


--
-- Dumping data for table `perfil`
--

LOCK TABLES `perfil` WRITE;
/*!40000 ALTER TABLE `perfil` DISABLE KEYS */;
INSERT INTO `perfil` VALUES (1,'Acceso total'),(2,'Maquinas virtuales Nro. 3'),(3,'Tres Maquinas en
Xen y 3 en VMWare'),(10,'Acceso a una maquina de Xen');
/*!40000 ALTER TABLE `perfil` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `perfilescenario`
--

DROP TABLE IF EXISTS `perfilescenario`;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `perfilescenario` (
  `iDPerfil` int(11) NOT NULL,
  `iDEscenario` int(11) NOT NULL,
  PRIMARY KEY  (`iDPerfil`,`iDEscenario`),
  KEY `FK_rescper` (`iDEscenario`),
  CONSTRAINT `FK_rescper` FOREIGN KEY (`iDEscenario`) REFERENCES `escenario`
(`iDEscenario`),
  CONSTRAINT `FK_rperesc` FOREIGN KEY (`iDPerfil`) REFERENCES `perfil` (`iDPerfil`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
SET character_set_client = @saved_cs_client;

--
-- Dumping data for table `perfilescenario`
--
```

```
LOCK TABLES `perfilescenario` WRITE;
/*!40000 ALTER TABLE `perfilescenario` DISABLE KEYS */;
INSERT INTO `perfilescenario` VALUES (1,1),(3,1),(2,4),(1,9),(3,9),(2,12),(1,16),(10,30);
/*!40000 ALTER TABLE `perfilescenario` ENABLE KEYS */;
UNLOCK TABLES;


--
-- Table structure for table `persona`
--

DROP TABLE IF EXISTS `persona`;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `persona` (
  `CEdualRucPasaporte` varchar(15) NOT NULL COMMENT ' @Property( name =
PROPERTY_C_EDUAL_RUC_PASAPORTE, kind = ReferenceHandler.ReferenceKind.ATTRIBUTE )',
  `Apellidos` varchar(30) NOT NULL COMMENT ' @Property( name = PROPERTY_APELLIDOS, kind
= ReferenceHandler.ReferenceKind.ATTRIBUTE )',
  `Nombres` varchar(30) NOT NULL COMMENT ' @Property( name = PROPERTY_NOMBRES, kind =
ReferenceHandler.ReferenceKind.ATTRIBUTE )',
  `Direccion` varchar(100) NOT NULL COMMENT '@Property( name =
PROPERTY_DIRECCIÃƒâ€œN, kind = ReferenceHandler.ReferenceKind.ATTRIBUTE )',
  PRIMARY KEY  (`CEdualRucPasaporte`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT=' * To change this template, choose Tools |
Templates\r\n ';
SET character_set_client = @saved_cs_client;


--
-- Dumping data for table `persona`
--

LOCK TABLES `persona` WRITE;
/*!40000 ALTER TABLE `persona` DISABLE KEYS */;
INSERT INTO `persona` VALUES ('1700000001','User 1','Xen','xx'),('1700000002','User
2','Xen','xx'),('1700000003','User 3','Xen','xx'),('1700000004','User 4','Xen','xx'),('1700000005','User
5','Xen','xx'),('1700000006','User 6','Xen','xx'),('1700000007','User 7','Xen','xx'),('1702122234','Fuertes
Diaz','Walter Marcelo','Desconocida'),('1703364453','Meneses Becerra','Fausto Honorato','Diego
Noboa Oe2-205, Cel 096107046, Tel 022074205'),('1705547527','Guerra Cruz','Luis Alberto','Av. de
los Conquistadores Nro. N24-507, La Floresta, Tel 5008146');
/*!40000 ALTER TABLE `persona` ENABLE KEYS */;
UNLOCK TABLES;


--
-- Table structure for table `usuario`
--

DROP TABLE IF EXISTS `usuario`;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `usuario` (
  `CEdualRucPasaporte` varchar(15) NOT NULL COMMENT ' @Property( name =
PROPERTY_C_EDUAL_RUC_PASAPORTE, kind = ReferenceHandler.ReferenceKind.ATTRIBUTE )',
  `id` varchar(30) NOT NULL,
  `contrasenia` varchar(50) default NULL,
  PRIMARY KEY  (`CEdualRucPasaporte`),
  CONSTRAINT `FK_rperusr` FOREIGN KEY (`CEdualRucPasaporte`) REFERENCES `persona`
(`CEdualRucPasaporte`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
SET character_set_client = @saved_cs_client;


--
-- Dumping data for table `usuario`
--
LOCK TABLES `usuario` WRITE;
/*!40000 ALTER TABLE `usuario` DISABLE KEYS */;
```

```
INSERT INTO `usuario` VALUES
('1700000001','uxen1','uxen1'),('1700000002','uxen2','uxen2'),('1700000003','uxen3','uxen3'),('1700000
004','uxen4','uxen4'),('1700000005','uxen5','uxen5'),('1700000006','uxen6','uxen6'),('1700000007','uxen
7','uxen7'),('1702122234','Walter','Walter'),('1703364453','fausto','fausto'),('1705547527','Luis','Luis');
/*!40000 ALTER TABLE `usuario` ENABLE KEYS */;
UNLOCK TABLES;


--
-- Table structure for table `usuarioperfil`
--

DROP TABLE IF EXISTS `usuarioperfil`;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `usuarioperfil` (
  `CEdualRucPasaporte` varchar(15) NOT NULL COMMENT ' @Property( name =
PROPERTY_C_EDUAL_RUC_PASAPORTE, kind = ReferenceHandler.ReferenceKind.ATTRIBUTE )',
  `iDPerfil` int(11) NOT NULL,
  PRIMARY KEY  (`CEdualRucPasaporte`,`iDPerfil`),
  KEY `FK_rperusrper` (`iDPerfil`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
SET character_set_client = @saved_cs_client;


--
-- Dumping data for table `usuarioperfil`
--

LOCK TABLES `usuarioperfil` WRITE;
/*!40000 ALTER TABLE `usuarioperfil` DISABLE KEYS */;
INSERT INTO `usuarioperfil` VALUES
('1703364453',1),('1705547527',2),('1702122234',3),('1700000001',10),('1700000002',10),('170000000
3',10),('1700000004',10),('1700000005',10),('1700000006',10),('1700000007',10);
/*!40000 ALTER TABLE `usuarioperfil` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2011-05-16  3:47:48
```

Para importar al servidor de base de datos destino, se emplea el siguiente comando linux: mysql -u root -p dbseguraves2 < dbseav150511.sql

ANEXO D. Desarrollo de Scripts, MOF, sh, cfg, vmx

Scripts Java

Clase Main:

```
/**
 *
 * @authors: Walter Fuertes - Fausto Meneses - Luis Guerra - FermÃn GalÃ¡n
 * date: FEB/2011
 *
 **/

package vne_manv2;
public class Main {

 //                              CIMOM
 private static String hst = "192.168.0.102";
/// User:
 private static String usr = "fausto";
// private static String usr = "walter";
 // Password:
 private static String pw = "fausto";
// private static String pw = "jr2605";

  public static void main(String[] args) {
    new loginDialog(hst, usr, pw).setVisible(true);
  }
}
```

Clase Login Dialog:

```
/*
 * loginDialog.java
 * Created on 10/11/2010, 09:49:53 PM
 */

/**
 * @authors: Walter Fuertes - Fausto Meneses - Luis Guerra - FermÃn GalÃ¡n
 * date: OCT/2011
 */
package vne_manv2;
public class loginDialog extends javax.swing.JDialog {
    /** A return status code - returned if Cancel button has been pressed */
    public static final int RET_CANCEL = 0;
    /** A return status code - returned if OK button has been pressed */
    public static final int RET_OK = 1;

    private static String hst;
 // User:
    private static String usr;
 // Password:
    private static String pw;
 // Objeto DB:
    private static VNE_Man_DataBase objDB;
 // Numero de escenarios:
    private static int nes;
 // Escenarios:
  //  private static String [] esc;

    /** Creates new form loginDialog */
    public loginDialog(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
    }
```

```
//    public loginDialog(String h1, String h2, String u, String p) {
    public loginDialog(String h, String u, String p) {
        hst = h;
        usr = u;
        pw = p;
        initComponents();
    }

    /** @return the return status of this dialog - one of RET_OK or RET_CANCEL */
    public int getReturnStatus() {
        return returnStatus;
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-   BEGIN:initComponents
    private void initComponents() {

        okButton = new javax.swing.JButton();
        cancelButton = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jLabel2 = new javax.swing.JLabel();
        jPasswordField1 = new javax.swing.JPasswordField();

        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                closeDialog(evt);
            }
        });

        okButton.setText("Continuar");
        okButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                okButtonActionPerformed(evt);
            }
        });
        okButton.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyPressed(java.awt.event.KeyEvent evt) {
                okButtonKeyPressed(evt);
            }
        });

        cancelButton.setText("Cancelar");
        cancelButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cancelButtonActionPerformed(evt);
            }
        });

        jLabel1.setText("Usuario:");

        jTextField1.setName("txtID"); // NOI18N

        jLabel2.setText("ContraseÃ±a:");

        jPasswordField1.setName("pwdClave"); // NOI18N

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addGap(121, 121, 121)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel2)
```

```
                .addComponent(jLabel1))
            .addGap(4, 4, 4)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 103,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 16,
Short.MAX_VALUE))
                .addComponent(jPasswordField1, javax.swing.GroupLayout.DEFAULT_SIZE, 119,
Short.MAX_VALUE)
                .addGroup(layout.createSequentialGroup()
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(okButton, javax.swing.GroupLayout.PREFERRED_SIZE, 113,
javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(cancelButton)
            .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addGap(115, 115, 115)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
              .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(jPasswordField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 79,
Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(cancelButton)
                    .addComponent(okButton))
                .addContainerGap())
        );
        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void okButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_okButtonActionPerformed
    try {
        String id = jTextField1.getText(),
                clave = jPasswordField1.getText();
       // ConecciÃ³n a la base de datos:
       //  VNE_Man_DataBase objDB = new VNE_Man_DataBase(hst[1]); Error: Host '192.168.0.103' is
not allowed to connect to this MySQL server
      objDB = new VNE_Man_DataBase("127.0.0.1");
       if (!objDB.connect()) {
          System.out.println("Error: no se pudo conectar a la base de datos");
          objDB.close();
          return;
       }
       if (!objDB.login(id, clave)) {
          System.out.println("Usuario: " + id + "o contraseÃ±a incorrectos");
          objDB.close();
          return;
       }
//       esc = new String[10];
//       nes = objDB.getDescEsc(esc);
       VNE_Man_Start vns =  new VNE_Man_Start(nes, hst, usr, pw, objDB);
       objDB.close();
    }
    catch (Exception ex ) {
      ex.getMessage();
      return;
    }
      // fmb  doClose(RET_OK);
```

```
    }//GEN-LAST:event_okButtonActionPerformed

    private void cancelButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_cancelButtonActionPerformed
        doClose(RET_CANCEL);
    }//GEN-LAST:event_cancelButtonActionPerformed

    /** Closes the dialog */
    private void closeDialog(java.awt.event.WindowEvent evt) {//GEN-FIRST:event_closeDialog
        doClose(RET_CANCEL);
    }//GEN-LAST:event_closeDialog

    private void okButtonKeyPressed(java.awt.event.KeyEvent evt) {//GEN-
FIRST:event_okButtonKeyPressed
        // TODO add your handling code here:
    }//GEN-LAST:event_okButtonKeyPressed

    private void doClose(int retStatus) {
        returnStatus = retStatus;
        setVisible(false);
        dispose();
    }

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
            loginDialog dialog = new loginDialog(new javax.swing.JFrame(), true);
                dialog.addWindowListener(new java.awt.event.WindowAdapter() {
                    public void windowClosing(java.awt.event.WindowEvent e) {
                        System.exit(0);
                    }
                });
                dialog.setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton cancelButton;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JPasswordField jPasswordField1;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JButton okButton;
    // End of variables declaration//GEN-END:variables

    private int returnStatus = RET_CANCEL;
}
```

Clase VNE_Man_DataBase:

```
package vne_manv2;

import java.sql.*;
/**
 * @authors: Walter Fuertes - Fausto Meneses - Luis Guerra - FermÃn GalÃ¡n
 * date: OCT/2011
 */

public class VNE_Man_DataBase {
    // Datos miembro:
    private String id;
    private String contrasenia;
    private String urlParcial;
    private String url;
    private Connection dbConn;
```

```
private ResultSet rs;

// Funciones miembro:
public VNE_Man_DataBase(String pIP)
{
    urlParcial = pIP;
}

public boolean connect()
{
    try {
       Class.forName("com.mysql.jdbc.Driver").newInstance();
    }
    catch (Exception e) {
       System.out.println("Error coneccion a db (1): "+ e.toString());
       return false;
   }
   try {
      url = "jdbc:mysql://" + urlParcial + "/dbseguraves";
      dbConn = DriverManager.getConnection(url, "root", "espel");
      System.out.println("Coneccion a db exitosa");
      return true;
   }
   catch (Exception e) {
       System.out.println("Error coneccion a db (2): "+ e.toString());
       return false;
   }
}

public boolean login(String pID, String pClave)
{
    String sqlSl;
    id = pID;
    contrasenia = pClave;
    try {
       Statement sqlCom = dbConn.createStatement();
       sqlSl = "SELECT p.CEdualRucPasaporte, p.Apellidos, p.Nombres, ";
       sqlSl += "e.descEsc ";
       sqlSl += "FROM persona p, usuario u, escenario e, perfil r, ";
       sqlSl += "usuarioperfil s, perfilescenario f ";
       sqlSl += "WHERE (p.CEdualRucPasaporte = u.CEdualRucPasaporte) AND ";
       sqlSl += "(u.CEdualRucPasaporte = s.CEdualRucPasaporte) AND ";
       sqlSl += "(r.iDPerfil = s.iDPerfil) AND ";
       sqlSl += "(r.iDPerfil = f.iDPerfil) AND ";
       sqlSl += "(e.iDEscenario = f.iDEscenario) AND ";
    sqlSl += "(u.id = '" + pID + "') AND (u.contrasenia = '" + pClave + "')";
       sqlSl += " ORDER BY e.descEsc DESC";
       rs = sqlCom.executeQuery(sqlSl);
       if (!rs.next()) return false;
   }
   catch (Exception e) {
       System.out.println("Error acceso a db (1): "+ e.toString());
       return false;
   }
   return true;
}

public boolean InsertPistaAuditoria(String esAc)
{
    String sqlIns;
    VNE_Man_GetEfficiency gEf = new VNE_Man_GetEfficiency();
    try {
       Statement sqlCom = dbConn.createStatement();
sqlIns = "INSERT INTO Pistas_Auditoria (CEdualRucPasaporte,
FechaPista, EstadoActual) VALUES ('";
       sqlIns += getCedula() + "', '";
       sqlIns += gEf.getDate() + "', '";
       sqlIns += esAc + "');";
       sqlCom.executeUpdate(sqlIns);
   }
```

```
  catch (Exception e) {
     System.out.println("Error acceso a db (2): "+ e.toString());
     return false;
  }
  return true;
}

public String getCedula()
{
   try {
     if (rs.first())
      return rs.getString("CEdualRucPasaporte");
     else
      return "";
   }
   catch (Exception e) {
     System.out.println("Error en getCedula(): "+ e.toString());
     return "";
  }
}

public String getApellidos()
{
   try {
     if (rs.first())
      return rs.getString("Apellidos");
     else
      return "";
   }
   catch (Exception e) {
     System.out.println("Error en getApellidos(): "+ e.toString());
     return "";
  }
}

public String getNombres()
{
   try {
     if (rs.first())
      return rs.getString("Nombres");
     else
      return "";
   }
   catch (Exception e) {
     System.out.println("Error en getNombres(): "+ e.toString());
     return "";
  }
}

public int getDescEsc(String [] de)
{
   int c = 0;
   boolean continua = true;
   try {
       if (!rs.first()) return c;
       while (continua) {
           de[c ++] = rs.getString("descEsc");
           if (!rs.next()) continua = false;
       }
     return c;
   }
   catch (Exception e) {
     System.out.println("Error en getXenID(): "+ e.toString());
     return 0;
  }
}

public void close()
{
  try {
```

```
        dbConn.close();
       }
       catch (Exception e) {
          System.out.println("Error al cerrar coneccion a db: "+ e.toString());
       }
    }
}
```

Clase VNE_Man_LoadBalance:

