UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMÁTICA

"SISTEMA DE GESTIÓN DE HOSTS BASADA EN AGENTES CON SOPORTE SNMP"

TESIS DE GRADO

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA

MENCIÓN: INGENIERÍA DE SISTEMAS INFORMÁTICOS

POSTULANTE:	Univ. Jorge Camilo Villca Limachi
TUTOR:	Lic. Germán Huanca Ticona
REVISOR:	Lic. José María Tapia Baltazar

La Paz – Bolivia

2011

Dedicatoria

A mi amada esposa y a mí querida hija quien me
enseño con sus pequeños pasos que levantarse cuesta más,
pero siempre hay una sonrisa una vez de pie.

A mis queridos padres y abuelitos quienes me
enseñaron a recorrer mi destino con amor.

A la vida, pues en ella disfruto cada día
la oportunidad de ser feliz al lado de mis seres amados.

Agradecimientos

A Dios por darme la inspiración que necesita mi vida y por todas sus bendiciones.

A los docentes de la carrera de Informática que con sus enseñanzas alumbraron mi camino hacia el conocimiento, en especial al Lic. José María Tapia por toda la confianza depositada y su apoyo incondicional al desarrollo de la presente tesis, al Lic. Germán Huanca que como tutor supo guiar mis conocimientos.

A mis queridos padres por todo el apoyo brindado en lo largo de mis estudios, a mi querida hermana que fue, es y será siempre mi mejor ejemplo y mejor amiga.

A mis abuelos que siempre apoyaron mis estudios con sus muestras de cariño y sus recomendaciones.

A mi querida esposa y mi beba adorada por su compañía y comprensión.

A todos mis amigos por el tiempo dedicado y todo el apoyo brindado

Resumen

Con el transcurso de los años, la evolución y el incremento del uso de la tecnología han influido en el crecimiento de las redes de telecomunicaciones y su uso en diversas situaciones, tal es el caso de las redes LAN, MAN y WAN que en estos días su implementación para oficinas, edificios y ciudades es muy común.

A partir de este crecimiento es que nace una duda en los administradores de red, de cómo tener información a la mano del estado de los host encontrados dentro de una red ya sea LAN o WAN, una tarea más a la administración de la red, que se acrecienta con el número de usuarios sumados y que por algún descuido de la PC de uno de ellos puede incurrir en la caída de un servicio de red (web, correo, etc.).

Viendo de esta manera que las redes son de una importancia crítica y creciente, una red no se puede instalar y gestionar sólo con el esfuerzo humano y en respuesta a ello se trata la gestión de redes que cubre los servicios, protocolo y la base de información de gestión mediante un agente.

En 1988 nace el protocolo SNMP (Simple Network Managment Protocol), protocolo sencillo de administración de redes, dicho protocolo es una llave de conexión entre un host-cliente y el servidor configurado, dando la oportunidad a un administrador de red de consultar de forma remota mediante consola una a una, el estado del hardware y uso de puertos de cada host mediante comandos, llevando a una tarea 253 consultas por día, trabajo que resulta tedioso y en muchos casos vano pues los problemas no se presentan a diario.

Planteando como solución a este problema se pretende la programación de un agente que entienda idioma SNMP bajo una interfaz web que actúe como sistema, para la gestión autónoma de hosts dentro de una red LAN y de este modo brindar información precisa al administrador de red.

Abstract

Over the years, the evolution and increased use of technology have influenced the growth of telecommunications networks and their use in various situations, such is the case of LAN, MAN and WAN these days implementation for offices, buildings and cities is very common.

From this growth is that a question arises in the network administrators, how to have information of the state of the host found within a network either LAN or WAN, a task in the administration of the network, which increases with the number of users added and that by some oversight of the PC of one of them may incur in the fall of a network service (web, mail, etc.).

Viewing in this way that networks are a critical and growing importance, a network can not only install and manage human effort and in response to this is managing networks that covers services, protocol and the information base management through an agent.

Born in 1988 SNMP protocol (Simple Network Managment Protocol), simple network management protocol that is a key connection between a host-client and the server configured, providing an opportunity for a network administrator to remotely view using console one at a hardware status and use of ports for each host through commands, leading to 253 queries a day job, work that is tedious and often futile because the problems do not occur daily.

Posing as a solution to this problem is to an agent programming language that understands SNMP on a Web interface that acts as a system for autonomous management of hosts within the LAN and thus provide accurate information to the network administrator.

ÍNDICE

Capítulo 1

Introducción

SISTEMA DE GESTIÓN DE HOSTS BASADA EN AGENTES CON SOPORTE SNMP

1.1 INTRODUCCIÓN

En la complejidad de las redes de hoy, donde se puede encontrar un sin fin de sistemas, acompañados de componentes de red cada ves más puestos a la supervisión de un administrador y debido al crecimiento mismo del uso de redes dentro de un ambiente, ya sea este pequeño o para la cobertura de muchos predios ubicados en lugares alejados geográficamente, es que nace el problema del monitoreo de redes y lo que implica la supervisión de todos los componentes que forman parte de una red, sean estos servidores, routers, swtich's, hosts[1], etc.

La realidad de hoy es un poco compleja para los administradores de una red, hasta para un subgrupo que puede trabajar con ellos, que es el área de soporte técnico, debido a la magnitud de componentes que deben ser administrados y/o supervisados, el personal con que se cuenta y la organización que se posee para la atención a posibles problemas, solucionando cualquier falla solo a petición de un usuario y no pudiendo realizar un seguimiento en línea o gestión por lo menos para la parte de host's.

Es así que en la necesidad de una solución nace el Protocolo SNMP (*Simple Network Management Protocol*), que fue una idea que introducida desde 1988 para satisfacer la necesidad creciente de una norma para la administración de dispositivos mediante el Protocolo de Internet (IP) y que fue madurando pasando por las versiones SNMP v1, SNMP v2 y la versión más reciente SNMP v3 que soluciona problemas de seguridad que tuvieron sus antecesores, dicho protocolo que posee una estructura ya establecida es una herramienta muy poco explotada en el ámbito de la administración de redes por lo menos en nuestro país.

[1] Host, dícese de una máquina conectada a una red de ordenadores y que tiene un nombre de equipo (en inglés, *hostname*).

Es conjuntamente con esta idea de administración de dispositivos que se rescata la necesidad de hacer un monitoreo general desatendido de los mismos, encontrándose para tal cometido un paradigma ya con años de madurez que es el concepto de Agentes; concepto que brinda todo lo necesario para optimizar la tarea que para este tema se ha atribuido, que es el monitoreo de hosts, rescatando las propiedades de:

- Aprender y actuar emulando el comportamiento del usuario.
- Ejecutar tareas rutinarias (consulta a una base de datos, búsqueda de información, etc.), de manera autónoma.
- Tener la capacidad de identificar qué actividad es más relevante que otro.

que poseen los Agentes, para reunirlos en un solo sistema.

Es así que para esta tesis se hace la fusión de dos áreas muy importantes y poderosas que tiene la informática que son las redes y la programación, combinadas para perseguir un solo fin que es el de anticiparse, adaptarse y buscar activamente maneras de dar mejor soporte al usuario.

Soporte que se realizara mediante un Sistema basado en agentes que sirva para el monitoreo y evaluación de conectividad, rescate de información (en cuanto a componentes que posee cada host) y a la detección de la ubicación del host dentro un red.

1.2 ANTECEDENTES

La necesidad de una investigación en la unión de estas dos ramas de la informática, que vienen a ser las redes y la programación, es muy importante para los objetivos que persigue un administrador de red, que es de tener al alcance de la mano una información detallada y en línea del estado de la red que administra como también el estado de cada uno de los componentes

administrados, esta última pudiendo ser switch, routers, hosts,etc., para la presente tesis solo se tomara en cuenta el monitoreo de hosts (pcs).

