ARQUITECTURA BASADA EN AGENTES INTELIGENTES PARA LA GESTIÓN DE CONFIGURACIÓN DE RED

Investigadores:
MARIA MACARENO MOJICA
EDELBERTO REYES TORREGROZA

UNIVERSIDAD DE CARTAGENA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
CARTAGENA DE INDIAS, 2012.

ARQUITECTURA BASADA EN AGENTES INTELIGENTES PARA LA GESTIÓN DE CONFIGURACIÓN DE RED

TESIS DE GRADO

GRUPO DE INVESTIGACION E-SOLUCIONES
Ingeniería De Software

Investigadores:
MARIA MACARENO MOJICA
EDELBERTO REYES TORREGROZA

Director De Proyecto:
Msc. MARTIN EMILIO MONROY RIOS

UNIVERSIDAD DE CARTAGENA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
CARTAGENA DE INDIAS, 2012.

UNIVERSIDAD DE CARTAGENA

Tesis de Grado: ARQUITECTURA BASADA EN AGENTES INTELIGENTES PARA LA GESTIÓN DE CONFIGURACIÓN DE RED

Autores: MARIA ISABEL MACARENO MOJICA
EDELBERTO ENRIQUE REYES TORREGROZA

Director: Msc. MARTIN EMILIO MONROY RIOS

Nota de Aceptación

_____

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Cartagena de Indias, ____ de _____ de 2012

RESUMEN

El crecimiento de las redes de computadoras y la heterogeneidad de los recursos actuales y servicios de telecomunicaciones, conlleva a que actividades de gestión de configuración de red, se tornen más complejas de ejecutar para el administrador de red, dada las diversas topologías o arquitecturas de servicios; lo que implica un proceso difícil de gestión, atado a retardos o pérdida de tiempo, susceptible al error humano y con un aumento de costos por la contratación de mayor personal cualificado; evidenciándose de esta manera, la necesidad de implementar una herramienta que de forma inteligente, optimice las labores de gestión de red.

Es por esto, que el presente proyecto de investigación, especificó una arquitectura basada en agentes inteligentes para facilitar la labor de gestión de configuración de red ejercida por el administrador. Para lograrlo, se optó por definir una ontología que propicie la comunicación y acceso a la información entre agentes, de tal forma que se comparta y reutilice conocimiento útil, en aras de automatizar dicha labor.

De esta forma, se obtuvo en calidad de resultado de la investigación, una arquitectura basada en agentes inteligentes para la gestión de configuración de red, cuya estructura se validó mediante la construcción, de un prototipo de agente inteligente, empleando JAVA como lenguaje de programación y apoyados en el framework JADE.

Palabras Clave:
Agente Inteligente, Arquitectura, Gestión de Configuración de Red, Ontología.

ABSTRACT

The growing of computer networks and the heterogeneity of existing resources and telecommunication services, this results that activities of management, it become more complex to implement for the network manager, given the various topologies and service architectures; which involves a difficult process management, tied to delays or lost time, susceptible to human mistakes and increased costs by hiring more qualified staff; thus evidencing, the need to implement a smart tool, for optimize network management tasks.

Therefore, that this research project, it specified an architecture based on intelligent agents to facilitate the tasks of network configuration management exercised by the administrator. To achieve it, it chose to make an ontology that encourages communication and access to information among agents, so that it shares and reuse useful knowledge, to automate this task.

Thus, it got as a result of the research, an architecture based on intelligent agents for configuration management, whose structure was validated by constructing a prototype intelligent agent.

Keywords:
Intelligent Agent, Architecture, Configuration Management, Ontology.

DEDICATORIA

A Dios el dueño de todo, para él siempre sea la gloria.

A nuestros padres por su amor, apoyo constante, consejos y los sacrificios que han tenido que realizar durante estos años.

A nuestros amigos, por estar en las buenas y en las malas.

A nuestros docentes, por enseñarnos que el verdadero valor de la educación no está en el dinero, si no en ser agentes de cambio para nuestra sociedad.

A nuestro tutor, Martin Monroy, por su inteligencia y paciencia en todo el proceso.

AGRADECIMIENTOS

A Dios nuestro creador, gracias por concedernos tanta fuerza, constancia, perseverancia, sabiduría y entendimiento, que hicieron posible el éxito y culminación de cada uno de los objetivos propuestos durante el transcurso de nuestra vida académica.

A nuestros padres, agradecemos infinitamente porque han sido un pilar fundamental durante la carrera, con su amor, motivación, apoyo incondicional, comprensión, consejos y llamados de atención, han logrado que hoy en día se vean los resultados de toda su entrega.

A nuestros profesores, gracias a todos ellos que sin esperar nada a cambio, han sido pilares en nuestro camino y así, forman parte de este logro que nos abre puertas inimaginables en nuestro desarrollo profesional, gracias por su intento de transmitir todos sus conocimientos, experiencia, buenos consejos, que fueron útiles tanto en nuestra vida académica como en nuestra formación personal. Sin lugar a duda este trabajo no pudo haberse realizado sin la formación recibida durante todos estos años en el programa de ingeniería de Sistemas.

A nuestro tutor Martin Emilio Monroy Ríos, a quien especialmente deseamos agradecer por su apoyo, paciencia, motivación y confianza en los buenos y malos momentos que pasamos en el desarrollo de este trabajo de grado; gracias a su orientación y dedicación llegamos a alcanzar este gran logro en nuestras vidas. En ocasiones fue la llave mágica que nos ayudó a abrir las puertas hacia el camino de las soluciones. De esta forma, más que nuestro tutor, se convirtió en nuestra inspiración para proyectarnos como el excelente profesional que queremos ser.

A nuestra colaboradora y amiga Aury Jaramillo, quien con su creatividad y empeño logró darle al prototipo ese diseño novedoso y llamativo que lo hiciese más amable.

Finalmente y no menos importante, A nuestros amigos y compañeros, Yessica Marrugo, Ronald Núñez, Beatriz Benítez, Verónica Gonzales, Iván Romero, José Llamas, Pedro

Ruiz, Ana Tarrá, Julissa Mendoza, Mayker Pájaro, Jesús Rodríguez, Camilo Velásquez, Alonso Montenegro, Francisco Carreño, Luis Villalobos, Gennys Carrasquilla y Hamid Pinilla; junto con quienes pasamos los momentos más memorables de nuestras vidas y que nos embargan hoy de muchas alegrías al recordarlos con tanta nostalgia. Locuras, alegrías, sufrimientos,

CONTENIDO

Pág.

ÍNDICE DE FIGURAS

Pág.

INTRODUCCIÓN

El concepto de gestión de red, ha tomado mayor acogida en aras de administrar eficaz y eficientemente los recursos de la misma con el propósito de asegurar su disponibilidad, debido al crecimiento en tamaño y complejidad de las redes de computadoras. El área funcional de la gestión de red, la gestión de configuración, se encarga de todas las tareas que tienen que ver con la recolección de información, modificación del comportamiento de los dispositivos y almacenamiento de la información que determina este comportamiento (GUERRERO CASTELEIRO, 2007).

Sin embargo, el crecimiento de las redes de computadoras y la heterogeneidad de los recursos actuales y servicios de telecomunicaciones, conlleva a que actividades relacionadas con la gestión de configuración (mantenimiento de inventario de red, salvaguarda y restauración de la configuración ante la detección de errores, control de estado, entre otras), se tornen más difíciles de efectuar para el administrador de red, dada las diversas topologías.

Teniendo en cuenta lo anterior, a comienzos de la década de los 90 aparecieron en escena los modelos de gestión de red integrada; como es el caso del protocolo simple de gestión de red o SNMP (Simple Network Management)(Cisco Systems, 2009); y la arquitectura Común de Intermediarios de Peticiones de Objetos o CORBA (Common Object Request Broker Architecture)(Object Management Group, 2012), y el modelo de gestión basada en Web de empresa o WBEM (Web Based Enterprise Management) que trata de establecer una arquitectura que permite la interoperabilidad entre modelos de gestión. También, con base en los avances alcanzados en la rama de la Inteligencia artificial, en relación con el uso de las ontologías (Web Semántica), se realizaron investigaciones relevantes como la realizada en el año 2003 por Jorge Enrique López de Vergara, en su tesis doctoral titulada: "*Especificación de modelos de información de*

Gestión de red integrada mediante el uso de ontologías y técnicas de representación del conocimiento"; y Antonio Guerrero Casteleiro en el año 2007, en su tesis "*Especificación del comportamiento de gestión de red mediante ontologías*".

En este orden de ideas, en el presente trabajo, se construye un prototipo que facilite la labor de gestión de configuración de red, basados en una revisión bibliográfica que incluye la existencia de modelos ontológicos para la gestión de red, arquitecturas de agentes inteligentes y tecnologías de gestión de red integrada; un modelo ontológico representativo; una arquitectura basada en agentes inteligentes que permita obtener la información de gestión; y un prototipo de agente como mecanismo de verificación de la arquitectura presentada.

Ahora bien, se desarrolló una investigación, en primera instancia *Aplicada*, puesto que no se pretende construir nuevo conocimiento; y en segunda instancia *Cualitativa*. Debido a la naturaleza cualitativa de la investigación, se utiliza un diseño *No Experimental*, ya que el estudio se realizó sin la manipulación deliberada de variables; por lo que la recolección de datos se desarrolló, partiendo de una revisión bibliográfica.

Finalmente, la memoria de esta tesis sigue el siguiente esquema:
En el capitulo 1, se presenta el planteamiento del problema, donde se resumen todos los aspectos concernientes a los antecedentes, formulación del problema y justificación. El capitulo 2, muestra los objetivos y alcance del proyecto. El capitulo 3, hace evidente el marco de referencia, que incluye el marco teórico y el estado del arte. El capitulo 4, muestra la metodología usada en el desarrollo del proyecto. En el capitulo 5, se presentan los resultados del proyecto, es decir, modelo ontológico, arquitectura, desarrollo y validación del prototipo de agente inteligente. Por último, se muestran las conclusiones y recomendaciones.

1. PLANTEAMIENTO DEL PROBLEMA

1.1. ANTECEDENTES

En un contexto marcado por el crecimiento en tamaño y complejidad de las redes de computadoras; el concepto de gestión de red, ha tomado mayor acogida en aras de administrar eficaz y eficientemente los recursos de la misma con el propósito de asegurar su disponibilidad. Gestión de la capacidad, aumento de la disponibilidad, supervisión y resolución de problemas, entre otras; conforman algunas de las tareas de administración de red, que ante la escasa existencia en el mercado de herramientas software que faciliten dicha actividad de forma inteligente, tornan la labor de gestión aún más compleja para el administrador de la red.

La Gestión de configuración, como área funcional de la gestión de red incluye todas las tareas que tienen que ver con la recolección de información, modificación del comportamiento de los dispositivos y almacenamiento de la información que determina este comportamiento (GUERRERO CASTELEIRO, 2007). En este punto, el crecimiento de las redes de computadoras y la heterogeneidad de los recursos actuales y servicios de telecomunicaciones, conlleva a que actividades de mantenimiento de inventario de red, salvaguarda y restauración de la configuración ante la detección de errores, versiones de configuración, control de estado, entre otras, se tornen más complejas de ejecutar para el administrador de red, dada las diversas topologías o arquitecturas de servicios; lo que implica un proceso difícil de gestión, atado a retardos o pérdida de tiempo, susceptible al error humano y con un aumento de costos por la contratación de mayor personal cualificado.

Ante la urgencia de solventar dicho problema y la necesidad de minimizar cada vez más la labor de administración ejercida por el hombre, surgen los agentes inteligentes como solución para la interacción con un entorno complejo y el manejo de la información, sin necesidad de supervisión constante por parte del usuario.

En este orden de ideas, se propone a través del presente trabajo, construir un prototipo que facilite la labor de gestión de configuración de red, a partir de una arquitectura basada en agentes inteligentes.

1.2. FORMULACIÓN DEL PROBLEMA

¿Cómo facilitar la labor de gestión de configuración de red ejercida por el administrador?

1.3. JUSTIFICACIÓN

La propuesta de una arquitectura que integre agentes inteligentes en un dominio de gestión de configuración de red, proporcionaría ventajas como, la sistematización del proceso de recolección de información en la red dejando atrás la práctica manual, la optimización de los tiempos de respuesta orientados a disminuir los lapsos de espera del administrador de red, el empleo de un dominio común para representar el conocimiento facilitando la comunicación entre agentes; el desarrollo de agentes inteligentes comunicativos, autónomos y adaptables; al igual que, la ejecución de tareas de gestión de configuración de forma independiente a los mecanismos de gestión de los recursos. Centrando toda su relevancia, en mejorar el acceso a la información y con ello la obtención de conocimiento útil.

En este sentido, con el desarrollo de la misma, se estimulan y sientan las bases hacia la construcción de herramientas software inteligentes que den solución a problemas determinados, y que provistas de una estructura jerárquica, siguiendo lineamientos orientados a las buenas prácticas de programación garanticen la reutilización de código, y así, contribuir al fortalecimiento de los procesos investigativos en el Programa de Ingeniería de Sistemas, adscrito a la Facultad de Ingeniería de la Universidad de Cartagena; que por otro lado, en la medida en que se implemente facilitaría la gestión de configuración de red de pequeñas, medianas y grandes empresas a nivel nacional.

Para tal fin, y en pos de concretar la base factible que garantice la ejecución del proyecto. Desde el punto de vista económico, la propuesta es respaldada por el empleo de tecnologías Open Source. A nivel científico; es sustentada con la elaboración del marco teórico, que proporciona la fundamentación cognitiva necesaria. Finalmente, se torna viable a nivel tecnológico, puesto que se apoya en tecnologías como OWL para la construcción de la ontología, PROTEGÉ como editor de ontologías, RDQL como lenguaje de consulta, JADE como plataforma de desarrollo, entre otras herramientas libres que se adaptan a los requerimientos del proyecto.

La no implementación de una arquitectura basada en agentes inteligentes para la gestión de configuración de red, constituiría un obstáculo para alcanzar una verdadera gestión de red integrada, teniendo en cuenta la actual convergencia de tecnologías de red y servicios, propagando un problema de interoperabilidad implícito en los diversos modelos de gestión existentes; prolongando la complejidad de la labor de gestión de configuración de red ejercida por el administrador.

2. OBJETIVOS Y ALCANCE

2.1. OBJETIVO GENERAL

Construir un prototipo que facilite la labor de gestión de configuración de red, a partir de una arquitectura basada en agentes inteligentes.

2.2. OBJETIVOS ESPECÍFICOS

- ✓ Establecer el estado del arte a partir de una revisión bibliográfica basada en la existencia de modelos ontológicos para la gestión de red, arquitecturas de agentes inteligentes y tecnologías de gestión de red integrada.
- ✓ Diseñar un modelo ontológico representativo para la gestión de configuración de red.
- ✓ Diseñar una arquitectura basada en agentes inteligentes que permita obtener la información de gestión de configuración de red.
- ✓ Desarrollar un prototipo de agente como mecanismo de verificación de la arquitectura propuesta.
- ✓ Validar el prototipo obtenido con el propósito de garantizar que cumple con los requerimientos establecidos.

3. MARCO DE REFERENCIA

3.1. ESTADO DEL ARTE

En sus inicios, las redes de computadores eran relativamente pequeñas, ya que su composición básica eran uno o más servidores centrales conectados a una mínima cantidad de periféricos pertenecientes a un mismo fabricante. Durante los años 80's se originó una expansión significativa en el área de implantación de sistemas, redes de telecomunicaciones, y en el desarrollo de dispositivos de red; propiciando así que la gestión de red se tornara compleja e ineficiente, debido a la diversidad de arquitecturas de dispositivos adelantadas por los diferentes fabricantes; que a la final, se traducía en emplear un modelo por fabricante para llevar a cabo el proceso de gestión.

Debido a lo anterior, se requería mayor inversión en los costos de mantenimiento, ya que se solicitaba más personal capacitado para administrar y gestionar los recursos de forma eficiente para asegurar la disponibilidad de los servicios; por tal motivo, surgió la necesidad de disponer de herramientas para facilitar las tareas de gestión de red, tales como: gestión de la capacidad, aumento de la disponibilidad, supervisión y resolución de problemas (Guerrero Casteleiro, 2007).

Con el objetivo de facilitar las labores de gestión, a comienzos de la década de los 90 aparecieron en escena los modelos de gestión de red integrada, con el fin de gestionar redes conformadas por equipos de distintos fabricantes, mediante mecanismos estandarizados para normalizar las comunicaciones y la información gestionada. Entre los modelos de gestión de red integrada más representativos se encuentran el protocolo simple de gestión de red o SNMP (Simple Network Management); el protocolo común de gestión de red o CMIP (Common Management Information protocol); la interfaz de gestión de ordenadores de sobremesa o DMI (Desktop Management Interface); y la arquitectura

Común de Intermediarios de Peticiones de Objetos o CORBA (Common Object Request Broker Architecture), entre otros.

Esta variedad de modelos, trae como consecuencia la necesidad de crear mecanismos que posibiliten la interoperabilidad entre los distintos dominios implicados, puesto que varios modelos deben trabajar en forma conjunta con los problemas que esto conlleva. La solución es posible si se puede definir un conjunto de reglas que sean comunes, de ahí nace la iniciativa de la gestión basada en Web de empresa o WBEM (Web Based Enterprise Management) que trata de establecer una arquitectura que permite la interoperabilidad entre modelos de gestión.

Aunque es una solución válida, solamente traduce directamente los componentes de la especificación de un mismo recurso en dos o más modelos, pero no entre el significado que contienen; es decir, solamente realiza una traducción sintáctica, lo que supone un obstáculo para alcanzar una verdadera gestión integrada.

Por otra parte, en la rama de la Inteligencia Artificial, las ontologías se han convertido en el medio para resolver problemas similares en otros ámbitos como el de la Web Semántica, en donde esta técnica de representación del conocimiento proporciona semántica a las páginas y servicios web (LÓPEZ DE VERGARA MÉNDEZ, 2003).

Tomando como base este avance, se han aplicado estas técnicas en la gestión de redes como medio para compartir la semántica entre los modelos de gestión existentes, investigaciones relevantes y relacionadas como la realizada en el año 2003 por Jorge Enrique López de Vergara, en su tesis doctoral titulada: "*Especificación de modelos de información de Gestión de red integrada mediante el uso de ontologías y técnicas de representación del conocimiento*", proponen una solución que tiene por objetivo perfeccionar la interoperabilidad de los dominios, mediante el estudio y aplicación de ontologías. Seguidamente, Rafael Fernández Gallego para el año 2006, propone un framework de gestión semántico basado en políticas, titulado "*Arquitecturas Emergentes para Gestión Integrada de Redes y Sistemas Distribuidos*", en el cual agrega la interacción

de agentes inteligentes autónomos para mejorar la disponibilidad y calidad del servicio. De igual forma, Antonio Guerrero Casteleiro en el año 2007, en su tesis "*Especificación del comportamiento de gestión de red mediante ontologías*", analiza la posibilidad de representar a través de ontologías los diferentes tipos de comportamientos de gestión de red, para definir los mecanismos que permitirán el uso de estas definiciones en un sistema de gestión y así lograr la interoperabilidad entre los distintos modelos existentes.

También, Luis Ernesto Rios y Mario Giovanny Sabogal en el año 2010, en su tesis denominada: "Sistema de Agentes Móviles para el Monitoreo en la Gestión de Redes" (Universidad Distrital Francisco José de Caldas, Bogotá), proponen el uso de agentes móviles para el monitoreo de redes LAN, para facilitar la obtención de información de cada una de las estaciones de trabajo y así presentar dicha información de forma clara y fácil de comprender, de manera que pueda ser usada como soporte en la gestión de aspectos de la red como lo son procesos, almacenamiento y memoria. Asimismo, en la investigación realizada en la Universidad Pontificia Bolivariana (seccional Medellín) por Sebastián Ortiz, Ana Isabel Oviedo y Cesar López, en su artículo denominado: "Sistema Multiagente para el monitoreo Distribuido de Inventario de Software y Hardware en una Red Corporativa", plantean el desarrollo de un prototipo de sistema multiagente distribuido que efectúe un monitoreo constante en tiempo real de los cambios en el software y hardware de las máquinas de los usuarios, para ejecutar de forma fácil y rápida la gestión de inventario de una red corporativa.

En resumen, las investigaciones expuestas colocan en evidencia unas falencias en cuanto al uso de los agentes de software, ya que por una parte se enuncian el uso de las ontologías como medio para la integración semántica de los distintos dominios de gestión existentes, y por otra se hace énfasis en el uso de sistemas multiagentes como medio para facilitar la gestión de algún aspecto concerniente a las redes, lo cual implica que no se está haciendo uso de todas las potencialidades generadas a partir de la integración de los agentes y las ontologías.

Por tanto, en la presente investigación, se propone el uso de agentes inteligentes, que es el resultado de la unión entre agentes software y ontologías, para facilitar el

desarrollo de una arquitectura para la gestión de configuración de red; la cual permitirá utilizar las ontologías como medio para integrar los distintos dominios de gestión existentes, y así obtener una mayor adaptabilidad en cuanto a los avances tecnológicos; y los agentes de software para obtener un mayor rendimiento, movilidad y eficiencia para obtener la información en las estaciones de trabajo que formen parte de la red. En ese orden de ideas, este proyecto constituirá las bases para futuras investigaciones concernientes con esta temática a nivel local, ya que no se encontraron evidencias de trabajos relacionados con esta rama de la Inteligencia artificial.

3.2. MARCO TEÓRICO

A continuación, se desarrolla el contexto teórico que sustentará la propuesta de investigación; temáticas como: Gestión de red, arquitectura gestor-agente, modelos de red integrada, ontologías y gestión de configuración de red, son las desarrolladas en esta sección.

3.2.1. Gestión De Red

Muchos autores han presentado numerosas definiciones para la gestión de red, entre las cuales se encuentran:

- ✓ La StandardsCommitte T1 Telecommunications, telecom glosary 2000 (American National Standard T1 523-2001(febrero 2001) la define como: "*La ejecución de un conjunto de funciones requeridas para controlar, planear, asignar, desplegar, coordinar y monitorizar los recursos de una red de telecomunicación, incluyendo funciones de rendimiento tales como planear la red inicial, asignación de frecuencias, encaminamiento de tráfico predeterminado para soportar balances de carga, automatización de la distribución de claves criptográficas, gestión de configuración, gestión de fallos, gestión de seguridad, gestión de rendimiento y gestión de contabilidad*"(Standards Committe Telecommunications, 2001).

- ✓ Jorge E. López de Vergara, en su tesis, menciona: "*La gestión de red es la planificación, organización, supervisión, y control de elementos de comunicaciones para garantizar un nivel de servicio*" (LÓPEZ DE VERGARA MÉNDEZ, 2003).

Como se puede observar, las definiciones tienen puntos en común, por tanto, se puede decir que la Gestión de Red es la ejecución de funciones, supervisión, control, planificación, asignación y coordinación de los recursos de una red para garantizar un nivel de servicio; usar en forma eficiente los recursos de red; ofrecer mecanismos para detectar, diagnosticar y recuperarse frente problemas; asignar recursos de red a los servicios que se implantan; asistir en el diseño y planificación de la implantación de nuevos recursos.

Estas herramientas o mecanismos descritos se denominan *sistemas de gestión de red;* y la persona encargada de usar las mismas basándose en unos procedimientos para gestionar los recursos de red se le conoce como *administrador de red.*

Teniendo en cuenta las definiciones anteriores, se puede notar que los recursos a gestionar no sólo se limitan a la red física, sino también a los servicios y sistemas finales. En este punto, se introduce el concepto de "*áreas funcionales de la gestión de red*", las cuales forman parte del modelo FCAPS.