```
package vne_manv2;
/**
 *
 * @authors: Walter Fuertes - Fausto Meneses - Luis Guerra - FermÃn GalÃ¡n
 * date: JUN/2011
 **/

import java.io.PrintStream;
import java.io.StringWriter;
import java.io.FileOutputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class VNE_Man_LoadBalance {

    // Datos miembro de entorno:
    private String pathre;
    private String hst, user, passwr;
    private VNE_Man_Transfer obtr;


    // Funciones miembro

    // Constructor:

     public VNE_Man_LoadBalance (String pr, String h, String u, String p)
     {
        pathre = pr;
        hst = h;
        user = u;
        passwr = p;
     }


    public void genStatXenSh (String vIpC)
     {
      try {
        PrintStream ps = new PrintStream(new FileOutputStream(pathre +  "statXen.sh"));
                StringWriter s = new StringWriter();
        s.write("xm list > " + pathre + "maqvirtact.dat\n");
         s.write("scp " + pathre + "maqvirtact.dat root@" + vIpC + ":" + pathre + "\n");
         ps.print(s.toString());
         ps.close();
         obtr = new VNE_Man_Transfer(hst, user, passwr);
         obtr.Transfer(pathre + "statXen.sh", pathre + "statXen.sh", 0);
       }
       catch (FileNotFoundException ex1) {
       ex1.printStackTrace();
      }
     }


    public int findInFile(String x, String f)
     {
       int dt = 1, lon = x.length();
       String sl = "";
      try {
```

```
        FileReader ent = new FileReader(pathre + f);
        while (dt > 0) {
          char [] line = new char[120];
                dt = ent.read(line);
          if (dt < 0) break;
          sl = sl + String.valueOf(line);
          if (sl.indexOf(x) >= 0) {
             ent.close();
             return 1;
          }
        }

    //    System.out.println(sl);
        ent.close();
        return 0;
       }
       catch (IOException ex1) {
         ex1.printStackTrace();
      }
     return dt;
  }

}
```

Clase VNE_Man_Start:

```
package vne_manv2;

/**
 * @authors: Walter Fuertes - Fausto Meneses - Luis Guerra - FermÃn GalÃ¡n
 * date: OCT/2011
 */
public class VNE_Man_Start {

 // User:
    private static String usr;
 // Password:
    private static String pw;
 // Scenarios:
    private static String [] implem;

       // Plataforms:
   /* When local machine is Linux */
   String pathfilesource = "/usr/local/VNE_ManV2/resultsource/";
   String pathfile = "/usr/local/VNE_ManV2/results/";
              String instanceID = "vne-values";

    // Funciones miembro:
public VNE_Man_Start(int nes, String h, String u, String contrasen, VNE_Man_DataBase obDB)
    {
     try {
       usr = u;
       pw = contrasen;
              String ncs = "http://" + h + ":5988//root/cimv2184";
       String aux;
       implem = new String[10];
       nes = obDB.getDescEsc(implem);
       // DefiniciÃ³n e inicializaciÃ³n del objve:
      VNE_ManV2 objve = new VNE_ManV2(ncs, usr, pw, pathfilesource);
       // DefiniciÃ³n e inicializaciÃ³n del re:
       VNE_Man_RemoteExec re = new VNE_Man_RemoteExec();
       VNE_Man_GetEfficiency gEf = new VNE_Man_GetEfficiency();

 long startTime, elapsed, sTgS, eTgS = 0, strTgS, etrTgS = 0, sasTgS, easTgS = 0;
       int i;
       String [] scen = {"xenImplem3", "xen2Implem2"};
       String [] scx1 = new String[10];
```

```
      String [] scx2 = new String[10];
      int nvma1 = 0, nvma2 = 0;
      if (implem[0].equals("xenAssign")) {
        // Balanceo de carga:
         // get VMs  from scenarios:
         // Compare scenarios:
         // Get scenario with more VMs
         // if VMs available number > 0
         //    Generate and transfer file sh and cfg
         //    Return Host and VM.
         nvma1 = 0; nvma2 = 0;
         for (i = 0; i < 2; i ++) {
           //  Se conecta desde el CIMMOM
           if (objve.connect(instanceID, pathfile) > 0) { // Coneccion exitosa
            // Generar archivos sh y de configuraciÃ³n para el entorno virtual:
              if (objve.toSh(scen[i], false) > 0) {
                 VNE_Man_LoadBalance lb = new VNE_Man_LoadBalance(pathfile, objve.hst, objve.user,
objve.passwr);
                 sTgS = gEf.getStartTime();
                 System.out.println("Reading:  "+ scen[i]+" -> scenario, in the host " + objve.hst + " ...");
                 re.assignAddr(objve.user + "@" + objve.hst, objve.passwr);
                 eTgS = gEf.getElapsed(sTgS);
                 // Delete file maqvirtact.dat:
                 strTgS = gEf.getStartTime();
                 String [] coma2 = {"sh", pathfile +"elimaqvir.sh"};
                 Process t2 = Runtime.getRuntime().exec(coma2);
                 // Generate and transfer statXen.sh:
                 lb.genStatXenSh(objve.valIpClient);
                 // Transfer maqvirtact.dat:
                 re.remoteExec(". " + pathfile +"statXen.sh", 0);
                 // Incorpora retardo hasta que llegue maqvirtact.dat:
                 System.out.println("waiting for data transfer");
                 while (true) {
                   String [] coma1 = {"sh", pathfile +"getLlegadaArch.sh"};
                   Process t1 = Runtime.getRuntime().exec(coma1);
                   if (lb.findInFile("maqvirtact.dat", "lsarchenv.dat") > 0) break;
                 }
                 etrTgS = gEf.getElapsed(strTgS);
                 // Asigna y levanta VM:
                 sasTgS = gEf.getStartTime();
                 for (int j = 0; j < objve.vmc; j++) {
                   if (lb.findInFile(objve.vmn[j] + ".example.com", "maqvirtact.dat")== 0)
                     if (i == 0)
                       scx1[nvma1 ++] = "xenImplem1" + (j + 1);
                     else
                       scx2[nvma2 ++] = "xen2Implem1" + (j + 1);
                 }
                 easTgS = gEf.getElapsed(sasTgS);
              }
             }
           }

           // Assign scenario with more VMs available:
//           System.out.println("n1, n2 = " + nvma1 + " y " + nvma2);
           if (nvma1 + nvma2 == 0) {
             aux = "No hay mÃ¡quinas virtuales disponibles";
             System.out.println(aux);
             obDB.InsertPistaAuditoria(aux);
             return;
           }
           nes = 1;
           if (nvma1 >= nvma2)
             implem[0] = scx1[0];
           else
             implem[0] = scx2[0];
           System.out.println("Tiempos medidos para:" + implem[0]);
           aux = "Obtener los escenarios del CIMOM: " + eTgS + " ms";
           System.out.println(aux);
           obDB.InsertPistaAuditoria(aux);
           aux = "Transferir el estado del Hipervisor Xen del servidor al cliente: " + etrTgS + " ms";
```

```
            System.out.println(aux);
            obDB.InsertPistaAuditoria(aux);
            aux = "Seleccionar mÃ¡quina virtual para balanceo de carga: " + easTgS + " ms";
            System.out.println(aux);
            obDB.InsertPistaAuditoria(aux);
          }
        // Para cada plataforma:
        for (i = 0; i < nes; i ++) {
          //  Se conecta desde el CIMMOM
          if (objve.connect(instanceID, pathfile) > 0) { // Coneccion exitosa
            // Generar archivos sh y de configuraciÃ³n para el entorno virtual:
            if (objve.toSh(implem[i], true) > 0) {
              VNE_Man_LoadBalance lb = new VNE_Man_LoadBalance(pathfile, objve.hst, objve.user,
objve.passwr);
              System.out.println("Generation:  "+ implem[i]+" -> scenario, in the host " + objve.hst + " ...");
              // Asignar la address para la plataforma:
              re.assignAddr(objve.user + "@" + objve.hst, objve.passwr);
             // Si se encuentra instalada plataforma:
              // Levantar en forma remota el escenario virtual:
              re.remoteExec(". /usr/local/VNE_ManV2/results/exporter.sh", 0);
              for (int j = 0; j < objve.vmc; j++)
                if (objve.valPlataform.equals("xen")) {
                   // Generate and transfer statXen.sh:
                  lb.genStatXenSh(objve.valIpClient);
                  re.remoteExec(". " + pathfile +"statXen.sh", 0);
                  startTime = gEf.getStartTime();
                  if (lb.findInFile(objve.vmn[j] + ".example.com", "maqvirtact.dat")== 0) {
                     re.remoteExec(". " + pathfile + objve.vmn[j] + ".sh", 0);
                     Process p = Runtime.getRuntime().exec("xterm -T "+ objve.vmn[j] + " -e ssh " + objve.hst
+ " xm console " + objve.vmncfg[j]);
                     elapsed = gEf.getElapsed(startTime);
                     aux = "Tiempo transcurrido para levantar la mÃ¡quina " + objve.vmn[j] + ": " + elapsed + "
milisegundos";
                     System.out.println(aux);
                     obDB.InsertPistaAuditoria(aux);
                   }
                  else {
                  aux = "La mÃ¡quina " + objve.vmn[j] + " se encuentra levantada";
                     System.out.println(aux);
                     obDB.InsertPistaAuditoria(aux);
                   }
                  }
                  else {
                    try {
                     startTime = gEf.getStartTime();
                     Process q = Runtime.getRuntime().exec("ssh-agent sh -c â€™ssh-add < /dev/null &&
bashâ€™");
                     Process p = Runtime.getRuntime().exec("ssh -X " + objve.hst);
                     String [] coman = {"sh", pathfile + objve.vmn[j] + ".sh"};
                     Process r = Runtime.getRuntime().exec(coman);
                     elapsed = gEf.getElapsed(startTime);
                     aux = "Tiempo transcurrido para levantar la mÃ¡quina " + objve.vmn[j] + ": " + elapsed + "
milisegundos";
                     System.out.println(aux);
                     obDB.InsertPistaAuditoria(aux);
                    }
                    catch (Exception ex ) {
                      ex.getMessage();
                      return;
                    }
                  }
          }
          else {
            aux = "Plataforma " + objve.valPlataform + " No disponible";
             System.out.println(aux);
             obDB.InsertPistaAuditoria(aux);
          }
          System.out.println("ConecciÃ³n exitosa");
        }
        else {
```

```
      aux = "No se tuvo la ConecciÃ³n al CIMOM";
       System.out.println(aux);
       obDB.InsertPistaAuditoria(aux);
       break;
     }
    }
   }
   catch (Exception ex ) {
    ex.getMessage();
   }
  }
}
```

Clase_VNE_ManV2:

```
package vne_manv2;
/**
 * @authors: Walter Fuertes - Fausto Meneses - Luis Guerra - FermÃn GalÃ¡n
 * date: JUN/2011
 **/

import java.util.Enumeration;
import java.io.PrintStream;
import java.io.StringWriter;
import java.io.FileOutputStream;
import java.io.FileNotFoundException;

import javax.wbem.cim.CIMException;
import javax.wbem.cim.CIMNameSpace;
import javax.wbem.client.CIMClient;
import javax.wbem.client.PasswordCredential;
import javax.wbem.client.UserPrincipal;
import javax.wbem.cim.CIMObjectPath;
import javax.wbem.cim.CIMInstance;
// import javax.wbem.cim.UnsignedInt8;

public class VNE_ManV2 {

    // Datos miembro de entorno:
    private CIMClient cc;
    private CIMNameSpace cnns;
    private UserPrincipal usp;
    private PasswordCredential pcr;
    private String fileNamePath;
    private String fileNamePathTarget;
    private PrintStream ps;
    public String [] vmn;
    public String [] vmncfg;
    public String [] vmip4;
    public int vmc;
    public String hst;
    public String user;
    public String passwr;
    public String valPlataform;

    // Datos miembro de configuraciÃ³n:
    private String startPlataform;
    private String valVersion;
    private String valExecMode;
    private String valFilesystemType;
    private String valFilesystem;
    private String valIp1;
    private String valIpMask;
    private String valKernel;
    private String valRamDisk;
    private String valConsole;
    private String pathvm;
```

```
 private String vname;
 private VNE_Man_Transfer obtr;
 public String valIpClient;

   // Constructor:
 public VNE_ManV2(String cn, String up, String pc, String fnp)
 {
     cnns = new CIMNameSpace(cn);
     usp = new UserPrincipal(up);
     pcr = new PasswordCredential(pc);
     fileNamePath = fnp;
     vmn = new String[10];
     vmncfg = new String[50];
     vmip4 = new String[10];
 }

 // Connection to CIMOM:
 public int connect(String sinID, String patarget)
 {
     Enumeration en;
     String insID;
     CIMInstance inst;
     try {
        fileNamePathTarget = patarget;
        cc = new CIMClient(cnns, usp, pcr);
        en = cc.enumerateInstances(new CIMObjectPath("VNE_Configuration"));

     while (en.hasMoreElements()) {
      inst = (CIMInstance)en.nextElement();
      insID = String)inst.getProperty("InstanceID").getValue().getValue();
       if (insID.compareTo(sinID) == 0) {
      valVersion = (String)inst.getProperty("valVersion").getValue().getValue();
         valExecMode = (String)inst.getProperty("valExecMode").getValue().getValue();
         valFilesystemType = (String)inst.getProperty("valFilesystemType").getValue().getValue();
         valIp1 = (String)inst.getProperty("valIp1").getValue().getValue();
         valIpClient = valIp1 + (String)inst.getProperty("valIpClient").getValue().getValue();
         valIpMask = (String)inst.getProperty("valSubnetMask").getValue().getValue();
          break;
      }
      }
      return 1;
     }
     catch (CIMException e) {
        e.printStackTrace();
        return -1;
     }
 }

 // Generarate sh and configuration files:
 public int toSh(String scenarioName, boolean write) throws CIMException {

/* Get the VNE_Network instance that match scenarioName */
 vmc = 0;
CIMInstance scenario = null;
Enumeration e;
try {
   e = cc.enumerateInstances(new CIMObjectPath("VNE_Network"));
}
 catch (CIMException ex) {
   ex.printStackTrace();
  throw new CIMException("Error");
}
while (e.hasMoreElements()) {
   CIMInstance inst = (CIMInstance)e.nextElement();
   String name = (String)inst.getProperty("Name").getValue().getValue();
   if (name.equals(scenarioName)) {
       scenario = inst;
       hst = valIp1 + (String)inst.getProperty("ipv4Srv").getValue().getValue();
       valPlataform = (String)inst.getProperty("plataform").getValue().getValue();
       user = (String)inst.getProperty("user").getValue().getValue();
```

```
          passwr = (String)inst.getProperty("passw").getValue().getValue();
          startPlataform = (String)inst.getProperty("startPlataform").getValue().getValue();
          pathvm = (String)inst.getProperty("pathPlataform").getValue().getValue();
          valFilesystem = (String)inst.getProperty("valFilesystem").getValue().getValue();
          valKernel = (String)inst.getProperty("valKernel").getValue().getValue();
         valRamDisk = (String)inst.getProperty("valRamDisk").getValue().getValue();
          valConsole = (String)inst.getProperty("valConsole").getValue().getValue();
         break;
       }
     }
     if (scenario == null) {
         System.out.println("No scenario with name "+scenarioName+" was found");
         return -1;
      //  throw new CIMException("No scenario with name "+scenarioName+" was found");
     }
      if (startPlataform.equals("0")) return -1;
     try {
         if (write) obtr = new VNE_Man_Transfer(hst, user, passwr);
        vneGenVms(scenario, scenarioName, write);
     }
          catch (CIMException ex) {
         ex.printStackTrace();
         throw new CIMException("Error");
     }
      return 1;
    }