Dentro de la Carrera de Informática se pueden apreciar dos tesis que se adentran a la investigación de uso de agentes en redes y que dan un gran aporte para la elaboración de la presente tesis, estas son:

- "Agentes autónomos para la detección de intrusos de red" T.953
- "Arquitectura de un sistema de agentes autónomos en la consulta de información distribuida" T.490

Esta ultima que propone una estructura para la elaboración de un sistema multiagente con dos funcionalidades muy importantes que son la comunicación y cooperación, para permitir recuperar información de diferentes centros de almacenamientos.

A nivel país se desconocen proyectos o existe muy poca información acerca del concepto de agentes combinados con el uso del protocolo SNMP, o aplicaciones hechas para la administración de redes.

A nivel internacional la empresa CISCO se ha encargado de crear su propio software para la administración de sus equipos que son routers[1], switchs[2] y otros, pero específicamente para el monitoreo[3] de Pcs existe software con licencia o muy poco software libre, pero con la limitante de que se desconoce su código y a consecuencia de esto se limita bastante su uso y la expansión en cuanto a sus alcances.

[1] Router, 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.

[2] Switch es un dispositivo digital de lógica de interconexión de redes de computadores 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.

[3] Monitoreo, Monitorizar, observar el curso de uno o varios parámetros para detectar posibles anomalías

1.3 PLANTEAMIENTO DEL PROBLEMA

1.3.1 PROBLEMA GENERAL

"Deficiencias en la gestión de hosts para brindar atención oportuna y registro adecuado de incidencias tanto a nivel de software y hardware, como a la conexión a una red".

1.3.2 PROBLEMAS ESPECÍFICOS

- Demora en el diagnóstico ante problemas que atraviesan los hosts.
- Demora en solución ante problemas que atraviesan los hosts.
- Limitado seguimiento a antecedentes de los hosts.
- Falencias en la supervisión de conexión de un host a una red.
- Desconocimiento del universo de hosts administrados.
- Ausencia de información adecuada de los hosts.

1.4 OBJETIVOS

1.4.1 OBJETIVO GENERAL

"Implementar un sistema de gestión y/o administración de hosts basado en agentes usando el protocolo SNMP, para identificar y evitar posibles problemas de hardware o software en terminales, que originen la caída de servicios dentro una red LAN".

1.4.2 OBJETIVOS ESPECÍFICOS

- Aviso oportuno para un diagnostico y una propuesta de solución rápida ante problemas que atraviesan los hosts.
- Correcto seguimiento de antecedentes de los hosts.
- Supervisión de quien, donde y cuando de la conexión del host a una red.
- Generación de información general oportuna de incidentes en hardware, software y red de las terminales.
- Generación de información individual de cada host conectado a una red.

- Hacer el uso de agentes para la solución a un problema de administración de red.
- Explotación del protocolo de gestión y/o administración SNMP, para así demostrar su gran utilidad dentro el área de las redes informáticas.

1.5 HIPÓTESIS

Este trabajo plantea la siguiente hipótesis:

La automatización de la supervisión del comportamiento de los hosts mediante agentes optimizará la administración de PCS en una red LAN y reducirá la caída de servicios.

1.6 JUSTIFICACIÓN

1.6.1 SOCIAL

La sociedad hoy en día atraviesa tiempos donde la administración de la información debe ser rápida, oportuna y fiable, una información que vaya en secuencia inmediata con los hechos ocurridos, es éste tipo de información valiosa para un administrador de red y es lo que cualquier usuario que forma parte de una red desearía obtener, por ello la presente investigación pretende cumplir con ese objetivo.

La programación de un sistema con dichas cualidades y con la supervisión constante de un agente reducirá los perjuicios a los usuarios de la red LAN y si es implementado en una empresa de servicios tratara de evitar problemas con los clientes si se trata de hosts ubicados en cajas tomando un ejemplo.

1.6.2 ECONÓMICA

Se reducirán costos en el caso de relevamiento de información host o más comúnmente conocido como inventario, además que se usará software libre para el desarrollo del agente y el sistema con el que trabajara, haciendo

uso de java, MySql y un servidor con sistema operativo Linux lo cual hace del tema de investigación accesible para su implementación pudiendo este sistema ser usado en una red LAN.

Se buscara evitar la degradación de equipos que a la larga representan un perjuicio para los usuarios de cualquier tipo, pues resta su jornada de trabajo una Pc en malas condiciones y genera pérdidas económicas en empresas.

1.6.3 TÉCNICA

Actualmente las actividades de gestión usan técnicas y herramientas muchas veces aisladas e incompatibles entre sí. Se requiere por tanto de una pronta solución que además de simple, se convierta en una plataforma unificada para la gestión de redes.

La Gestión basada en agentes es un acercamiento promisorio que puede proveer una solución de gestión realmente integrada.

Este trabajo plantea inspeccionar el área de la Gestión de redes basada en agentes, y desarrollar una primera fase que sea rápidamente implementable en el entorno para la cual será desarrollada.

1.7 METODOLOGÍA

Se usa el método científico que incluye tanto la inducción y deducción, así como la experimentación que es la parte medular y de la que arrojará resultados que luego serán usados.

Para la realización del presente trabajo se tomará como base la Metodología de la Programación Extrema, que se adapta al proceso de creación del sistema debido a sus particularidades, donde se realizará el análisis, diseño e implementación a base de prototipos por cada etapa de presentación.

Para el diseño del Agente se hará uso de la metodología MAS Common – KADS, y se trabajará bajo el framework JADE (Java Agent DEvelopment Framework), donde dicho framework de desarrollo de agentes hacen uso de JAVA como lenguaje de programación.

1.8 LIMITES Y ALCANCES

- El sistema basado en agentes trabaja solo bajo la arquitectura de una red LAN.

- El sistema basado en agentes tendrá alcance a la información de hardware de los hosts hablando específicamente de PCS, no tomando en cuenta switchs, routers o impresoras que trabajen bajo red.

- Para el caso de la elaboración de un diagnostico lo hará de forma general y no específica por que se estaría adentrando al área de los sistemas expertos que escapa a los objetivos de la presente tesis.

- Para el caso de diagnóstico y relevamiento de información en el área de hardware solo se tomara en cuenta los datos de capacidad, uso de los recursos y temperatura de los componentes administrables.

1.9 APORTES

En cuanto a los aportes que se brindara:

- ☑ Se dará al área de Informática un nuevo modelo de agente que interactúe con las redes de computadoras, para este cometido también se brindara información sobre el uso del protocolo de gestión de red que es el SNMP.

- ☑ Se dejará la estructura de un sistema de gestión y/o administración abierto a cualquier modificación, para que sea de ayuda o base de futuras investigaciones.

Capítulo 2

Marco Teórico

2.1 METODOLOGÍAS AGILES Y PROGRAMACIÓN EXTREMA

2.1.1 HISTORIA

En febrero de 2001, tras una reunión celebrada en Utah - EEUU, nace el término "ágil" aplicado al desarrollo de software. En esta reunión participa un grupo de 17 expertos de la industria del software, incluyendo a algunos de los creadores o impulsores de metodologías de desarrollo del software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto.

Se pretendía ofrecer una alternativa a los procesos de desarrollo del software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas (Canos J,2003).

La Tabla 1 recoge esquemáticamente las principales diferencias de la metodología ágil con respecto a las metodologías tradicionales (denominado en este contexto como "no ágil"). Estas diferencias que afectan no sólo al proceso en sí, sino también al contexto del equipo así como a su organización (Canos J, 2003).