Conforman éste modelo, la gestión de: **Fallos (Fault),** encargada de ejecutar el monitoreo o seguimiento del sistema en cuestión, con el objeto de detectar posibles problemas, y determinar las políticas a efectuar para solucionar dichos problemas; **Configuración (Configuration),** definida como el conjunto de tareas concernientes a la recopilación de información de la red, modificación del comportamiento de los dispositivos y almacenamiento de la información que determina este comportamiento, por ejemplo, inventario de red, control de versiones, salvaguarda y restauración de la configuración de los elementos; **Contabilidad (Accouting),** la cual tiene como objetivo medir la capacidad de la red y distribuir los costos entre los distintos usuarios de acuerdo a políticas establecidas; **Rendimiento (Performance),** se encarga de realizar un seguimiento del rendimiento de la red manteniendo la calidad de servicio deseada, a través de estadísticas

que incluyen el uso de la red y la eficiencia; y finalmente, **Seguridad (security),** que tiene como finalidad controlar el acceso a los recursos por parte de los posibles usuarios y proteger la integridad de la información transmitida.(GUERRERO CASTELEIRO, 2007)

En síntesis, tomando los elementos presentes en las definiciones presentadas, se nota que la gestión de red es un factor vital en las empresas, porque sus herramientas permiten reducir los costos, mejorar la velocidad y calidad del servicio, para aumentar eficiencia y la competitividad.

3.2.2. Arquitectura Gestor-Agente

Tomando como base el concepto anterior, se hace necesario explicar cómo se gestionan los recursos de una red. Actualmente la mayor parte de los sistemas de gestión usan la arquitectura gestor-agente (*Ver Figura 1*); donde los agentes son los componentes lógicos o físicos que operan con la información de los elementos de red que pueden ser gestionados; y los gestores son los elementos que interactúan con los administradores y permiten a estos llevar a cabo las operaciones de gestión tomando como base la información proporcionada por los agentes.

Figura 1. Gráfica que ilustra la arquitectura Gestor-Agente.

Fuente: GUERRERO CASTELEIRO, 2007

En este orden de ideas, los recursos de una red que posean un gestor recibirán el nombre de nodos gestores y los que tengan un agente se denominaran nodos gestionados. Generalmente, el principio de funcionamiento de los sistemas de gestión se basa en el intercambio de información entre nodos gestores y gestionados; en este punto los agentes mantienen en cada nodo gestionado información acerca del estado y las características de funcionamiento de un recurso de red en particular, y los gestores piden a los agentes que realicen ciertas operaciones con la información que poseen, para conocer el estado del recurso y así poder alterar el comportamiento a través de la modificación de algún dato de gestión (GUERRERO CASTELEIRO, 2007).

3.2.3. Modelos De Gestión De Re Integrada

En la actualidad se han presentado varias propuestas de modelos de gestión de red integrada, pero en las siguientes secciones se hará un énfasis especial en los modelos de gestión OSI, internet o SNMP, y WBEM.

- ***Modelo de gestión OSI y TMN***

Con el objetivo de integrar la gestión en el modelo de comunicaciones OSI de 7 capas, la ISO (Organización Internacional de Estandarización, International Standarization Organization), definió el modelo OSI-SM (Interconexión de sistemas abiertos-Gestión de sistemas, Open systems Interconnection-Systems Management).

Sin embargo, su implementación se debe al modelo TMN (*Red de gestión de las telecomunicaciones, Telecommunication management network*) definido por la ITU (*Unión internacional de telecomunicaciones, International telecommunication Union*), el cual está constituido por una arquitectura física (estructura y entidades de la red), un modelo organizativo de gestión (niveles de gestión, *ver Figura 2*), un modelo funcional (servicios, componentes y funciones de gestión), y un modelo de información (definición de recursos gestionados).

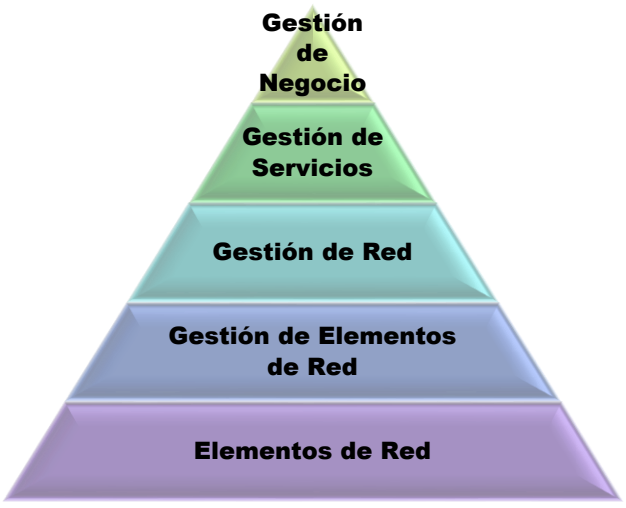

Figura 2. Niveles de Gestión según TMN.

Fuente: LÓPEZ DE VERGARA MÉNDEZ, 2003

En estos modelos, la comunicación se basa en el Servicio Común de información de gestión (CMIS, Common Management Information Service), encargado de definir los servicios de operación (solicitar, modificar o añadir información) y notificación (notificar eventos relacionados con los objetos gestionados); el protocolo común de información de gestión(CMIP, Common Management Information protocol), define la arquitectura y protocolo de comunicación; y las directrices para la definición de objetos gestionados o lenguaje GDMO, que la realiza la tarea de definir la información de gestión a través de un conjunto de plantillas(Clases, atributos y acciones).(GUERRERO CASTELEIRO, 2007)

- ***Modelo de Gestión de Internet o SNMP***

Este modelo de gestión de red integrada desarrollado por el Grupo de trabajo de Ingeniería para Internet (IETF, Internet Engineering Task Force) también conocido como SNMP, está constituido por el protocolo simple de gestión de red o SNMP, el cual proporciona a los gestores la obtención, modificación de información y notificación de eventos, además, se encarga de la comunicación entre éstos y los agentes; la estructura de la información de gestión o SMI (Structure of Management Information), que define el lenguaje de especificación de información de gestión para normalizar la sintaxis; y un modelo de información basado en las bases de datos de información de gestión (MIBs, Management Information Bases) definidas con el fin de brindar una serie de conceptos comunes.(GUERRERO CASTELEIRO, 2007)

- ***Modelo de Gestión WBEM***

El modelo de gestión de empresa basado en web, como su nombre lo indica, reutiliza las tecnologías como HTTP (Protocolo de transferencia de hipertexto) y XML (lenguaje de marcas extensibles), ya que sobre ellas se maneja una gran parte del conocimiento ofreciendo buenos resultados.

Por otra parte, su principio de funcionamiento se basa en el uso de los esquemas CIM (Modelo de información común) para definir el modelo de información; HTTP/ XML

o CIM-XML, como protocolo de comunicaciones, el cual utiliza xmlCIM y HTTP para codificar y transportar la información respetivamente; y en el lenguaje para las definiciones de la información de gestión (CIM), donde se especifican clases, métodos, propiedades, relaciones, asociaciones y calificadores. (GUERRERO CASTELEIRO, 2007)

3.2.4. Gestión De Configuración

Es un área funcional de la gestión de red que tiene por objeto proveer funciones para operar, controlar, identificar y recopilar datos de una red. Su principio de funcionamiento está basado en dos tipos de configuración: dinámica y estática.

La primera implica la asignación, registro, indicación, muestra y reporte de la topología de red, listado de equipos de red en conjunto con sus parámetros de sistema, tales como el tipo, localización, direcciones físicas y lógicas. Además en el caso de redes pequeñas y simples, involucra el control de inventario.(ROSERO & PROAÑO)

La segunda incluye el establecimiento de las rutas actuales para las interconexiones requeridas por la red, establecimiento de nuevas rutas o la cancelación de las mismas.

Con base en la definición preliminar, se debe tener en cuenta que la gestión de configuración implica:

Estado actual de la red: registro estático y dinámico de la topología.

Gestión de inventario: incluye base de datos de los elementos de la red e historial de cambios y problemas.

Mantenimiento de directorios: todos los nodos y sus aplicaciones, y base de datos de nombres de dominios.

Coordinación del esquema de nombres para nodos y aplicaciones.

Control operacional de la red: incluye cambiar la configuración de dispositivos, iniciar y detener elementos individuales, entre otras.

3.2.5. Agentes Inteligentes

En la actualidad, las tecnologías de agentes se han convertido en una de las áreas de

investigación de mayor popularidad en campos como la Inteligencia Artificial, sistemas distribuidos, y comercio electrónico, debido a la necesidad de desarrollar nuevos paradigmas que permitan modelar problemas de forma transparente y eficiente. En este sentido los sistemas multiagentes, en donde dos o más agentes interactúan para cumplir una determinada tarea, se han convertido en la respuesta a la necesidad creciente de un nuevo paradigma de programación.

En cuanto a la definición formal de agente, se pueden encontrar en la literatura un gran número de ellas, sin que ninguna haya sido completamente aceptada por la comunidad científica. Entre estas se encuentra la presentada por Russell, quien dice que un agente "*es una entidad que percibe y actúa sobre un entorno*"(RUSSELL, 1996), es decir, se pueden crear diversos agentes dependiendo del problema en cuestión.

También la presentada por Carla Salazar Serrudo en su libro "Agentes y multiagentes Inteligentes: conceptos, arquitecturas y aplicaciones", que dice que un agente "*es un sistema informático, situado en algún entorno, dentro del cual actúa de forma autónoma y flexible para así cumplir sus objetivos; donde un agente recibe entradas de su entorno y a la vez ejecuta acciones para intentar cambiarlo a su gusto*". (SALAZAR SERRUDO)

Además de la propuesta por Barba Antoni y Hesselbach Xavier, quienes plasman que un agente inteligente se puede definir como: "*una entidad software autónoma que se comporta de acuerdo a una inteligencia autocontenida*". (BARBA & HESSELBACH, 2002)

En general se puede decir que un agente es todo sistema software que cumple con las siguientes características (BOTTI):

Autonomía: es la capacidad que tiene de adaptarse a los cambios en el entorno y actuar con base en esto. Es decir, que partiendo de sus conocimientos, puedan tomar decisiones de acuerdo a un evento determinado sin la intervención de un usuario.

Sociabilidad: todo agente debe tener un mecanismo de interacción con otros agentes u otras entidades mediante algún tipo de lenguaje de comunicación.

Pro-actividad: el agente debe ser capaz de reconocer, controlar y cumplir sus propios objetivos, de acuerdo a los cambios en el entorno o sus propias decisiones.

Racionalidad: el agente lleva a cabo sus tareas tomando como punto de partida los datos percibidos del entorno.

Adaptabilidad: se relaciona con el cambio que puede tener un agente en su comportamiento, en relación con el aprendizaje adquirido a través del tiempo.

Movilidad: como su nombre lo indica, es la capacidad que posee el agente de trasladarse a distintos puntos que conforman una red.

Benevolencia: todo agente debe estar dispuesto a brindar su ayuda a otros agentes, solamente si esto no afecta el cumplimiento de las tareas que tiene asignadas.

Coherencia: con base al entorno el agente debe almacenar su base de conocimientos teniendo en cuenta la coherencia entre los datos almacenados, para que su comportamiento sea el esperado.

- ***Estados De Un Agente***

El ciclo de vida de un agente software se encuentra determinado por los siguientes estados (*ver Figura 3*) (ROA):

Figura 3. Gráfico que ilustra el ciclo de vida de un agente.

Fuente: ROA

***Estacionario*:** también conocido como estado cero. En este estado el agente se encuentra inmóvil, es decir, no envía mensajes, ante los mensajes recibidos solo responde "no entendido" (not –understood).

***Salida*:** en este estado el agente ha sido inicializado por primera vez (estado 1); es decir, el agente es móvil, ya que recibe el mensaje recibido y pasa al estado 2.

***Movido*:** en este estado 2, el agente es móvil ha llegado a su destino y pasa al siguiente estado.

***Terminando*:** el agente aun es móvil y sencillamente ha retornado a su localización original (estado 3).

- ***Arquitectura De Construcción De Agentes***

En general, las arquitecturas usadas para construir agentes especifican qué rol cumple cada agente en un conjunto de módulos que interactúan entre sí para lograr la funcionalidad deseada.

A continuación se presentan tres arquitecturas de agentes, teniendo en cuenta el modelo de razonamiento utilizado, entre las cuales se destacan (SALAZAR SERRUDO):

Deliberativas: son aquellas que se basan en la teoría de planificación, en la cual existe un estado inicial, un conjunto de planes y un estado final; en otras palabras, utiliza modelos de representación simbólica del conocimiento. En este tipo de sistemas, es necesario proveer a los agentes de un mecanismo de planificación que determine los pasos que se deben realizar para alcanzar los objetivos. Teniendo en cuenta lo anterior, un agente deliberativo, es aquel que contiene un modelo simbólico del mundo real, que toma las decisiones tomando como base el razonamiento lógico orientado por la concordancia entre los patrones y la manipulación simbólica.

Reactivas: se caracterizan porque no basan su razonamiento en un modelo simbólico y en la utilización del razonamiento simbólico complejo.

Hibridas: debido a las ventajas y desventajas de utilizar una arquitectura reactiva o deliberativa en el desarrollo de agentes, se han propuesto sistemas híbridos, es decir, la combinación de las características que conforman varios modelos, logrando minimizar las desventajas y maximizar las ventas. Una arquitectura hibrida puede ser en la que un agente se encuentra formado por dos subsistemas: uno deliberativo, que use un modelo simbólico y que genere planes en el sentido expuesto anteriormente, y otro reactivo basado en reaccionar de acuerdo a los eventos que tengan lugar en el ambiente y que no requieran de un razonamiento complejo.

- ***Tipos De Agentes***

Los agentes se pueden clasificar teniendo en cuenta diversos criterios. Pero en esta sección se expondrá una clasificación basada en el rol de los agentes en la gestión de información (CALLE, 1997):

Agentes suministradores: son los encargados de recopilar la información y mostrarla de forma ordenada o desordenada para los destinatarios o agentes peticionarios dependiendo de sus necesidades.

Agentes peticionarios: son aquellos cuyo objetivo es buscar y recopilar la información necesaria, es decir, dependen directamente de los agentes suministradores.

Agentes facilitadores: se encargan de adaptar las peticiones realizadas por agentes peticionarios con base en las capacidades de los agentes suministradores.

- ***Comunicación Entre Agentes***

En los sistemas multiagentes, la comunicación juega un papel crucial, debido a que permite la coordinación entre un grupo de agentes de forma que se ejecuten solo las acciones necesarias para efectuar una tarea determinada. Ahora bien, en la actualidad existe un rango amplio de formas de comunicación entre las cuales se pueden destacar (SALAZAR SERRUDO):

No hay comunicación: este modelo es recomendable cuando los agentes no están interrelacionados, ya que interactúan sin comunicarse, deduciendo las intenciones de los otros agentes.

Comunicación primitiva: en este modelo la comunicación esta limitada a un número de señales fijas con una interpretación establecida con antelación.

Arquitectura de pizarra: es comúnmente utilizada en las ramas de la inteligencia artificial para reutilizar conocimiento y memoria. Su uso se basa en el intercambio de información a través de una pizarra conocida por todos los agentes, donde pueden dejar resultados parciales o consultar la información suministrada por otros agentes.

Intercambio de planes e información: en este modelo, los agentes pueden intercambiarse planes de información, para adaptar sus estrategias.

Intercambio de mensajes: son agentes que modifican o no su comportamiento en respuesta al procesamiento de una comunicación.

Comunicación de alto nivel: permite la comunicación directa entre dos agentes para generar e interpretar un lenguaje, con el objetivo de comunicar información que el emisor conoce, para que el receptor pueda alcanzar el estado que tiene el emisor.

3.2.6. Frameworks Para El Desarrollo De Sistemas Multiagentes

En la actualidad existen diversas herramientas que permiten el desarrollo de sistemas multiagentes, entre las cuales se destacan (ROA):

Aglets Software DevelopmentKit(ASDK), es un framework creado por IBM, el cual presenta una interfaz gráfica identificada como una agencia, en la cual los agentes existen y se ejecutan. Además utiliza el método serializarían de Java (JOS) que facilita la movilidad y el cifrado del código, y el protocolo de transporte de Aglets (ATP) para trasladar los agentes a través de la red, mensajería, creación, clonación, expedición, eliminación, activación y paso de mensajes entre agentes.

Java AgentDevelopment Framework, es un framework desarrollado en Italia, cumple con el estándar FIPA. Además las aplicaciones desarrolladas bajo este framework se pueden ejecutar en una amplia gama de dispositivos como PDA's y teléfonos celulares, posee librerías adicionales y permite la integración con otras aplicaciones como Java Expert System Shell, la cual permite representar el conocimiento heurístico de un experto humano.

FIPA-OS, como su nombre lo indica cumple con el estándar FIPA para el diseño e implementación de sistemas multi-agentes y desarrollos en Java. Este framework se centra en proveer mecanismos de comunicación entre agentes. Permite la integración con Java Expert System Shell(JESS), para el manejo del conocimiento y movilidad de los agentes.

3.2.7. Ontologías

Desde inicio de los años 90's el desarrollo de las tecnologías web ha alcanzado niveles sorprendentes, ya que se ha convertido en el medio de mayor uso para la publicación de documentos científicos e información clasificada.

Pero lo anterior, ha traído como consecuencia, la necesidad de que toda esta información pueda ser compartida, reutilizada, interpretada y aplicada en cualquier ámbito del conocimiento; es decir, que toda esta información posea una estructura común, que permita el procesamiento automático de la información en forma correcta. Ante estas necesidades, aparecen en escena las ontologías, que son las encargadas de suministrar los

metadatos necesarios para una representación declarativa del conocimiento de un dominio; proporcionar una definición formal de conceptos consensuados; y brindar un vocabulario común.

Ahora bien, la definición formal de lo que es una ontología, es una de las grandes ausentes en la literatura, ya que muchos expertos en esta área de la Inteligencia artificial han propuesto muchas definiciones sin haber estandarizado. Entre las definiciones de mayor aceptación se encuentran:

- La presentada por Natalya Noy y Deborah McGuinness en su Guía para crear tu primera Ontología, la definen como: "*Una descripción explicita y formal de conceptos en un dominio de discurso (conceptos o clases), propiedades de cada concepto describiendo varias características y atributos del concepto (slots o roles), y restricciones sobre los slots (facetas o restricciones de un slot)*".(NOY & McGUINNESS, 2005)

- La de Gruber que dice: "*Una ontología es la especificación explicita de una conceptualización*".(GRUBER, 1995)

Sin embargo, las anteriores definiciones se pueden resumir en la presentada por Studer, V.R. Benjamins, D. Fensel en Knowledge Engineering: Principles and Methods, donde exponen que "*Una ontología es una especificación explicita y formal de una conceptualización compartida*" (STUDER, BENJAMINS, & FENSEL, 1998). A partir de este concepto se puede inferir que (LÓPEZ DE VERGARA MÉNDEZ, 2003):

- ✓ Es explícita ya que define conceptos, propiedades, relaciones, funciones, axiomas y restricciones que la componen.
- ✓ Es formal, pues puede ser interpretada por máquinas.
- ✓ Es una conceptualización, porque es una abstracción de fenómenos que ocurren en un dominio del conocimiento que se desee representar.
- ✓ Es compartida, es decir, la información tiene una estructura aceptada entre distintos grupos de expertos.

- ***Clasificación de Ontologías***

La clasificación de ontologías se realiza teniendo en cuenta diversos criterios emitidos por algunos autores; el grado de axiomatización, la dependencia del contexto y el sujeto de contextualización, son algunos de ellos.

De acuerdo al **Grado De Axiomatización** que tenga la definición de sus categorías, las ontologías se pueden clasificar en ***Terminológicas*** y ***Formales***. Las primeras, define términos y sus relaciones en taxonomías, que involucran tanto relaciones de subtipo y supertipo; disponiendo así de menos información del universo modelado, lo que permite igualmente la construcción de ontologías de gran tamaño por la relativa simplicidad de su especificación. Las segundas en cambio, poseen sus categorías restringidas por axiomas y definiciones expresadas en lógica formal o en algún tipo de lenguaje procesable por la computadora, éstas ontologías suelen tener menos cantidad de conceptos que las terminológicas, pero sus axiomas y definiciones pueden soportar computaciones e inferencias más complejas.

Al clasificarse conforme al **Grado De Dependencia Del Contexto**, donde las menos dependientes del contexto serán las más reusadas, se encuentra: ***ontologías de dominio***, cuyo principal objetivo es permitir el reuso de la ontología para diferentes aplicaciones que involucren al mismo dominio**.** ***Generales o De Sentido Común***, que definen un vocabulario relacionado a cosas, eventos, tiempo, espacio, causalidad, comportamiento, función, entre otras; y las ***Meta ontologías*** u ***Ontologías Genéricas***, que son similares a las de dominio, pero los conceptos que definen se consideran genéricos a través de diferentes áreas del conocimiento y por ello reusable en diferentes dominios.

Siguiendo con el criterio de Sujeto de Conceptualización, las ontologías se clasifican en: ***De Tareas***, que proveen un sistemático vocabulario de los términos usados para resolver problemas asociados con tareas particulares, ya sean dependientes o no del dominio. ***De Aplicación***, contiene todas las definiciones que son necesarias para modelar el conocimiento requerido para una aplicación en particular para un dominio dado; y finalmente las ***De Representación de Conocimiento***, junto con las ***De Más Alto Nivel***.(MARTIN, 2004)

- ***Lenguajes de Definición de Ontologías***

En la última década se ha definido un gran número de lenguajes de definición de ontologías, partiendo de paradigmas de representación de conocimiento (*Red semántica, Representación basada en Datos, Lógica Descriptiva y Orientación a Objetos*). Herramientas como Ontolingua, Loom, F-Logic, entre otros, ya se utilizaban para representar conocimiento en aplicaciones, otros se adaptaron a partir de lenguajes ya existentes y otros se crearon específicamente para la representación de ontologías. Este conjunto de lenguajes llamados tradicionales, usan una sintaxis en texto plano que en muchos casos es similar LIPS (Procesador de Lista) desarrollado para programar aplicaciones de inteligencia artificial.

Con el surgimiento de la Web Semántica, se han desarrollado otros lenguajes, tales como: RDF (*Resource Description Framework,* Marco de Descripción de Recusos), RDFS (*RDF Schema*, Esquema RDF), OIL (*Ontology Inference Layer*, Capa de Inferencias de Ontologías), DAML+OIL (fusión de los lenguajes DAML y OIL) y últimamente OWL (*Web Ontology Lenguage,* Lenguaje de Ontologías para la Web). Estos lenguajes basan su sintaxis en XML, ampliamente adoptado para el intercambio de información web.(LÓPEZ DE VERGARA MÉNDEZ, 2003)

- ***Pasos En El Desarrollo De Una Ontología***

A continuación se presentará de forma resumida una sencilla metodología de desarrollo de ontologías, siguiendo en gran medida una propuesta elaborada por las autoras Noy y McGuiness. Los pasos propuestos son los siguientes:

1) *Determinar el dominio y el alcance de la ontología.* Básicamente, esta actividad comprende delimitar el dominio que va a abarcar la ontología, para qué se va a usar, para qué cuestiones debe proporcionar una respuesta la ontología y/o quién la utilizará y la mantendrá. Algunos de estos aspectos puede cambiar durante el diseño, pero conocerlos ayudará a acotar la ontología.
2) *Considerar reutilizar otras ontologías existentes.* En algunos casos, la reutilización puede contribuir a ahorrar trabajo a los desarrolladores, ya que, por ejemplo, se parte de decisiones de diseño que ya han sido estudiadas y adoptadas. La reutilización puede

llegar a ser un requisito si el sistema debe interactuar con otras aplicaciones que han adoptado alguna ontología particular. En este sentido, los sistemas de representación de conocimiento proveen herramientas que ayudan a exportar/importar otras ontologías.