     // Generarate sh header file:
    private String vneGenHeader(String scenarioName) {
     StringWriter s = new StringWriter();
     /* Generate the usual file header */
     /* FIXME: it can be improved */
     s.write("# version:"+valVersion+"\n");
     s.write("# simulation_name:"+ scenarioName +"\n");
     s.write("# vm_defaults exec_mode='"+valExecMode+"'\n");
     s.write("# filesystem type='"+valFilesystemType+"'  "+valFilesystem+"\n");
     s.write("# kernel = '"+valKernel+"'\n");
     s.write("# console id = " + valConsole + "\n");
     return s.toString();
    }

     // Generate Virtual Machines:
     private void vneGenVms(CIMInstance scenario, String scenarioName, boolean wr) throws
CIMException {
         String  nvm, nvmsp, nvmaux;
         /* VNE_Network->VNE_ComputerSystem to generate <vm> */
         Enumeration e = cc.associators(scenario.getObjectPath(),
                 "VNE_SystemComponent",
                         "VNE_ComputerSystem",
                         "GroupComponent",
                         "PartComponent", false, false, null);
         while(e.hasMoreElements()) {
         CIMInstance cs = (CIMInstance)e.nextElement();
         vname = (String)cs.getProperty("Name").getValue().getValue();
          vmn[vmc] = vname;
          if (wr) {
          ps = null;
          try {
        ps = new PrintStream(new FileOutputStream(fileNamePath + vname + ".sh"));
          }
           catch (FileNotFoundException ex1) {
           ex1.printStackTrace();
           throw new CIMException("Error");
          }
           ps.print(vneGenHeader(scenarioName));
           // if VMWare plataform:
           nvmaux = "/usr/bin/vmplayer " + pathvm + vname + "/" + vname + ".vmx &\n" +
                 "sleep 10\n\n";
           nvmsp = vname + ".vmx";
           nvm = pathvm + "/" +  vname + "/" +  nvmsp;
```

```
        // else:
        if (valPlataform.compareTo("xen") == 0) {
          nvmsp = vname + ".example.com";
          nvm = pathvm + nvmsp;
          nvmaux = "";
          /* fmb 10/04/2011 19:25
          nvmaux =   "xm list > mqxenac.dat\n";
          nvmaux +=   "./localizar.sh mqxenac.dat " + nvmsp + "\n";
          nvmaux +=   "if [ $? -gt 0 ] \n";
          nvmaux +=   " then #localizado\n";
          nvmaux +=   "   return 2\n";
          nvmaux +=   "else #levantar la mÃ¡quina\n";
           */
          nvmaux +=   " xm create /etc/xen/" + nvmsp + ".cfg &\n " +
valConsole + " -T " + vname + " -e xm console " + vname + ".example.com &\n" +
               " sleep 15\n";
          /* fmb 10/04/2011 19:25
          nvmaux +=   " return 1\n";
          nvmaux +=   "fi\n";
          */
        }
        vmncfg[vmc ++] = nvmsp;
       ps.print(nvmaux);
        ps.close();
        /* fmb 10/04/2011 19:25
        try  {
final Process p = Runtime.getRuntime().exec("chmod +x " + fileNamePath + vname + ".sh");
        }
        catch (Exception ex)  {
        }
         * */
        obtr.Transfer(fileNamePath + vname + ".sh", fileNamePathTarget + vname + ".sh", vmc > 1 ? 1 :
0);
       vneGenVmNics(cs, nvm, nvmsp);
       }
       else vmc ++;
      }
    }

    // Generate Nics of Virtual Machines:
private void vneGenVmNics(CIMInstance vm, String nvima, String nvimasp)throws CIMException {
      PrintStream pst;
     pst = null;
      String aux = "";
      if (valPlataform.compareTo("xen") == 0) aux = ".cfg";

     try {
 pst = new PrintStream(new FileOutputStream(fileNamePath + nvimasp + aux));
     }
      catch (FileNotFoundException e) {
       e.printStackTrace();
       throw new CIMException("Error");
     }

      if (valPlataform.compareTo("xen") == 0) {
          pst.print("kernel      = '" + valKernel + "'\n");
          pst.print("ramdisk     = '" + valRamDisk + "'\n");
          pst.print("memory      = '96'\n\n");
      }
      else { // if VMWare plataform:
          pst.print("#!/usr/bin/vmware\n");
          pst.print(".encoding = \"UTF-8\"\n");
          pst.print("config.version = \"8\"\n");
          pst.print("virtualHW.version = \"7\"\n");
          pst.print("maxvcpus = \"4\"\n");
          pst.print("scsi0.present = \"TRUE\"\n");
          pst.print("scsi0.virtualDev = \"lsilogic\"\n");
          pst.print("memsize = \"128\"\n");
          pst.print("scsi0:0.present = \"TRUE\"\n");
          pst.print("scsi0:0.fileName = \"" + vname + ".vmdk\"\n");
```

```
        pst.print("ide1:0.present = \"TRUE\"\n");
        pst.print("ide1:0.fileName = \"/home/walter/Escritorio/debianiso/debian-503-i386-
businesscard.iso\"\n");
        pst.print("ide1:0.deviceType = \"cdrom-image\"\n");
        pst.print("floppy0.startConnected = \"FALSE\"\n");
        pst.print("floppy0.fileName = \"\"\n");
        pst.print("floppy0.autodetect = \"TRUE\"\n");
        pst.print("ethernet0.present = \"TRUE\"\n");
        pst.print("ethernet0.connectionType = \"bridged\"\n");
        pst.print("ethernet0.wakeOnPcktRcv = \"FALSE\"\n");
        pst.print("ethernet0.addressType = \"generated\"\n");
        pst.print("usb.present = \"TRUE\"\n");
        pst.print("ehci.present = \"TRUE\"\n");
        pst.print("sound.present = \"TRUE\"\n");
        pst.print("sound.fileName = \"-1\"\n");
        pst.print("sound.autodetect = \"TRUE\"\n");
        pst.print("pciBridge0.present = \"TRUE\"\n");
        pst.print("pciBridge4.present = \"TRUE\"\n");
        pst.print("pciBridge4.virtualDev = \"pcieRootPort\"\n");
        pst.print("pciBridge4.functions = \"8\"\n");
        pst.print("pciBridge5.present = \"TRUE\"\n");
        pst.print("pciBridge5.virtualDev = \"pcieRootPort\"\n");
        pst.print("pciBridge5.functions = \"8\"\n");
        pst.print("pciBridge6.present = \"TRUE\"\n");
        pst.print("pciBridge6.virtualDev = \"pcieRootPort\"\n");
        pst.print("pciBridge6.functions = \"8\"\n");
        pst.print("pciBridge7.present = \"TRUE\"\n");
        pst.print("pciBridge7.virtualDev = \"pcieRootPort\"\n");
        pst.print("pciBridge7.functions = \"8\"\n");
        pst.print("vmci0.present = \"TRUE\"\n");
        pst.print("roamingVM.exitBehavior = \"go\"\n");
        pst.print("displayName = \"" + vname + "\"\n");
        pst.print("guestOS = \"debian5\"\n");
        pst.print("nvram = \"" + vname + ".nvram\"\n");
        pst.print("virtualHW.productCompatibility = \"hosted\"\n");
        pst.print("gui.exitOnCLIHLT = \"FALSE\"\n");
        pst.print("extendedConfigFile = \"" + vname + ".vmxf\"\n");
        pst.print("checkpoint.vmState = \"\"\n");
       }

  /* Generate <nic> tags */
  Enumeration e = cc.associators(vm.getObjectPath(),
                        "VNE_HostedAccessPoint",
                        "VNE_IPProtocolEndpoint",
                        "Antecedent",
                        "Dependent", false, false, null);
  int ifId = 1;
  int ifLoopId = 1;

          while(e.hasMoreElements()) {
    CIMInstance ipe = (CIMInstance)e.nextElement();

    Enumeration ee = cc.associators(ipe.getObjectPath(),
                                    "VNE_MemberOfLink",
                                    "VNE_ConnectivityCollection",
                                    "Member",
                                    "Collection", false, false, null);
    /* We are considering that the Enumeration has only zero
     * (if loopback if) or one (if conventional if)
     * element, which is consistent with the MOF Class
     * definition for VNE
    */
    if (ee.hasMoreElements()) {
      CIMInstance lcc = (CIMInstance)ee.nextElement();
      String net = (String)lcc.getProperty("InstanceID").getValue().getValue();
    }
     // Search for <ipv4> addresses
    ee = cc.associators(ipe.getObjectPath(),
                        "VNE_ElementSettingData",
                        "VNE_StaticIPAssignmentSettingData",
```

```
                        "ManagedElement",
                        "SettingData", false, false, null);
     int i = 0;
    while(ee.hasMoreElements()) {
       CIMInstance ip4 = (CIMInstance)ee.nextElement();
        String ipAddr = valIp1 + (String)ip4.getProperty("IPv4Address").getValue().getValue();
        vmip4[++i] = ipAddr;
        String mask = valIpMask;
    }

    // Search for <ipv6> addresses
    ee = cc.associators(ipe.getObjectPath(),
                                  "VNE_ElementSettingData",
                                  "VNE_CompleteAddr",
                                  "ManagedElement",
                                  "SettingData", false, false, null);
//                                  "VNE_IPv6AssignmentSettingData",
     i = 0;
    while(ee.hasMoreElements()) {
      CIMInstance ip6 = (CIMInstance)ee.nextElement();
       String ipAddr = (String)ip6.getProperty("ipAddr2").getValue().getValue();
       if (valPlataform.compareTo("xen") == 0)
          ipAddr = "ip=" + vmip4[++i] + ",mac=" + ipAddr;
       String dev1 = (String)ip6.getProperty("device1").getValue().getValue();
       String dev2 = (String)ip6.getProperty("device2").getValue().getValue();
       if (valPlataform.compareTo("xen") == 0) {
           pst.print("root        = '/dev/xvda2 ro'\n" +
               "disk        = [\n" +
               "              '" + dev1 + ",\n" +
               "              '" + dev2 + ",\n" +
               "             ]\n\n");
           pst.print("name        = '" + nvimasp + "'\n\n");
           pst.print("vif         = [ '" + ipAddr + "' ]\n\n");
           pst.print("on_poweroff = 'destroy'\n" +
                   "on_reboot   = 'restart'\n" +
                   "on_crash    = 'restart'\n");
         }
         else { // if VMWare plataform:
           pst.print("ethernet0.generatedAddress = \"" + ipAddr + "\"\n");
           pst.print("uuid.location = \"" + dev1 + "\"\n");
           pst.print("uuid.bios = \"" + dev2 + "\"\n");
           pst.print("cleanShutdown = \"TRUE\"\n");
           pst.print("replay.supported = \"FALSE\"\n");
           pst.print("replay.filename = \"\"\n");
           pst.print("scsi0:0.redo = \"\"\n");
           pst.print("pciBridge0.pciSlotNumber = \"17\"\n");
           pst.print("pciBridge4.pciSlotNumber = \"21\"\n");
           pst.print("pciBridge5.pciSlotNumber = \"22\"\n");
           pst.print("pciBridge6.pciSlotNumber = \"23\"\n");
           pst.print("pciBridge7.pciSlotNumber = \"24\"\n");
           pst.print("scsi0.pciSlotNumber = \"16\"\n");
           pst.print("usb.pciSlotNumber = \"32\"\n");
           pst.print("sound.pciSlotNumber = \"34\"\n");
           pst.print("ehci.pciSlotNumber = \"35\"\n");
           pst.print("vmci0.pciSlotNumber = \"36\"\n");
           pst.print("vmotion.checkpointFBSize = \"7733248\"\n");
           pst.print("ethernet0.generatedAddressOffset = \"0\"\n");
           pst.print("vmci0.id = \"271222168\"\n");
           pst.print("tools.remindInstall = \"TRUE\"\n");
           pst.print("ethernet0.linkStatePropagation.enable = \"TRUE\"\n");
           pst.print("tools.syncTime = \"FALSE\"\n");
           pst.print("debugStub.linuxOffsets =
\"0x0,0xffffffff,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0\"\n");
           pst.print("svga.autodetect = \"FALSE\"\n");
           pst.print("svga.maxWidth = \"1600\"\n");
           pst.print("svga.maxHeight = \"1200\"\n");
           pst.print("svga.vramSize = \"7680000\"\n");
           pst.print("numvcpus = \"1\"\n");
           pst.print("vmi.present = \"TRUE\"\n");
           pst.print("vmi.pciSlotNumber = \"37\"\n");
```

```
            pst.print("ethernet0.pciSlotNumber = \"33\"\n");
          }
         }
  }
      if (valPlataform.compareTo("xen") == 0)
        obtr.Transfer(fileNamePath + nvimasp + aux, nvima + aux, 1);
      else {
        obtr.Transfer(fileNamePath + nvimasp + aux, nvima + aux, 1);
      }
   }

   // Destructor:
   protected void finalize() throws CIMException
   {
      cc.close();
   }
}
```

Clase_VNE_Man_RemoteExec:

```
package vne_manv2;
/**
* @authors: Walter Fuertes - Fausto Meneses - Luis Guerra - FermÃn GalÃ¡n
 * date: FEB/2011
**/

import com.jcraft.jsch.*;
import java.io.*;

public class VNE_Man_RemoteExec {
   // Datos Miembro:
   private String host, user, pwd;
   private Session session;
      // Funciones miembro:
   public void assignAddr(String ht, String p) {
      host = ht;
      user = host.substring(0, host.indexOf('@'));
      host = host.substring(host.indexOf('@')+1);
      pwd = p;
   }

   public void remoteExec(String command, int getsec) {
     try{
      JSch jsch=new JSch();
      session=jsch.getSession(user, host, 22);
      // username and password will be given via UserInfo interface.
      UserInfo ui=new VNE_Man_UserInfo(pwd);
      session.setUserInfo(ui);
      if (getsec == 0)
        session.connect();
      Channel channel=session.openChannel("exec");
      ((ChannelExec)channel).setCommand(command);
      // X Forwarding
      // channel.setXForwarding(true);
      //channel.setInputStream(System.in);
      channel.setInputStream(null);
      //channel.setOutputStream(System.out);
      //FileOutputStream fos=new FileOutputStream("/tmp/stderr");
      //((ChannelExec)channel).setErrStream(fos);
      ((ChannelExec)channel).setErrStream(System.err);
      InputStream in=channel.getInputStream();
      channel.connect();
      byte[] tmp=new byte[1024];
      while(true){
        while(in.available()>0){
          int i=in.read(tmp, 0, 1024);
          if(i<0)break;
          System.out.print(new String(tmp, 0, i));
```

```
          }
          if(channel.isClosed()){
            System.out.println("exit-status: "+channel.getExitStatus());
            break;
          }
          try{Thread.sleep(1000);}catch(Exception ee){}
        }
        channel.disconnect();
        if (getsec == 2)
          session.disconnect();
      }
      catch(Exception e){
        System.out.println(e);
      }
    }
}
```

Clase_VNE_Man_Transfer:

```
package vne_manv2;
/**
 * @authors: Walter Fuertes - Fausto Meneses - Luis Guerra - FermÃn GalÃ¡n
 * date: FEB/2011
 **/

import com.jcraft.jsch.*;
import java.io.*;

public class VNE_Man_Transfer {

    // Datos miembro de entorno:

    private String host, user, pwd;
    private Session session;
    private Channel channel;
    private UserInfo ui;