Metodología Ágil	Metodologías Tradicionales
Basada en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuesta internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños de menos de 10 integrantes y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos
Pocos artefactos.	Más artefactos
Pocos roles.	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 1: Diferencias entre la metodología ágil y las tradicionales.

Fuente: [Canos J, 2003].

2.1.2 MANIFIESTO ÁGIL

El manifiesto ágil es un resumen de los principios sobre los que se basan los métodos alternativos de desarrollo del software, y en su firma participaron: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland y Dave Thomas, según este manifiesto ágil se valora (Canos J,2003):

1. **Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.** La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo con base en sus necesidades.

2. **Desarrollar software que funciona más que conseguir una buena documentación.** La regla a seguir es "no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante". Estos documentos deben ser cortos y centrarse en lo fundamental.

3. **La colaboración con el cliente más que la negociación de un contrato.** Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

4. **Responder a los cambios más que seguir estrictamente un plan.** La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo.

2.1.2.1 PRINCIPIOS ÁGILES

Los valores anteriores inspiran los 12 principios del manifiesto, estos principios son características que diferencian un proceso ágil de uno tradicional. Los dos primeros son generales y resumen gran parte del espíritu ágil. Los restantes tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto a las metas a seguir y la organización del mismo.

Según (Canos J, 2003), los 12 principios son:

1. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas del software que le aporte un valor.

2. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.

3. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.

4. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.

5. Construir el proyecto en torno a individuos motivados. Proporcionarles el entorno, el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.

6. El diálogo cara a cara es el método más eficiente para comunicar información al interior de un equipo de desarrollo.

7. El software que funciona es la medida principal de progreso.

8. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una armonía de trabajo constante.

9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.

10. La simplicidad es esencial.

11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.

12. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

2.1.3 MÉTODOS ÁGILES

Los métodos ágiles actualmente constituyen una tendencia bastante aceptada para el desarrollo de productos software, debido a que estos se están convirtiendo en métodos populares entre los desarrolladores (Choque G.,2005).

2.1.3.1 MÉTODOS ÁGILES REPRESENTATIVOS

Aunque los creadores e impulsores de los métodos ágiles más populares han suscrito el manifiesto ágil y coinciden con los principios ya mencionados, cada uno de estos métodos tiene características propias y enfatizan en algunos aspectos específicos (Canos J,2003).

A continuación se resumen los métodos ágiles más importantes:

1. Métodos Cristal - Crystal Methods

Desarrollados por Alistair Cockburn, Crystal no se refiere a un método cerrado, sino a un conjunto de ellos, con los criterios para seleccionar y adecuar los más apropiados a cada proyecto, (Enrich M,2003),(Palacio J,2007).

Estos métodos están centrados en las personas del equipo de desarrollo que representa un factor clave, por lo que es necesario invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener definidas políticas de trabajo en equipo. Estas políticas dependerán del tamaño del equipo. De esta

manera se ha establecido una clasificación por colores, es decir, que Crystal identifica con colores diferentes cada método y su elección debe ser consecuencia de los parámetros de critica así como del tamaño del sistema, que también involucra a la cantidad de miembros en el equipo de desarrollo (Enrich M,2003),(Palacio J,2007).

En la figura 1.1, se observan los métodos cristal en función de los factores de criticidad y tamaño.

Figura 1: Métodos cristal en función a la criticidad y tamaño.

Fuente: [Palacio J, 2007].

Las características de criticidad y dimensión de este método, figura 1.1, son descritas a continuación (Enrich M, 2003), (Palacio J, 2007).

a. Criticidad, es la medida cuantitativa del número de pérdidas que un mal funcionamiento del sistema ocasionaría al desarrollo del mismo. De abajo hacia arriba, los criterios que corresponden a los niveles de integridad y estos son: Pérdida de confort o usabilidad. Pérdidas económicas moderadas. Pérdidas económicas graves. Pérdida de vidas humanas.

b. Dimensión, determina el tamaño del sistema por el número de personas empleadas en su desarrollo, Cuantas más personas estén implicadas, más grande será el proyecto, y en este caso un error no detectado puede ser crítico además que implica un costo considerable. De

izquierda a derecha, la dimensión está descrita por el número de miembros del equipo desde el más pequeño de 6 personas hasta el más grande de 80 personas, diferenciándose así la dimensión a través de colores.

Es de esta manera que la familia de métodos Crystal selecciona al más adecuado para el proyecto a realizar. La familia Crystal se fundamenta en (Enrich M, 2003), (Palacio J, 2007).

a. Desarrollo iterativo e incremental

b. Duración máxima de una iteración de 1 a 3 meses.

c. Énfasis en la importancia de las personas que en los procesos.

d. Énfasis en la comunicación directa.

e. Es un modelo abierto a la adaptación e introducción de prácticas de otros métodos ágiles, tales como Programación Extrema, Scrum, etc.

2. Desarrollo veloz en internet - Internet Speed Development (ISD)

El desarrollo veloz en Internet (Internet Speed Development (ISD)), surge de las conclusiones del coloquio celebrado en Octubre de 2001, que reunió a especialistas de las universidades Carneige Mellon, Georgia y Copenhagen. Sienta bases de gestión para abordar el desarrollo de sistemas de software, de tamaño pequeño que requieren tiempos de desarrollo muy rápidos (Enrich M, 2003).

3. Desarrollo de software adaptativo - Adaptative Software Development (ASD)

El Desarrollo de Software Adaptativo (Adaptative Software Development (ASD)) tiene su impulsor en Jim Highsmith, es un modelo de implementación de

patrones ágiles para el desarrollo de software (Enrich M, 2003), (Palacio J, 2007).

El ciclo de vida que propone tiene tres fases esenciales (Enrich M, 2003), (Palacio J, 2007):

a. Especulación, fase en la que se inicia el proyecto y se planifican las características del software, esta fase está compuesta por 5 pasos:

    i. Inicio para determinar la misión del proyecto.
    ii. Determinación del marco temporal del proyecto.
    iii. Determinación del número de iteraciones y la duración de cada una.
    iv. Determinación del objetivo de cada iteración.
    v. Asignación de funcionalidad a cada iteración.

b. Colaboración, fase en la que se realiza un desarrollo concurrente de la construcción y gestión del producto

c. Aprendizaje, fase en la que:

    i. Se revisa su calidad, con criterios del cliente y técnicos. La revisión de los imponentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.
    ii. Se entrega al cliente el producto

Sus principales características son:

a. Iterativo,

b. Orientado a los componentes software más que a las tareas y

c. Tolerante a los cambios.

2.1.3.2 PROGRAMACIÓN EXTREMA[1]

La Programación Extrema (Extreme Programming XP) es un método ágil centrado en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, la comunicación fluida entre todos los participantes, la simplicidad en las soluciones implementadas y el coraje para enfrentar los cambios, es especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Las características de la programación extrema, son:

Comunicación: programadores en constante comunicación con los clientes para satisfacer los requisitos propuestos por ellos y responder rápidamente a los cambios de los mismos. Este valor requiere de una interacción total por parte de todo el equipo, es muy importante ya que apoya y acelera el proceso de retroalimentación en el proyecto.

Simplicidad: aplicación de diseños simples y claros en cuanto al código del programa, permiten un mantenimiento estable, y desarrollo de futuras implementaciones sin mucho esfuerzo.

Retroalimentación: mediante la retroalimentación se ofrece al cliente la posibilidad de conseguir un sistema apto a las necesidades que presenta, ya que se muestra el proyecto a tiempo para ser cambiado y retroceder a una fase anterior para rediseñarlo según se vea conveniente. Este proceso está muy ligado a la comunicación donde el cliente provee las variables de entrada en forma de requerimientos y necesidades a una siguiente fase donde es mejorado el software.