3) *Enumerar los términos relevantes de la ontología.* Contar con una lista donde aparezcan los términos sobre los que se van a hacer declaraciones en la ontología, sus propiedades, relaciones, etc., contribuyen a no olvidarlos y tener que "rescatarlos" en fases posteriores del proceso de desarrollo con un coste mayor. Esta lista de términos y relaciones puede elaborarse sin atender a consideraciones sobre si existen solapamientos entre unos y otros, o la oportunidad de modelar un concepto como una clase o una propiedad, que podrán perfilarse en fases posteriores.
4) *Definir las clases y la jerarquía de clases.* En este proceso se distinguen tres enfoques fundamentales: *top-down* o "*de arriba hacia abajo*", en el que se define el concepto más general y, a partir de éstos, todos los demás por especialización del mismo; *bottom-up* o "*de abajo hacia arriba*", al contrario que en el anterior, se comienza por definir los conceptos más específicos para posteriormente agruparlos en clases de conceptos más generales, *combinación,* en el que se combinan los dos enfoques anteriores, de tal forma que se comienza definiendo los conceptos más destacados del dominio para especializarlos y generalizarlos convenientemente.

   Ninguno de los enfoques anteriores se puede calificar como mejor que otro, algo que dependerá también de cada diseñador concreto. En cualquier caso, el proceso comienza con la definición de una serie de clases. Para ello, se seleccionarán preferentemente aquellos términos de la lista creada en el paso anterior que se refieren a objetos que tengan una existencia independiente, antes que términos que se utilicen para describir otros objetos. Las clases se organizarán en una taxonomía jerárquica, de tal forma que, si una clase *A* es subclase de otra clase *B*, entonces cada instancia de *A* es también instancia de *B*
5) *Definir las propiedades (también llamadas relaciones o slots) de las clases.* Por sí solas, las clases no contienen información suficiente para responder a las cuestiones que se plantearon en el paso 1. Por esta razón, es necesario describir la estructura de los conceptos por medio de *propiedades*. Una vez seleccionadas las clases en el paso

anterior, la gran mayoría de los términos restantes de la lista elaborada en el paso 3, corresponderán a propiedades de dichas clases. Todas las subclases de una clase heredan los *slots* de esta última. Por esta razón, conviene asociar cada propiedad a la clase más general que pueda tener dicha propiedad.

6) *Definir facetas y/o restricciones sobre los slots o relaciones.* Los *slots* pueden presentar diferentes facetas que permiten definir el tipo al que deben pertenecer los valores de una propiedad, los valores permitidos o el número de dichos valores (conocido también como *cardinalidad de slots*). Por ejemplo, es normal definir una faceta que limite los valores de la propiedad *nombre* de una clase a valores de tipo cadena.
7) *Definir instancias.* La primera iteración del proceso de desarrollo concluye con la definición de instancias de las clases de la jerarquía. La definición de una instancia individual de una clase requiere, en primer lugar, seleccionar una clase, a continuación crear una instancia individual de esa clase, y por último, asignar valores a sus *slots*.

3.2.8. Arquitectura 4+1

Antes de iniciar a describir el modelo de Vistas 4+1, se hace necesario definir el concepto de arquitectura de software, pero no es novedad que en la literatura se han propuesto muchas definiciones sin que todavía haya sido tomada alguna como estándar por parte de los arquitectos; teniendo en cuenta lo anterior, muchos autores han propuesto sus definiciones agregando su enfoque personal en cada una, entre los cuales se encuentra:

- (Pressman, 2002), quien la define como: *"la estructura jerarquica de los componentes del programa(módulos), la manera en que los componentes interactúan y la estructura de datos que van a utilizar los componentes".*
- (Clements, 1996), describe la arquitectura de software como: *"una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema."*

Sin embargo, a lo largo de la historia de la Ingeniería de Software, se ha observado que la arquitectura del software tiene falencias al plasmar el diseño de un sistema, ya que fue muy lejos en particionar prematuramente el software, o hace un énfasis excesivo de algunos aspectos del desarrollo del software tales como: ingeniería de datos, eficiencia en tiempo de ejecución, estrategias de desarrollo y organización de equipos; además la arquitectura tampoco aborda los intereses de todos sus clientes.

Debido a lo anterior, Krichten propuso el modelo de Vistas 4+1, el cual describe la arquitectura del software usando cinco vistas concurrentes tal como se muestra en la *Figura 4*, donde cada vista hace referencia a un conjunto de intereses de diferentes interesados en el sistema. (Krichten, s.f.)

A continuación se dará una breve descripción de cada una de las vistas que conforman dicho modelo.

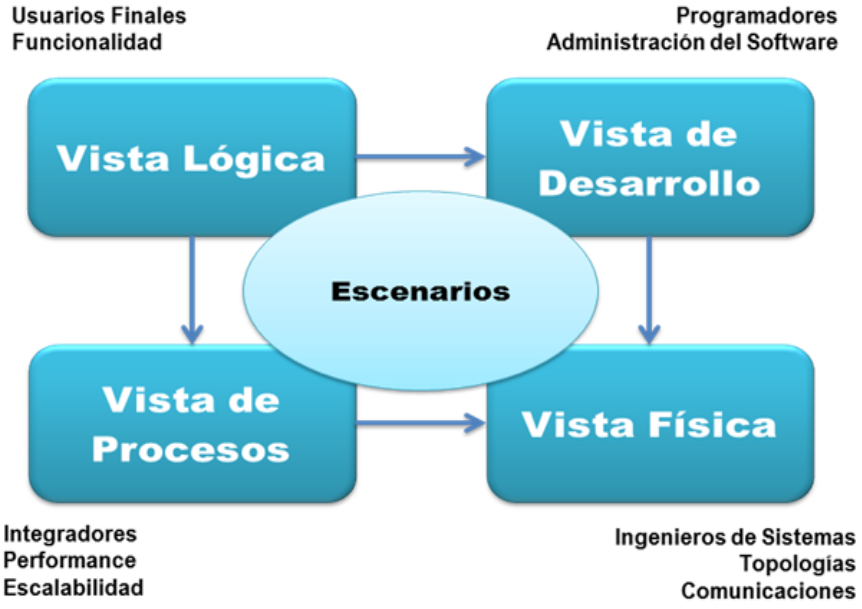

Figura 4. Gráfico que las vistas de arquitectura 4+1.

Fuente: Kruchten

- ***Vista Lógica***

Se basa en los requisitos funcionales y describe el modelo de objetos del diseño cuando es aplicada para un diseño orientado a objetos; o modelo entidad-relación cuando se va a diseñar un sistema orientado a los datos; en resumen, dicha vista representa todo lo que el sistema debe brindar en términos de servicios a sus usuarios.

En primer lugar, para diseñar esta vista, se hace un análisis funcional que consiste en descomponer el sistema en una serie de abstracciones clave, que se toman del dominio del problema en forma de objetos o clases de objetos; en segundo lugar, se aplican los principios de abstracción, encapsulamiento y herencia; además esta descomposición permite identificar mecanismos y elementos de diseño comunes a diversas partes del sistema.

Por último, para representar el diseño, se puede hacer uso del enfoque Booch/Rational a través de un diagrama de clases para mostrar el conjunto de clases y sus relaciones lógicas (asociaciones, uso, composición, herencia, entre otras), además las clases relacionadas pueden agruparse en categorías; o los templates de Clases que hacen énfasis en las operaciones principales de cada clase e identifican las características principales de sus instancias (objetos). Además si se requiere definir el comportamiento interno de un objeto, se usa el diagrama de transición de estados.

- ***Vista De Procesos***

Esta vista toma como base los requerimientos no funcionales, en especial el rendimiento y la disponibilidad. Aquí se representan los flujos de trabajo de cada uno de los componentes que conforman el software, es decir, aspectos concernientes a la concurrencia y distribución, integridad del sistema, tolerancia a fallos; además especifica en qué hilo de control se ejecuta efectivamente una operación de una clase identificada en la vista lógica.

En este punto, se describen varios niveles de abstracción que responden a distintos intereses, donde el nivel más alto se puede representar como un conjunto de redes lógicas

de procesos(programas comunicantes) ejecutándose en forma independiente, y distribuidos a lo largo de un conjunto de recursos de hardware interconectados(un bus, una LAN o WAN); y el nivel más bajo se representa por los procesos, ya que con ellos se puede controlar la arquitectura de procesos; por otra parte, un proceso se define como una agrupación de tareas que forman una unidad ejecutable.

Para construir esta vista, se divide el sistema en un conjunto de tareas independientes (hilo de control separado que puede planificarse para su ejecución independiente en un nodo de procesamiento), denominadas tareas mayores y menores. Donde las primeras son elementos arquitectónicos que se pueden manejar en forma univoca y se comunican a través de mecanismos de comunicación inter-tarea (servicios de comunicación síncronos y asíncronos basados en mensajes); y las segundas son tareas adicionales introducidas localmente por motivos de implementación como son actividades cíclicas, por ejemplo almacenamiento en un buffer, time-out, etc.

- ***Vista De Desarrollo***

Dicha vista se enfoca en la organización real de los módulos de sistema en el ambiente de desarrollo del software; es decir, el software se empaqueta en partes pequeñas conocidas como subsistemas o bibliotecas, para facilitar la codificación por parte de los programadores. Se debe tener en cuenta que los subsistemas se organizan en una jerarquía de capas, cada una de las cuales brinda una interfaz bien definida hacia las capas superiores.

Ahora bien, en esta vista, se tienen en cuenta los requisitos internos relacionados con la facilidad de desarrollo, administración de software, reutilización y restricciones impuestas por las herramientas o el lenguaje de desarrollo usado; también la asignación de requisitos y trabajo al equipo de desarrollo, la evaluación de costos, la planificación, el monitoreo del proyecto, para analizar el porcentaje de reúso, portabilidad y seguridad.

Por otro lado, la arquitectura de desarrollo de un sistema se representa a través de diagramas de módulos o subsistemas que muestra las relaciones de importación y uso. No

obstante, para poder describir la vista en su totalidad deben haberse identificado todos los elementos que conforman el software.

- ***Vista Física***

En esta vista se tienen en cuenta los requisitos no funcionales relacionados con la disponibilidad, confiabilidad, rendimiento (performance) y escalabilidad; en otras palabras, es el mapeo del software al hardware.

En la práctica, se debe tener en cuenta si el software se ejecuta sobre una red de computadoras o en nodos de procesamiento; es decir, mapear los elementos identificados (redes, procesos, tareas y objetos) sobre los nodos donde se van a ejecutar. Por ejemplo, se pueden usar configuraciones para desarrollo, pruebas y concurrencia de usuarios.

En resumen, el mapeo de software en los nodos debe ser flexible y minimizar el impacto sobre el código fuente.

- ***Escenarios***

Teniendo en cuenta las vistas anteriores, se puede notar que trabajan de manera conjunta en forma natural a través del uso de un conjunto reducido de escenarios relevantes, para los cuales se describe una secuencia de interacciones entre objetos y procesos; por lo tanto, los escenarios se pueden ver como una abstracción de los requisitos más importantes del sistema.

Por otra parte, esta vista tiene un cierto grado de redundancia en comparación con las 4 anteriores, pero sirve en el diseño del sistema como:

- Una guía que permite descubrir elementos arquitectónicos durante el diseño de la arquitectura.
- Un rol de validación e ilustración después de completar el diseño de la arquitectura, ya que facilita la formulación de pruebas de un prototipo basado en la arquitectura.

4. METODOLOGÍA

Teniendo en cuenta la naturaleza de los objetivos, se desarrolló una investigación, en primera instancia *Aplicada*, puesto que no se pretende construir nuevo conocimiento; sino aplicar el existente a través de la búsqueda y consolidación del saber, empleando tecnologías concretas para dar solución a un problema específico de gestión de red (gestión de configuración); y en segunda instancia *Cualitativa*, en la medida en que se pretende solucionar el problema planteado a partir de la descripción de procesos que comprenden dicho problema. Es conveniente precisar, que debido a la propia naturaleza cualitativa de la investigación, se utiliza un diseño *No Experimental*, ya que el estudio se realizó sin la manipulación deliberada de variables, optando así, por la observación de los fenómenos (gestión de configuración de red) en su ambiente natural (red de computadoras) para su posterior análisis; por lo que la recolección de datos se desarrolló, partiendo de una revisión bibliográfica compuesta por 4 libros, 10 tesis y 13 artículos.

De esta forma y teniendo en cuenta dichos criterios metodológicos, a continuación se describe la manera como se ejecutó la presente propuesta desarrollada en la Universidad de Cartagena; dando cumplimiento, a cada una de las etapas iniciadas en el segundo semestre del año 2011 hasta el segundo semestre del 2012.

En la etapa inicial y para la consecución del primer objetivo específico, se efectuó una revisión literaria en aras de construir un referente teórico que propició delimitar y desarrollar el proyecto de investigación, en forma coherente con los cambios y avances presentes en la actualidad. Es por tal, que la temática desarrollada a lo largo del proyecto se basó en la gestión de configuración de red, modelos de gestión de red integrada (OSI y TMN, Internet o SNMP, y WBEM), arquitectura 4+1 con sus vistas, modelos ontológicos, herramientas de gestión de configuración de red, lenguaje de definición y pasos para el desarrollo de ontologías, tecnologías de comunicación entre agentes inteligentes, arquitectura de agentes inteligentes, y en especial, estudios de casos de aplicación relacionados con ésta temática.

En la segunda etapa, teniendo como fundamentado la información especificada en el referente teórico; se procedió a diseñar y desarrollar el modelo ontológico de gestión de configuración de red, tomando como guía la propuesta presentada por NOY & McGUINNESS para el año 2005. La funcionalidad del modelo ontológico en mención, fue planteada a partir de la definición de clases, la organización de las clases en jerarquía taxonómica (subclase-superclase), la definición de slots y descripción de valores permitidos para esos slots, componentes, relaciones, funciones, instancias y axiomas, logrando el cumplimiento del segundo objetivo específico.

En la tercera etapa, apoyado en el modelo 4+1 vistas y tomando como base la literatura desarrollada en el marco teórico relacionada con Agentes de software, ontologías y gestión de configuración de redes; se construye describiendo cada una de las vistas (Lógica, Procesos, Desarrollo o implementación, Física y escenarios), la arquitectura de software basada en agentes inteligentes que permite la obtención y manejo de información para gestionar la configuración de red. Con lo cual se obtuvo el tercer objetivo específico.

En la cuarta etapa, se implementó y desarrolló el prototipo de agente inteligente amparado en el Modelo de Construcción de Prototipos de software (Pressman, Ingenieria de software: Un enfoque práctico., 2002) y en el empleo del framework JADE. Lo anterior, propició la creación de un producto reutilizable y de calidad; cuya codificación, configuración e instalación, probaron la arquitectura planteada, que apoyada en tecnologías de comunicación entre agentes, permitió un manejo eficaz y eficiente de la información de configuración de red. Esto permite la consecución del cuarto objetivo específico.

Para concluir el proceso, se procedió a validar el prototipo obtenido, a través de pruebas de caja blanca y caja negra en un entorno de ejecución caracterizado por la simulación en una máquina. Teniendo en cuenta que el prototipo soporta los cinco aspectos que engloban la gestión de configuración de red, se optó por seleccionar el escenario *Estado actual de la red* para el caso práctico de la validación, por constituirse en el punto de partida para el desarrollo de los demás niveles que implica la gestión de configuración. De esta forma, se dio cumplimiento al quinto objetivo específico.

5. RESULTADOS DEL PROYECTO

Este capítulo muestra la consecución de los objetivos específicos durante el desarrollo del proyecto. Con el ánimo de dar cumplimiento a los mismos; inicialmente fue establecido en el capítulo 4 el estado del arte, cumpliendo así con el primer objetivo específico; posteriormente y conforme se desarrolla el presente capítulo, se mostrará cada una de las secciones donde se alcanzan los demás objetivos específicos.

5.1. DISEÑO DEL MODELO ONTOLÓGICO PARA LA GESTIÓN DE CONFIGURACIÓN DE RED

Tomando como apoyo la literatura relacionada en el marco teórico, y para fines de resaltar aspectos relevante que permitan desarrollar el modelo ontológico, conviene tener presente que; una ontología, es una descripción explícita y formal de conceptos en un dominio de discurso (**clases** (a veces llamadas **conceptos**)), propiedades de cada concepto describiendo varias características y atributos del concepto (**slots** (a veces llamados **roles** o **propiedades**)), y restricciones sobre los slots (**facetas** (algunas veces llamados **restricciones de rol**)). Una ontología junto con un conjunto de individuos de clases constituye una **base de conocimiento.**

En términos prácticos, desarrollar una ontología incluye: definir clases en la ontología, organizar las clases en una jerarquía taxonómica (subclase-superclase), y finalmente, definir slots y describir valores permitidos para esos slots. (NOY & McGUINNESS, 2005)
En este sentido, no existe ni una sola forma o ni una sola metodología "correcta" para desarrollar ontologías; sin embargo, aquí se plantean los puntos generales que deben ser tomados en consideración, y se ofrece uno de los procedimientos posibles para desarrollarlas. Por tanto, describiendo un enfoque iterativo en el desarrollo de la ontología: se comienza por abordar la ontología de manera frontal. A continuación, se retorna a la ontología, considerada en proceso de evolución, afinándola y completándola con detalles. A

lo largo de este proceso se discuten las decisiones de modelización que toma el diseñador, así como los pros, los contras y las implicaciones de diferentes soluciones.

A esta instancia, se quiere enfatizar algunas reglas fundamentales en el diseño de ontologías a las cuales se hace referencia en reiteradas oportunidades. Esas reglas pueden parecer algo dogmáticas. Ellas pueden ayudar, sin embargo, para tomar decisiones de diseño en muchos casos.

1. No hay una forma correcta de modelar un dominio - siempre hay alternativas viables. La mejor solución casi siempre depende de la aplicación que se tiene en mente y las extensiones a las que se anticipa.
2. El desarrollo de ontologías es un proceso necesariamente iterativo.
3. Los conceptos en la ontología deben ser cercanos a los objetos (físicos o lógicos) y relaciones en el dominio de interés. Esos son muy probablemente sustantivos (objetos) o verbos (relaciones) en oraciones que describen el dominio. (NOY & McGUINNESS, 2005)

Por tanto, para la creación de esta primera ontología cuyo dominio es la gestión de configuración de red (OntologyTesis.owl ubicada en la carpeta Prototipo/Ontologia del CD de instalación), se sigue el proceso de desarrollo propuesto por Noy y McGuinness, que se encuentra descrito en el marco teórico del presente documento.

Paso1. Determinar el dominio y el alcance de la Ontología

Como primera instancia, se tratan de dar solución a interrogantes como:

- *¿Cuál es el dominio que la ontología cubrirá?*

Considerando el modelo ontológico, el dominio lo constituye la gestión de configuración de red; naturalmente, los conceptos que describan el estado actual de la red, la gestión de inventario, el mantenimiento de directorios, la coordinación del esquema de nombres para nodos y aplicaciones, y el control operacional de la red, permitieron delimitar el dominio que abarca la ontología (*Ver Figura 5*).

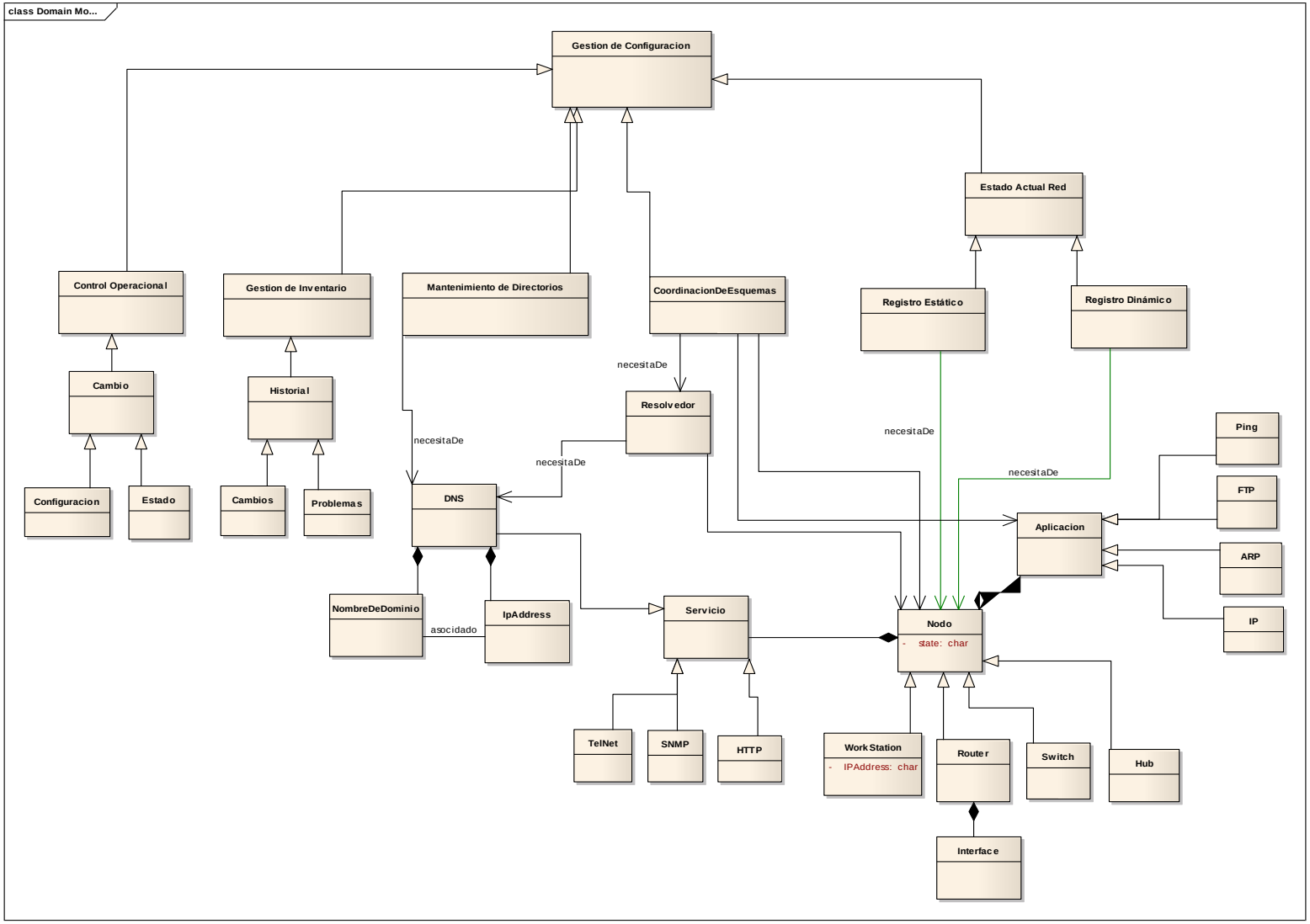

Figura 5. Conceptos y relaciones del marco conceptual Gestión De Configuración De Red.

Fuente: Los Autores.

- *¿Para qué se usa la ontología?*

La Ontología se usa para proveer funciones que permitan operar, controlar, identificar y recopilar datos de una red y así, facilitar la labor de gestión de configuración ejercida por un administrador de red.

- *¿Para qué tipo de preguntas de información en la ontología deberá proveer respuestas?*

Con el fin de dar solución a preguntas de información del tipo: ¿Cuántos PCs se encuentran activos y/o inactivos?, ¿Cuáles son las direcciones IP de las estaciones de trabajo?, ¿Cuántas son las interfaces de un Router?, ¿En qué periodo (mes, semana, horas) se han encontrado mayor número de PCs inactivos?, ¿Qué aplicaciones se encuentran activas en un nodo?, ¿Cuáles son las aplicaciones más usadas por un nodo?, ¿Cuál es el esquema de nombres establecidos para nombres y aplicaciones? O ¿Cuántos dispositivos deseas iniciar y/o detener?.