    // Funciones miembro

    // Constructor:

    public VNE_Man_Transfer (String h, String u, String p)
     {
        host = h;
        user = u;
        pwd = p;
     }

    public void Transfer(String lfile, String rfile, int getsec) {
      FileInputStream fis=null;
      try{

//      String lfile="/home/walter/arch1.txt";
       // String lfile="/home/fausto/Comandosbk";
       // String user="root";
//      String host="10.1.14.84";
       // String host="192.168.0.3";
       // String rfile="/home/scpPrueba1.txt";
        JSch jsch=new JSch();
        if (getsec == 0) {
          session=jsch.getSession(user, host, 22);
          // username and password will be given via UserInfo interface.
          ui=new VNE_Man_UserInfo(pwd);
          session.setUserInfo(ui);
          session.connect();
        }
        // exec 'scp -t rfile' remotely
        String command="scp -p -t "+rfile;
```

```
        channel=session.openChannel("exec");
        ((ChannelExec)channel).setCommand(command);

        // get I/O streams for remote scp
        OutputStream out=channel.getOutputStream();
        InputStream in=channel.getInputStream();

        channel.connect();

        if(checkAck(in)!=0){
                System.exit(0);
        }

// send "C0644 filesize filename", where filename should not include '/'
        long filesize=(new File(lfile)).length();
        command="C0644 "+filesize+" ";
        if(lfile.lastIndexOf('/')>0){
          command+=lfile.substring(lfile.lastIndexOf('/')+1);
        }
        else{
          command+=lfile;
        }
        command+="\n";
        out.write(command.getBytes()); out.flush();
        if(checkAck(in)!=0){
      System.exit(0);
        }

        // send a content of lfile

        fis=new FileInputStream(lfile);
        byte[] buf=new byte[1024];

        while(true){
          int len=fis.read(buf, 0, buf.length);
      if(len<=0) break;
          out.write(buf, 0, len); //out.flush();
        }


        fis.close();
        fis=null;

        // send '\0'
        buf[0]=0; out.write(buf, 0, 1); out.flush();
        if(checkAck(in)!=0){
      System.exit(0);
        }
        out.close();


        channel.disconnect();

        if (getsec == 2)
          session.disconnect();


//        System.exit(0);
      }
      catch(Exception e){
        System.out.println(e);
        try{if(fis!=null)fis.close();}catch(Exception ee){}
      }
    }


    static int checkAck(InputStream in) throws IOException{
      int b=in.read();
      // b may be 0 for success,
      //          1 for error,
```

```
      //          2 for fatal error,
      //          -1
      if(b==0) return b;
      if(b==-1) return b;

      if(b==1 || b==2){
        StringBuffer sb=new StringBuffer();
        int c;
        do {
     c=in.read();
     sb.append((char)c);
        }
        while(c!='\n');
        if(b==1){ // error
     System.out.print(sb.toString());
        }
        if(b==2){ // fatal error
     System.out.print(sb.toString());
        }
      }
      return b;
  }
 }
```

Clase_VNE_Man_UserInfo:

```
package vne_manv2;
/**
 * @authors: Walter Fuertes - Fausto Meneses, Luis Guerra - FermÃn GalÃ¡n
 * date: FEB/2011
 **/

import com.jcraft.jsch.*;
import java.awt.*;
import javax.swing.*;

public class VNE_Man_UserInfo implements UserInfo, UIKeyboardInteractive{
    // Datos miembro:
    private String passwd;


    // Funciones miembro:
    public VNE_Man_UserInfo(String p)
    {
        passwd = p;
    }

    public String getPassword(){ return passwd; }
    public boolean promptYesNo(String str){
      Object[] options={ "yes", "no" };
      /* int foo=JOptionPane.showOptionDialog(null, FMB
            str,"Warning",
            JOptionPane.DEFAULT_OPTION,
            JOptionPane.WARNING_MESSAGE,
            null, options, options[0]);
       */
       int foo = 0;
       return foo==0;
    }


   // String passwd;
    JTextField passwordField=(JTextField)new JPasswordField(20);

    public String getPassphrase(){ return null; }
    public boolean promptPassphrase(String message){ return true; }
    public boolean promptPassword(String message){
      Object[] ob={passwordField};
```

```
  /* int result= FMB
    JOptionPane.showConfirmDialog(null, ob, message,
                            JOptionPane.OK_CANCEL_OPTION);
  if(result==JOptionPane.OK_OPTION){
    passwd=passwordField.getText();
    return true;
  }
  else{
    return false;
  }
   */

  // FMB
  //passwd="faustor";
  return true;
  // FMB
}

public void showMessage(String message){
  JOptionPane.showMessageDialog(null, message);
}
final GridBagConstraints gbc =
  new GridBagConstraints(0,0,1,1,1,1,
                         GridBagConstraints.NORTHWEST,
                         GridBagConstraints.NONE,
                         new Insets(0,0,0,0),0,0);
private Container panel;
public String[] promptKeyboardInteractive(String destination,
                                  String name,
                                  String instruction,
                                  String[] prompt,
                                  boolean[] echo){
  panel = new JPanel();
  panel.setLayout(new GridBagLayout());

  gbc.weightx = 1.0;
  gbc.gridwidth = GridBagConstraints.REMAINDER;
  gbc.gridx = 0;
  panel.add(new JLabel(instruction), gbc);
  gbc.gridy++;

  gbc.gridwidth = GridBagConstraints.RELATIVE;

  JTextField[] texts=new JTextField[prompt.length];
  for(int i=0; i<prompt.length; i++){
    gbc.fill = GridBagConstraints.NONE;
    gbc.gridx = 0;
    gbc.weightx = 1;
    panel.add(new JLabel(prompt[i]),gbc);

    gbc.gridx = 1;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.weighty = 1;
    if(echo[i]){
      texts[i]=new JTextField(20);
    }
    else{
      texts[i]=new JPasswordField(20);
    }
    panel.add(texts[i], gbc);
    gbc.gridy++;
  }


  if(JOptionPane.showConfirmDialog(null, panel,
                             destination+": "+name,
                             JOptionPane.OK_CANCEL_OPTION,
                             JOptionPane.QUESTION_MESSAGE)
     ==JOptionPane.OK_OPTION){
    String[] response=new String[prompt.length];
```

```
      for(int i=0; i<prompt.length; i++){
        response[i]=texts[i].getText();
      }
  return response;
    }
    else{
      return null;  // cancel
    }
   }
 }
```

ANEXO E. Scripts MOF

Archivo configuration.mof:

```
// configuracion.mof, v1
//
// Copyright (C) 2011/JUN Walter Fuertes - Fausto Meneses - Luis Guerra


[Version("1.0"),
 Description(
    "VNE_Configuration describe los parÃ¡metros base." )]
class VNE_Configuration : CIM_SettingData {
   string valVersion;
   string valIp1;
     string valSubnetMask;
   string valExecMode;
   string valFilesystemType;
     string valIpClient;
};
```

Archivo main.mof:

```
// main.mof, v1
// Copyright (C) 2011/MAY Walter Fuertes - Fausto Meneses - Luis Guerra
// Main classes


[Version("1.0"),
 Description(
    "VNE_Network models the scenario itself, as a specialization of CIM_Network." )]
class VNE_Network : CIM_Network {
  string ipv4Srv;
  string plataform;
  string user;
  string passw;
  string startPlataform;
  string pathPlataform;
  string valFilesystem;
  string valKernel;
  string valRamDisk;
  string valConsole;
};

[Version("1.0"),
 Description("VNE_ComputerSystem , as a specialization of CIM_ComputerSystem." )]
class VNE_ComputerSystem : CIM_ComputerSystem {
};

[Version("1.0"),
 Description("VNE_SystemComponent , as a specialization of CIM_SystemComponent." )]
class VNE_SystemComponent : CIM_SystemComponent {
};
```

```
[Version("1.0"),
 Description("VNE_HostedAccessPoint , as a specialization of CIM_HostedAccessPoint." )]
class VNE_HostedAccessPoint : CIM_HostedAccessPoint {
};

[Version("1.0"),
 Description("VNE_IPProtocolEndpoint as a specialization of CIM_IPProtocolEndpoint." )]
class VNE_IPProtocolEndpoint : CIM_IPProtocolEndpoint {
};


[Version("1.0"),
 Description("VNE_StaticIPAssignmentSettingData as a specialization of
CIM_StaticIPAssignmentSettingData." )]
class VNE_StaticIPAssignmentSettingData : CIM_StaticIPAssignmentSettingData {
};


[Version("1.0"),
 Description("VNE_ElementSettingData , as a specialization of CIM_ElementSettingData." )]
class VNE_ElementSettingData : CIM_ElementSettingData {
};


[Version("1.0"),
 Description("VNE_ConnectivityCollection models the scenario links (i.e. node"
    "interconnections)." )]
class VNE_ConnectivityCollection : CIM_ConnectivityCollection {

    [Description("The max number of connection allowed in the VNE_ConnectivityCollection. "
        "For example, MaxConnections=2 implies a point-to-point connection." )]
    uint16 MaxConnections;
};


[Version("1.0"),
 Description("VNE_CompleteAddr objects model network interface"
    "IPv6 addresses. It is needed due to the existing VNE_StaticIPAssignmentSettingData" )]
class VNE_CompleteAddr : CIM_IPAssignmentSettingData {

    [Description ("The complete address that will be assigned to the ProtocolEndpoint.")]
    string ipAddr2;
    string generatedAddressOffset;
    string device1;
    string device2;
};


// association classes


[Association, Version("1.0"),
 Description("VNE_MemberOfCollection associate a VNELinkConnectivityConnection to"
    "the VNE_IPProtocolEndpoints belonging to it." )]
class VNE_MemberOfLink : CIM_MemberOfCollection {

    [Description("The LinkConnectivityCollection modeling the link." ),
     Override ("Collection")]
    VNE_ConnectivityCollection REF Collection;

    [Description("The IPProtocolEndPoint member of the link."),
     Override ("Member")]
    VNE_IPProtocolEndpoint REF Member;
};
```

Archivo valConfiguration.mof:

```
// valConfiguration.mof, v1
//
// Copyright (C) 2011/JUN Walter Fuertes - Fausto Meneses - Luis Guerra

instance of VNE_Configuration {
    InstanceID = "vne-values";
  valVersion = "3.3.1";
    valIp1 = "192.168.0.";
    valSubnetMask = "255.255.255.0";
  valExecMode = "wfconsole";
  valFilesystemType = "ext3";
    valIpClient = "102";
};
```

Archivo xenImplem3.mof:

```
// xenImplem3.mof, v1
//
// Copyright (C) 2011/FEB Walter Fuertes - Fausto Meneses - Luis Guerra
// Network (1)


instance of VNE_Network as $ENVIRONMENT {

  // key properties
  CreationClassName = "VNE_Network";
  Name = "xenImplem3";
    ipv4Srv ="101";
    plataform = "xen";
    user = "root";
    passw = "espe2009";
    startPlataform = "1";
    pathPlataform = "/etc/xen/";
    valFilesystem = "/def/sda";
    valKernel = "/boot/vmlinuz-2.6.27-14-server";
    valRamDisk = "/boot/initrd.img-2.6.27-14-server";
    valConsole = "xterm";
};


// the virtual machine (3)

instance of VNE_ComputerSystem as $VM1 {

  // key properties
  CreationClassName = "VNE_ComputerSystem";
  Name = "vm1";
};

instance of VNE_ComputerSystem as $VM2 {
  // key properties
  CreationClassName = "VNE_ComputerSystem";
  Name = "vm2";
};

instance of VNE_ComputerSystem as $VM3 {
  // key properties
  CreationClassName = "VNE_ComputerSystem";
  Name = "vm3";
};
// NICS (3)

instance of VNE_IPProtocolEndpoint as $VM1_NIC1 {
  // key properties
```

```
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vm1-nic1";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vm1";
};


instance of VNE_IPProtocolEndpoint as $VM2_NIC2 {
  // key properties
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vm2-nic2";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vm2";
};


instance of VNE_IPProtocolEndpoint as $VM3_NIC3 {
  // key properties
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vm3-nic3";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vm3";
};


// Complete Address (3):

instance of VNE_CompleteAddr as $VM1_CPAD_1 {

  // key properties
  InstanceID = "vm1-ip2";

// completeAddr = "ip=192.168.0.231,mac=00:16:3E:5C:6D:87";
  ipAddr2 = "00:16:3E:5C:6D:87";
      generatedAddressOffset = "0";
      device1 = "file:/home/xen/domains/xen1.example.com/swap.img,xvda1,w";
      device2 = "file:/home/xen/domains/xen1.example.com/disk.img,xvda2,w";
};


instance of VNE_CompleteAddr as $VM2_CPAD_1 {

  // key properties
  InstanceID = "vm2-ip2";

  //completeAddr = "ip=10.1.14.232,mac=00:16:3E:5C:6D:88";
  ipAddr2 = "00:16:3E:5C:6D:88";
      generatedAddressOffset = "0";
      device1 = "file:/home/xen/domains/xen2.example.com/swap.img,xvda1,w";
      device2 = "file:/home/xen/domains/xen2.example.com/disk.img,xvda2,w";
};


instance of VNE_CompleteAddr as $VM3_CPAD_1 {
  // key properties
  InstanceID = "vm3-ip2";

  // completeAddr = "ip=10.1.14.233,mac=00:16:3E:5C:6D:89";
  ipAddr2 = "00:16:3E:5C:6D:89";
      generatedAddressOffset = "0";
      device1 = "file:/home/xen/domains/xen3.example.com/swap.img,xvda1,w";
      device2 = "file:/home/xen/domains/xen3.example.com/disk.img,xvda2,w";
};
// IPV4 (3):

instance of VNE_StaticIPAssignmentSettingData as $VM1_IPV4_1 {

  // key properties
  InstanceID = "vm1-ip1";
  IPv4Address = "121";
```

```
  SubnetMask = "";
};

instance of VNE_StaticIPAssignmentSettingData as $VM2_IPV4_1 {

  // key properties
  InstanceID = "vm2-ip1";
  IPv4Address = "122";
  SubnetMask = "";
};

instance of VNE_StaticIPAssignmentSettingData as $VM3_IPV4_1 {

  // key properties
  InstanceID = "vm3-ip1";
  IPv4Address = "123";
  SubnetMask = "";
};

// the interconnection networks (1)

instance of VNE_ConnectivityCollection as $NET0 {

  // key properties
  InstanceID = "net0";
};

// interfaces associations to networks vm (3):

instance of VNE_SystemComponent {
  GroupComponent = $ENVIRONMENT;
  PartComponent = $VM1;
};

instance of VNE_SystemComponent {
  GroupComponent = $ENVIRONMENT;
  PartComponent = $VM2;
};

instance of VNE_SystemComponent {
  GroupComponent = $ENVIRONMENT;
  PartComponent = $VM3;
};

// interfaces associations to networks (3):

instance of VNE_MemberOfLink {
  Collection = $NET0;
  Member = $VM1_NIC1;
};

instance of VNE_MemberOfLink {
  Collection = $NET0;
  Member = $VM2_NIC2;
};

instance of VNE_MemberOfLink {
  Collection = $NET0;
  Member = $VM3_NIC3;
};
// interfaces associations to nodes (3):


instance of VNE_HostedAccessPoint {
  Antecedent = $VM1;
  Dependent = $VM1_NIC1;
};


instance of VNE_HostedAccessPoint {
```

```
  Antecedent = $VM2;
  Dependent = $VM2_NIC2;
};


instance of VNE_HostedAccessPoint {
  Antecedent = $VM3;
  Dependent = $VM3_NIC3;
};



// IPv4 associations to interfaces (3):


instance of VNE_ElementSettingData {
  ManagedElement = $VM1_NIC1;
  SettingData = $VM1_IPV4_1;
};


instance of VNE_ElementSettingData {
  ManagedElement = $VM2_NIC2;
  SettingData = $VM2_IPV4_1;
};


instance of VNE_ElementSettingData {
  ManagedElement = $VM3_NIC3;
  SettingData = $VM3_IPV4_1;
};