[1] PROGRAMACIÓN EXTREMA, artículo publicado por la Asociación de Investigación en Software Inteligente A.I.S.I. Simposio 2006.

Coraje: valor esencial, ya que la metodología Programación Extrema requiere de valentía o coraje para cumplir con los tres puntos anteriores en beneficio del proyecto. Se debe tener valor para comunicarse con el cliente y enfatizar algunos puntos, y se debe tener coraje para mantener un diseño.

Existe un conjunto de doce lineamientos basados en los cuatro valores para la correcta aplicación de la Programación Extrema, los cuales son en su mayoría un compendio de buenas prácticas de desarrollo de software que se fueron perfeccionando a lo largo del tiempo, y que en esta metodología son aplicadas con frecuencia y de manera optima, dichos lineamientos o buenas prácticas se muestran en la tabla 2:

Nro	Practica
1	Juego de planificación
2	Diseño simple.
3	Prueba continua
4	Estándares de codificación.
5	Entregas pequeñas y frecuentes
6	Metáfora del sistema
7	Refactorización continua.
8	Programación en pares
9	Propiedad colectiva del código
10	Integración continua
11	Ritmo sostenible, trabajando un máximo de 8 horas por día
12	Todo el equipo en el mismo lugar

Tabla 2: Prácticas de la Programación Extrema.
Fuente: [Beck, 2000].

La metodología Programación Extrema se divide en 4 fases principales en el desarrollo que son la exploración, planificación, iteraciones y la producción, donde en la primera que es la fase de exploración se definen los requerimientos iniciales con los que comenzara el proyecto, luego en la fase de planificación se priorizan las tareas más importantes a ser ejecutadas, se planifica el desarrollo del proyecto, y se estiman los tiempos iniciales de

desarrollo, luego en la fase de iteración se desarrolla en software en sí, teniendo siempre al cliente disponible, esta fase junto con la de planificación y de producción siguen un ciclo incremental repetitivo, que se repite n-veces de acuerdo a la magnitud del sistema, se alcanza la conclusión del proyecto cuando el cliente ya no tiene ningún requerimiento y está conforme con el software desarrollado.

Las fases de Planificación, Iteración, y Producción son iterativas entre sí, ya que la salida de una es la entrada de otra y siguen un ciclo evolutivo hasta que no existen necesidades por parte del usuario respecto al software, su ciclo de vida se lo detalla en la figura 1.

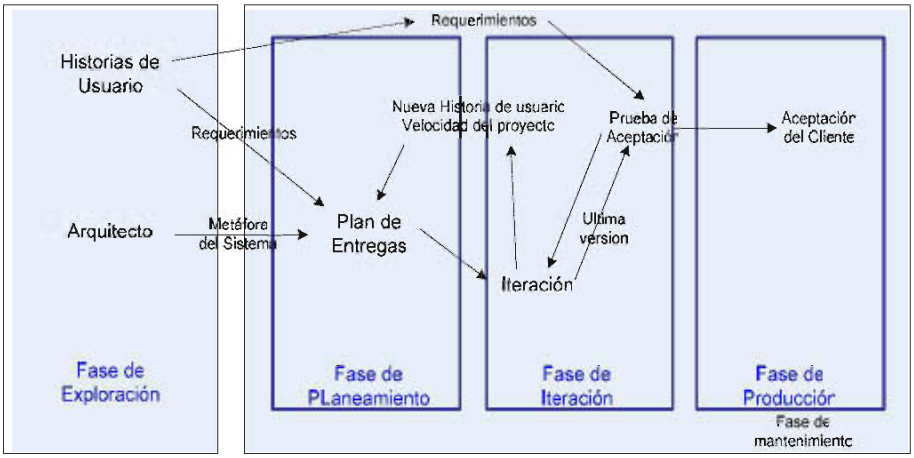

Figura 2: Ciclo de vida del proyecto XP.
Fuente: [Ambler, 2002].

En la fase de Exploración es donde se realiza el análisis de requerimientos previo al inicio para el desarrollo del sistema, es muy importante que el cliente se acople a la dinámica de trabajo a realizarse en el proyecto, proveyendo de historias de usuario aptas para la codificación de los programadores, es la única que no participara en el ciclo iterativo de planificación, iteración y producción del proyecto citadas anteriormente.

Una vez que se llega a la fase de Producción se inicia una nueva Iteración empezando nuevamente con la fase de Planificación, para seguir un nuevo ciclo.

2.1.3.2.1 FASE DE EXPLORACIÓN (Exploration Phase)

En esta fase, los clientes plantean a grandes rasgos las historias de usuarios que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizaran en el proyecto. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores.

El desarrollo de esta fase debe cumplir con los siguientes elementos:

a) Historias de usuario (HU): Constan de 3 a 5 líneas escritas por el cliente en lenguaje no técnico, sin hacer énfasis en los detalles, no se debe hablar de posibles algoritmos, diseños de base de datos adecuados, ni de otras características técnicas.

Solamente se proporciona los detalles sobre la estimación del riesgo y del tiempo de duración para la implementación de dicha historia de usuario. El nivel de detalle debe ser el mínimo posible, para que permita plasmar una ligera idea de cuánto costará implementar el sistema. Cuando se llegue a la fase de implementación, los desarrolladores tienen la posibilidad de acudir al cliente para ampliar detalles.

Cuando llega la hora de implementar una historia de usuario, el cliente y los desarrolladores se reúnen para concretar y detallar lo que tiene que hacer dicha historia, el tiempo ideal de desarrollo e implementación para una historia de usuario está entre 1 y 3 semanas (Beck, 2000).

Historia de Usuario	Nº : 1	7
Nombre: 2		Prioridad: 3
Descripción : 4		
Estimación: 5		Dependencia: 6

Descripción:		
	1	Numero de historia de usuario
	2	Nombre de la historia de usuario
	3	Prioridad en planificación de entregas
	4	Descripción de la historia de usuario
	5	Estimación del tiempo de desarrollo
	6	Dependencia del programador
	7	Nombre del sistema

Figura 3: Historia de Usuario.

Fuente: [Beck, 2000].

b) Metáfora: una metáfora para el sistema es una historia que se cuenta acerca de cómo el sistema funciona.

c) Diseño Arquitectónico: la arquitectura no es solo el software operacional, es la representación de la estructura que capacita al ingeniero para: analizar la efectividad del diseño para la obtención de los objetivos fijados, considerar las alternativas arquitectónicas en una fase en la cual hacer un cambio en el diseño es relativamente fácil y reducir los riesgos asociados a la construcción de software.

d) Elección de herramientas tecnológicas y estrategias: proceso de escoger herramientas y estrategias aptas para resolver un problema determinado, de acuerdo a las características y alcances. Proveerán de facilidades y un mejor alcance para el objetivo del proyecto.

2.1.3.2.2 FASE DE PLANIFICACIÓN (Planning Phase)

El cliente establece la prioridad de cada historia de usuario y correspondientemente los programadores realizan la estimación del esfuerzo necesario da cada una de ellas. Tanto cliente como equipo de desarrollo

determinan el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería de obtenerse en no más de tres meses. Esta fase dura unos pocos días.

2.1.3.2.3 FASE DE ITERACIONES (Iteration to Release Phase)

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El plan de entrega está compuesto por iteraciones que como máximo duran tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias de usuarios que contengan elementos necesarios que den lugar a la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que el cliente decide que historias se implementaran en cada iteración. Al final de la última iteración el sistema estará listo para entrar en producción.