Para el caso práctico de estudio (desarrollo del prototipo), se diseñó la ontología teniendo en cuenta el concepto de *estado actual de la red*. Con el fin de dar solución a cuestionamientos relacionados, en primer lugar, al registro estático, tales como: ¿Qué está instalado?, ¿Dónde está instalado?, ¿Cómo está conectado?, ¿Quién responde por cada cosa? y ¿Cómo comunicarse con los responsables?. Y en segundo lugar, aquellos vinculados al registro dinámico; donde se pretendió resolver interrogantes con respecto al Estado operacional de los elementos de la red y al Estado de conexión de cada dispositivo de infraestructura.

- *¿Quién usa y mantiene la ontología?*

La ontología es usada por el prototipo de Agente de gestión de configuración de red, y su mantenimiento corre por cuenta del Administrador del sistema.

Paso2. Considerar la reutilización de Ontologías existentes

Luego de una búsqueda exhaustiva en bibliotecas de ontologías reusables en la web y en la literatura, tales como: Ontolingua y DAML; y de acuerdo a la investigación

expuesta en el estado del arte del presente trabajo, se evidenció un progreso lento en cuanto al uso de ontologías como medio para la integración semántica de los distintos dominios de gestión de red existentes.

Por esta razón, en el momento de comenzar la definición de la ontología; de entre las contadas propuestas, la que mayor valor aportó fue la escrita por Samira Abar, titulada: *DomainOntology-NMS.owl*; una ontología de dominio para el protocolo TCP/IP basada en SNMP administrado el sistema de comunicaciones de gestión de red (Protegé, 2012).

Este enfoque, ayudó a agilizar el proceso de codificación de la ontología que se presenta en este proyecto. Si bien no se reutilizó en un sentido estricto, con base en ella, se tomaron algunos atributos que debían contener las clases, por ejemplo, para la clase nodo: IPAddress, MAC_Address, DomainName, entre otras. Al igual que, las relaciones de composición y uso entre las clases; por ejemplo, el slot NecesitaDe, fue basado en esta ontología.

Paso 3. Enumerar términos importantes para la ontología

La especificación del marco conceptual relaciona una serie de términos comúnmente usados a la hora de describir la gestión de red, tales como:

1. *Gestión de Configuración*,
2. *Coordinación de esquemas*,
3. *Estado actual de la Red*,
4. *Registro Estático* y *Dinámico*,
5. *Gestión de inventario*,
6. *Mantenimiento de directorios*,
7. *control operacional de la Red*, *dispositivos*, *aplicaciones* y *servicios*.

Cada uno de estos términos representa un concepto para el que la metodología define cómo ha de interpretarse en cada modelo (*Ver ANEXO A. GLOSARIO DE TÉRMINOS DE LA* ***ONTOLOGÍA***). En la definición de cada concepto se descubren otros nuevos, por ejemplo:

1. *Nodo*,
2. *repetidores*,
3. *concentradores*,
4. *puentes*,
5. *conmutadores*,
6. *enrutadores* y *puertas de enlace*,
7. *Workstation*,
8. *ftp*,
9. *Smtp*,
10. *ARP, SEND IP PACKET, RECEIVE IP PACKET*
11. *DNS*,
12. *Telnet*,
13. *SNMP*,
14. *dirección ip*,
15. *red*, *directorios*,
16. *esquema de nombres*,
17. *inventario*,
18. *localización*,
19. *topología*,
20. *host*,
21. *resolvedor*,
22. *MAC Address*, *SNMP*,
23. *NMS* y
24. *MIB*;

Y también, otros términos que se utilizan para expresar relaciones entre dichos conceptos, como por ejemplo, un nodo *tiene* ip adress, los registros *necesitan de* nodos, y así sucesivamente.

Paso 4. Definir la clase y la jerarquía de clases

Al contar con un marco conceptual ya definido, esta etapa y la siguiente se simplifican considerablemente. En efecto, para determinar las clases de la ontología se seleccionan los términos de la etapa anterior para los que la metodología define un concepto. Asimismo, la representación del marco conceptual en un diagrama de clases UML (*Ver Figura 5*), aunque no recoge dicho marco de forma completa, constituye un bueno punto de partida que ayuda al proceso de creación de la ontología, puesto que las clases UML pueden traducirse directamente a clases de la ontología.

Partiendo de un enfoque combinado, empleado para desarrollar la jerarquía de clases; inicialmente se definieron los conceptos más sobresalientes y luego, se generalizaron y especializaron apropiadamente.

Tomando como base el modelo conceptual presentado en el paso 1, se codificó la Ontología usando la herramienta Protégé.

En la *Figura 6*, se evidencia la idea de definir clases que provean un marco de referencia para la Gestión de Configuración de Red. De esta forma, se muestrael conjunto de clases que resulta y la descomposición entre los diferentes niveles de generalidad.

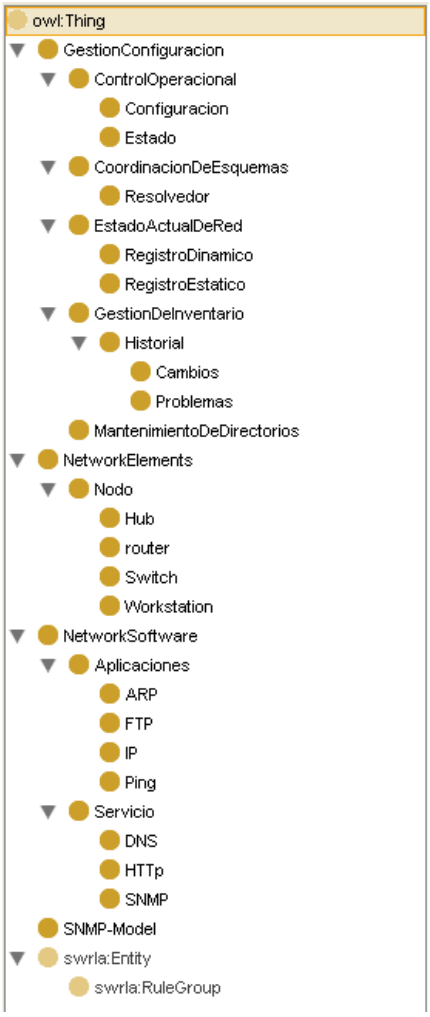

Figura 6. Ontología que representa el sistema de Gestión de Configuración de Red.

Fuente: Los Autores.

Paso 5. Definir las propiedades de las clases: Slots.

Por medio de las propiedades se estructuran los conceptos, lo que a su vez, configura la arquitectura del sistema de gestión de configuración de red.

En la lista de términos obtenida en el *paso 3*, aparecían bastante delimitados, por un lado, aquellos que correspondían a conceptos, y por otro, los utilizados para relacionarlos o describir sus propiedades. Tras haber seleccionado de dicha lista los términos que conformarán conceptos (*paso 4*), la mayor parte de los que queden representarán propiedades. Por esta razón, en la ontología se necesitan los slots mostrados en la *Figura 7*.

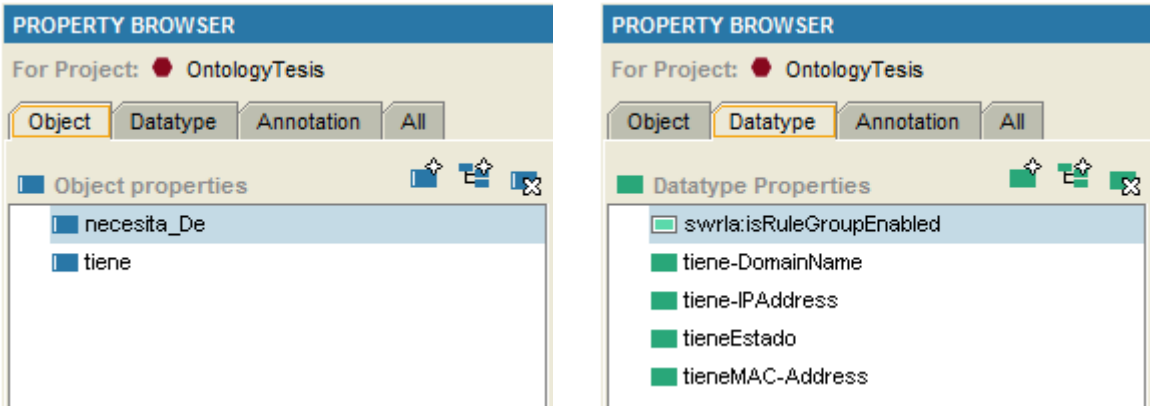

Figura 7. Gráfico que ilustra los Slots en nuestra ontología.

Fuente: Los Autores.

Ahora bien, la clase Nodo, necesita añadir los siguientes slots: DomainName, IPAddress, Estado, MAC-address (*ver Figura 8).* Estas propiedades serán heredadas por las subclases de Nodo.

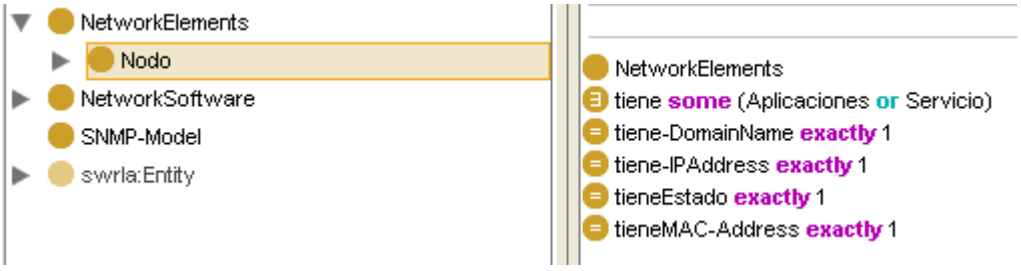

Figura 8. Gráfico que ilustra los Slots asociados a la clase Nodo.

Fuente: Los Autores.

También, las subclases de EstadoActualDeRed, necesitan añadir el slot "necesitaDe", donde se relacionan con Nodo, en la *Figura 9* se ilustra.

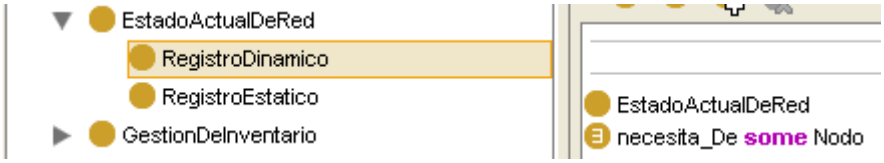

Figura 9. Gráfico que ilustra los Slots asociados a la clase EstadoActualDeRed.

Fuente: Los Autores.

Paso 6. Definir facetas y/o restricciones sobre los slots o relaciones

Los slots pueden tener diferentes facetas que describen el tipo de valor; valores admitidos, el número de los valores (cardinalidad), y otras características de los valores que pueden tomar.

De acuerdo a lo indicado en el paso anterior, se establece lo siguiente:

- ✓ *String*: Son aquellos atributos que se representan por cadenas de caracteres, por ejemplo: tiene-DomainName, tiene-IpAddress, tieneMAC-Address.
- ✓ *Boolean*: Son las propiedades que pueden tomar dos valores (true or false), por ejemplo, el slot tiene Estado.
- ✓ *Instance*: necesita_De.

Paso 7. Definir instancias.

A este nivel, basados en los resultados de los pasos anteriores, se procede a definir las instancias para las clases más relevantes en la jerarquía. Esta definición requiere seleccionar una clase, crear una instancia individual y rellenar los valores del slot. Con el fin de dar ejemplo, y para el caso particular; se definen las instancias RegistroEstatico, RegistroDinamico y las subclases de Nodo (*Figura 10, Figura 11 y Figura 12*)

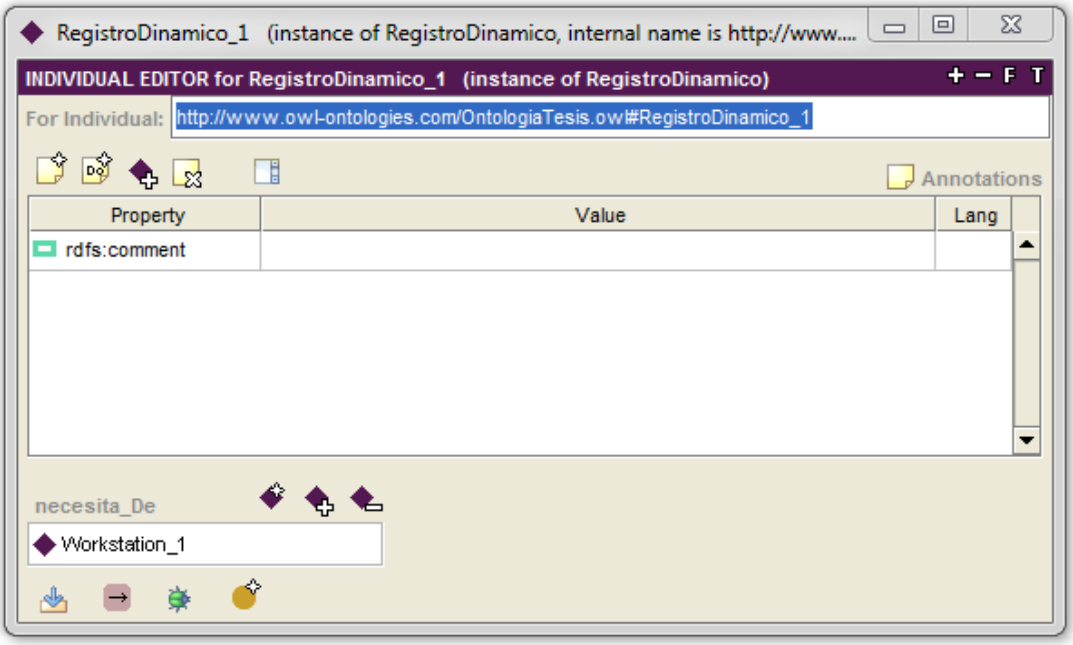

Figura 10. Gráfico que ilustra la relación entre las instancias RegistroDinamico_1 y Workstation_1.

Fuente los Autores.

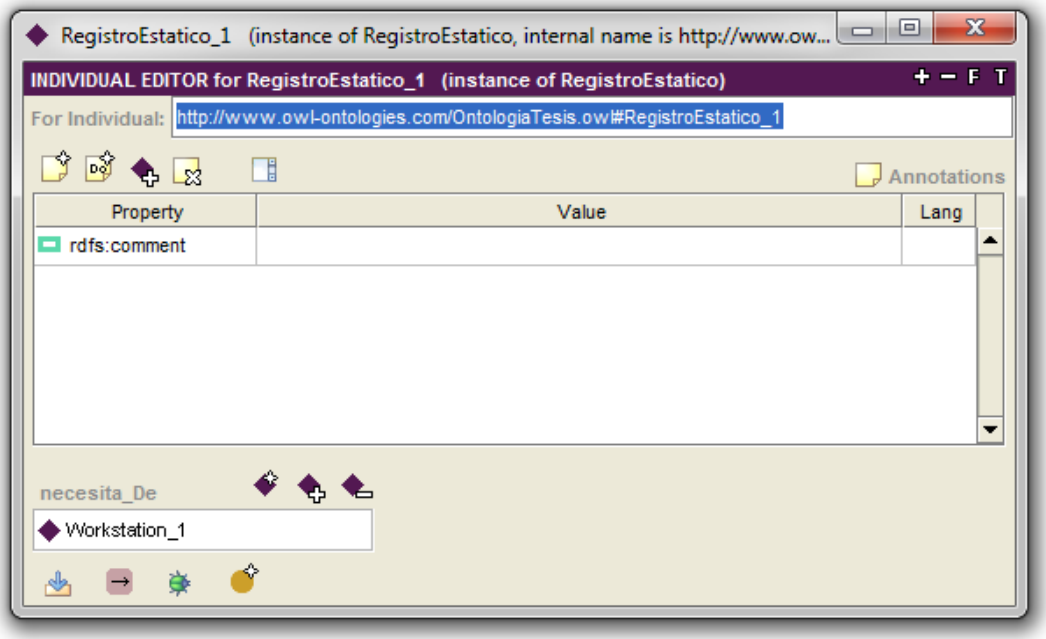

Figura 11. Gráfico que ilustra la relación entre las instancias RegistroEstatico_1 y Workstation_1.

Fuente Los Autores.

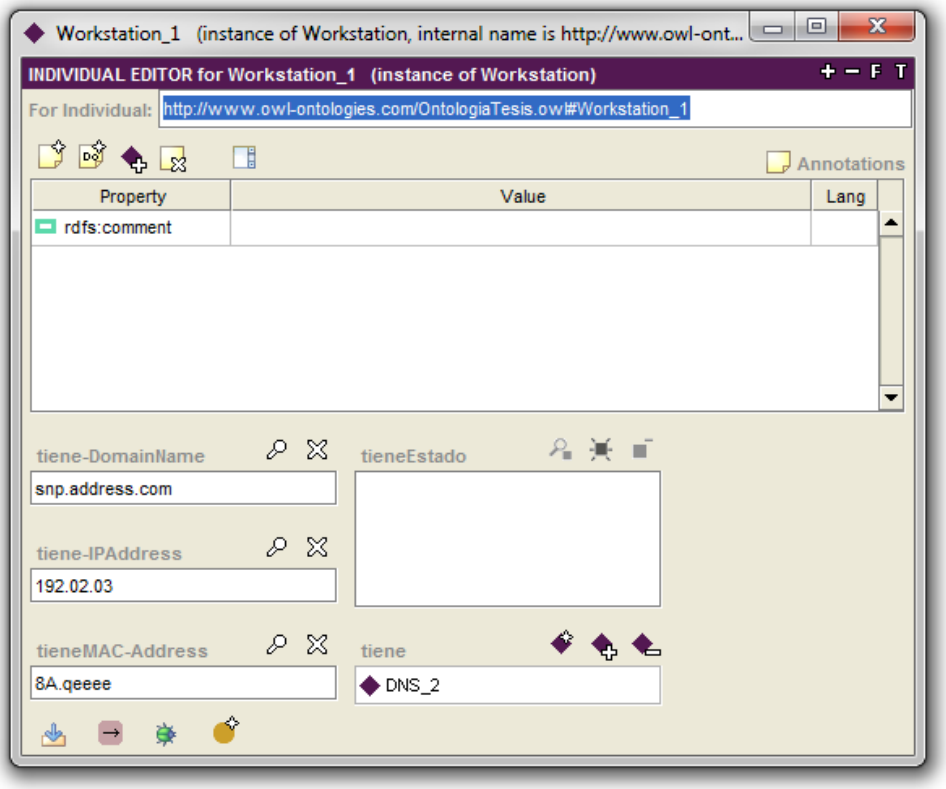

Figura 12. Gráfico que ilustra los campos concernientes a la instancia Workstation_1, debido a su asociación con las instancias RegistroEstatico_1 y RegistroDinámico_1.

Fuente: Los Autores.

5.2. DISEÑO DE LA ARQUITECTURA BASADA EN AGENTES INTELIGENTES

Tomando como base la literatura desarrollada en el marco teórico relacionada con Agentes de software, ontologías y gestión de configuración de redes, en esta sección se presenta la arquitectura propuesta con la descripción de cada una de las Vistas(Lógica, Procesos, Desarrollo o implementación, Física y escenarios) que la conforman, para cumplir con el tercer objetivo especifico.

5.2.1. Vista Lógica

Esta vista es conformada por tres componentes que implementan la funcionalidad principal del sistema, que consiste en facilitar la gestión de configuración de la red. Los componentes son: Pizarra, Mensajero y Gestor (*Ver Figura 13*); donde el Componente Pizarra es el encargado de solicitar un servicio, el Componente Mensajero gestiona la petición y la lleva esta solicitud al Componente Gestor, y este a su vez, ofrece la interfaz gestionar red y de ahí se realiza el proceso inverso.

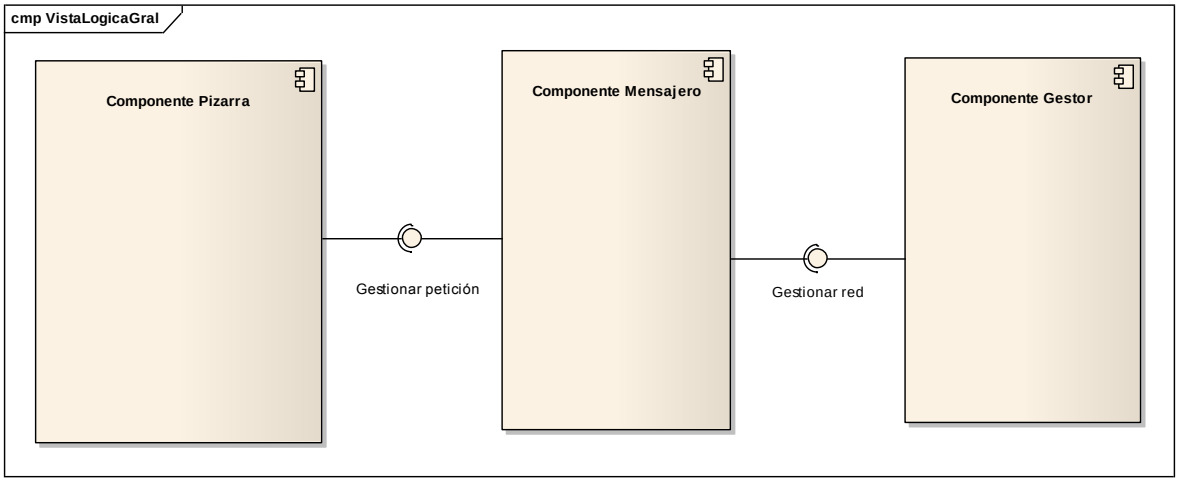

Figura 13 . Gráfico que ilustra la vista lógica general.

Fuente: Los Autores.

❖ ***Componente Pizarra***

Este componente es el encargado de manejar la interfaz gráfica del usuario y está conformado por los siguientes subsistemas como se ilustra en la Figura 14:

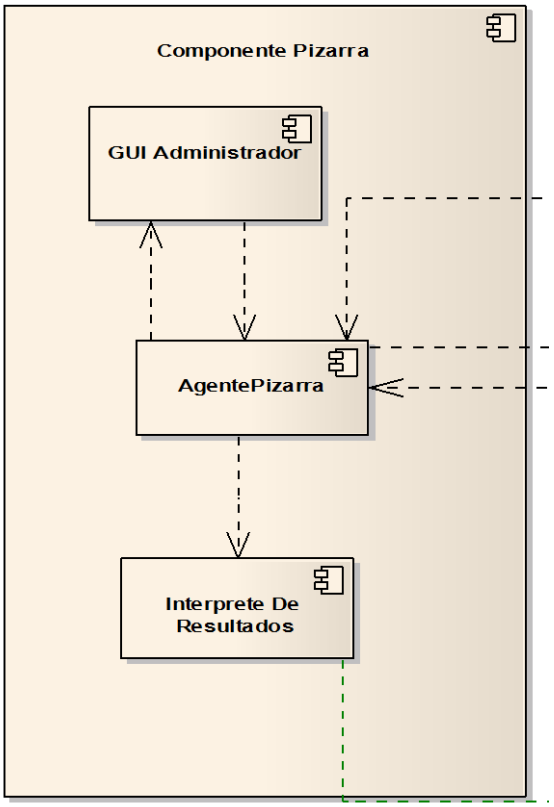

Figura 14. Gráfico que ilustra el componente pizarra.

Fuente: Los Autores.

GUI Administrador: Contiene todas las clases y paquetes que permiten el desarrollo del entorno gráfico en donde el usuario realiza las peticiones al sistema.

Agente Pizarra: Contiene al AgenteMovilPizarra, encargado de obtener las peticiones realizadas por el Usuario e interactuar con el Agente Mensajero a través del ReceptorDePeticiones para gestionar las solicitudes. Además, interactúa con el InterpreteDeResultados para obtener la información solicitada.

InterpreteDeResultados: Este subsistema se encarga de recorrer la ontología del sistema para descifrar los resultados que suministra el Agente Mensajero para que sean usados por el AgentePizarra de una manera entendible para el Usuario.