// Complete address associations to interfaces (3):


instance of VNE_ElementSettingData {
  ManagedElement = $VM1_NIC1;
  SettingData = $VM1_CPAD_1;
};


instance of VNE_ElementSettingData {
  ManagedElement = $VM2_NIC2;
  SettingData = $VM2_CPAD_1;
};


instance of VNE_ElementSettingData {
  ManagedElement = $VM3_NIC3;
  SettingData = $VM3_CPAD_1;
};
```

Archivo xenImplem11.mof:

```
// xenImplem11.mof, v1
// Copyright (C)2011/FEB Walter Fuertes-Fausto Meneses-Luis Guerra
// Network (1)

instance of VNE_Network as $ENVIRONMENT {
  // key properties
  CreationClassName = "VNE_Network";
  Name = "xenImplem11";
    ipv4Srv ="101";
    plataform = "xen";
    user = "root";
```

```
    passw = "espe2009";
    startPlataform = "1";
    pathPlataform = "/etc/xen/";
  valFilesystem = "/def/sda";
    valKernel = "/boot/vmlinuz-2.6.27-14-server";
    valRamDisk = "/boot/initrd.img-2.6.27-14-server";
    valConsole = "xterm";
};


// the virtual machine (1)

instance of VNE_ComputerSystem as $VM1 {
  // key properties
  CreationClassName = "VNE_ComputerSystem";
  Name = "vm1";
};


// NICS (1)

instance of VNE_IPProtocolEndpoint as $VM1_NIC1 {
  // key properties
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vm1-nic1";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vm1";
};


// Complete Address (1):

instance of VNE_CompleteAddr as $VM1_CPAD_1 {
  // key properties
  InstanceID = "vm1-ip2";

// completeAddr = "ip=192.168.0.231,mac=00:16:3E:5C:6D:87";
  ipAddr2 = "00:16:3E:5C:6D:87";
      generatedAddressOffset = "0";
      device1 = "file:/home/xen/domains/xen1.example.com/swap.img,xvda1,w";
      device2 = "file:/home/xen/domains/xen1.example.com/disk.img,xvda2,w";
};

// IPV4 (1):

instance of VNE_StaticIPAssignmentSettingData as $VM1_IPV4_1 {
  // key properties
  InstanceID = "vm1-ip1";
  IPv4Address = "121";
  SubnetMask = "";
};

// the interconnection networks (1)

instance of VNE_ConnectivityCollection as $NET0 {

  // key properties
  InstanceID = "net0";
};

// interfaces associations to networks vm (1):

instance of VNE_SystemComponent {
  GroupComponent = $ENVIRONMENT;
  PartComponent = $VM1;
};

// interfaces associations to networks (1):

instance of VNE_MemberOfLink {
```

```
  Collection = $NET0;
  Member = $VM1_NIC1;
};

// interfaces associations to nodes (1):

instance of VNE_HostedAccessPoint {
  Antecedent = $VM1;
  Dependent = $VM1_NIC1;
};

// IPv4 associations to interfaces (1):

instance of VNE_ElementSettingData {
  ManagedElement = $VM1_NIC1;
  SettingData = $VM1_IPV4_1;
};
// Complete address associations to interfaces (1):

instance of VNE_ElementSettingData {
  ManagedElement = $VM1_NIC1;
  SettingData = $VM1_CPAD_1;
};
```

Archivo xenImplem12.mof:

```
// xenImplem12.mof, v1
// Copyright (C) 2011/FEB Walter Fuertes - Fausto Meneses - Luis Guerra
// Network (1)
instance of VNE_Network as $ENVIRONMENT {

  // key properties
  CreationClassName = "VNE_Network";
  Name = "xenImplem12";
    ipv4Srv ="101";
    plataform = "xen";
    user = "root";
    passw = "espe2009";
    startPlataform = "1";
    pathPlataform = "/etc/xen/";
  valFilesystem = "/def/sda";
    valKernel = "/boot/vmlinuz-2.6.27-14-server";
    valRamDisk = "/boot/initrd.img-2.6.27-14-server";
    valConsole = "xterm";
};

// the virtual machine (1)

instance of VNE_ComputerSystem as $VM2 {

  // key properties
  CreationClassName = "VNE_ComputerSystem";
  Name = "vm2";
};


// NICS (1)

instance of VNE_IPProtocolEndpoint as $VM2_NIC2 {

  // key properties
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vm2-nic2";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vm2";
};
```

```
// Complete Address (1):

instance of VNE_CompleteAddr as $VM2_CPAD_1 {

  // key properties
  InstanceID = "vm2-ip2";

  //completeAddr = "ip=10.1.14.232,mac=00:16:3E:5C:6D:88";
  ipAddr2 = "00:16:3E:5C:6D:88";
      generatedAddressOffset = "0";
      device1 = "file:/home/xen/domains/xen2.example.com/swap.img,xvda1,w";
      device2 = "file:/home/xen/domains/xen2.example.com/disk.img,xvda2,w";
};

// IPV4 (1):

instance of VNE_StaticIPAssignmentSettingData as $VM2_IPV4_1 {

  // key properties
  InstanceID = "vm2-ip1";
  IPv4Address = "122";
  SubnetMask = "";
};

// the interconnection networks (1)

instance of VNE_ConnectivityCollection as $NET0 {

  // key properties
  InstanceID = "net0";
};

// interfaces associations to networks vm (1):

instance of VNE_SystemComponent {
  GroupComponent = $ENVIRONMENT;
  PartComponent = $VM2;
};

// interfaces associations to networks (1):

instance of VNE_MemberOfLink {
  Collection = $NET0;
  Member = $VM2_NIC2;
};
// interfaces associations to nodes (1):

instance of VNE_HostedAccessPoint {
  Antecedent = $VM2;
  Dependent = $VM2_NIC2;
};

// IPv4 associations to interfaces (1):

instance of VNE_ElementSettingData {
  ManagedElement = $VM2_NIC2;
  SettingData = $VM2_IPV4_1;
};

// Complete address associations to interfaces (1):

instance of VNE_ElementSettingData {
  ManagedElement = $VM2_NIC2;
  SettingData = $VM2_CPAD_1;
};
```

Archivo xenImplem13.mof:

```
// xenImplem13.mof, v1
// Copyright (C) 2011/FEB Walter Fuertes - Fausto Meneses - Luis Guerra

// Network (1)

instance of VNE_Network as $ENVIRONMENT {

  // key properties
  CreationClassName = "VNE_Network";
  Name = "xenImplem13";
    ipv4Srv ="101";
    plataform = "xen";
    user = "root";
    passw = "espe2009";
    startPlataform = "1";
    pathPlataform = "/etc/xen/";
  valFilesystem = "/def/sda";
    valKernel = "/boot/vmlinuz-2.6.27-14-server";
    valRamDisk = "/boot/initrd.img-2.6.27-14-server";
    valConsole = "xterm";
};

// the virtual machine (1)

instance of VNE_ComputerSystem as $VM3 {

  // key properties
  CreationClassName = "VNE_ComputerSystem";
  Name = "vm3";
};

// NICS (1)

instance of VNE_IPProtocolEndpoint as $VM3_NIC3 {

  // key properties
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vm3-nic3";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vm3";
};

// Complete Address (1):

instance of VNE_CompleteAddr as $VM3_CPAD_1 {

  // key properties
  InstanceID = "vm3-ip2";

  // completeAddr = "ip=10.1.14.233,mac=00:16:3E:5C:6D:89";
  ipAddr2 = "00:16:3E:5C:6D:89";
      generatedAddressOffset = "0";
      device1 = "file:/home/xen/domains/xen3.example.com/swap.img,xvda1,w";
      device2 = "file:/home/xen/domains/xen3.example.com/disk.img,xvda2,w";
};

// IPV4 (1):

instance of VNE_StaticIPAssignmentSettingData as $VM3_IPV4_1 {

  // key properties
  InstanceID = "vm3-ip1";
  IPv4Address = "123";
  SubnetMask = "";
};
```

```
// the interconnection networks (1)

instance of VNE_ConnectivityCollection as $NET0 {

  // key properties
  InstanceID = "net0";
};

// interfaces associations to networks vm (1):

instance of VNE_SystemComponent {
  GroupComponent = $ENVIRONMENT;
  PartComponent = $VM3;
};

// interfaces associations to networks (1):

instance of VNE_MemberOfLink {
  Collection = $NET0;
  Member = $VM3_NIC3;
};

// interfaces associations to nodes (1):

instance of VNE_HostedAccessPoint {
  Antecedent = $VM3;
  Dependent = $VM3_NIC3;
};

// IPv4 associations to interfaces (1):

instance of VNE_ElementSettingData {
  ManagedElement = $VM3_NIC3;
  SettingData = $VM3_IPV4_1;
};

// Complete address associations to interfaces (1):

instance of VNE_ElementSettingData {
  ManagedElement = $VM3_NIC3;
  SettingData = $VM3_CPAD_1;
};
```

Archivo xen2Implem2.mof:

```
// xen2Implem2.mof, v1
// Copyright (C) 2011/FEB Walter Fuertes - Fausto Meneses - Luis Guerra

// Network (1)


instance of VNE_Network as $ENVIRONMENT {

  // key properties
  CreationClassName = "VNE_Network";
  Name = "xen2Implem2";
    ipv4Srv ="101";
    plataform = "xen";
    user = "root";
    passw = "espe2009";
    startPlataform = "1";
    pathPlataform = "/etc/xen/";
  valFilesystem = "/def/sda";
    valKernel = "/boot/vmlinuz-2.6.27-14-server";
    valRamDisk = "/boot/initrd.img-2.6.27-14-server";
    valConsole = "xterm";
};
```

```
// the virtual machine (2)


instance of VNE_ComputerSystem as $VM4 {

  // key properties
  CreationClassName = "VNE_ComputerSystem";
  Name = "vm4";
};

instance of VNE_ComputerSystem as $VM5 {

  // key properties
  CreationClassName = "VNE_ComputerSystem";
  Name = "vm5";
};


// NICS (2)


instance of VNE_IPProtocolEndpoint as $VM4_NIC1 {

  // key properties
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vm4-nic1";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vm4";
};

instance of VNE_IPProtocolEndpoint as $VM5_NIC2 {

  // key properties
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vm5-nic2";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vm5";
};

// Complete Address (2):

instance of VNE_CompleteAddr as $VM4_CPAD_1 {

  // key properties
  InstanceID = "vm4-ip2";

// completeAddr = "ip=192.168.0.231,mac=00:16:3E:5C:6D:87";
  ipAddr2 = "00:16:3E:5C:6D:87";
      generatedAddressOffset = "0";
      device1 = "file:/home/xen/domains/xen4.example.com/swap.img,xvda1,w";
      device2 = "file:/home/xen/domains/xen4.example.com/disk.img,xvda2,w";
};

instance of VNE_CompleteAddr as $VM5_CPAD_1 {

  // key properties
  InstanceID = "vm5-ip2";

  //completeAddr = "ip=10.1.14.232,mac=00:16:3E:5C:6D:88";
  ipAddr2 = "00:16:3E:5C:6D:88";
      generatedAddressOffset = "0";
      device1 = "file:/home/xen/domains/xen5.example.com/swap.img,xvda1,w";
      device2 = "file:/home/xen/domains/xen5.example.com/disk.img,xvda2,w";
};


// IPV4 (2):

instance of VNE_StaticIPAssignmentSettingData as $VM4_IPV4_1 {
```

```
  // key properties
  InstanceID = "vm4-ip1";
  IPv4Address = "131";
  SubnetMask = "";
};

instance of VNE_StaticIPAssignmentSettingData as $VM5_IPV4_1 {

  // key properties
  InstanceID = "vm5-ip1";
  IPv4Address = "132";
  SubnetMask = "";
};


// the interconnection networks (1)

instance of VNE_ConnectivityCollection as $NET0 {

  // key properties
  InstanceID = "net1";
};

// interfaces associations to networks vm (2):

instance of VNE_SystemComponent {
  GroupComponent = $ENVIRONMENT;
  PartComponent = $VM4;
};

instance of VNE_SystemComponent {
  GroupComponent = $ENVIRONMENT;
  PartComponent = $VM5;
};


// interfaces associations to networks (2):

instance of VNE_MemberOfLink {
  Collection = $NET0;
  Member = $VM4_NIC1;
};

instance of VNE_MemberOfLink {
  Collection = $NET0;
  Member = $VM5_NIC2;
};

// interfaces associations to nodes (2):

instance of VNE_HostedAccessPoint {
  Antecedent = $VM4;
  Dependent = $VM4_NIC1;
};
instance of VNE_HostedAccessPoint {
  Antecedent = $VM5;
  Dependent = $VM5_NIC2;
};

// IPv4 associations to interfaces (2):

instance of VNE_ElementSettingData {
  ManagedElement = $VM4_NIC1;
  SettingData = $VM4_IPV4_1;
};

instance of VNE_ElementSettingData {
  ManagedElement = $VM5_NIC2;
  SettingData = $VM5_IPV4_1;
```

```
};

// Complete address associations to interfaces (2):

instance of VNE_ElementSettingData {
  ManagedElement = $VM4_NIC1;
  SettingData = $VM4_CPAD_1;
};

instance of VNE_ElementSettingData {
  ManagedElement = $VM5_NIC2;
  SettingData = $VM5_CPAD_1;
};
```

Archivo xen2Implem11.mof:

```
// xen2Implem11.mof, v1
// Copyright (C) 2011/FEB Walter Fuertes - Fausto Meneses - Luis Guerra

// Network (1)
instance of VNE_Network as $ENVIRONMENT {

  // key properties
  CreationClassName = "VNE_Network";
  Name = "xen2Implem11";
    ipv4Srv ="101";
    plataform = "xen";
    user = "root";
    passw = "espe2009";
    startPlataform = "1";
    pathPlataform = "/etc/xen/";
  valFilesystem = "/def/sda";
    valKernel = "/boot/vmlinuz-2.6.27-14-server";
    valRamDisk = "/boot/initrd.img-2.6.27-14-server";
    valConsole = "xterm";
};


// the virtual machine (1)

instance of VNE_ComputerSystem as $VM4 {

  // key properties
  CreationClassName = "VNE_ComputerSystem";
  Name = "vm4";
};


// NICS (1)

instance of VNE_IPProtocolEndpoint as $VM4_NIC1 {

  // key properties
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vm4-nic1";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vm4";
};


// Complete Address (1):

instance of VNE_CompleteAddr as $VM4_CPAD_1 {

  // key properties
  InstanceID = "vm4-ip2";

// completeAddr = "ip=192.168.0.231,mac=00:16:3E:5C:6D:87";
```

```
  ipAddr2 = "00:16:3E:5C:6D:87";
      generatedAddressOffset = "0";
      device1 = "file:/home/xen/domains/xen4.example.com/swap.img,xvda1,w";
      device2 = "file:/home/xen/domains/xen4.example.com/disk.img,xvda2,w";
};

// IPV4 (1):

instance of VNE_StaticIPAssignmentSettingData as $VM4_IPV4_1 {

  // key properties
  InstanceID = "vm4-ip1";
  IPv4Address = "131";
  SubnetMask = "";
};

// the interconnection networks (1)

instance of VNE_ConnectivityCollection as $NET0 {

  // key properties
  InstanceID = "net1";
};

// interfaces associations to networks vm (1):

instance of VNE_SystemComponent {
  GroupComponent = $ENVIRONMENT;
  PartComponent = $VM4;
};
// interfaces associations to networks (1):

instance of VNE_MemberOfLink {
  Collection = $NET0;
  Member = $VM4_NIC1;
};

// interfaces associations to nodes (1):

instance of VNE_HostedAccessPoint {
  Antecedent = $VM4;
  Dependent = $VM4_NIC1;
};