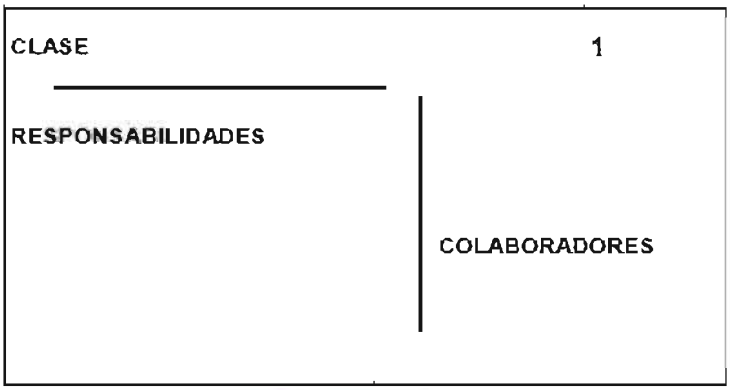

Figura 4: Formato de Tarjeta CRC.
Fuente: [Beck, 2000].

En esta fase es común el uso de las tarjetas CRC (Clase responsabilidad y Colaboración) que permiten al programador centrarse y apreciar el desarrollo orientado a objetos, representan objetos. La clase a la que pertenece el objeto se escribe en la parte superior de la tarjeta, en una columna a la izquierda se escriben las responsabilidades u objetivos que debe cumplir el objeto, a la derecha las clases que colaboran con cada responsabilidad.

El nombre del objeto debe ser muy descriptivo, las responsabilidades especifican las funciones que le objeto debe cumplir, y las colaboraciones muestran otras clases (CRC) que colaboraran para efectuar las tareas asignadas. Para tener una mayor organización respecto a las historias de usuario se pude colocar en la parte superior derecha el número de historia de usuario a la que pertenece.

2.1.3.2.4 FASE DE PRODUCCIÓN (Production)

Esta fase requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se toman decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase

2.2. AGENTES INTELIGENTES [1]

El término agente es cada vez más conocido y se emplea en campos tan diversos como Internet, sistemas distribuidos, inteligencia artificial o la interacción persona-computador. En la actualidad, se habla de agentes inteligentes, agentes móviles, agentes software, agentes autónomos, sistemas multiagente. El campo de los agentes ha atraído a científicos procedentes de áreas diferentes: psicología, sociología, ingeniería del software, inteligencia artificial, etc. y cada uno de los miembros de estas comunidades tiende a ver el problema desde su perspectiva. Por tanto, realizar una definición de agente o agencia es complicado, debido a la diversidad de opiniones que existen en la comunidad científica sobre este tema. (Molina, Garcia, Bernardos, 2004, p. 3)

2.2.1 DEFINICIONES

a) Definición de Michael Wooldridge:

"Un agente es un sistema informático que está situado en un entorno y

[1] Agentes Inteligentes, artículo publicado por la Asociación de Investigación en Software Inteligente A.I.S.I. Simposio 2006.

es capaz de actuar autónomamente sobre él a fin de conseguir sus objetivos de diseño". (Wooldridge, 2002: 15)

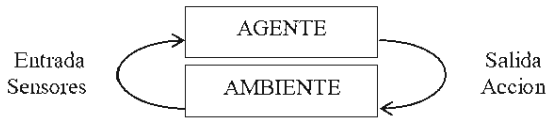

Figura 5: Un agente en su ambiente. El agente toma entradas de sensores del ambiente, y produce como acciones del rendimiento que lo afecta.
Fuente: [Wooldridge, 2002: 16].

La Figura 5 da una vista abstracta de un agente. En este diagrama, se puede ver el rendimiento de acción generado por el agente para afectar su ambiente. En la mayoría de los dominios de complejidad razonable, un agente no tendrá mando completo encima de su ambiente. ¿Cuándo considerar que un agente es inteligente? La pregunta, no es fácil de contestar. Una manera de contestar la pregunta es listando las características que se tiene a continuación: reactivo, proactivo y social. (Wooldridge, 2002, p.16)

b) Definición de Russell S. y Norvig P.:

"Un agente es todo aquello que puede considerarse que percibe su ambiente mediante sensores y que responde o actúa en tal ambiente por medio de efectores" (Russell, Norvig, 1995, p. 33).

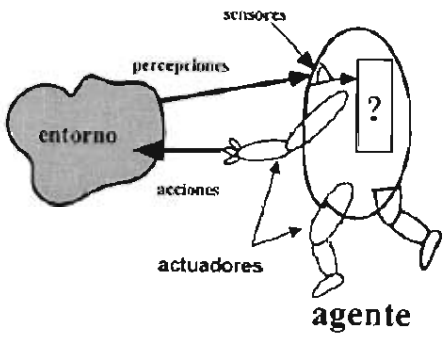

Figura 6: Visión Esquemática de un agente
Fuente: [Russell, Norvig, 1995, p. 34].

2.2.2 PROPIEDADES DE AGENTES

En las varias definiciones encontradas, se identifica la similitud de conceptos entre ellas, ya que algunas enfatizan las mismas características; sin embargo, otras agregan nuevas características o extienden definiciones anteriores.

a) Guillermo Choque indica que para que un agente pueda llamarse inteligente además de ser autónomo tiene que ser flexible. El termino flexible en la autonomía del agente engloba las siguientes características: (Choque, 2004, p.2)

1. **Sensible o reactivo (responsive):** el agente tiene que percibir su entorno y responder en un tiempo adecuado ante los cambios que se produzca en dicho entorno.

2. **Oportunista o pro-activo (proactive):** el agente no sólo ha de reaccionar frente a su entorno, sino que el agente tiene que aprovechar aquellas situaciones que puedan darse en dicho entorno si estas favorecen la consecución de sus objetivos ofreciendo una conducta de iniciativa propia.

3. **Social (social):** el agente tiene que ser capaz de interactuar, cuando lo considere oportuno, con otros agentes y humanos con el objetivo de completar sus actividades o ayudar a la realización de las tareas de otros.

b) Franklin S. y Graesser A., indican que luego de analizar una serie de definiciones relacionadas con el concepto de agente, de manera natural definen una característica, la autonomía la cual permite distinguirlos de los programas tradicionales. Afirman que: La autonomía, en este caso, se relaciona con la no dependencia para la realización de algo, y con respecto de alguien. La

capacidad de autonomía por sí sola no basta para considerar a los agentes como entidades inteligentes, sino más bien es su capacidad para responder de manera adecuada ante las situaciones cambiantes en el entorno.

c) Hermans Björn describe las características de los agentes:

Autonomía: capacidad de operar sin intervención directa de los humanos o de otros agentes, con cierto tipo de control sobre sus acciones. Después del conocimiento integrado definitivamente que la autonomía es la característica más importante de los agentes inteligentes dado que, esta le permitirá definir su conducta basado en su propia experiencia.

Sensibilidad: los agentes tienen la capacidad de interactuar con humanos u otros agentes mediante un lenguaje en particular.

Reactividad: las percepciones captadas de su ambiente producen una acción específica.

Pro actividad: los agentes tienen capacidad de exhibir un comportamiento particular dependiendo de los objetivos planteados.

Continuidad: los agentes están constantemente ejecutando procesos (captando percepciones y ejecutando acciones).

Benevolencia: capacidad de satisfacer solicitudes.

Racionalidad: el agente actuará de manera tal de satisfacer sus objetivos.

Colaboración: al interactuar de manera constante con el usuario, los agentes solicitan colaboración de estos constantemente con la finalidad de ejecutar acciones eficaces y eficientes.

2.2.3 CARACTERÍSTICAS DE AGENTES

La confusión respecto al término “agente” viene del hecho de que cada investigador asocia con los agentes un juego de características algo diferentes. De todas maneras, casi todas las definiciones contienen alguna combinación de características comunes que se indican a continuación:

a) Capacidad de trabajo asíncrono y de manera autónoma.