- ***Componente Mensajero***

Este componente se encarga de comunicar al Componente Pizarra y al Componente Gestor, está conformado por los siguientes subsistemas (*Ver Figura 15*):

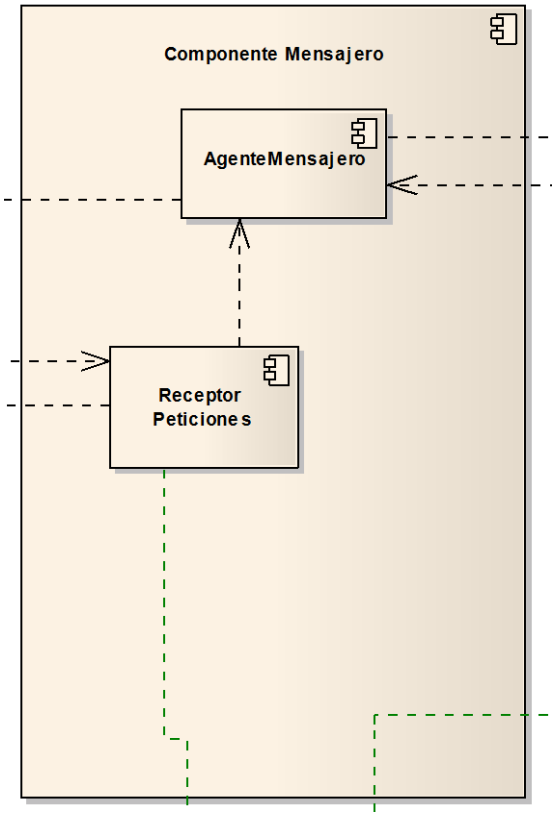

Figura 15. Gráfico que ilustra el componente Mensajero.

Fuente: Los Autores.

AgenteMensajero: este subsistema se encarga de interactuar con el AgenteGestor a través del InterpreteDePeticiones, y le proporciona la petición solicitada por el Usuario.

ReceptorDePeticiones: este subsistema recibe las peticiones realizadas por el Usuario y las suministra al AgenteMensajero; es decir, es el intermediario entre el AgentePizarra y el Agente Mensajero, disminuyendo el acoplamiento en el sistema.

- ***Componente Gestor***

Este componente se encarga de realizar las peticiones solicitadas por el administrador y está conformado por los siguientes subsistemas (*Ver Figura 16*):

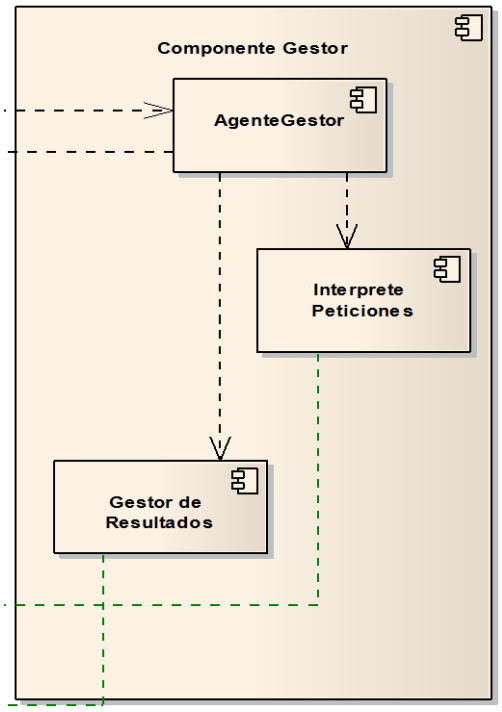

Figura 16. Gráfico que ilustra el componente Gestor.

Fuente: Los Autores.

AgenteGestor: contiene las clases y paquetes que permiten recorrer las estaciones de trabajo para obtener la información solicitada por el Usuario.

InterpreteDePeticiones: Se encarga de recibir las solicitudes que proporciona el AgenteMensajero, y consultar la Ontología para conocer la naturaleza de la petición realizada por el Usuario.

GestorDeResultados: obtiene y consulta la Ontología de Gestión para tabular la información consultada por el AgenteGestor, para que sea enviada por el AgenteMensajero al Usuario.

Por último, se muestra la Vista lógica detallada (*Ver Figura 17*), que presenta la interacción de cada uno de los subsistemas identificados en los requisitos.

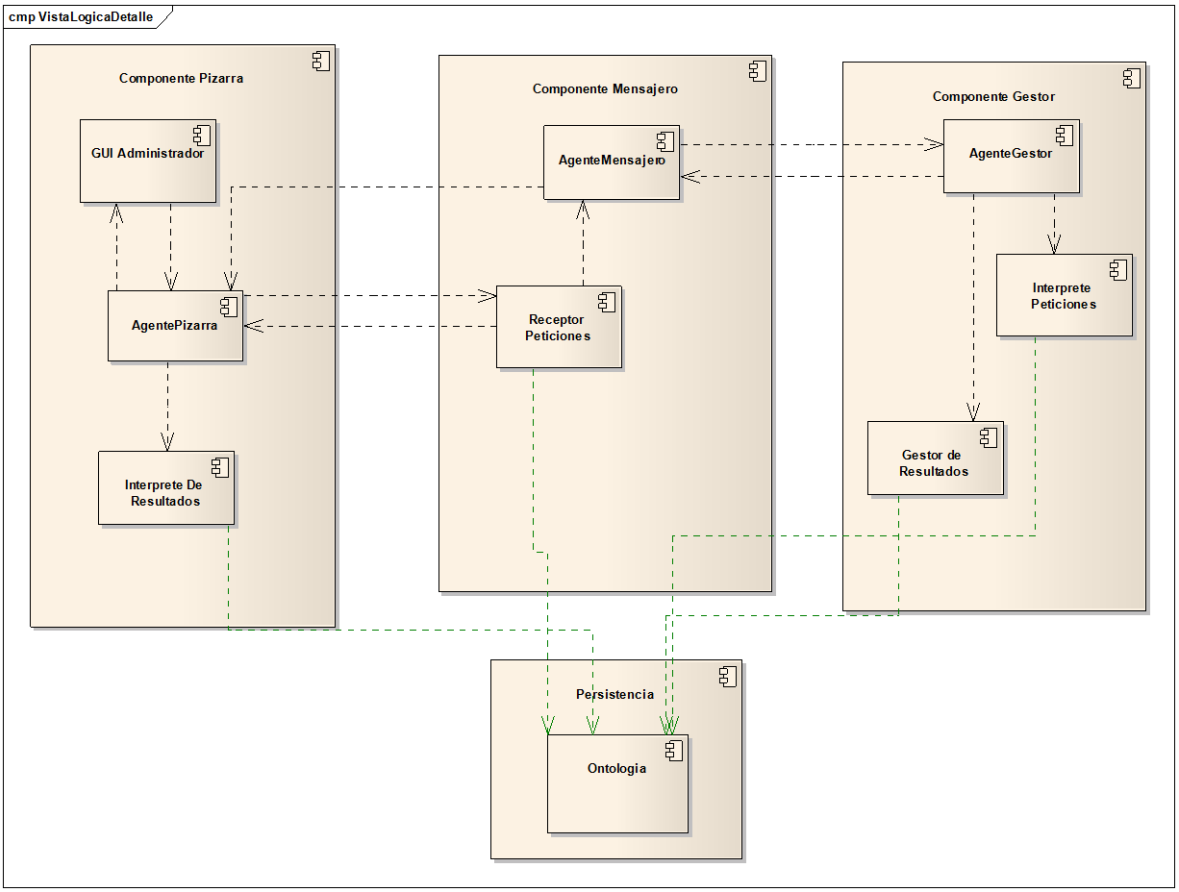

Figura 17. Gráfico que ilustra la Vista Lógica detallada.

Fuente: Los Autores.

5.2.2. Vista De Procesos

Para llevar a cabo el proceso de Gestión de Configuración por parte del Usuario, se deben cumplir los siguientes pasos (*Ver Figura 18*):

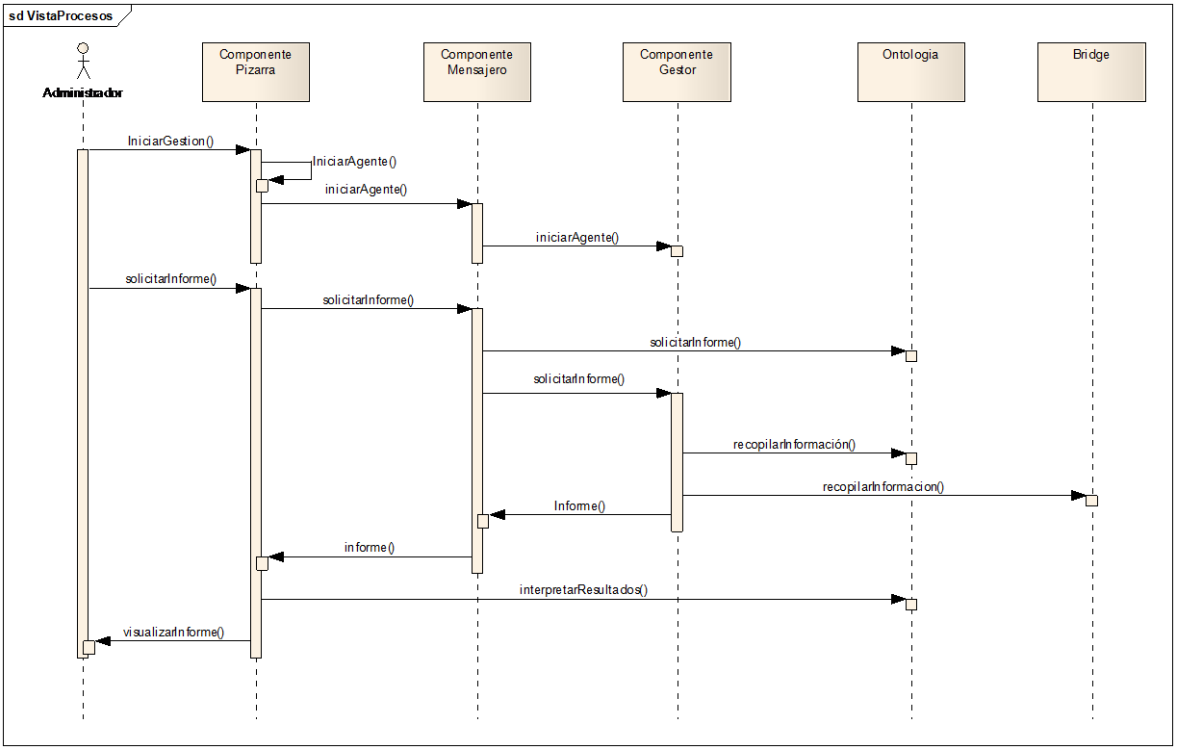

Figura 18. Gráfico que ilustra la Vista de Procesos (gestión solicitada por el administrador).

Fuente Los Autores

1) El Administrador inicia la Gestión interactuando con el ComponentePizarra
2) El componente pizarra inicia el Agente Pizarra.
3) El componente pizarra solicita el inicio del Agente Mensajero
4) El Agente Mensajero solicita al Componente Gestor iniciar el Agente Gestor.
5) El Administrador solicita un informe de Gestión al Componente Pizarra
6) El Agente Pizarra solicita el informe de Gestión al Componente Mensajero.
7) El componente Mensajero consulta a la Ontología de Gestión para interpretar la solicitud.
8) El componente Mensajero solicita el informe al Componente Gestor.
9) El componente Gestor consulta a la Ontología de Gestión para interpretar la solicitud.
10) El componente Gestor recopila la información en las estaciones de trabajo.
11) El componente Gestor retorna el informe al Componente Mensajero.
12) El componente Mensajero retorna el informe al componente Pizarra.

13) El componente Pizarra retorna el informe de Gestión al Usuario.

14) El componente Pizarra consulta la Ontología de Gestión, para interpretar los resultados.

15) El Administrador visualiza el informe generado.

Para realizar la gestión automática se llevan a cabo los siguientes pasos (*Ver Figura 19*):

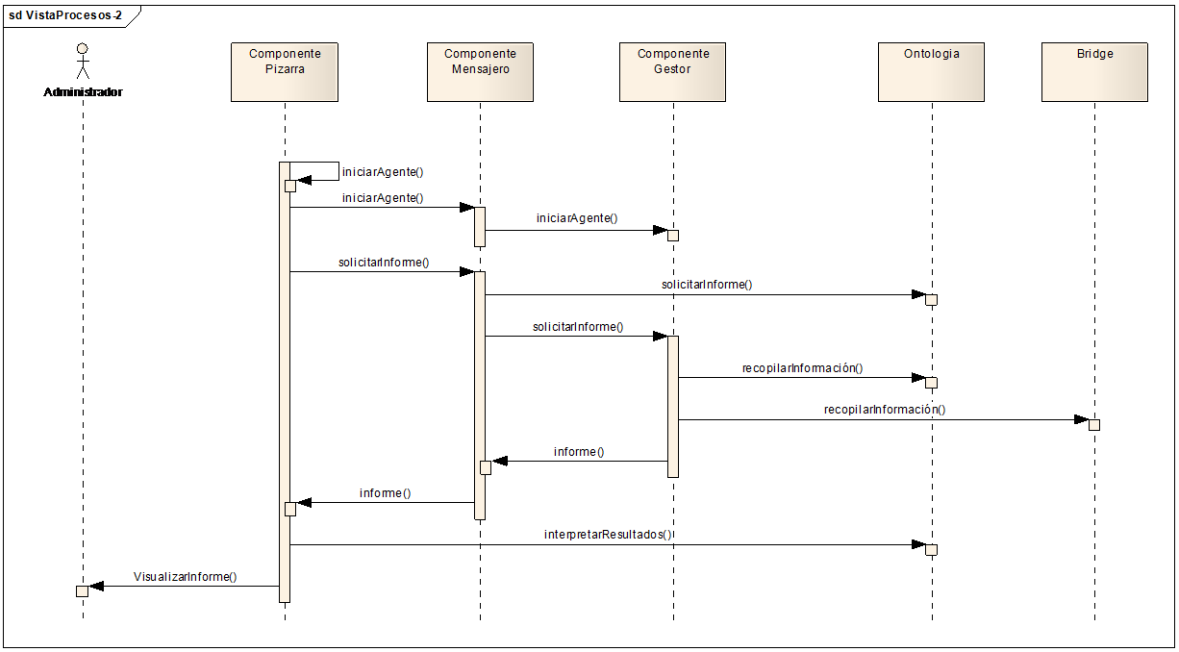

Figura 19. Gráfico que ilustra la Vista de Procesos (gestión automática).

Fuente Los Autores.

1) El componente pizarra inicia el Agente Pizarra.

2) El componente pizarra solicita el inicio del Agente Mensajero

3) El Agente Mensajero solicita al Componente Gestor iniciar el Agente Gestor.

4) El Administrador solicita un informe de Gestión al Componente Pizarra

5) El Agente Pizarra solicita el informe de Gestión al Componente Mensajero.

6) El componente Mensajero consulta a la Ontología de Gestión para interpretar la solicitud.

7) El componente Mensajero solicita el informe al Componente Gestor.

8) El componente Gestor consulta a la Ontología de Gestión para interpretar la solicitud.
9) El componente Gestor recopila la información en las estaciones de trabajo.
10) El componente Gestor retorna el informe al Componente Mensajero.
11) El componente Mensajero retorna el informe al componente Pizarra.
12) El componente Pizarra retorna el informe de Gestión al Usuario.
13) El componente Pizarra consulta la Ontología de Gestión, para interpretar los resultados.
14) El Administrador visualiza el informe generado.

5.2.3. Vista De Implementación

En esta vista se describe la estructura de clases que conforman cada componente; es decir, se muestran los patrones de micro arquitectura usados en cada uno de los subsistemas identificados en las secciones anteriores.

- ***Vista de Implementación – Componente Pizarra***

En esta sección se hace una descripción de las clases y paquetes que conforman el Componente Pizarra. A continuación se muestran los subsistemas que lo conforman:

GUIAdministrador. Este subsistema se encarga de realizar las interacciones con el Administrador y la manera como se presentan los resultados; para lograr esto, se aplica el patrón decorator, implementado en las clases: *(1)*GUIAdministrador, *(2)*ResultadoGrafico, *(3)*ResultadoGrafico, *(4)*DecoradorGrafico y *(5)* Tabla. (*Ver Figura 20*):

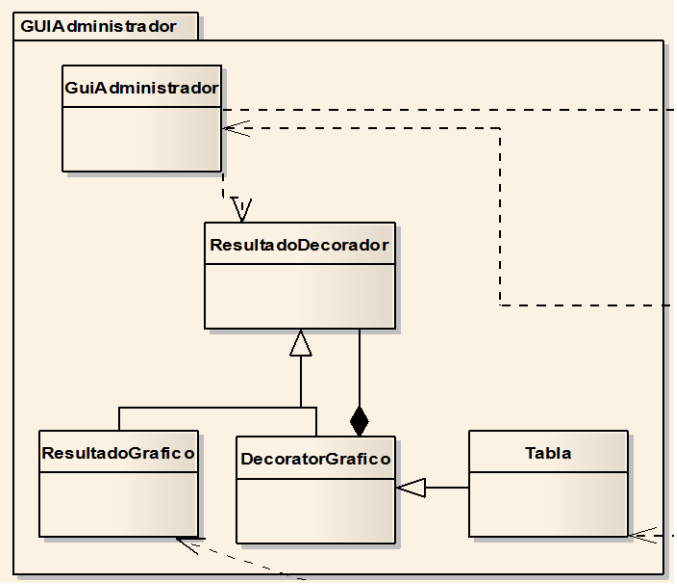

Figura 20. Gráfico que ilustra el paquete GUIAdministrador.

Fuente Los Autores.

AgentePizarra. Este subsistema como su nombre lo indica, contiene las clases para el manejo del Agente, este componente aplica el patrón proxy, para que otros objetos no accedan directamente al AgenteMovil, esto se describe con las siguientes clases:*(1)*Agente, *(2)*AgenteMovilPizarra y *(3)*ProxyAgente. (*Ver Figura 21*)

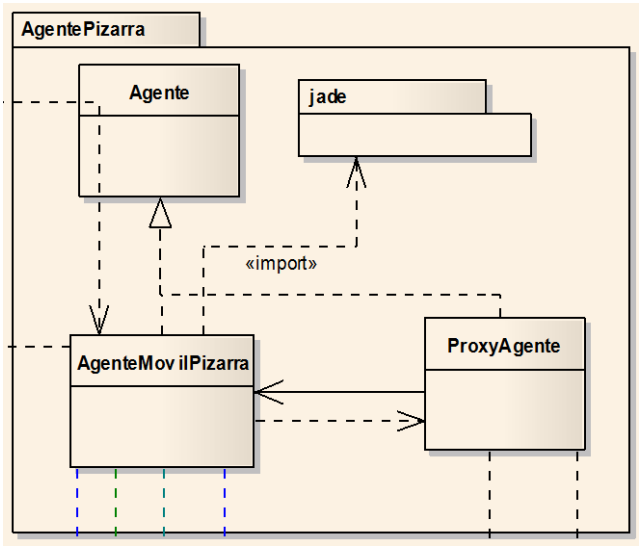

Figura 21. Gráfico que ilustra el paquete AgentePizarra.

Fuente Los Autores.

InterpreteDeResultados. Este subsistema como su nombre lo indica, contiene las clases que manejan los resultados obtenidos y le da un formato entendible para el usuario; este componente aplica los patrones Iterator y Bridge, se describe con las siguientes clases:*(1)*Iterador e IteradorConcreto, *(2)*Agregate y AgregadoResultado, *(3)*Resultado y ResultadoExt, *(4)*ResultadoImpl y *(5)*ResultadoFichero, ResultadoTabla y ResultadoLista. (*Ver Figura 22*):

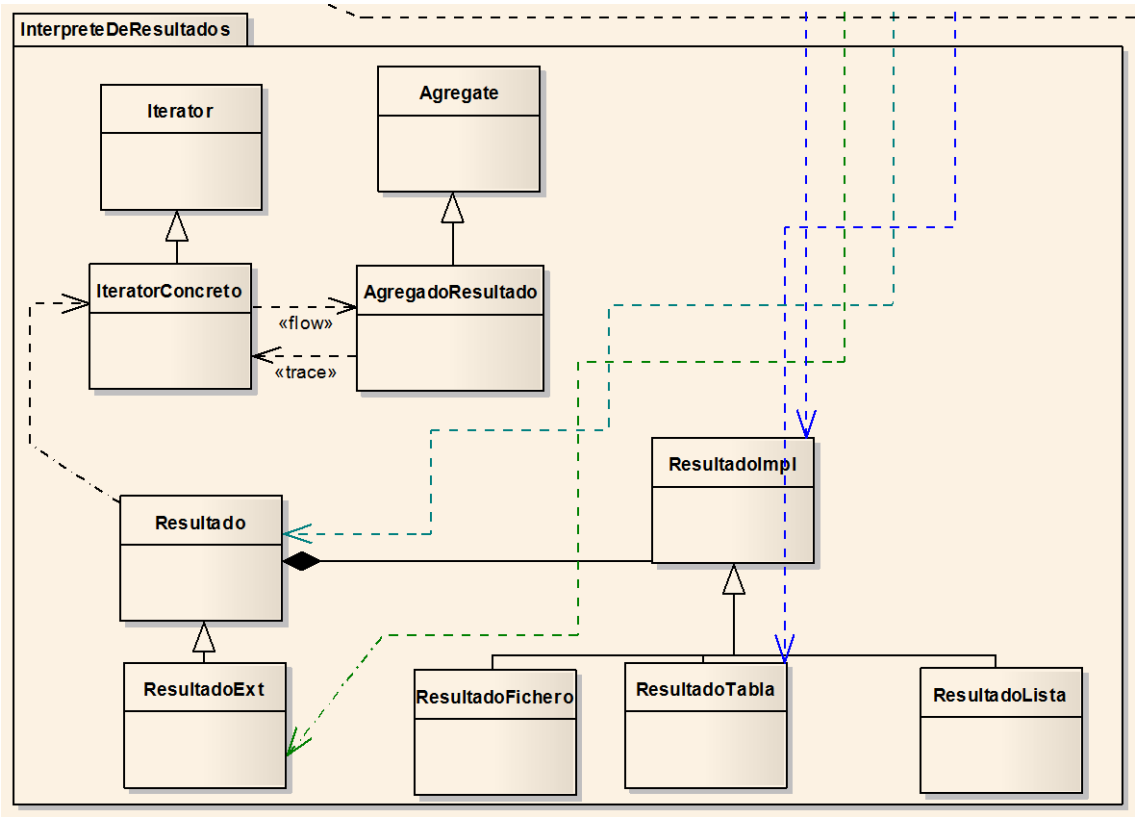

Figura 22. Gráfico que ilustra el paquete InterpreteDeResultados.

Fuente Los Autores.

- ***Vista de Implementación –Componente Mensajero***

En esta sección se hace una descripción de las clases y paquetes que conforman el Componente Mensajero. A continuación se muestran los subsistemas que lo conforman:

AgenteMensajero: Este subsistema como su nombre lo indica, contiene las clases para el manejo del Agente, este componente aplica el patrón proxy, para que otros objetos no accedan directamente al AgenteMovil, esto se describe con las siguientes clases: *(1)*Agente, *(2)*AgenteMovilMensajero y *(3)*ProxyAgenteMensajero. (*Ver Figura 23*):

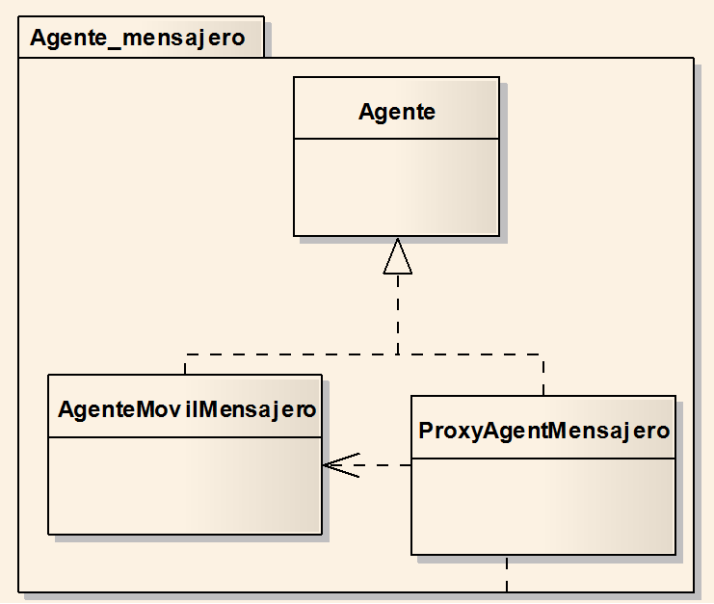

Figura 23. Gráfico que ilustra el paquete AgenteMensajero.