// IPv4 associations to interfaces (1):

instance of VNE_ElementSettingData {
  ManagedElement = $VM4_NIC1;
  SettingData = $VM4_IPV4_1;
};

// Complete address associations to interfaces (1):

instance of VNE_ElementSettingData {
  ManagedElement = $VM4_NIC1;
  SettingData = $VM4_CPAD_1;
};
```

Archivo xen2Implem12.mof:

```
// xen2Implem12.mof, v1
// Copyright (C) 2011/FEB Walter Fuertes - Fausto Meneses - Luis Guerra


// Network (1)
```

```
instance of VNE_Network as $ENVIRONMENT {

  // key properties
  CreationClassName = "VNE_Network";
  Name = "xen2Implem12";
    ipv4Srv ="101";
    plataform = "xen";
    user = "root";
    passw = "espe2009";
    startPlataform = "1";
    pathPlataform = "/etc/xen/";
    valFilesystem = "/def/sda";
    valKernel = "/boot/vmlinuz-2.6.27-14-server";
    valRamDisk = "/boot/initrd.img-2.6.27-14-server";
    valConsole = "xterm";
};


// the virtual machine (1)


instance of VNE_ComputerSystem as $VM5 {

  // key properties
  CreationClassName = "VNE_ComputerSystem";
  Name = "vm5";
};

// NICS (1)

instance of VNE_IPProtocolEndpoint as $VM5_NIC2 {

  // key properties
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vm5-nic2";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vm5";
};


// Complete Address (1):

instance of VNE_CompleteAddr as $VM5_CPAD_1 {

  // key properties
  InstanceID = "vm5-ip2";

  //completeAddr = "ip=10.1.14.232,mac=00:16:3E:5C:6D:88";
  ipAddr2 = "00:16:3E:5C:6D:88";
      generatedAddressOffset = "0";
      device1 = "file:/home/xen/domains/xen5.example.com/swap.img,xvda1,w";
      device2 = "file:/home/xen/domains/xen5.example.com/disk.img,xvda2,w";
};


// IPV4 (1):

instance of VNE_StaticIPAssignmentSettingData as $VM5_IPV4_1 {

  // key properties
  InstanceID = "vm5-ip1";
  IPv4Address = "132";
  SubnetMask = "";
};


// the interconnection networks (1)

instance of VNE_ConnectivityCollection as $NET0 {
```

```
  // key properties
  InstanceID = "net1";
};


// interfaces associations to networks vm (1):

instance of VNE_SystemComponent {
  GroupComponent = $ENVIRONMENT;
  PartComponent = $VM5;
};

// interfaces associations to networks (1):

instance of VNE_MemberOfLink {
  Collection = $NET0;
  Member = $VM5_NIC2;
};

// interfaces associations to nodes (1):

instance of VNE_HostedAccessPoint {
  Antecedent = $VM5;
  Dependent = $VM5_NIC2;
};
// IPv4 associations to interfaces (1):

instance of VNE_ElementSettingData {
  ManagedElement = $VM5_NIC2;
  SettingData = $VM5_IPV4_1;
};


// Complete address associations to interfaces (1):

instance of VNE_ElementSettingData {
  ManagedElement = $VM5_NIC2;
  SettingData = $VM5_CPAD_1;
};
```

Archivo vmwareImplem3.mof:

```
// vmwareImplem3.mof, v1
// Copyright (C)2011/FEB Walter Fuertes-Fausto Meneses-Luis Guerra

// Network (1)

instance of VNE_Network as $ENVIRONMENT {

  // key properties
  CreationClassName = "VNE_Network";
  Name = "vmwareImplem3";
    ipv4Srv ="106";
    plataform = "vmware";
    user = "root";
    passw = "fausto";
    startPlataform = "1";
    pathPlataform = "/var/lib/vmware/Virtual Machines/";
    valFilesystem = "";
    valKernel = "otherlinux";
    valRamDisk = "";
    valConsole = "pc1";
};


// the virtual machine (3)
```

```
instance of VNE_ComputerSystem as $VMW1 {

  // key properties
  CreationClassName = "VNE_ComputerSystem";
  Name = "vmw1";
};

instance of VNE_ComputerSystem as $VMW2 {

  // key properties
  CreationClassName = "VNE_ComputerSystem";
  Name = "vmw2";
};

instance of VNE_ComputerSystem as $VMW3 {

  // key properties
  CreationClassName = "VNE_ComputerSystem";
  Name = "vmw3";
};

// NICS (3)


instance of VNE_IPProtocolEndpoint as $VMW1_NIC1 {

  // key properties
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vmw1-nic1";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vmw1";
};

instance of VNE_IPProtocolEndpoint as $VMW2_NIC2 {

  // key properties
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vmw2-nic2";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vmw2";
};

instance of VNE_IPProtocolEndpoint as $VMW3_NIC3 {

  // key properties
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vmw3-nic3";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vmw3";
};


// IPV4 (3):

instance of VNE_StaticIPAssignmentSettingData as $VMW1_IPV4_1 {

  // key properties
  InstanceID = "vmw1-ip1";
  //IPv4Address = "10.1.14.241";
  IPv4Address = "221";
  SubnetMask = "";
};

instance of VNE_StaticIPAssignmentSettingData as $VMW2_IPV4_1 {

  // key properties
  InstanceID = "vmw2-ip1";
  IPv4Address = "222";
```

```
  SubnetMask = "";
};

instance of VNE_StaticIPAssignmentSettingData as $VMW3_IPV4_1 {

  // key properties
  InstanceID = "vmw3-ip1";
  IPv4Address = "223";
  SubnetMask = "";
};


// Complete Address (3):


instance of VNE_CompleteAddr as $VMW1_CPAD_1 {

  // key properties
  InstanceID = "vmw1-ip2";

  //completeAddr = "00:0c:29:65:7b:55";
  ipAddr2 = "00:0c:29:65:7b:55";
      generatedAddressOffset = "0";
      device1 = "56 4d c1 4b 0b 2a 10 ce-4c 78 96 0c 87 65 7b 55";
      device2 = "56 4d c1 4b 0b 2a 10 ce-4c 78 96 0c 87 65 7b 55";
};

instance of VNE_CompleteAddr as $VMW2_CPAD_1 {

  // key properties
  InstanceID = "vmw2-ip2";

  //completeAddr = "00:0c:29:a3:0d:df";
  ipAddr2 = "00:0c:29:a3:0d:df";
      generatedAddressOffset = "0";
      device1 = "56 4d 46 1d 2e 1b 0e 8d-9f ef 4b 7c 14 a3 0d df";
      device2 = "56 4d 46 1d 2e 1b 0e 8d-9f ef 4b 7c 14 a3 0d df";
};

instance of VNE_CompleteAddr as $VMW3_CPAD_1 {

  // key properties
  InstanceID = "vmw3-ip2";

  // completeAddr = "00:0c:29:c8:17:68";
  ipAddr2 = "00:0c:29:c8:17:68";
      generatedAddressOffset = "0";
      device1 = "56 4d a6 0e 6e 3f f3 9c-60 65 80 56 6d c8 17 68";
      device2 = "56 4d a6 0e 6e 3f f3 9c-60 65 80 56 6d c8 17 68";
};


// the interconnection networks (1)


instance of VNE_ConnectivityCollection as $NET1 {

  // key properties
  InstanceID = "net1";
};


// interfaces associations to networks vm (3):


instance of VNE_SystemComponent {
  GroupComponent = $ENVIRONMENT;
  PartComponent = $VMW1;
};
```

```
instance of VNE_SystemComponent {
   GroupComponent = $ENVIRONMENT;
   PartComponent = $VMW2;
};

instance of VNE_SystemComponent {
   GroupComponent = $ENVIRONMENT;
   PartComponent = $VMW3;
};

// interfaces associations to networks (3):

instance of VNE_MemberOfLink {
   Collection = $NET1;
   Member = $VMW1_NIC1;
};

instance of VNE_MemberOfLink {
   Collection = $NET1;
   Member = $VMW2_NIC2;
};

instance of VNE_MemberOfLink {
   Collection = $NET1;
   Member = $VMW3_NIC3;
};


// interfaces associations to nodes (3):


instance of VNE_HostedAccessPoint {
   Antecedent = $VMW1;
   Dependent = $VMW1_NIC1;
};

instance of VNE_HostedAccessPoint {
   Antecedent = $VMW2;
   Dependent = $VMW2_NIC2;
};

instance of VNE_HostedAccessPoint {
   Antecedent = $VMW3;
   Dependent = $VMW3_NIC3;
};


// IPv4 associations to interfaces (3):


instance of VNE_ElementSettingData {
   ManagedElement = $VMW1_NIC1;
   SettingData = $VMW1_IPV4_1;
};

instance of VNE_ElementSettingData {
   ManagedElement = $VMW2_NIC2;
   SettingData = $VMW2_IPV4_1;
};

instance of VNE_ElementSettingData {
   ManagedElement = $VMW3_NIC3;
   SettingData = $VMW3_IPV4_1;
};


// Complete address associations to interfaces (3):


instance of VNE_ElementSettingData {
```

```
  ManagedElement = $VMW1_NIC1;
  SettingData = $VMW1_CPAD_1;
};

instance of VNE_ElementSettingData {
  ManagedElement = $VMW2_NIC2;
  SettingData = $VMW2_CPAD_1;
};

instance of VNE_ElementSettingData {
  ManagedElement = $VMW3_NIC3;
  SettingData = $VMW3_CPAD_1;
};
```

Archivo vmwareImplem11.mof:

```
// vmwareImplem11.mof, v1
// Copyright (C) 2011/FEB Walter Fuertes - Fausto Meneses - Luis Guerra
// Network (1)

instance of VNE_Network as $ENVIRONMENT {

  // key properties
  CreationClassName = "VNE_Network";
  Name = "vmwareImplem11";
    ipv4Srv ="106";
    plataform = "vmware";
    user = "root";
    passw = "fausto";
    startPlataform = "1";
    pathPlataform = "/var/lib/vmware/Virtual Machines/";
    valFilesystem = "";
    valKernel = "otherlinux";
    valRamDisk = "";
    valConsole = "pc1";
};

// the virtual machine (1)

instance of VNE_ComputerSystem as $VMW1 {

  // key properties
  CreationClassName = "VNE_ComputerSystem";
  Name = "vmw1";
};

// NICS (1)

instance of VNE_IPProtocolEndpoint as $VMW1_NIC1 {

  // key properties
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vmw1-nic1";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vmw1";
};

// IPV4 (1):

instance of VNE_StaticIPAssignmentSettingData as $VMW1_IPV4_1 {

  // key properties
  InstanceID = "vmw1-ip1";
  //IPv4Address = "10.1.14.241";
  IPv4Address = "221";
  SubnetMask = "";
};
```

```
// Complete Address (1):

instance of VNE_CompleteAddr as $VMW1_CPAD_1 {

  // key properties
  InstanceID = "vmw1-ip2";

  //completeAddr = "00:0c:29:65:7b:55";
  ipAddr2 = "00:0c:29:65:7b:55";
      generatedAddressOffset = "0";
      device1 = "56 4d c1 4b 0b 2a 10 ce-4c 78 96 0c 87 65 7b 55";
      device2 = "56 4d c1 4b 0b 2a 10 ce-4c 78 96 0c 87 65 7b 55";
};

// the interconnection networks (1)

instance of VNE_ConnectivityCollection as $NET1 {

  // key properties
  InstanceID = "net1";
};

// interfaces associations to networks vm (1):

instance of VNE_SystemComponent {
  GroupComponent = $ENVIRONMENT;
  PartComponent = $VMW1;
};

// interfaces associations to networks (1):

instance of VNE_MemberOfLink {
  Collection = $NET1;
  Member = $VMW1_NIC1;
};

// interfaces associations to nodes (1):

instance of VNE_HostedAccessPoint {
  Antecedent = $VMW1;
  Dependent = $VMW1_NIC1;
};
// IPv4 associations to interfaces (1):

instance of VNE_ElementSettingData {
  ManagedElement = $VMW1_NIC1;
  SettingData = $VMW1_IPV4_1;
};

// Complete address associations to interfaces (1):

instance of VNE_ElementSettingData {
  ManagedElement = $VMW1_NIC1;
  SettingData = $VMW1_CPAD_1;
};
```

Archivo vmwareImplem12.mof:

```
// vmwareImplem12.mof, v1
// Copyright (C) 2011/FEB Walter Fuertes - Fausto Meneses - Luis Guerra

// Network (1)

instance of VNE_Network as $ENVIRONMENT {

  // key properties
  CreationClassName = "VNE_Network";
  Name = "vmwareImplem12";
```

```
    ipv4Srv ="106";
    plataform = "vmware";
    user = "root";
    passw = "fausto";
    startPlataform = "1";
    pathPlataform = "/var/lib/vmware/Virtual Machines/";
    valFilesystem = "";
    valKernel = "otherlinux";
    valRamDisk = "";
    valConsole = "pc1";
};


// the virtual machine (1)


instance of VNE_ComputerSystem as $VMW2 {

  // key properties
  CreationClassName = "VNE_ComputerSystem";
  Name = "vmw2";
};


// NICS (1)


instance of VNE_IPProtocolEndpoint as $VMW2_NIC2 {

  // key properties
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vmw2-nic2";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vmw2";
};


// IPV4 (1):


instance of VNE_StaticIPAssignmentSettingData as $VMW2_IPV4_1 {

  // key properties
  InstanceID = "vmw2-ip1";
  IPv4Address = "222";
  SubnetMask = "";
};


// Complete Address (1):


instance of VNE_CompleteAddr as $VMW2_CPAD_1 {

  // key properties
  InstanceID = "vmw2-ip2";

  //completeAddr = "00:0c:29:a3:0d:df";
  ipAddr2 = "00:0c:29:a3:0d:df";
      generatedAddressOffset = "0";
      device1 = "56 4d 46 1d 2e 1b 0e 8d-9f ef 4b 7c 14 a3 0d df";
      device2 = "56 4d 46 1d 2e 1b 0e 8d-9f ef 4b 7c 14 a3 0d df";
};


// the interconnection networks (1)


instance of VNE_ConnectivityCollection as $NET1 {
```

```
  // key properties
  InstanceID = "net1";
};


// interfaces associations to networks vm (1):

instance of VNE_SystemComponent {
  GroupComponent = $ENVIRONMENT;
  PartComponent = $VMW2;
};

// interfaces associations to networks (1):

instance of VNE_MemberOfLink {
  Collection = $NET1;
  Member = $VMW2_NIC2;
};

// interfaces associations to nodes (1):

instance of VNE_HostedAccessPoint {
  Antecedent = $VMW2;
  Dependent = $VMW2_NIC2;
};

// IPv4 associations to interfaces (1):

instance of VNE_ElementSettingData {
  ManagedElement = $VMW2_NIC2;
  SettingData = $VMW2_IPV4_1;
};

// Complete address associations to interfaces (1):

instance of VNE_ElementSettingData {
  ManagedElement = $VMW2_NIC2;
  SettingData = $VMW2_CPAD_1;
};
```

Archivo vmwareImplem13.mof:

```
// vmwareImplem13.mof, v1
// Copyright (C)2011/FEB Walter Fuertes-Fausto Meneses-Luis Guerra

// Network (1)
instance of VNE_Network as $ENVIRONMENT {

  // key properties
  CreationClassName = "VNE_Network";
  Name = "vmwareImplem13";
    ipv4Srv ="106";
    plataform = "vmware";
    user = "root";
    passw = "fausto";
    startPlataform = "1";
    pathPlataform = "/var/lib/vmware/Virtual Machines/";
  valFilesystem = "";
    valKernel = "otherlinux";
    valRamDisk = "";
    valConsole = "pc1";
};

// the virtual machine (1)
instance of VNE_ComputerSystem as $VMW3 {

  // key properties
  CreationClassName = "VNE_ComputerSystem";
```

```
  Name = "vmw3";
};
// NICS (1)
instance of VNE_IPProtocolEndpoint as $VMW3_NIC3 {