La característica manejada por la mayoría de los agentes presumiblemente inteligentes es que pueden funcionar con autonomía, y sin la intervención del ser humano .

b) Capacidad para cambiar su comportamiento según el conocimiento acumulado.

La capacidad de "aprender" puede ser, generalmente, la segunda característica más asociada con los agentes inteligentes. Por ejemplo, la "ayuda adaptativa" se ha caracterizado como una característica central en arquitecturas de interfaz inteligente. Similarmente, un uso frecuente de los agentes está en preveer las necesidades de información del usuario buscando qué bases de datos mantienen un "perfil de interés" basándose en preferencias pasadas de búsqueda. Cualquier cambio de interés del usuario deberá ser tomado en cuenta por el agente.

c) Capacidad para tomar iniciativas.

Un agente verdaderamente inteligente debe tener la capacidad para ejecutar tareas de acuerdo con su propia meta, separada de la del usuario. Los ambientes que condujeron a esta propuesta eran esos en los que el agente “sabe” más que el usuario sobre una materia y sobre las estrategias a utilizar para resolver el problema.

d) Capacidad inferencial.

Una característica frecuentemente asociada con agentes es la capacidad para realizar inferencias. Se define como la capacidad para ir más allá de las instrucciones concretas y específicas del usuario y resolver problemas utilizando alguna forma de abstracción simbólica (Shoham, 1993). En algunos casos, estas abstracciones estarán en forma de reglas (como las utilizadas en sistemas expertos). En otros casos, las inferencias pueden estar basadas en razonamientos probabilísticos.

e) Conocimiento anterior de metas generales y métodos preferidos

Esta característica puede verse como una extensión de las características "aprendizaje" e "inferencia". Tiene que ver con el reconocimiento de que las soluciones a muchos problemas mundiales verdaderos requieren una comprensión de metas generales.

Este nivel de comprensión es el requerido para las versiones más sofisticadas de agentes inteligentes. Un agente inteligente no solamente deber ser capaz de generar su propia meta prioritaria, sino que debe comprender también la meta del usuario para el que actúa como agente.

f) Lenguaje natural

Para problemas simples, los lenguajes de scripting actuales pueden resultar suficientes para que el usuario especifique el método y el resultado deseado al agente. Sin embargo, cuando los agentes tienen que resolver una variedad amplia de problemas cotidianos, la especificación de esos problemas llega a ser una tarea compleja. Debido a la complejidad y a causa de las preferencias naturales de un usuario, muchos investigadores asumen una interfaz de lenguaje natural (Shoham, 1993).

g) Personalidad

Algunos investigadores han encontrado atractivo adjuntar características humanas a los agentes, como la personalidad. La suposición aquí parece ser

que como el problema a resolver y los requerimientos interactivos de comunicación pueden llegar a ser muy grandes y complejos, únicamente un humano como entidad podría gestionar el problema. De forma similar a los humanos, la comunicación de algunos tipos de información va a estar estrechamente relacionada con la personalidad. El concepto de agente inteligente, como se puede comprobar, va ser una complicada mezcla de características. A pesar del acuerdo sobre la sustancia del concepto y una lista final de características comunes de agente, permanece la confusión sobre el término porque el subconjunto necesario y suficiente de características no está acordado. El programa agente, para poder realizar sus actividades de forma autónoma, necesitará ser un experto en el dominio, tener conocimiento de métodos y estrategias para dividir el problema en subtareas, habilidades inferenciales y capacidades para adoptar medidas. El que los agentes puedan tener distintos grados de parecido al ser humano contribuye a aumentar la confusión del término. Se pueden imaginar agentes que trabajan bien pero que no están personificados. Sin embargo, como la delegación ha sido una actividad fundamentalmente interpersonal, la gente puede encontrar más fácil delegar a, y trabajar con, agentes que son, más o menos, como el ser humano.

2.2.4. TAXONOMÍA DE AGENTES

La taxonomía es una ciencia que ayuda a ordenar, describir y clasificar a todos los seres vivos, en el caso de los agentes inteligentes, se la tomara para ordenar, describir y clasificar de acuerdo a sus características principales. Mientras los Agentes inteligentes todavía son algo nuevos en los ambientes de computación de comercial, ha sido el enfoque de investigadores por años. Durante ese tiempo, han sido propuestas muchas maneras diferentes de clasificar o categorizar a los agentes. Una manera es poner al agente en el contexto de la inteligencia, el organismo, y la movilidad. Otro enfoque es concentrarse en la estrategia de procesamiento principal del agente.(Bigus, Bigus, 2001, p.9)

2.2.4.1 TIPOLOGÍA

Liliana Santacruz indica que la tipología se refiere al estudio de tipos de entidades de agentes. (Santacruz, 2005) Existen maneras de clasificar a los Agentes Software existentes: (Nwana, 1996, p.6)

- En primer lugar pueden ser clasificados por su movilidad es decir por su habilidad para moverse por la red. Se originan dos tipos de agentes: estáticos o móviles.

- En segundo lugar pueden ser clasificados como deliberativos o reactivos. Los deliberativos se derivan del paradigma del pensamiento deliberativo: los agentes poseen un modelo de razonamiento simbólico interno comprometido en la planeación y negociación para realizar coordinación con otros agentes. Los agentes reactivos al contrario no tienen ningún modelo simbólico interno de su entorno y actúan utilizando un tipo de comportamiento de estímulo/respuesta para responder al estado presente en el entorno en el que están embebidos.

- En tercer lugar, los agentes se pueden clasificar a lo largo de varios atributos primarios que deben exhibir como mínimo autonomía, aprendizaje y cooperación.

  La autonomía se refiere al principio de que los agentes pueden operar por ellos mismos sin intervención humana o de otros agentes.

  La cooperación es la razón para tener múltiples agentes y para que ellos cooperen es necesario que los agentes posean una habilidad social, por ejemplo la de interactuar con otros agentes y posiblemente con humanos a través de algún lenguaje de comunicación. Finalmente, los agentes inteligentes, ellos pueden aprender a cómo reaccionar, interactuar con su medio ambiente.

A partir de tres características mínimas de agente en la Figura 6 se derivan cuatro tipos de agentes que se pueden añadir a la tipología de agentes. Los agentes colaborativos, agentes de aprendizaje colaborativo, agentes de interface y agentes inteligentes. (Nwana, 1996, p.6)

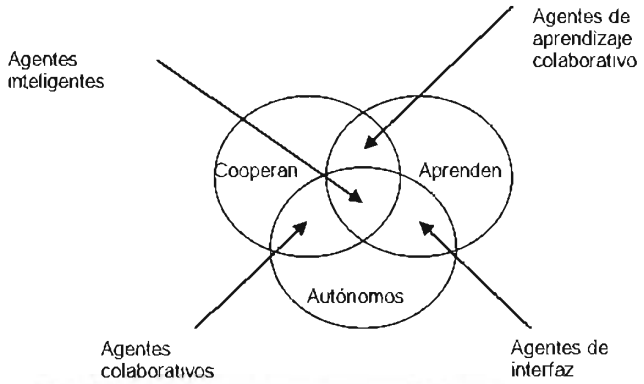

Figura 7: Tipología de los Agentes según sus características.
Fuente: [Nwana, 1996, p.6]

Los agentes cooperativos se enfatizan más en la cooperación y en la autonomía que en el aprendizaje, pero esto no implica que nunca aprendan. (Nwana, 1996, p.10)

Por otra parte, los agentes de interfaz hacen más énfasis en la autonomía y en el aprendizaje que en la cooperación.