Fuente Los Autores.

ReceptorDePeticiones. Este subsistema como su nombre lo indica, contiene las clases que manejan las peticiones solicitadas por el Administrador; aplicando los patrones Iterator, Command, Singleton y Facade, usando las siguientes clases: *(1)*Iterador e IteradorConcreto, *(2)*Agregate y Receptor, *(3)*Invoker y Command, *(4)*Receiver y ConcretaPeticion y *(5)*ReceptorPeticion. (*Ver Figura 24*):

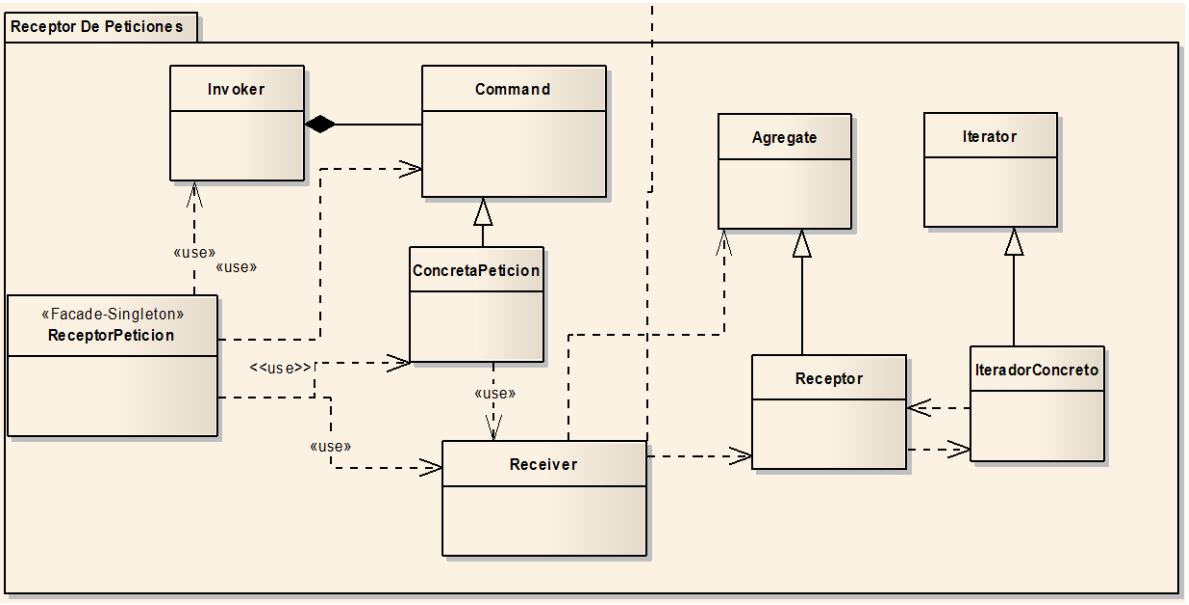

Figura 24. Gráfico que ilustra el paquete ReceptorDePeticiones.

Fuente Los Autores.

- ***Vista de Implementación – Componente Gestor***

En esta sección se hace una descripción de las clases y paquetes que conforman el Componente Gestor. A continuación se muestran los subsistemas que lo conforman:

AgenteGestor. Este subsistema como su nombre lo indica, contiene las clases para el manejo del Agente, este componente aplica el patrón proxy, para que otros objetos no accedan directamente al AgenteMovil, esto se describe con las siguientes clases:*(1)*Agente, *(2)*AgenteMovilGestor y *(3)*ProxyAgenteGestor. (*Ver Figura 25*):

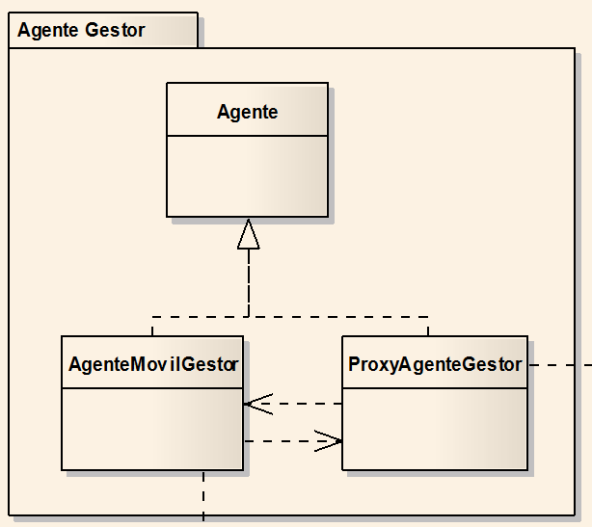

Figura 25. Gráfico que ilustra el paquete AgenteGestor.

Fuente Los Autores.

GestorDeResultados. Este subsistema como su nombre lo indica, contiene las clases que manejan los resultados obtenidos aplicando los patrones Facade, Singleton y Method Factory, usando las siguientes clases: *(1)*CreadorAbstracto y CreadorResultado y *(2)*ResultadoAbstracto y ResultadoConcreto. (*Ver Figura 26*):

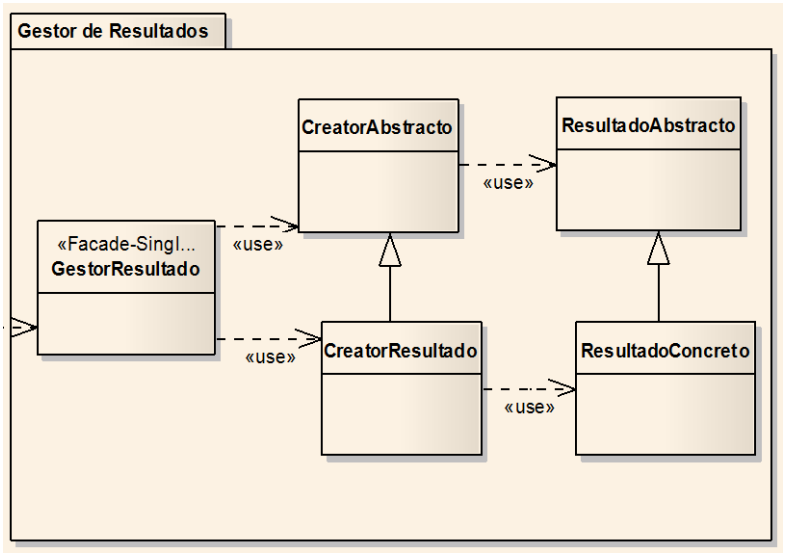

Figura 26. Gráfico que ilustra el paquete GestorDeResultados.

Fuente Los Autores.

InterpreteDePeticiones. Este subsistema como su nombre lo indica, contiene las clases que manejan las peticiones solicitadas por el Administrador a través del AgenteMensajero; aplicando los patrones Iterator, Command, Singleton y Facade, usando las siguientes clases: *(1)*Iterador e IteradorConcreto,*(2)*Agregado y Agregado,*(3)*Invoker y Command,*(4)*Receiver y ComandoInterpretey *(5)*InterpretePeticion. (*Ver Figura 27*):

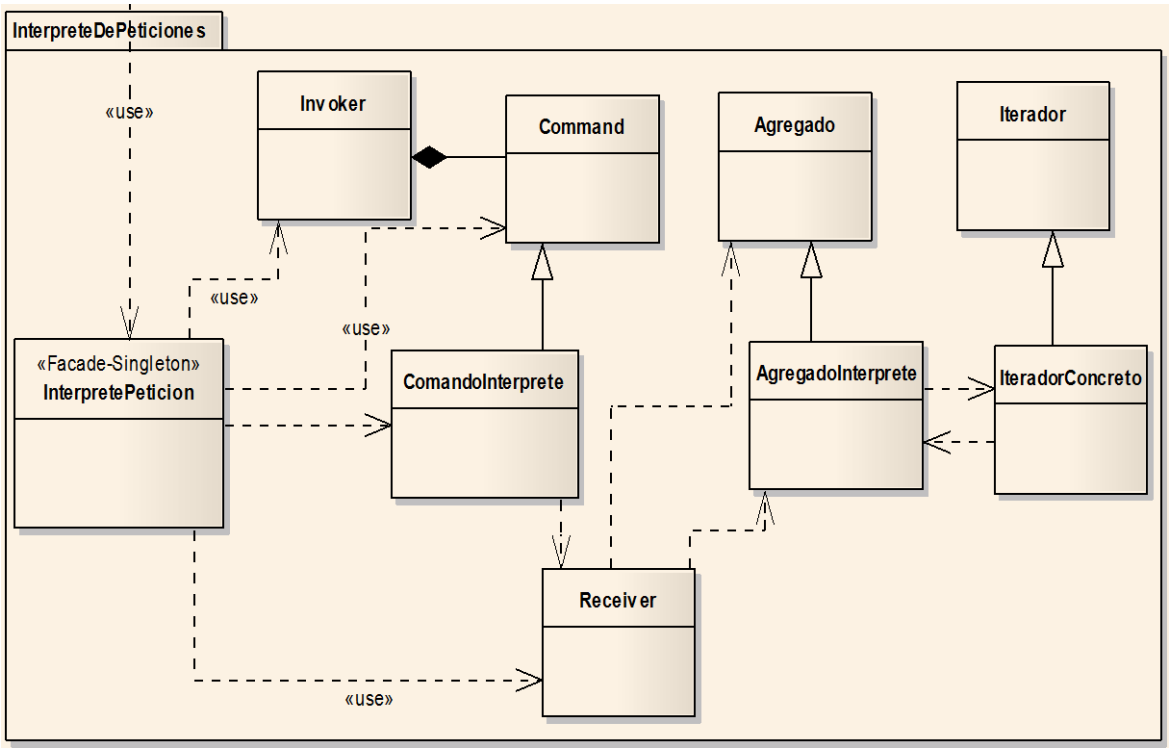

Figura 27. Gráfico que ilustra el paquete InterpreteDePeticiones.

Fuente Los Autores.

5.2.4. Vista De Despliegue

En esta vista se encuentran dos nodos, la WorkStation y el Server. El primero es el encargado de alojar el componente Gestor; y el segundo aloja los Componentes Pizarra, Mensajero y la Ontología (*Ver Figura 28*).

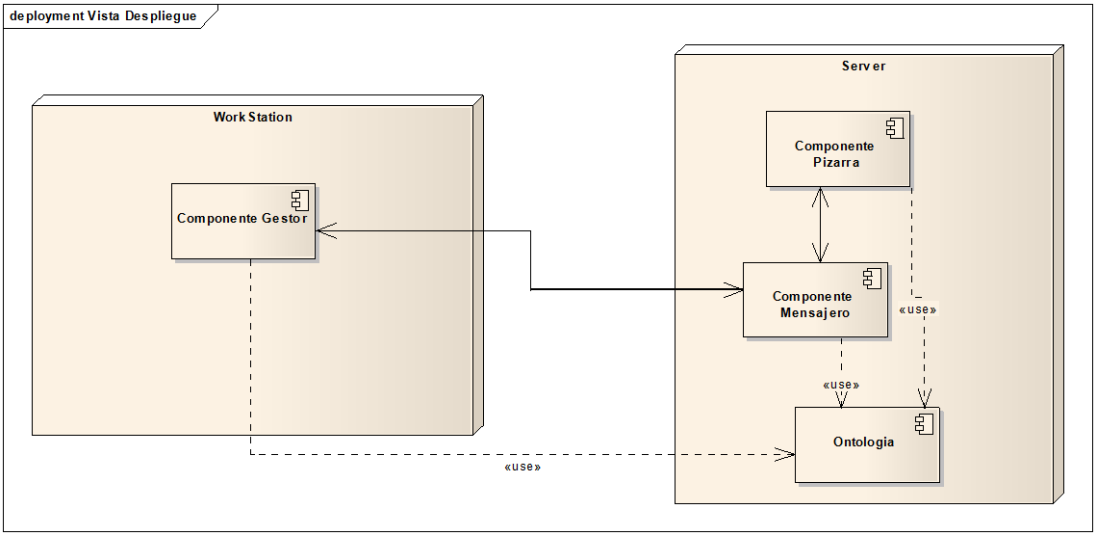

Figura 28. Gráfico que ilustra la Vista de despliegue general.

Fuente Los Autores.

A continuación, se muestran los entornos de ejecución de cada componente (*Ver Figura 29*):

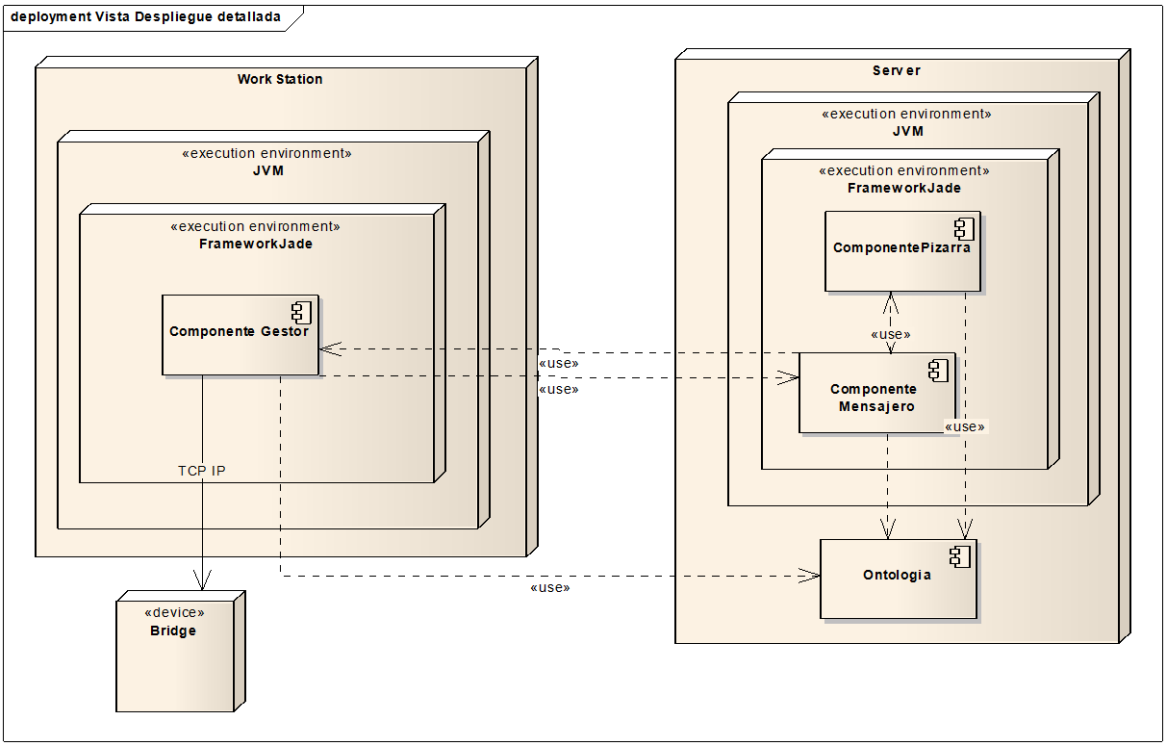

Figura 29. Gráfico que ilustra la Vista de despliegue detallada.

Fuente Los Autores.

- ✓ El componente Gestor (alojado en la Workstation), se ejecuta en la máquina virtual de Java, y en el Framework Jade (Java Agent Development).
- ✓ El componente Mensajero y Pizarra, tienen su entorno de ejecución con la JVM (Maquina virtual de Java) y el framework Jade en el Server.

5.2.5. Escenarios

En esta vista se muestran los requerimientos funcionales, identificados a lo largo del análisis y diseño de la arquitectura (*Ver Figura 30*).

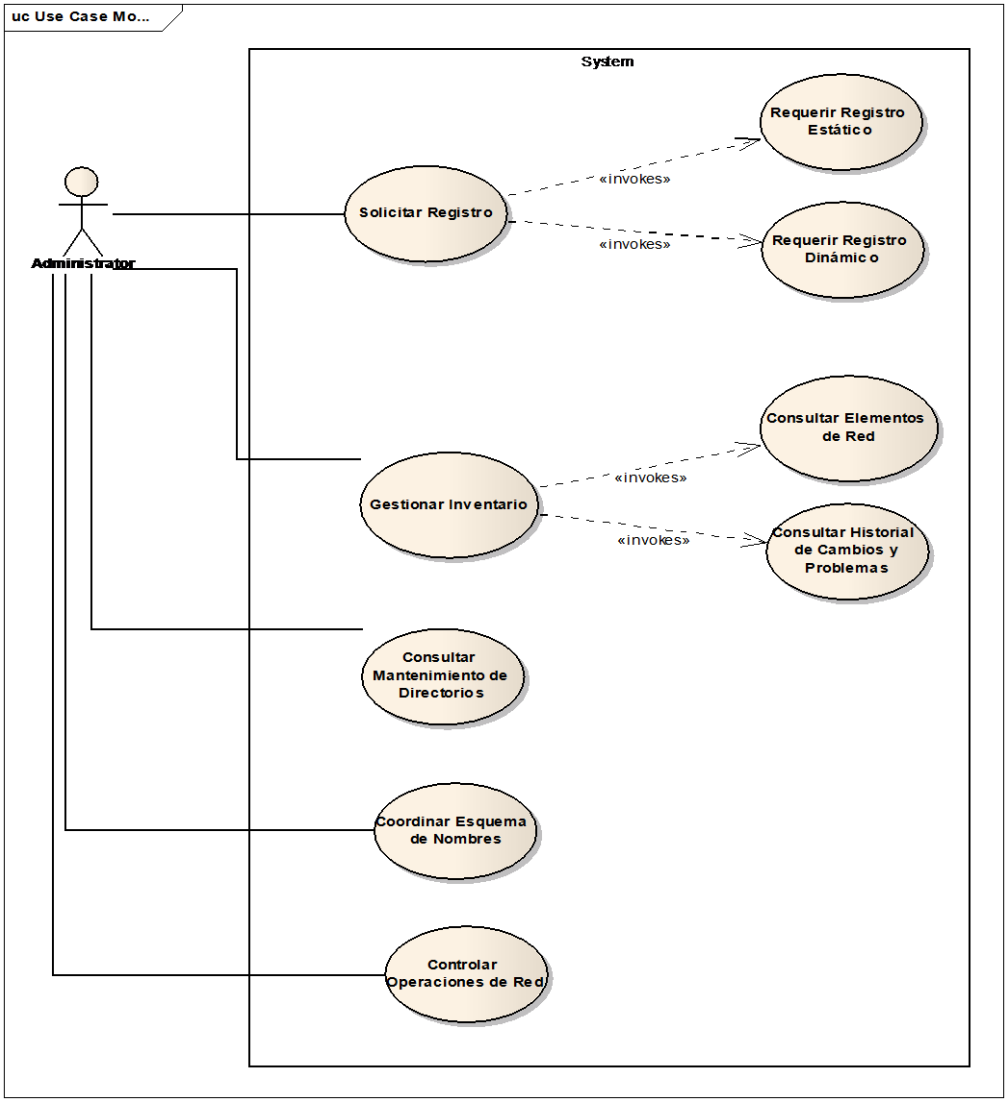

Figura 30. Gráfico que ilustra los escenarios del sistema.

Fuente Los Autores.

Solicitar Registro: permite que el Administrador requiera al sistema el registro estático o dinámica de la red gestionada; es decir, cuáles son los elementos que cambian de estado en un momento determinado o permanecen constantes.

Gestionar Inventario: permite al Administrador obtener la información correspondiente a los elementos que pertenecen a la red (Consultar elementos de red) y el historial de cambios y problemas en un tiempo determinado.

Consultar mantenimiento de directorios: permite al Administrador conocer los nombres de dominio de las estaciones de trabajo gestionadas y sus aplicaciones, y la base de datos de nombres de dominios.

Coordinar esquemas de nombres: permite al Administrador conocer los esquemas de clasificación de los nombres de dominios en la red gestionada.

Controlar operaciones de red: permite al Administrador cambiar la configuración de dispositivos, iniciar y detener elementos individuales, entre otras.

5.3. DESARROLLO DEL PROTOTIPO DE AGENTE INTELIGENTE

Tomando como base la arquitectura que se presentó en la sección anterior, y las vistas que la conforman (Lógica, Procesos, Desarrollo o implementación, Física y escenarios) en este apartado se presenta el Prototipo desarrollado con la descripción de cada uno de los aspectos concernientes a la metodología de desarrollo empleada, su codificación, configuración de la máquina, e instalación de la aplicación, para cumplir con el cuarto objetivo especifico.

❖ Metodología de desarrollo

Para implementar el prototipo se siguió los pasos recomendados en el modelo de construcción de Prototipos(Pressman, Ingenieria de Software: Un enfoque práctico, 2002):

- Se identifican los requerimientos
- Se construye y se revisa la maqueta
- El cliente prueba la maqueta.
- Se repiten los pasos anteriores.

❖ Codificación.

Para iniciar la codificación del prototipo, se tuvo en cuenta los distintos frameworks existentes para el desarrollo de agentes de software, entre los cuales los más usados son Aglets y JADE. El primero ofrece una interfaz gráfica donde se ejecutan y existen los agentes, denominada agencia; servicios de movibilidad cifrando el código y los datos de un aglet (agente aglet) usando el método de serialización de Java (JOS); el protocolo para el transporte de Aglets (ATP) para trasladar agentes. El segundo ofrece una interface gráfica para la administración remota de agentes; una plataforma distribuida; herramientas de debugging; movilidad de agentes inter-plataforma; ejecutar múltiples agentes en paralelo; transportar mensajes ACL dentro de la plataforma; una interface para que aplicaciones

externas inicien agentes autónomos; servicio de nombres(GUID: GloballyUniqueIdentifier); ejecutar las aplicaciones en un rango amplio de ambientes, entre ellos dispositivos móviles(PDA's y celulares).

Por otra parte, en cuento al uso e instalación del framework, Aglets resulta difícil de usar, ya que está ligado a la versión del sistema operativo que se utilice, y por lo tanto su instalación varía de acuerdo a esto, además la ejecución de la interface gráfica donde se ejecutan los agentes, es susceptible a fallos. En contraposición, JADE es fácil de instalar en cualquier sistema operativo, el uso de la interface gráfica para el manejo de agentes se despliegue sin ningún tipo de fallos; por esta razón en el desarrollo del prototipo se usa el Framework Jade.

Ahora bien, para crear los agentes con el framework JADE, se debe usar la clase *Agent* incluida en el paquete `jade.core`, la cual define las características básicas para que el agente interactúe con la plataforma(registro, configuración, administración remota) y los métodos para implementar el comportamiento del agente(envío y recepción de mensajes, uso de protocolos de interacción estándar.).

- ✓ Crear una clase hija de Agent (por ejemplo: AgentePizarraextendsAgent

❖ Configuración

Para la correcta ejecución del Prototipo, es necesario que la máquina en la cual se instale cuente con las siguientes características:

- ✓ Procesador Pentium 4 de 2.8 GHz.
- ✓ 2 Gigas de RAM o mínimo 1GB de RAM.
- ✓ Disco duro con espacio libre de 100 MB.
- ✓ Tarjeta de Red.