  // key properties
  CreationClassName = "VNE_IPProtocolEndpoint";
  Name = "vmw3-nic3";
  SystemCreationClassName = "VNE_ComputerSystem";
  SystemName = "vmw3";
};

// IPV4 (1):
instance of VNE_StaticIPAssignmentSettingData as $VMW3_IPV4_1 {

  // key properties
  InstanceID = "vmw3-ip1";
  IPv4Address = "723";
  SubnetMask = "";
};

// Complete Address (1):
instance of VNE_CompleteAddr as $VMW3_CPAD_1 {
  // key properties
  InstanceID = "vmw3-ip2";

  // completeAddr = "00:0c:29:c8:17:68";
  ipAddr2 = "00:0c:29:c8:17:68";
      generatedAddressOffset = "0";
      device1 = "56 4d a6 0e 6e 3f f3 9c-60 65 80 56 6d c8 17 68";
      device2 = "56 4d a6 0e 6e 3f f3 9c-60 65 80 56 6d c8 17 68";
};

// the interconnection networks (1)
instance of VNE_ConnectivityCollection as $NET1 {

  // key properties
  InstanceID = "net1";
};

// interfaces associations to networks vm (1):
instance of VNE_SystemComponent {
  GroupComponent = $ENVIRONMENT;
  PartComponent = $VMW3;
};

// interfaces associations to networks (1):
instance of VNE_MemberOfLink {
  Collection = $NET1;
  Member = $VMW3_NIC3;
};

// interfaces associations to nodes (1):
instance of VNE_HostedAccessPoint {
  Antecedent = $VMW3;
  Dependent = $VMW3_NIC3;
};

// IPv4 associations to interfaces (1):
instance of VNE_ElementSettingData {
  ManagedElement = $VMW3_NIC3;
  SettingData = $VMW3_IPV4_1;
};

// Complete address associations to interfaces (1):
instance of VNE_ElementSettingData {
  ManagedElement = $VMW3_NIC3;
  SettingData = $VMW3_CPAD_1;
};
```

ANEXO F. Scripts sh

Archivo para generar el repositorio CIMOM (reset_rootcimomv2.sh):

```
#!/bin/sh -x
#
#This program is free software; you can redistribute it and/or modify it #under the terms of the GNU General Public License as published by the #Free Software Foundation; either version 2 of the License, or (at your #option) any later version.
#
#This program is distributed in the hope that it will be useful, but #WITHOUT ANY WARRANTY; without even the implied warranty of #MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU #General Public License for more details.
#
#You should have received a copy of the GNU General Public License along #with this program; if not, write to the Free Software Foundation, Inc., #59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
#
#An online copy of the licence can be found at #http://www.gnu.org/copyleft/gpl.html
#
#Copyright (C) 2008 Fermin Galan Marquez
#Copyright (C) 2010 Walter Fuertes - Edison Lascano - Fausto Meneses - #Luis Guerra
#
# Reset the CIMOM repository.
#
# WARNINGS:
# - we asume that the file /usr/local/src/wbemservices-1.0.2.bin.zip exists
# - ensure the CIMOM server is stopped before using this script

# FIXME: unhardwire to generic variables in the preamble

PROJECT=/usr/local/VNE_ManV2
HOST=127.0.0.1
NS=root/cimv2184

# Reseting
cd /usr/local/src/
cp wbemservices-1.0.2.bin.zip /tmp/
cd /tmp/ && unzip wbemservices-1.0.2.bin.zip
cd /usr/local/src/wbemservices/cimom
rm -f logr/*
mv /tmp/wbemservices/cimom/logr/* logr/
rm -rf /tmp/wbemservices

# Starting the CIMOM
cd /usr/local/src/wbemservices/cimom/bin
sh start_cimom.sh

# Compiling the MOF Schema
cd /usr/local/src/wbemservices/mof/dmtf/
mkdir 2.18.2
#cd 2.18.2
#wget http://www.dmtf.org/standards/cim/cim_schema_v2181/cim_schema_2.18.1-Final-MOFs.zip
cd /usr/local/VNE_ManV2/mof_zip/
cp cim_schema_2.18.1-Final-MOFs.zip /usr/local/src/wbemservices/mof/dmtf/2.18.2/
cd /usr/local/src/wbemservices/mof/dmtf/2.18.2/
unzip cim_schema_2.18.1-Final-MOFs.zip
cd /usr/local/src/wbemservices/bin
sh mofcomp -u root -p x -n $NS -c $HOST /usr/local/src/wbemservices/mof/dmtf/2.18.2/cimv218.mof
rm -rf /usr/local/src/wbemservices/mof/dmtf/2.18.2

# Copying MOF to tmp

cd $PROJECT
rm -rf /tmp/mof
cp -r mof /tmp

# Expanding MOF
```

```
$PROJECT/scripts/expand_alias.pl /tmp/mof/xenImplem3.mof /tmp/mof/xenImplem3_e.mof
http://$HOST/$NS > /dev/null
# $PROJECT/scripts/expand_alias.pl /tmp/mof/xenImplem212.mof /tmp/mof/xenImplem212_e.mof
http://$HOST/$NS > /dev/null
# $PROJECT/scripts/expand_alias.pl /tmp/mof/xenImplem213.mof /tmp/mof/xenImplem213_e.mof
http://$HOST/$NS > /dev/null
# $PROJECT/scripts/expand_alias.pl /tmp/mof/xenImplem223.mof /tmp/mof/xenImplem223_e.mof
http://$HOST/$NS > /dev/null
$PROJECT/scripts/expand_alias.pl /tmp/mof/xenImplem11.mof /tmp/mof/xenImplem11_e.mof
http://$HOST/$NS > /dev/null
$PROJECT/scripts/expand_alias.pl /tmp/mof/xenImplem12.mof /tmp/mof/xenImplem12_e.mof
http://$HOST/$NS > /dev/null
$PROJECT/scripts/expand_alias.pl /tmp/mof/xenImplem13.mof /tmp/mof/xenImplem13_e.mof
http://$HOST/$NS > /dev/null
$PROJECT/scripts/expand_alias.pl /tmp/mof/xen2Implem2.mof /tmp/mof/xen2Implem2_e.mof
http://$HOST/$NS > /dev/null
$PROJECT/scripts/expand_alias.pl /tmp/mof/xen2Implem11.mof /tmp/mof/xen2Implem11_e.mof
http://$HOST/$NS > /dev/null
$PROJECT/scripts/expand_alias.pl /tmp/mof/xen2Implem12.mof /tmp/mof/xen2Implem12_e.mof
http://$HOST/$NS > /dev/null
$PROJECT/scripts/expand_alias.pl /tmp/mof/vmwareImplem3.mof /tmp/mof/vmwareImplem3_e.mof
http://$HOST/$NS > /dev/null
# $PROJECT/scripts/expand_alias.pl /tmp/mof/vmwareImplem212.mof
/tmp/mof/vmwareImplem212_e.mof http://$HOST/$NS > /dev/null
# $PROJECT/scripts/expand_alias.pl /tmp/mof/vmwareImplem213.mof
/tmp/mof/vmwareImplem213_e.mof http://$HOST/$NS > /dev/null
# $PROJECT/scripts/expand_alias.pl /tmp/mof/vmwareImplem223.mof
/tmp/mof/vmwareImplem223_e.mof http://$HOST/$NS > /dev/null
# $PROJECT/scripts/expand_alias.pl /tmp/mof/vmwareImplem11.mof
/tmp/mof/vmwareImplem11_e.mof http://$HOST/$NS > /dev/null
# $PROJECT/scripts/expand_alias.pl /tmp/mof/vmwareImplem12.mof
/tmp/mof/vmwareImplem12_e.mof http://$HOST/$NS > /dev/null
$PROJECT/scripts/expand_alias.pl /tmp/mof/vmwareImplem13.mof /tmp/mof/vmwareImplem13_e.mof
http://$HOST/$NS > /dev/null

# Compiling

cd /usr/local/src/wbemservices/bin

sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/main.mof
sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/configuration.mof
sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/valConfiguration.mof
sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/xenImplem3_e.mof
# sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/xenImplem212_e.mof
# sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/xenImplem213_e.mof
# sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/xenImplem223_e.mof
sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/xenImplem11_e.mof
sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/xenImplem12_e.mof
sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/xenImplem13_e.mof
sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/xen2Implem2_e.mof
sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/xen2Implem11_e.mof
sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/xen2Implem12_e.mof
sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/vmwareImplem3_e.mof
# sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/vmwareImplem212_e.mof
# sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/vmwareImplem213_e.mof
# sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/vmwareImplem223_e.mof
# sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/vmwareImplem11_e.mof
# sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/vmwareImplem12_e.mof
sh mofcomp -u root -p x -n $NS -c $HOST /tmp/mof/vmwareImplem13_e.mof
```

Archivo auxiliar para generar el repositorio CIMOM (expand_alias.pl):

```
#!/usr/bin/perl
#
# This program is free software; you can redistribute it and/or modify it #under the terms of the GNU General Public License as #published by the #Free Software Foundation; either version 2 of the #License, or (at your #option) any later version.
#
```

```
#This program is distributed in the hope that it will be useful, but #WITHOUT ANY WARRANTY; without even the implied warranty of #MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU #General Public License for more details.
#
#You should have received a copy of the GNU General Public License
#along with this program; if not, write to the Free Software Foundation, #Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
#
#An online copy of the licence can be found at http://www.gnu.org/copyleft/gpl.html
#
#Copyright (C) 2008 Fermin Galan Marquez
#
#The MOF speficiation document (DSP0004, )allows alias utilization #sections ), but it seems that some compilers (in particular, the mofcomp #that comes with WBEM Services)have problems when such alias are used.
#
#In order to help those compilers, this script peform automatic alias #expansion. Thus,it takes an input MOF files as alias and generate an #output MOF file in which the alias has been resolved. It needs as #additional argument the URL namespace to use in the expansion (eg., #"http://phoenix:5988/cimv2181").

#Usage: ./expand_alias.pl input_file output_file namespace_url

use strict;

my $ifile = shift;
my $ofile = shift;
my $url = shift;

#First, build a hash for the aliases. The key properties are identified #after the "key properties" line, after finding a empty line(this is  #ugly because of it involves implicit restrictions in the MOF file #format, but checking the CIM classes would be overwhelming for a simple #script like this one)

my %aliases;
open INPUT, "$ifile";

my $in_keys  = 0;

my $current_alias;

while (<INPUT>) {
  if (/(\w+) as \$(\w+)/) {
              $current_alias = $2;
              $aliases{$current_alias} = "$1.";
              print("alias $current_alias (class $1): ");
  }
  if (/key properties/) {
              $in_keys = 1;
  }
  if ((/(\w+)\W?=\W?"(.+)";/) && $in_keys) {
              my $l = $1;
              my $r = $2;
              if ($aliases{$current_alias} =~ /\.$/) {
                        $aliases{$current_alias} .= "$l=$r";
              }
              else {
                        $aliases{$current_alias} .= ",$l=$r";
              }
  }

            if ((/^$/) && $in_keys) {
              $in_keys = 0;
              print "$aliases{$current_alias}\n";
  }
}
close INPUT;

# Second, expand each alias with the string calculated in the previous step
# In addition, remove the "as $ALIAS" string
```

```
my $n_e = 0;
my $n_v = 0;
open INPUT, "$ifile";
open OUT, ">$ofile";
while (<INPUT>) {
  if (/(.*) as \$\w+(.*)/) {
        print OUT "$1$2\n";
  }
  else if (/(\w+)\W?=\W?"?\$(.+)"?;/) {
        foreach my $key (keys %aliases) {
              if ($2 eq $key) {
                    print "expanding $key\n";
                    $n_e++;
                    print OUT "\t$1 = \"".$url.":".$aliases{$key}."\";\n";
                    last;
              }
        }
  }
  else {
        print OUT "$_";
        $n_v++;
  }
}
close INPUT;
close OUT;
print "STATS: $n_e expansions, $n_v verbatim lines\n";
```

Archivo para levantar el repositorio CIMOM (StartCIMOM.sh):

```
#!/bin/sh -x
# Este programa levanta el repositorio de clases VNE dentro del CIMOM:
# Copyright (C) 2009 Walter Fuetes
# Starts the CIMOM repository.
#

export JAVA_HOME="/usr/lib/jvm/java-1.6.0-openjdk/"
cd /usr/local/src/wbemservices/cimom/bin/
sh start_cimom.sh
cd ../../bin
sh cimworkshop.sh
```

Archivo auxiliar para levantar la máquina virtual 1 de XEN (vm1.sh):

```
# version:3.3.1
# simulation_name:xenImplem11
# vm_defaults exec_mode='wfconsole'
# filesystem type='ext3'  /def/sda
# kernel = '/boot/vmlinuz-2.6.27-14-server'
# console id = xterm
 xm create /etc/xen/vm1.example.com.cfg &
 xterm -T vm1 -e xm console vm1.example.com & sleep 15
```

Archivo auxiliar para levantar la máquina virtual 2 de XEN (vm2.sh):

```
# version:3.3.1
# simulation_name:xenImplem12
# vm_defaults exec_mode='wfconsole'
# filesystem type='ext3'  /def/sda
# kernel = '/boot/vmlinuz-2.6.27-14-server'
# console id = xterm

 xm create /etc/xen/vm2.example.com.cfg &
 xterm -T vm2 -e xm console vm2.example.com & sleep 15
```

Archivo auxiliar para levantar la máquina virtual 3 de XEN (vm3.sh):

```
# version:3.3.1
# simulation_name:xenImplem13
# vm_defaults exec_mode='wfconsole'
# filesystem type='ext3'  /def/sda
# kernel = '/boot/vmlinuz-2.6.27-14-server'
# console id = xterm

 xm create /etc/xen/vm3.example.com.cfg &
 xterm -T vm3 -e xm console vm3.example.com & sleep 15
```

Archivo auxiliar para levantar la máquina virtual 4 de XEN (vm4.sh):

```
# version:3.3.1
# simulation_name:xen2Implem2
# vm_defaults exec_mode='wfconsole'
# filesystem type='ext3'  /def/sda
# kernel = '/boot/vmlinuz-2.6.27-14-server'
# console id = xterm

 xm create /etc/xen/vm4.example.com.cfg &
 xterm -T vm4 -e xm console vm4.example.com & sleep 15
```

Archivo auxiliar para levantar la máquina virtual 5 de XEN (vm5.sh):

```
# version:3.3.1
# simulation_name:xen2Implem2
# vm_defaults exec_mode='wfconsole'
# filesystem type='ext3'  /def/sda
# kernel = '/boot/vmlinuz-2.6.27-14-server'
# console id = xterm

 xm create /etc/xen/vm5.example.com.cfg &
 xterm -T vm5 -e xm console vm5.example.com & sleep 15
```

Archivo auxiliar para levantar la máquina virtual 1 de VMWare (vmw1.sh):

```
# version:3.3.1
# simulation_name:vmwareImplem3
# vm_defaults exec_mode='wfconsole'
# filesystem type='ext3'
# kernel = 'otherlinux'
# console id = pc1

/usr/bin/vmplayer /var/lib/vmware/Virtual\ Machines/vmw1/vmw1.vmx &
sleep 10
```

Archivo auxiliar para levantar la máquina virtual 2 de VMWare (vmw2.sh):

```
# version:3.3.1
# simulation_name:vmwareImplem3
# vm_defaults exec_mode='wfconsole'
# filesystem type='ext3'
# kernel = 'otherlinux'
# console id = pc1

/usr/bin/vmplayer /var/lib/vmware/Virtual\ Machines/vmw2/vmw2.vmx &
sleep 10
```

Archivo auxiliar para levantar la máquina virtual 3 de VMWare (vmw3.sh):

```
# version:3.3.1
# simulation_name:vmwareImplem13
# vm_defaults exec_mode='wfconsole'
# filesystem type='ext3'
# kernel = 'otherlinux'
# console id = pc1