Además los agentes pueden ser clasificados por los roles que desempeñan, por ejemplo Agentes de Información o Agentes Internet, en esta categoría los agentes se dedican a explorar Internet ayudando con su actividad a manejar gran cantidad de información pueden ser estáticos, móviles o deliberativos. Finalmente, se tiene una clase de agentes híbridos, los cuales combinan dos o más filosofías de agentes en una clase de agente. A partir de la tipología de agentes definida se definirá brevemente las metáforas esenciales, hipótesis, objetivos, motivaciones, roles, ejemplos prototípicos, beneficios potenciales, cambios claves y algunos otros usos generales acerca de cada tipo particular de agentes. (Nwana, 1996, p.6)

2.2.4.1.1 AGENTES COLABORATIVOS

Los agentes colaborativos se enfatizan en la autonomía y la cooperación para ejecutar tareas por sí mismos, pueden aprender pero este no es un aspecto relevante. Las características generales de estos agentes incluyen autonomía, sociabilidad, responsabilidad y pro actividad. Sin embargo ellos deben ser capaces de actuar racional y autónomamente en entornos multi-agente en un menor tiempo y en entornos abiertos. Tienden a ser estáticos, pueden ser benevolentes, racionales, efectivos, algunas combinaciones de ellos o ninguna. (Nwana, 1996, p.10) Los sistemas de Agentes Colaborativos tienen una de las siguientes especialidades:

a) Resolver problemas en un sistema distribuido, ya que para un solo agente resultaría muy complejo debido a las limitaciones de recursos y evitar exponerse a tener un sistema centralizado.

b) Permitir la interconexión e ínter operación de múltiples sistemas por ejemplo sistemas expertos, sistemas para el soporte de decisión, etc.

c) Proporcionar soluciones a problemas inherentemente distribuidos, por ejemplo redes con sensores distribuidos, controladores de tráfico aéreo.

d) Proporcionar soluciones a partir de las fuentes de información distribuidas, por ejemplo en fuentes de información distribuidas on-line.

e) Proporcionar soluciones donde la experiencia es distribuida por ejemplo en provisión de cuidados sanitarios.

f) Proporcionar modularidad (la cual reduce la complejidad), velocidad (debido al paralelismo), confiabilidad (debido a la redundancia), flexibilidad (se realizan nuevas tareas más fácilmente desde una organización más modular) y reusabilidad a nivel del conocimiento (recursos compartidos).

2.2.4.1.2 AGENTES DE INTERFAZ

Se enfatizan en la autonomía y el aprendizaje para la realización de las tareas.

Esencialmente los agentes de Interfaz soportan y proporcionan asistencia, para que el usuario aprenda a utilizar una aplicación particular tal como un sistema operativo. El agente de interfaz observa y supervisa las acciones efectuadas por el usuario en la interfaz, aprende nuevos atajos y sugiere mejores formas para realizar las tareas. Así el agente de interfaz actúa como un asistente personal autónomo que coopera con el usuario en el cumplimiento de algunas tareas en la aplicación. Los agentes de interfaz aprenden a asistir al usuario de formas diferentes, por ejemplo: (Nwana, 1996, p.14)

a) Observando e imitando al usuario

b) Por realimentación positiva y negativa desde el usuario.

c) Recibiendo instrucciones explícitas por parte del usuario

d) Solicitando consejo a agentes.

El objetivo de los agentes de interfaz es migrar de una metáfora de manipulación directa a una que delegue alguna de las tareas a Agentes de Interfaz Software (proactivo y útil), para conformar a los usuarios nuevos. La motivación para este tipo de agentes es que pueden realizar tareas que para los humanos resultan en cierta medida tediosa y además reducen el trabajo y la sobrecarga de información. Los beneficios aportados por este tipo de agentes son: primero, reducen el trabajo para el usuario final y para el desarrollador de la aplicación. En segundo lugar, pueden adaptarse sobre la marcha a las preferencias y hábitos de los usuarios. Finalmente sabe cómo entre los diferentes usuarios de la comunidad será compartido. En general los agentes de interfaz pueden ser usados en el desarrollo de aplicaciones reales en

términos breves porque son simples, operan en dominios limitados y no requieren en general, cooperación con otros agentes. (Nwana, 1996, p.14)

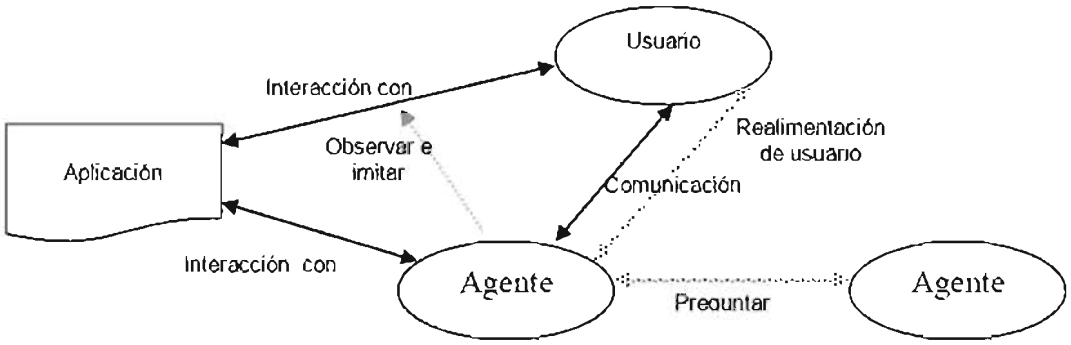

Figura 8: Funcionamiento de un Agente de Interfaz.
Fuente: [Nwana, 1996, p. 14].

2.2.4.1.3 AGENTES MÓVILES

Los Agentes Móviles son procesos software (computacionales) capaces de recorrer o vagar por redes de WAN tales como World Wide Web (WWW), interactuando con host extraños, recogiendo información en nombre de su propietario y realizando las obligaciones impuestas por sus usuarios. Esas obligaciones pueden variar desde hacer una reserva de avión hasta manejar una red de telecomunicaciones.

Los agentes móviles son implementaciones de programas remotos es decir programas que se desarrollan en una máquina y se distribuyen en otra máquina para su subsecuente ejecución.

Además son agentes porque son autónomos y cooperan, a diferencia de los agentes colaborativos, por ejemplo ellos pueden cooperar y comunicarse con otros agentes ubicando algunos de los objetos y métodos internos de otros agentes, de esta manera pueden intercambiar datos e información sin necesidad de dar toda la información. (Nwana, 1996, p.19) La utilización de agentes móviles supone los siguientes beneficios:

a) Reducción en el costo de la comunicación: existe información de la cual es necesaria determinar su relevancia y la transferencia de esta información puede llevar mucho tiempo y congestiona la red.

b) Limitación de los recursos locales: el poder de procesamiento y almacenamiento de una máquina local puede ser muy limitado.

c) Coordinación sencilla: resulta fácil coordinar un número de requisitos remotos e independientes y comparar todos los resultados localmente.

d) Cálculo asíncrono: que el agente móvil realice tareas y que los resultados se guarden en el buzón de correo por decir algo en algún momento después. Los agentes pueden operar cuando no esté el usuario conectado a la red.

e) Proporcionan entornos de desarrollo natural para implementar servicios de libre comercio. Estos servicios pueden co-existir con otros de nivel inferior proporcionando al consumidor más alternativas.

f) Una arquitectura de cómputo distribuida flexible: proporcionan una arquitectura de cómputo distribuido única cuyas funciones son diferentes de las asignaciones estáticas permitiendo una forma nueva de hacer computación distribuida.

g) Los agentes móviles representan una oportunidad para hacer un replanteamiento radical y atractivo de los procesos de diseño en general.