Además se deben instalar las siguientes librerías (buscar en el CD la carpeta Manuales, y ver el archivo Manual del Usuario.docx):

- ✓ JDK 1.6 (Java Deveplopment Kit, Kit de desarrollo de Java) o JRE(Java RuntimeEnvironment) suministrado en el CD(carpeta Instaladores/JAVA)
- ✓ JADE 4.1.1(Java AgentDeveplopment Framework) suministrado en el CD(carpeta Instaladores/JADE).

❖ Instalación

Para usar correctamente la aplicación se requiere:

- Elegir la máquina que se usará como Pizarra-Servidor y Ejecutar el archivo RunAgents.bat (Se encuentra ubicado en el CDPrototipo/RepositorioTesis46).
- En las máquinas Clientes elegidas, se debe copiar la carpeta RepositorioTesis46, y modificar el archivo RunGestorCliente.bat(Se encuentra ubicado en el CDPrototipo/RepositorioTesis46) en la línea donde dice –host y seguido el nombre de la máquina que se escogió como Pizarra-Servidor:

```
   java -jardist/PrototypoAgentes.jar -container -gui-host DELL -
   guienriqueGestor:ComponenteGestor.AgenteGestor.AgenteGestor

pause
```

- Después de modificar, se ejecuta el archivo.

Para más detalles buscar en el CD la carpeta Manuales, y ver el archivo Manual del Usuario.docx.

5.4. VALIDACIÓN DEL PROTOTIPO DE AGENTE INTELIGENTE

En las secciones anteriores, se presentaron los aspectos concernientes a: Marco teórico, Modelo Ontológico, Arquitectura del sistema y desarrollo del prototipo; en este apartado se muestra la validación del Prototipo, la cual se realizó mediante la explicación detallada del software en su entorno de ejecución. Se presentan los escenarios relacionados con el Estado actual de la red(registro estático y registro dinámico), que se llevaron a cabo inicialmente a través de la simulación en una máquina, para así culminar con el proceso de validación en el laboratorio del Programa de Ingeniería de Sistemas de la universidad de Cartagena. Dando cumplimiento al quinto objetivo especifico.

Cabe mencionar, que tanto el diseño de la arquitectura como el modelo ontológico propuestos, soportan los cinco aspectos que engloban la gestión de configuración de red: Estado actual de la red, Gestión de inventario, Mantenimiento de directorios, Coordinación del esquema de nombres para nodos y aplicaciones y Control operacional de la red; y que para el caso práctico de la validación de la misma, se seleccionó el escenario Estado actual de la red, por constituirse en el punto de partida para el desarrollo de los demás niveles que implica la gestión de configuración.

5.4.1. Escenario de Prueba No.1: Máquina Virtual

Para llevar a cabo la prueba, se procedió a ejecutar el prototipo en una máquina virtual, escenario seleccionado con el fin de comprobar el correcto funcionamiento del mismo y evidenciar las interacciones ocurridas a nivel de clases software.

Como criterio de aceptación de la prueba, se tomó el caso de uso: Solicitar Registro, el cual se encuentra conformado por dos casos: Requerir Registro Estático y Requerir Registro Dinámico (*Ver Figura 31*), los cuales, a continuación se presenta la descripción detallada de cada uno.

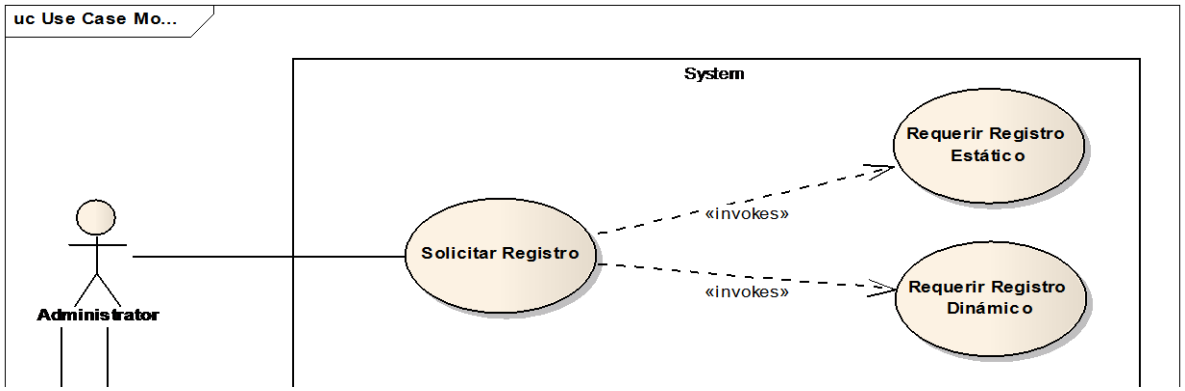

Figura 31. Gráfico que ilustra el escenario de pruebas.

Fuente Los Autores

Caso de Uso 1. Requerir Registro Estático	
Actor Principal	El Administrador
Personal Involucrado e Intereses	El administrador: quiere solicitar información sobre el registro estático de la red
Precondiciones	
Garantías de éxito (Postcondiciones)	Se selecciona la opción que se refiere al registro estático de la red, y se obtiene la información tabulada
Escenario principal de éxito (o flujo básico)	

1. El administrador ingresa a la aplicación GestConfig.Red. El prototipo, muestra la interfaz que se ilustra en la *Figura 32*, donde el Administrador puede realizar la solicitud del estado actual de la red.

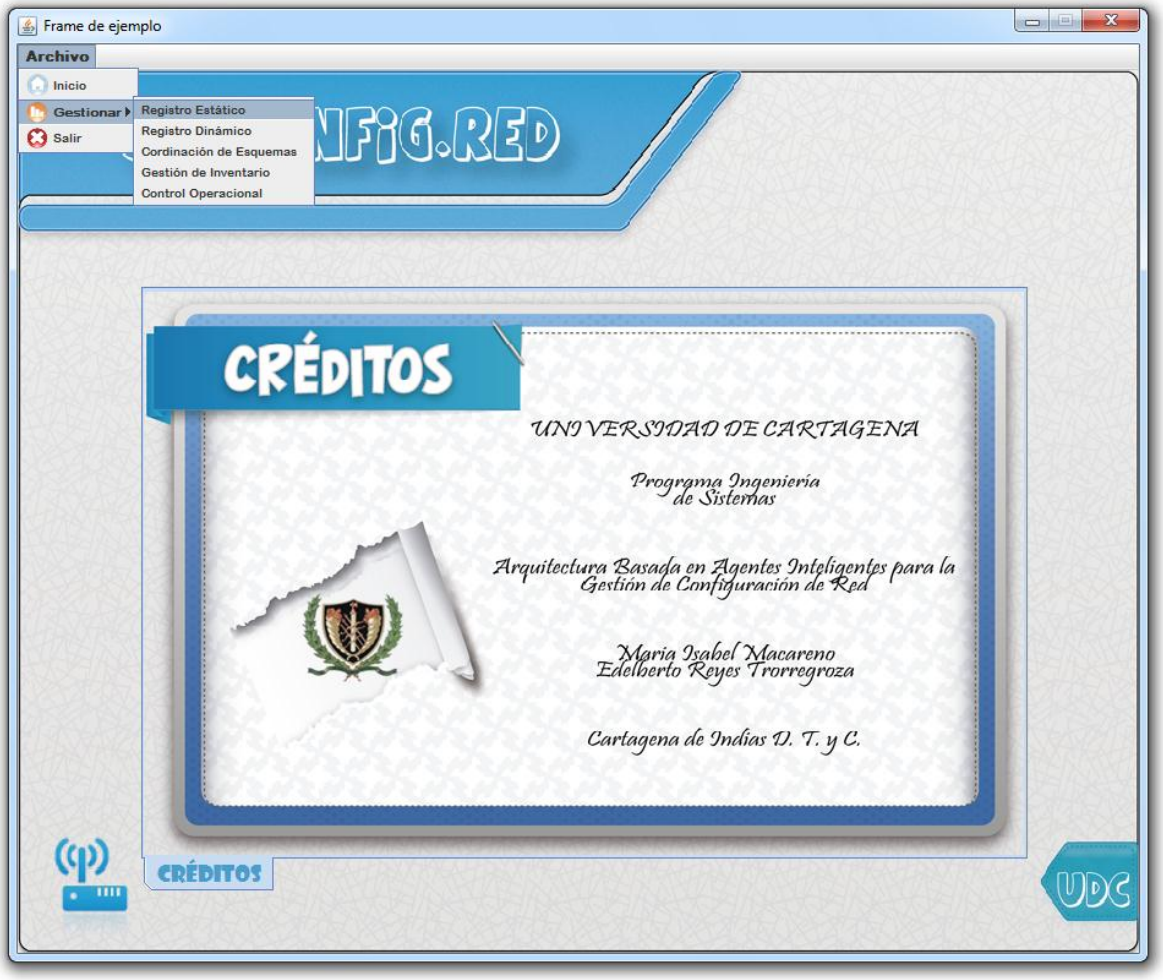

Figura 32. Gráfico que ilustra el prototipo en ejecución.

Fuente Los Autores

2. El sistema le permite ver las opciones disponibles para el Administrador.

a. En el `AgentePizarra`se inicia la `GUIAdministrador`(*Ver Figura 33*).

```
49      @Override
50      protected void setup(){
51          // Create and show the GUI
52          guiAdministrador =new GUIAdministrador(this);
53
54          // guiAdministrador.showGUI();
55          this.guiAdministrador.setVisible(true);
56          proxy=new ProxyAgente(this);
57
58          // Register the wait-info-gestion service in the yellow pages
59          DFAgentDescription dfd = new DFAgentDescription();
60          dfd.setName(getAID());
61          ServiceDescription sd = new ServiceDescription();
62          sd.setType("wait-info-gestion");
63          sd.setName("JADE-wait-info");
64          dfd.addServices(sd);
65
66          try {
67              DFService.register(this, dfd);
68          }catch (FIPAException fe) {
69              fe.printStackTrace();
          }
```

Figura 33. Fragmento de código que evidencia como el AgentePizarra inicia la GUIAdministrador.

Fuente Los Autores.

3. El administrador señala la opción concerniente al registro estático de la red.

4. El sistema reconoce la petición solicitada por el administrador.

a) En el `AgentePizarra`, se inicia el `AgenteMensajero`, pasando como parámetro la consulta hecha por el Administrador (*Ver Figura 34*).

```
257     public boolean sendPeticion(String mensaje){
258         // javax.swing.JOptionPane.showMessageDialog(null, "esto fue lo q mandaste "+mensaje);
259
260         this.mensaje=mensaje;
261         boolean pet=ReceptorPeticion.createInstance(mensaje, this).getTipoPeticion();
262         System.out.println("procesando peticion......");
263
264         if(pet==false){
265             javax.swing.JOptionPane.showMessageDialog(null, "PEticion invalida");
266             return false;
267         }
268         return true;
269     }
```

Figura 34. Fragmento de código que evidencia como el AgentePizarra comprueba la validez de la petición antes de iniciar elAgenteMensajero.

Fuente Los Autores

b) El receptor de peticiones del `AgenteMensajero`, comprueba la validez de la consulta, recorriendo la ontología (*Ver Figura 35* y *Figura 36*).

```
49
50      public boolean actionCommand(String mensaje) throws StaleProxyException{
51
52          Agregate receptor=new Receptor();
53
54          if(receptor.isCommandOnOntology(mensaje)){
55              ProxyAgentMensajero proxy = new ProxyAgentMensajero(agent);
56              proxy.startAgenteMovilMensajero(mensaje);
57              return true;
58          }
59          return false;
60
61      }
62
63      /**
64       * @return the resultado
65       */
66      public ResultadoAbstracto getResultado() {
67          return resultado;
68      }
69
```

Figura 35. Fragmento de código que valida consultas del AgenteMensajero

Fuente. Los Autores

```
58          * aqui verificamos si la peticion es valida, buscando en la ontology
59          */
            @Override
61          public boolean isCommandOnOntology(String peticion) {
62              String aux=this.searchClass(peticion);
63              System.out.println("------RECEPTOR DE PETICIONES-------- ");
64              System.out.println("se evalua "+peticion+" y tiene "+aux);
65              if(aux!=null)
66                  return true;
67              return false;
68
69              //AQUI DEBEMOS LEER LA ONTOLOGY Y HACER UNA BUSQUEDA TERMINIO A ATERMINO
70          }
71
            @Override
73          public String searchClass(String clase) {
74              for(int i=0;i<this.listaObjetos.size();i++){
75                  if(clase.equalsIgnoreCase(this.listaObjetos.get(i).toString())){
76                      return clase;
77                  }
78              }
79              return null;
80          }
81
```

Figura 36. Fragmento de código que ilustra como la petición solicitada es comparada con la ontología, para así retornar su validez.

Fuente Los Autores.

c) Al comprobar que la petición es válida, el `AgenteMensajero` le pasa como parámetro la consulta a las instancias de `AgenteGestor` iniciadas en los equipos que van a ser gestionados (*Ver Figura 37* y *Figura 38*).

```
49
                @Override
51              protected void onTick() {
52
53                  DFAgentDescription template = new DFAgentDescription();
54                  ServiceDescription sd = new ServiceDescription();
55                  sd.setType("gestion_configuracion");
56                  template.addServices(sd);
57
58                  try {
59                      //aqui se BUSCAN LOS AGENTES GESTORES CONECTADOS
60                      DFAgentDescription[] result = DFService.search(myAgent, template);
61                      System.out.println("Found the following gestion agents:");
62                      agentesGestores = new AID[result.length];
63                      respuestaGestion=new ArrayList<ArrayList>();
64
65                      for (int i = 0; i < result.length; ++i) {
66                          agentesGestores[i] = result[i].getName();
67                          System.out.println(agentesGestores[i].getName());
68                      }
69                  } catch (FIPAException fe) {
70                      fe.printStackTrace();
71                  }
72                  //add behaviour que PERMITE SOLICITAR LA INFO A CADA GESTOR INSTALADO
73                  myAgent.addBehaviour(new SolicitarInformacionGestion());
74              }
```

Figura 37. Fragmento de código que evidencia la búsqueda de los AgenteGestor

Fuente. Los Autores

```
93
94              case 0:
95                  // Send the cfp to all agents
96                  ACLMessage cfp = new ACLMessage(ACLMessage.CFP);
97                  for (int i = 0; i < agentesGestores.length; ++i) {
98                      cfp.addReceiver(agentesGestores[i]);
99                  }
100                 cfp.setContent(mensaje);
101                 cfp.setConversationId("gestion_configuracion");
102                 cfp.setReplyWith("cfp" + System.currentTimeMillis()); // Unique value
103                 System.out.println(" enviamos el mensaje a los gestores");
104                 myAgent.send(cfp);
105                 // Prepare the template to get proposals
106                 mt = MessageTemplate.and(MessageTemplate.MatchConversationId("gestion_configuracion"),
107                         MessageTemplate.MatchInReplyTo(cfp.getReplyWith()));
108                 step = 1;
109                 break;
110
```

Figura 38. Fragmento de código que evidencia el envío de mensajes alosAgenteGestor.

Fuente Los Autores.

d) El `AgenteGestor` usa el intérprete de peticiones, comprueba la validez de la consulta recibida del `AgenteMensajero`, utilizando la ontología (*Ver Figura 39*).

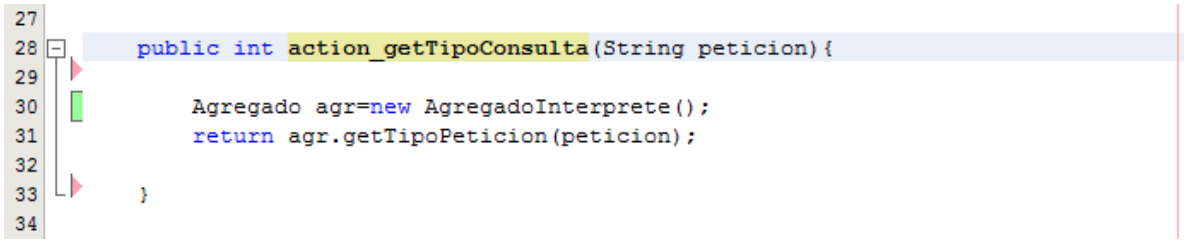

Figura 39. Fragmento de código que evidencia como el AgenteGestor conoce el tipo de consulta solicitada por el AgenteMensajero.

Fuente Los Autores.

e) El `AgenteGestor`, inicia la tarea asignada, obtener el registro estático de la red.

5. El sistema genera una tabla que se ilustra en la *Figura 40*, donde se muestran los resultados arrojados por la consulta.

Figura 40. Gráfico que ilustra el Registro Estático solicitado por el Administrador.

Fuente Los Autores

Extensiones (o Flujos alternativos)	
a. El administrador ya se encuentra registrado: ✓ La cuenta ya fue concedida previamente	
Requisitos especiales	Interfaz de Administrador con excelente diseño gráfico
Lista de Tecnología y variaciones de datos	Interfaz gráfica que permita desplegar el formato de la información en forma clara y precisa
Frecuencia	Cada vez que un administrador solicite información sobre el registro estático al Sistema

Caso de Uso 2. Requerir Registro Dinámico	
Descripción	Luego, el administrador puede solicitar el registro Dinámico de la red gestionada. Seguidamente aparece una tabla, donde se muestran los resultados arrojados por la consulta.
Actor Principal	El Administrador
Personal Involucrado e Intereses	El administrador: quiere solicitar información sobre el registro dinámico de la red
Precondiciones	
Garantías de éxito (Postcondiciones)	Se selecciona la opción que se refiere al registro dinámico de la red, y se obtiene la información tabulada
Escenario principal de éxito (o flujo básico)	

1.El administrador ingresa a la aplicación GestConfig.Red, opción Registro Dinámico a través del menú archivo, submenú Gestionar. El prototipo, muestra la interfaz donde el Administrador puede realizar la solicitud del estado actual de la red.

2. El sistema le permite ver las opciones disponibles para el Administrador.

a) En el AgentePizarra se inicia la GUIAdministrador(*Ver Figura 33).*

3. El administrador señala la opción concerniente al registro dinámico de la red.

4. El sistema reconoce la petición solicitada por el administrador.

a) En el AgentePizarra, se inicia el AgenteMensajero(*Ver Figura 34*), pasando como parámetro la consulta hecha por el Administrador.
b) El receptor de peticiones del AgenteMensajero, comprueba la validez de la consulta, recorriendo la ontología (*Ver Figura 35 y Figura 36*).
c) Al comprobar que la petición es válida, el AgenteMensajero le pasa como parámetro la consulta a las instancias de AgenteGestor iniciadas en los equipos que van a ser gestionados (*Ver Figura 37 y Figura 38*).
d) El AgenteGestor usa el interprete de peticiones, comprueba la validez de la consulta recibida del AgenteMensajero, utilizando la ontología (*Ver Figura 39*).
e) El AgenteGestor, inicia la tarea asignada, obtener el registro dinámico de la red.

5. El sistema genera una tabla que se ilustra en la *Figura 41*, donde se muestran los resultados arrojados por la consulta.

Figura 41. Gráfico que ilustra el Registro Dinámico solicitado por el Administrador.

Fuente Los Autores

Extensiones (o Flujos alternativos)	
a. El administrador ya se encuentra registrado: ✓ La cuenta ya fue concedida previamente	
Requisitos especiales	Interfaz de Administrador con excelente diseño gráfico
Lista de Tecnología y variaciones de datos	Interfaz gráfica que permita desplegar el formato de la información en forma clara y precisa
Frecuencia	Cada vez que un administrador solicite información sobre el registro estático al Sistema

5.4.2. Escenario de Prueba No.2: Laboratorio de Investigación

Culminada la fase de simulación anterior, se procede a realizar la prueba de la arquitectura diseñada mediante la ejecución del prototipo en los laboratorios de investigación del programa de Ingeniería de Sistemas de la Universidad de Cartagena, escenario seleccionado por tratarse de un ambiente de red carente de un control a nivel de gestión de configuración de los nodos allí establecidos.

La prueba tuvo como propósito obtener el registro estático y dinámico de los nodos que conforman la red del laboratorio de investigación, cuya infraestructura se constituye por: dos (2) computadores clientes en el Semillero de Investigaciones (INV01, INV02) y un (1) computador servidor (INV03). Al finalizar el proceso, se obtuvo una tabla que consignó la información, en primera instancia, referente al registro estático: nombre del host, IP, mac, netmask, gateway, Versión SO, nombre SO, y arquitectura del sistema; y en segunda instancia, los datos correspondientes al registro dinámico: memoria total, memoria usada y memoria libre del equipo gestionado.

A continuación se mencionará paso a paso, el proceso de prueba realizado en el laboratorio de semilleros de investigación.

- **Detalles de la Prueba en el Laboratorio de Investigación**

Inicialmente, se procedió a copiar la carpeta RepositorioTesis46 (ubicada en el CD en la carpeta Prototipo) en las máquinas clientes (INV01 y INV02) y en la Servidora (INV03); teniendo en cuenta que las máquinas tuvieran JAVA incluido en la variable de entorno del sistema (PATH), para ejecutar en forma correcta la aplicación.

Seguidamente, se obtuvo el nombre de host de la máquina Servidora, para el caso particular la máquina elegida fue INV03.

De esta forma, se continuó con la modificación del archivo RunClienteGestor.bat (ubicado en la carpeta Prototipo/RepositorioTesis46 en el CD de instalación) en cada

máquina cliente. Para ello, en la línea donde aparece el NOMBRE_HOST se remplazó por INV03, incluyendo además, el nombre del agente teniendo en cuenta que el nombramiento de los mismos debe ser diferente entre si (entre clientes); tal como se muestra a continuación.

```
java -jar dist/PrototypoAgentes.jar -container -gui -host  INV03
-guiNOMBRE_AGENTE:ComponenteGestor.AgenteGestor.AgenteGestor
```

Por otra parte, se ejecutó el archivo RunAgents.bat (ubicado en la carpeta Prototipo/RepositorioTesis46 del CD de instalación) en la máquina servidora, mostrando la interfaz que se aprecia en la Figura 42.

Figura 42. Prototipo en ejecución en la máquina servidora.

Fuente Los Autores

Realizado lo anterior, se ejecuta el archivo RunClienteGestor.bat en cada máquina elegida como clientes, apareciendo en pantalla la ventana mostrada en la *Figura 43*.

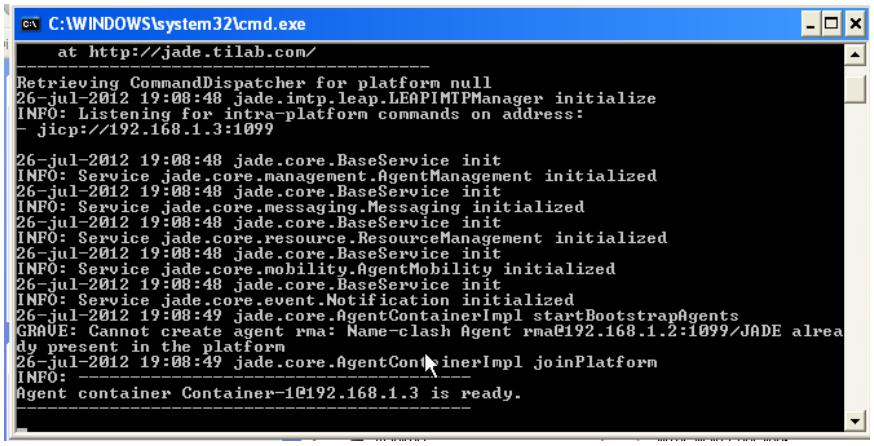

Figura 43. Ilustra el AgenteGestor ejecutándose en la máquina cliente.

Fuente Los Autores

Es preciso mencionar que para el caso particular, las computadoras clientes escogidas son: INV01 (*ver Figura 44*) y INV02 (*ver Figura 45*).