/usr/bin/vmplayer /var/lib/vmware/Virtual\ Machines/vmw3/vmw3.vmx &
sleep 10
```

Archivo para eliminar el archivo maqvirtact.dat (elimaqvir.sh):

```
rm /usr/local/VNE_ManV2/results/maqvirtact.dat
```

Archivo para exportar remotamente la pantalla de XEN (exporter.sh):

```
export DISPLAY=192.168.0.101:0.0
```

Archivo para generar el archivo maqvirtact.dat y transferirlo a la estación cliente (statXen.sh):

```
xm list > /usr/local/VNE_ManV2/results/maqvirtact.dat
scp /usr/local/VNE_ManV2/results/maqvirtact.dat root@192.168.0.102:/usr/local/VNE_ManV2/results/
```

ANEXO G. Scripts cfg

Archivo para levantar la máquina virtual 1 de XEN (vm1.example.com.cfg):

```
kernel      = '/boot/vmlinuz-2.6.27-14-server'
ramdisk     = '/boot/initrd.img-2.6.27-14-server'
memory      = '96'

root        = '/dev/xvda2 ro'
disk        = [
          'file:/home/xen/domains/xen1.example.com/swap.img,xvda1,w',
            'file:/home/xen/domains/xen1.example.com/disk.img,xvda2,w',
            ]
name        = 'vm1.example.com'
vif         = [ 'ip=192.168.0.121,mac=00:16:3E:5C:6D:87' ]

on_poweroff = 'destroy'
on_reboot   = 'restart'
on_crash    = 'restart'
```

Archivo para levantar la máquina virtual 2 de XEN (vm2.example.com.cfg):

```
kernel      = '/boot/vmlinuz-2.6.27-14-server'
ramdisk     = '/boot/initrd.img-2.6.27-14-server'
memory      = '96'

root        = '/dev/xvda2 ro'
disk        = [
            'file:/home/xen/domains/xen2.example.com/swap.img,xvda1,w',
            'file:/home/xen/domains/xen2.example.com/disk.img,xvda2,w',
            ]
name        = 'vm2.example.com'

vif         = [ 'ip=192.168.0.122,mac=00:16:3E:5C:6D:88' ]

on_poweroff = 'destroy'
on_reboot   = 'restart'
on_crash    = 'restart'
```

Archivo para levantar la máquina virtual 3 de XEN (vm3.example.com.cfg):

```
kernel      = '/boot/vmlinuz-2.6.27-14-server'
ramdisk     = '/boot/initrd.img-2.6.27-14-server'
memory      = '96'

root        = '/dev/xvda2 ro'
disk        = [
            'file:/home/xen/domains/xen3.example.com/swap.img,xvda1,w',
            'file:/home/xen/domains/xen3.example.com/disk.img,xvda2,w',
            ]
name        = 'vm3.example.com'
vif         = [ 'ip=192.168.0.123,mac=00:16:3E:5C:6D:89' ]

on_poweroff = 'destroy'
on_reboot   = 'restart'
on_crash    = 'restart'
```

Archivo para levantar la máquina virtual 4 de XEN (vm4.example.com.cfg):

```
kernel      = '/boot/vmlinuz-2.6.27-14-server'
ramdisk     = '/boot/initrd.img-2.6.27-14-server'
memory      = '96'

root        = '/dev/xvda2 ro'
disk        = [
              'file:/home/xen/domains/xen4.example.com/swap.img,xvda1,w',
              'file:/home/xen/domains/xen4.example.com/disk.img,xvda2,w',
              ]

name        = 'vm4.example.com'

vif         = [ 'ip=192.168.0.131,mac=00:16:3E:5C:6D:87' ]

on_poweroff = 'destroy'
on_reboot   = 'restart'
on_crash    = 'restart'
```

Archivo para levantar la máquina virtual 5 de XEN (vm5.example.com.cfg):

```
kernel      = '/boot/vmlinuz-2.6.27-14-server'
ramdisk     = '/boot/initrd.img-2.6.27-14-server'
memory      = '96'

root        = '/dev/xvda2 ro'
disk        = [
              'file:/home/xen/domains/xen5.example.com/swap.img,xvda1,w',
              'file:/home/xen/domains/xen5.example.com/disk.img,xvda2,w',
              ]

name        = 'vm5.example.com'

vif         = [ 'ip=192.168.0.132,mac=00:16:3E:5C:6D:88' ]

on_poweroff = 'destroy'
on_reboot   = 'restart'
on_crash    = 'restart'
```

ANEXO H. Scripts vmx

Archivo para levantar la máquina virtual 1 de VMWare (vmw1.vmx):

```
#!/usr/bin/vmware.encoding = "UTF-8"
config.version = "8"
virtualHW.version = "7"
maxvcpus = "4"
scsi0.present = "TRUE"
scsi0.virtualDev = "lsilogic"
memsize = "128"
scsi0:0.present = "TRUE"
scsi0:0.fileName = "vmw1.vmdk"
ide1:0.present = "TRUE"
ide1:0.fileName = "/home/walter/Escritorio/debianiso/debian-503-i386-businesscard.iso"
ide1:0.deviceType = "cdrom-image"
floppy0.startConnected = "FALSE"
floppy0.fileName = ""
floppy0.autodetect = "TRUE"
ethernet0.present = "TRUE"
ethernet0.connectionType = "bridged"
ethernet0.wakeOnPcktRcv = "FALSE"
ethernet0.addressType = "generated"
usb.present = "TRUE"
ehci.present = "TRUE"
sound.present = "TRUE"
sound.fileName = "-1"
sound.autodetect = "TRUE"
pciBridge0.present = "TRUE"
pciBridge4.present = "TRUE"
pciBridge4.virtualDev = "pcieRootPort"
pciBridge4.functions = "8"
pciBridge5.present = "TRUE"
pciBridge5.virtualDev = "pcieRootPort"
pciBridge5.functions = "8"
pciBridge6.present = "TRUE"
pciBridge6.virtualDev = "pcieRootPort"
pciBridge6.functions = "8"
pciBridge7.present = "TRUE"
pciBridge7.virtualDev = "pcieRootPort"
pciBridge7.functions = "8"
vmci0.present = "TRUE"
roamingVM.exitBehavior = "go"
displayName = "vmw1"
guestOS = "debian5"
nvram = "vmw1.nvram"
virtualHW.productCompatibility = "hosted"
gui.exitOnCLIHLT = "FALSE"
extendedConfigFile = "vmw1.vmxf"
checkpoint.vmState = ""
ethernet0.generatedAddress = "00:0c:29:65:7b:55"
uuid.location = "56 4d c1 4b 0b 2a 10 ce-4c 78 96 0c 87 65 7b 55"
uuid.bios = "56 4d c1 4b 0b 2a 10 ce-4c 78 96 0c 87 65 7b 55"
cleanShutdown = "TRUE"
replay.supported = "FALSE"
replay.filename = ""
scsi0:0.redo = ""
pciBridge0.pciSlotNumber = "17"
pciBridge4.pciSlotNumber = "21"
pciBridge5.pciSlotNumber = "22"
pciBridge6.pciSlotNumber = "23"
pciBridge7.pciSlotNumber = "24"
scsi0.pciSlotNumber = "16"
usb.pciSlotNumber = "32"
sound.pciSlotNumber = "34"
ehci.pciSlotNumber = "35"
vmci0.pciSlotNumber = "36"
vmotion.checkpointFBSize = "7733248"
```

```
ethernet0.generatedAddressOffset = "0"
vmci0.id = "271222168"
tools.remindInstall = "TRUE"
ethernet0.linkStatePropagation.enable = "TRUE"
tools.syncTime = "FALSE"
debugStub.linuxOffsets = "0x0,0xffffffff,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0"
svga.autodetect = "FALSE"
svga.maxWidth = "1600"
svga.maxHeight = "1200"
svga.vramSize = "7680000"
numvcpus = "1"
vmi.present = "TRUE"
vmi.pciSlotNumber = "37"
ethernet0.pciSlotNumber = "33"
```

Archivo para levantar la máquina virtual 2 de VMWare (vmw2.vmx):

```
#!/usr/bin/vmware.encoding = "UTF-8"
config.version = "8"
virtualHW.version = "7"
maxvcpus = "4"
scsi0.present = "TRUE"
scsi0.virtualDev = "lsilogic"
memsize = "128"
scsi0:0.present = "TRUE"
scsi0:0.fileName = "vmw2.vmdk"
ide1:0.present = "TRUE"
ide1:0.fileName = "/home/walter/Escritorio/debianiso/debian-503-i386-businesscard.iso"
ide1:0.deviceType = "cdrom-image"
floppy0.startConnected = "FALSE"
floppy0.fileName = ""
floppy0.autodetect = "TRUE"
ethernet0.present = "TRUE"
ethernet0.connectionType = "bridged"
ethernet0.wakeOnPcktRcv = "FALSE"
ethernet0.addressType = "generated"
usb.present = "TRUE"
ehci.present = "TRUE"

sound.present = "TRUE"
sound.fileName = "-1"
sound.autodetect = "TRUE"
pciBridge0.present = "TRUE"
pciBridge4.present = "TRUE"
pciBridge4.virtualDev = "pcieRootPort"
pciBridge4.functions = "8"
pciBridge5.present = "TRUE"
pciBridge5.virtualDev = "pcieRootPort"
pciBridge5.functions = "8"
pciBridge6.present = "TRUE"
pciBridge6.virtualDev = "pcieRootPort"
pciBridge6.functions = "8"
pciBridge7.present = "TRUE"
pciBridge7.virtualDev = "pcieRootPort"
pciBridge7.functions = "8"
vmci0.present = "TRUE"
roamingVM.exitBehavior = "go"
displayName = "vmw2"
guestOS = "debian5"
nvram = "vmw2.nvram"
virtualHW.productCompatibility = "hosted"
gui.exitOnCLIHLT = "FALSE"
extendedConfigFile = "vmw2.vmxf"
checkpoint.vmState = ""
ethernet0.generatedAddress = "00:0c:29:a3:0d:df"
uuid.location = "56 4d 46 1d 2e 1b 0e 8d-9f ef 4b 7c 14 a3 0d df"
uuid.bios = "56 4d 46 1d 2e 1b 0e 8d-9f ef 4b 7c 14 a3 0d df"
cleanShutdown = "TRUE"
```

```
replay.supported = "FALSE"
replay.filename = ""
scsi0:0.redo = ""
pciBridge0.pciSlotNumber = "17"
pciBridge4.pciSlotNumber = "21"
pciBridge5.pciSlotNumber = "22"
pciBridge6.pciSlotNumber = "23"
pciBridge7.pciSlotNumber = "24"
scsi0.pciSlotNumber = "16"
usb.pciSlotNumber = "32"
sound.pciSlotNumber = "34"
ehci.pciSlotNumber = "35"
vmci0.pciSlotNumber = "36"
vmotion.checkpointFBSize = "7733248"
ethernet0.generatedAddressOffset = "0"
vmci0.id = "271222168"
tools.remindInstall = "TRUE"
ethernet0.linkStatePropagation.enable = "TRUE"
tools.syncTime = "FALSE"
debugStub.linuxOffsets = "0x0,0xffffffff,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0"
svga.autodetect = "FALSE"
svga.maxWidth = "1600"
svga.maxHeight = "1200"
svga.vramSize = "7680000"
numvcpus = "1"
vmi.present = "TRUE"
vmi.pciSlotNumber = "37"
ethernet0.pciSlotNumber = "33"
```

Archivo para levantar la máquina virtual 3 de VMWare (vmw3.vmx):

```
#!/usr/bin/vmware.encoding = "UTF-8"
config.version = "8"
virtualHW.version = "7"
maxvcpus = "4"

scsi0.present = "TRUE"
scsi0.virtualDev = "lsilogic"
memsize = "128"
scsi0:0.present = "TRUE"
scsi0:0.fileName = "vmw3.vmdk"
ide1:0.present = "TRUE"
ide1:0.fileName = "/home/walter/Escritorio/debianiso/debian-503-i386-businesscard.iso"
ide1:0.deviceType = "cdrom-image"
floppy0.startConnected = "FALSE"
floppy0.fileName = ""
floppy0.autodetect = "TRUE"
ethernet0.present = "TRUE"
ethernet0.connectionType = "bridged"
ethernet0.wakeOnPcktRcv = "FALSE"
ethernet0.addressType = "generated"
usb.present = "TRUE"
ehci.present = "TRUE"
sound.present = "TRUE"
sound.fileName = "-1"
sound.autodetect = "TRUE"

pciBridge0.present = "TRUE"
pciBridge4.present = "TRUE"
pciBridge4.virtualDev = "pcieRootPort"
pciBridge4.functions = "8"
pciBridge5.present = "TRUE"
pciBridge5.virtualDev = "pcieRootPort"
pciBridge5.functions = "8"
pciBridge6.present = "TRUE"
pciBridge6.virtualDev = "pcieRootPort"
pciBridge6.functions = "8"
pciBridge7.present = "TRUE"
```

```
pciBridge7.virtualDev = "pcieRootPort"
pciBridge7.functions = "8"

vmci0.present = "TRUE"

roamingVM.exitBehavior = "go"
displayName = "vmw3"
guestOS = "debian5"
nvram = "vmw3.nvram"

virtualHW.productCompatibility = "hosted"
gui.exitOnCLIHLT = "FALSE"

extendedConfigFile = "vmw3.vmxf"
checkpoint.vmState = ""
ethernet0.generatedAddress = "00:0c:29:c8:17:68"

uuid.location = "56 4d a6 0e 6e 3f f3 9c-60 65 80 56 6d c8 17 68"
uuid.bios = "56 4d a6 0e 6e 3f f3 9c-60 65 80 56 6d c8 17 68"

cleanShutdown = "TRUE"
replay.supported = "FALSE"
replay.filename = ""
scsi0:0.redo = ""

pciBridge0.pciSlotNumber = "17"
pciBridge4.pciSlotNumber = "21"
pciBridge5.pciSlotNumber = "22"
pciBridge6.pciSlotNumber = "23"
pciBridge7.pciSlotNumber = "24"
scsi0.pciSlotNumber = "16"
usb.pciSlotNumber = "32"
sound.pciSlotNumber = "34"
ehci.pciSlotNumber = "35"
vmci0.pciSlotNumber = "36"
vmotion.checkpointFBSize = "7733248"
ethernet0.generatedAddressOffset = "0"
vmci0.id = "271222168"
tools.remindInstall = "TRUE"

ethernet0.linkStatePropagation.enable = "TRUE"

tools.syncTime = "FALSE"
debugStub.linuxOffsets = "0x0,0xffffffff,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0"

svga.autodetect = "FALSE"
svga.maxWidth = "1600"
svga.maxHeight = "1200"
svga.vramSize = "7680000"

numvcpus = "1"
vmi.present = "TRUE"
vmi.pciSlotNumber = "37"
ethernet0.pciSlotNumber = "33"
```

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

MAESTRÍA EN INGENIERÍA DE SOFTWARE

CERTIFICADO

Se certifica que el presente trabajo que se encuentra en esta Tesis denominada “SISTEMA DE CONTROL Y SEGURIDADES PARA EL PROYECTO AVES (APLICACIÓN DE TECNOLOGÍAS DE VIRTUALIZACIÓN PARA LA ESPE) EMPLEANDO LA METODOLOGÍA AUP”, fue desarrollado por los señores Ingeniero Fausto Honorato Meneses Becerra e Ingeniero Luis Alberto Guerra Cruz, bajo nuestra supervisión.

En la ciudad de Latacunga, a los 03 día del mes de marzo de 2015.

_____
Ing. Walter Fuertes Díaz, Ph.D
Director de Tesis

_____
Ing. Javier Montaluisa, MsC
Coordinador (E) de la Maestría

_____
Dr. Rodrigo Vaca
Secretario Académico