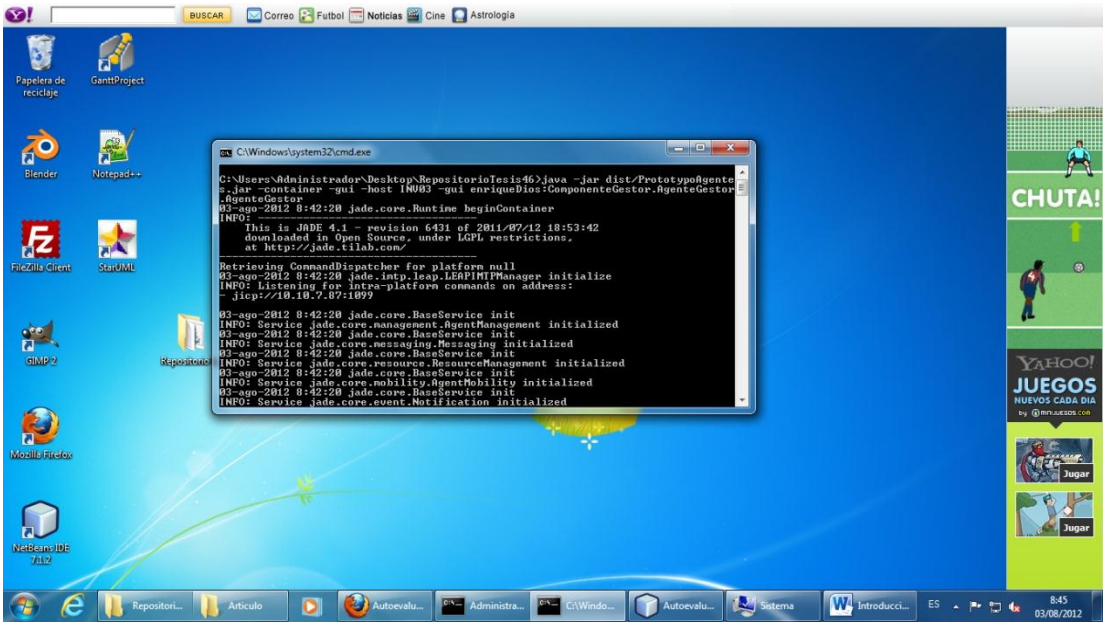

Figura 44. Ilustra el AgenteGestor ejecutándose en la máquina INV01.

Fuente los Autores

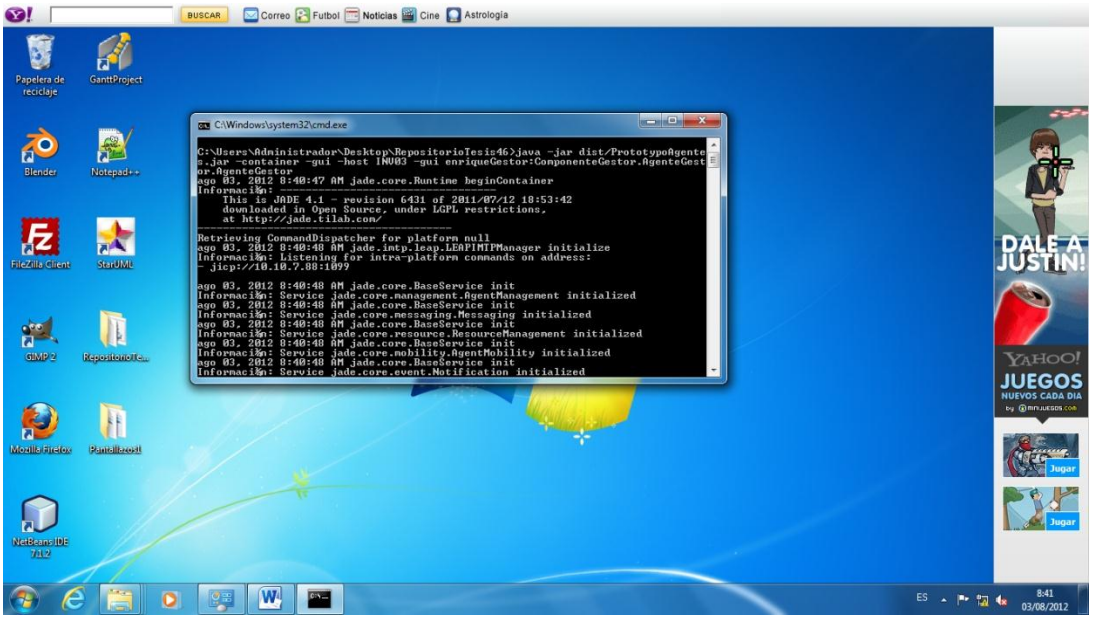

Figura 45. Ilustra el AgenteGestor ejecutándose en la máquina INV02.

Fuente los autores

En este punto, se verificó que las máquinas clientes estuviesen sincronizadas con el Servidor (INV03). Comprobado esto, se procedió a hacer peticiones de Registro estático y Dinámico (*ver Figura 46*).

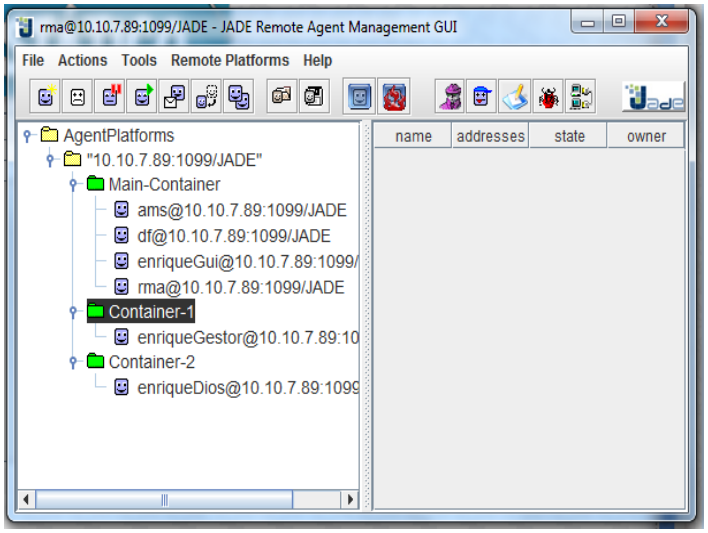

Figura 46. Ilustra la plataforma con los agentes gestores sincronizados.

Fuentes los autores

Al finalizar el proceso, el resultado de la petición de registro estático se muestra en la *Figura* 47, mientras que el resultado del Registro Dinámico se muestra en la *Figura 48.*

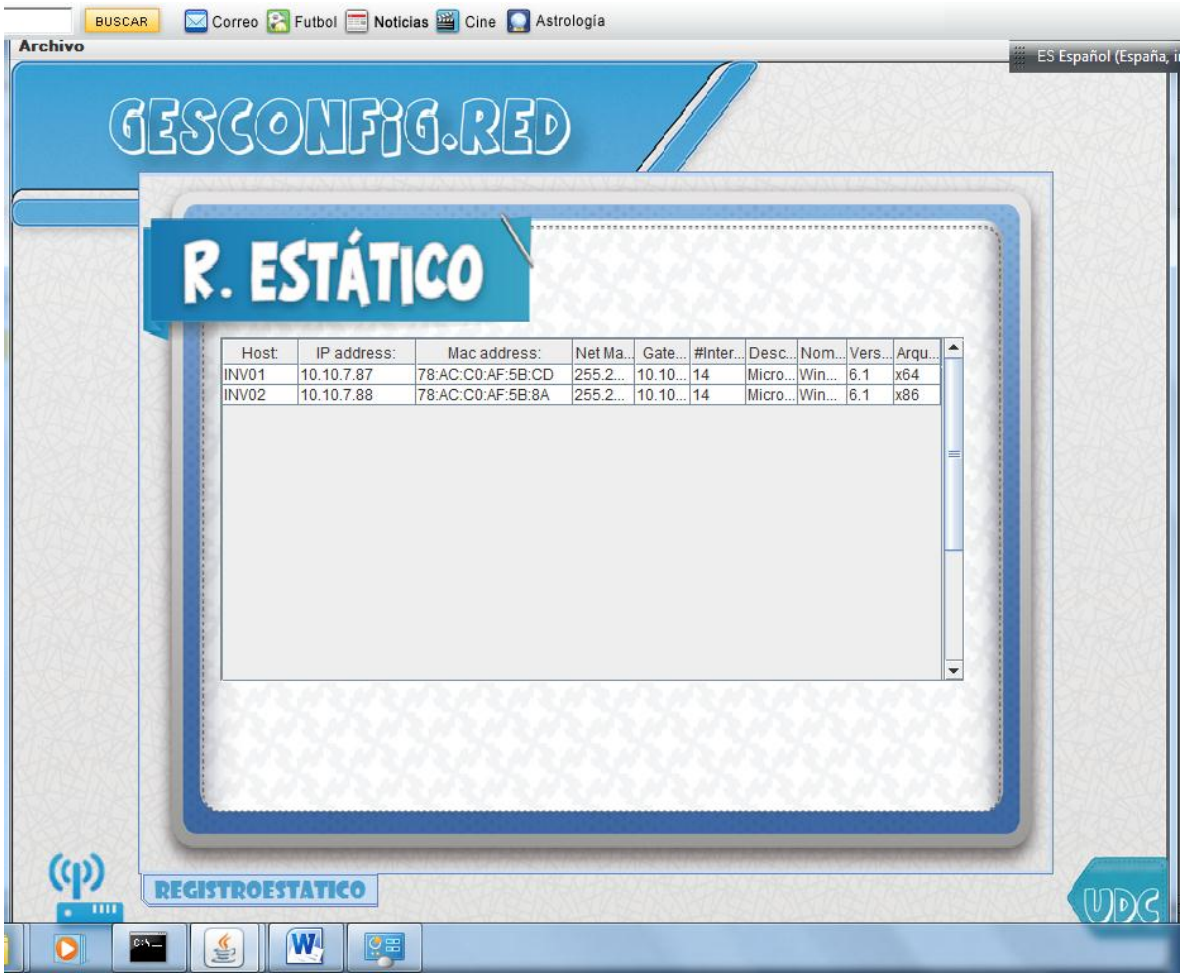

Figura 47. Ilustra el resultado de la petición de Registro Estático para las máquinas clientes.

Fuente los autores.

Figura 48. Ilustra el resultado de la petición de Registro Dinámico para las máquinas clientes.

Fuente los autores

De esta forma, se concluyó el proceso de prueba en el laboratorio de investigaciones de la Universidad de Cartagena; para un mayor detalle de la misma, se recomienda observar el video de la prueba (ubicado la carpeta Videos del CD).

7. CONCLUSIONES Y RECOMENDACIONES

La labor realizada en el presente trabajo de grado, así como sus principales aportes y conclusiones de acuerdo a los objetivos planteados, se pueden resumir en los siguientes puntos:

En primer lugar, las revisiones de investigaciones previas basadas en la existencia de modelos ontológicos para la gestión de red, arquitecturas de agentes inteligentes y tecnologías de gestión de red integrada; evidencian un crecimiento desacoplado entre estos campos de conocimiento. Por una parte se enuncian el uso de las ontologías como medio para la integración semántica de los distintos dominios de gestión existentes, y por otra, se hace énfasis en el uso de sistemas multiagentes como medio para facilitar la gestión de algún aspecto concerniente a las redes; lo que implica, que no se está haciendo uso de todas las potencialidades generadas a partir de la integración de los agentes y las ontologías.

En segunda instancia, la aplicación de la técnica de representación del conocimiento llamada Ontología, ha mejorado la definición de la información de Gestión de configuración de red, ya que ha permitido extraer los aspectos más relevantes del dominio. Así mismo, codificar la información de gestión con un lenguaje de ontologías como OWL mejora la expresividad de dicha información, debido a que brinda muchas facilidades para el manejo de relaciones entre las clases que conforman el dominio estudiado, donde además, el uso de la herramienta Protégé aporta un gran valor al momento del desarrollo.

Por otra parte, aunque no existe una regla general para el desarrollo de ontologías, tomar como base uno o varios de esos métodos agilizan el proceso y permiten que el producto final sea reusable. Sin embargo, durante la experimentación se ha constatado que su aplicación a modelos de información de gran tamaño puede llevar bastante tiempo

debido a la necesidad de intervención humana para comprobar la validez de las reglas. Al aprovechar las características de OWL, la posibilidad de definir reglas se obtiene de forma inmediata; lo cual trate como consecuencia que el modelo obtenido sea independiente del dominio de gestión usado.

En tercer lugar, el uso del modelo de vistas 4+1, para el diseño de la arquitectura planteada en el presente trabajo, brindó un aporte significativo, puesto que la subdivisión en varias vistas, separó cada uno de los aspectos relevantes en el proceso de diseño arquitectónico; permitiendo a los distintos stakeholders la remisión a aspectos puntuales de la arquitectura, por ejemplo, los ingenieros de sistemas se enfocarán en la vista física y de procesos; los usuarios finales, los clientes y los especialistas en datos en vista lógica; y los administradores de proyectos se fijarán en la vista de desarrollo.

Lo anterior, proporcionó como resultado una arquitectura robusta, flexible y escalable a los agentes inteligentes implementados, que comparada con los estándares para el desarrollo de agentes estudiados, corrobora sus similitudes con las arquitecturas híbridas. Evidenciando a su vez, una comunicación entre agentes que facilita la reutilización de conocimiento y memoria, debido al uso de la arquitectura de pizarra, tornando el proceso de gestión de configuración mas eficiente.

En cuarto lugar, la validación de la arquitectura software que soporta el proceso de gestión de configuración de red, otorgó como resultado el desarrollo de un prototipo; que en su primera instancia, empleando el paradigma de programación orientada a objetos aplicada al lenguaje de programación JAVA, propició flexibilidad en cuanto a la codificación y manejo de los objetos. Y que en segunda instancia, utilizó el Framework JADE para construir los agentes, debido a la facilidad de instalación y uso, eficiencia y eficacia en el manejo de los recursos informáticos, permitiendo incluso que el prototipo pueda ser extendido a dispositivos como PDA’s y smartphones.

Finalmente, realizar las pruebas al prototipo desarrollado comprobó que: los agentes son funcionales y cumplen con los parámetros establecidos en la arquitectura; la información el resultado obtenido, es el esperado; y que la ontología usada es funcional, pues permite que los agentes lleven a cabo sus tareas con base en ella.

BIBLIOGRAFÍA

BARBA, A., & HESSELBACH, X. (2002). *Inteligenica de Red.* UPC.

BOTTI, J. (s.f.). *Asociación de Técnicos de Informática.* Recuperado el 5 de Octubre de 2010, de Agentes Inteligentes: El siguiente paso en la Inteligencia Artificial: http://www.ati.es/novatica/2000/145/vjulia-145.pdf

CALLE, G. (1997). *Reingeniería y Seguridad en el Ciberespacio.* España: Diaz de Santos.

Clements, P. (1996). A Survey of Architecture Description Languages. En P. Clements. Alemania.

GRUBER, T. (1995). Towards principles for the design of ontologies used for knowledge sharing.

GUERRERO CASTELEIRO, A. (2007). *Biblioteca Universitaria Politécnica, Archivo Digital UPM.* Recuperado el 1 de Agosto de 2010, de Especificación del comportamiento de gestión de red mediantes ontologías: http://oa.upm.es/909/1/ANTONIO_GUERRERO_CASTELEIRO.pdf

Guerrero Casteleiro, A. (2007). Especificación del comportamiento de gestión de red mediantes ontologías. Madrid, España.

Krichten, P. (s.f.). *Planos Arquitectónicos: El Modelo de "4+1" Vistas de la Arquitectura del Software*. Recuperado el 21 de Febrero de 2012, de http://www.oscargallardo.com/wp-content/uploads/2011/04/Modelo4_1Krutchen.pdf

LÓPEZ DE VERGARA MÉNDEZ, J. E. (2003). *Scientific Literature Digital Library and Search Engine.*Recuperado el 5 de Agosto de 2010, de Especificación de Modelos de información de gestión de red integrada mediante el uso de ontologías y técnicas de representación del conocimiento: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.77.8694&rep=rep1&type=pdf

MARTIN, M. D. (Septiembre de 2004). *Sistema de catalogación de métricas e indicadores con potencia de web semántica.* Recuperado el 20 de Septiembre de 2010, de Sistema de catalogación de métricas e indicadores con potencia de web semántica: http://gidis.ing.unlpam.edu.ar/home/downloads/Tesis-Martin.pdf

NOY, N., & McGUINNESS, D. (19 de Septiembre de 2005). *Desarollo de Ontologías 101: Guía para crear tu primera ontología.* Recuperado el 2 de Septiembre de 2010, de Desarollo de Ontologías 101: Guía para crear tu primera ontología: http://protege.stanford.edu/publications/ontology_development/ontology101-es.pdf

Pressman, R. (2002). Ingenieria de software: Un enfoque práctico. En R. Pressman, *Ingenieria de software: Un enfoque práctico.* (pág. 265). Madrid, España: Mc Graw Hill.

Protegé. (05 de Abril de 2012). *OWL Ontologies.* Recuperado el 15 de Marzo de 2012, de Protegé. OWL Ontologies: http://protegewiki.stanford.edu/images/7/7c/DomainOntology-NMS.zip

ROA, O. L. (s.f.). *Agentes de Software: Tecnologías, herramientas y aplicaciones.* Recuperado el 1 de Agosto de 2010, de Agentes de Software: Tecnologías, herramientas y aplicaciones: http://dialnet.unirioja.es/servlet/articulo?codigo=2877339

ROSERO, O., & PROAÑO, D. (s.f.). *Biblioteca de Ingeniería Eléctrica y Electrónica.* Recuperado el 18 de Septiembre de 2010, de Estudio y Desarrollo de una metodología para la implementación de un modelo de gestión y administración de red: http://bieec.epn.edu.ec:8180/dspace/bitstream/123456789/1384/5/T%2011276%20CAPITULO%201.pdf

RUSSELL, S. (1996). *Inteligencia Artifical: Un enfoque moderno.* México: Prentice Hall.

SALAZAR SERRUDO, C. (s.f.). *Universidad Mayor de San Simón, Facultad de Ciencias y Tecnología.* Recuperado el 18 de Septiembre de 2010, de Agentes y Multiagentes Inteligentes: conceptos, arquitecturas y aplicaciones: http://www.fcyt.umss.edu.bo/docentes/269/publicacion/LibroIA.pdf

Standards Committe Telecommunications. (2001). Telecom Glosary 2000.

STUDER, R., BENJAMINS, R., & FENSEL, D. (1998). Knowledge Engineering: Principles and Methods.

W3C. (s.f.). Recuperado el 5 de Septiembre de 2010, de Guía breve de web semántica: http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica

WOOLDRIDGE, M., & JENNINGS, N. (1995). Intelligent Agents: Theory and Practice.

ANEXOS

ANEXO A. GLOSARIO DE TÉRMINOS DE LA ONTOLOGÍA

***ARP**:* Son las siglas en inglés de Address Resolution Protocol (Protocolo de resolución de direcciones).Es un protocolo de la capa de enlace de datos responsable de encontrar la dirección hardware (Ethernet MAC) que corresponde a una determinada dirección IP.

***Concentrador**:* Es un elemento de hardware que permite concentrar el tráfico de red que proviene de múltiples hosts y regenerar la señal.

***Conmutador**:* Dispositivo de interconexión de redes que opera en la capa de enlace de datos del modelo OSI. Su función es interconectar dos o más segmentos de red, de manera similar a los puentes de red, pasando datos de un segmento a otro de acuerdo con la dirección MAC de destino de las tramas en la red.

Control operacional de la Red: Permite iniciar y detener componentes individuales, alterar la configuración de los dispositivos, cargar y configurar versiones de dispositivos, entre otras.

Coordinación de esquemas: Permite organizar los esquemas de nombres y aplicaciones de manera que dicha información sea fácil de encontrar.

***Dirección IP**:* Etiqueta numérica que identifica de manera lógica y jerárquica, a un interfaz (elemento de comunicación/conexión) de un dispositivo dentro de una red que utilice el protocolo IP (*Internet Protocol*).

***Directorio**:* Aplicación o un conjunto de aplicaciones, que almacena y organiza la información concerniente a los usuarios de una red de ordenadores y los recursos de la misma, permitiendo a los administradores gestionar el acceso de usuarios a los recursos la red.

***DNS**:* Domain Name System (en español: sistema de nombres de dominio) es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada.

Enrutador: Es un dispositivo de hardware usado para la interconexión de redes informáticas que permite asegurar el direccionamiento de paquetes de datos entre ellas o determinar la mejor ruta que deben tomar.

Esquema de nombres: Es la representación de nombres que proporciona la estructura jerárquica para la base de dates DNS. Cada nodo representa un partición de la base de datos DNS. A estos nodos se les denomina dominios.

Estado actual de la Red: Permite obtener el registro de la topología, se encuentra conformado por Registro estático y dinámico.

FTP: Es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (Transmission Control Protocol), basado en la arquitectura cliente-servidor.

Gestión de Configuración: Es un área funcional de la gestión de red que tiene por objeto proveer funciones para operar, controlar, identificar y recopilar datos de una red.

Gestión de inventario: Es la función encargada de hacer una base de datos de los elementos de la red, incluyendo además el historial de cambios y problemas.

Host: Computador que funciona como punto de inicio y fin de las transferencias de datos.

Inventario: Base de datos que contiene todos los elementos de red, y el historial de cambios y problemas.

MAC Address: Identificador de 48 bits (6 bloques hexadecimales) que corresponde de forma única a una tarjeta o dispositivo de red.

Mantenimiento de directorios: Brinda toda la información relacionada con todos los nombres de dominio de la red, es decir, la base de datos de los nombres de dominio.

MIB: Base de Información de Administración (Management Information Base). Es una colección de información que está organizada jerárquicamente.

Nodo: Elemento de red, por ejemplo: Repetidores, Puente, concentrador, enrutador, entre otros.

NMS: Conocido como sistema administrador de red (en ingles *Network Management System*), se encarga de ejecutar aplicaciones que supervisan y controlan a los dispositivos administrados. Los NMS's proporcionan el volumen de recursos de procesamiento y memoria requeridos para la administración de la red.

Puente: Dispositivo de interconexión de redes de ordenadores que opera en la capa 2 (nivel de enlace de datos) del modelo OSI. Este interconecta segmentos de red (o divide una red en segmentos) haciendo la transferencia de datos de una red hacia otra con base en la dirección física de destino de cada paquete.

Puerta de enlace: Hardware y software que permite las comunicaciones entre la red local y grandes computadoras (mainframes).

Red: También conocida como red de computadores, es un conjunto de computadoras conectadas entre si, con la finalidad de compartir información, recursos y ofrecer servicios.

Registro Dinámico: Brinda información acerca del estado operacional de la red y el estado de conexión de cada dispositivo.

Registro Estático: Brinda información acerca de aquellos aspectos de la red que no cambian. Por ejemplo, los nodos están instalados y el numero de interfaces de un router.

Repetidor: Dispositivo electrónico que recibe una señal débil o de bajo nivel y la retransmite a una potencia o nivel más alto, de tal modo que se puedan cubrir distancias más largas sin degradación o con una degradación tolerable.

Resolvedor: Procedimiento de biblioteca que relaciona un nombre con una dirección IP.

SMTP: Protocolo Simple de Transferencia de Correo (en ingles, Simple Mail Transfer Protocol). Es un protocolo de la capa de aplicación, basado en textos utilizados para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (PDA's, teléfonos móviles, etc.)

SNMP: Protocolo simple de gestión de red (en ingles Simple Network Management Protocol). Es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red.

Telnet:Es un protocolo de red a otra máquina, que permite manejarla remotamente como si se estuviese sentado frente a ella.

Topología: Se define como la cadena de comunicación usada por los nodos que conforman

Workstation: Son los computadores que tienen a su disposición los recursos que ofrece la red así como los servicios que proporcionan los Servidores a los cuales pueden acceder a una red para comunicarse.