2.2.4.1.4 AGENTES DE INFORMACIÓN / INTERNET

Los agentes de información surgen de la necesidad de manejar el crecimiento de información que se encuentra en la Internet y obtener de la red los mayores beneficios. Los agentes de información se encargan de manejar, manipular y coleccionar información de muchas fuentes distribuidas. Los agentes de información tienen varias características: pueden ser estáticos o móviles, no cooperativos o sociales y pueden o no aprender, de allí que no existe un modelo estándar que defina su modo de operación. Los agentes

Internet móviles, son capaces de atravesar el WWW y recolectar información. Los agentes de información pueden estar asociados a un índice o índices particulares que son capaces de buscar en el WWW, almacenando la topología del WWW en un Sistema Manejador de Base de Datos (DBMS). Los buscadores como Lycos o Webcrawler pueden emplearse para construir el índice. El usuario del agente de información requiere información con ciertas características y solicita varias búsquedas a una o más máquinas de búsqueda de Uniform Resource Locator (URLs) para encontrar la petición, algunas de esas búsquedas pueden hacerse localmente se tiene una cache local. La información se localiza y se envía al usuario.

Los agentes de información son similares a los agentes de interfaz o a los agentes móviles. Si los agentes de información son estáticos, entonces se les aplica los cambios de los agentes de interfaz, sin embargo si son móviles se aplica los cambios de los agentes móviles. Por otra parte, las dificultades que presentan los agentes de información son similares a los agentes de interfaz y los móviles dependiendo de si ellos son estáticos o móviles respectivamente. (Nwana, 1996, p.24)

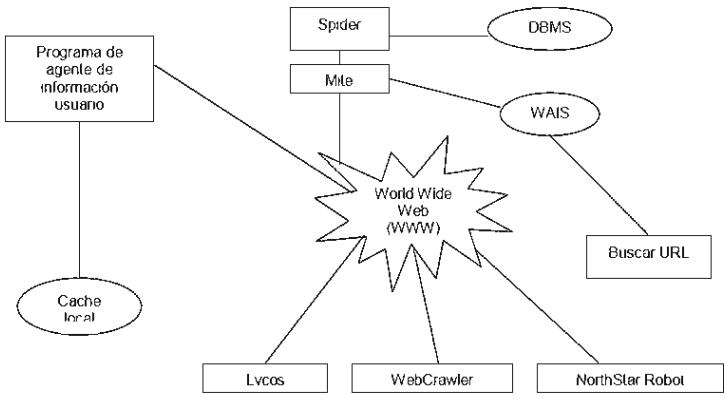

Figura 9: Funcionamiento de los Agentes de Información.
Fuente: [Nwana, 1996, p.24].

2.2.4.1.5 AGENTES SOFTWARE REACTIVOS

Los agentes reactivos representan una categoría especial de agentes que no poseen modelos simbólicos internos de sus entornos, en su lugar reaccionan o responden en modo de estímulo respuesta para representar el estado del entorno en el que están empotrados. Algunas de las características soportadas por los agentes reactivos son: (Nwana, 1996, p.26)

a) Funcionalidad: No existe una especificación a priori del comportamiento de los agentes reactivos.

b) Descomposición de tareas: Un agente reactivo es visto como un conjunto de módulos que operan autónomamente y que son responsables de tareas especificas.

c) Operan sobre representaciones que se aproximan a datos en bruto, en contraste a las representaciones simbólicas de alto nivel que están presentes en otros tipos de agentes.

Los sistemas de agentes inteligentes (reactivos), se desarrollan a partir de agentes simples que no tienen modelos simbólicos internos y el que sean inteligentes se deriva del comportamiento que surge de las interacciones entre varios módulos. Es importante notar que los agentes reactivos actuales no poseen necesariamente actores y sensores que los conecten al mundo físico.

Los agentes reactivos son simples y fáciles de entender, su recurso cognitivo es bastante bajo, ya que tienen que recordar muy pocas cosas. Ellos no hacen planes futuros o revisan algunos modelos del mundo, sus acciones dependen de lo que pasa en el momento presente.

La ventaja de trabajar con agentes reactivos es que son más robustos y tolerantes que otros sistemas basados en agentes, por ejemplo un agente se puede perder, pero sin efectos catastróficos. Otros beneficios que se incluyen son la flexibilidad y la adaptabilidad en contraste con la inflexibilidad, tiempo de

respuesta lento y la vulnerabilidad de los sistemas de Inteligencia Artificial clásicos. Otro beneficio es que este tipo de trabajo puede direccionar el problema de estructura que ha sido difícilmente abordable por medio de las técnicas tradicionales de inteligencia artificial tales como el razonamiento no monotónico. (Nwana, 1996, p.26)

2.2.4.1.6 AGENTES HÍBRIDOS

Hasta ahora los tipos de agentes estudiados poseen distintas fortalezas, deficiencias y lo que se busca es maximizar las fortalezas y minimizar las deficiencias de las técnicas más relevantes para propósitos particulares. Una forma de hacerlo es adoptar una aproximación híbrida. Los agentes híbridos se refieren a aquellos cuya constitución es una combinación de dos o más filosofías de agentes para formar un agente único. Para el desarrollo de algunas aplicaciones el tener una arquitectura de agentes híbrida puede ser de ayuda si se quiere tener mayores beneficios de la arquitectura, ya que se aprovecha lo mejor de la filosofía de cada agente, por ejemplo en el caso de agentes reactivos se puede aprovechar la robustez, tiempo de respuesta rápida y adaptabilidad. (Nwana, 1996, p.29)

2.2.4.2 CLASIFICACIÓN

Todos los tipos de agentes se pueden asignar a una matriz multidimensional construida a partir de las diferentes características, pero en este caso se consideran tres criterios de gran relevancia. Los criterios seleccionados son los adecuados para realizar la clasificación de los actuales y futuros tipos de agentes y permiten además que todas las características mencionadas anteriormente se puedan representar mediante uno o más criterios. Los criterios de los que se habla son Inteligencia, movilidad y número de agentes. (Santacruz, 2005). Para un mejor entendimiento, la clasificación de agentes es mostrada en la Figura 7.

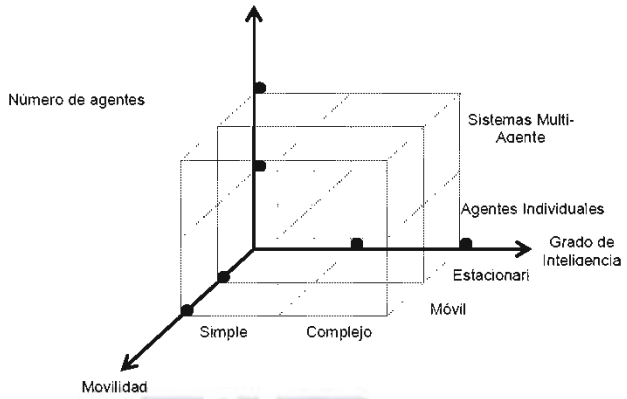

Figura 10: Matriz de Clasificación para los Sistemas de Agentes
Fuente: [Santacruz, 2005].

Se utilizan los términos simple y complejo para referirse al grado de inteligencia de los agentes. Los agentes simples tienen una cantidad limitada de inteligencia, a comparación de los sistemas complejos que demuestran un alto grado de comportamiento inteligente. Se diferencian también dos tipos de agentes móviles dentro del criterio de clasificación de movilidad objetos móviles y códigos móviles. Los códigos móviles son enviados a otra computadora antes de que se ejecuten en esa computadora. En contraste los objetos móviles pueden cambiar su posición en cualquier momento durante su ejecución, en este caso no solo se transfiere sino también el estado del sistema.

Los agentes individuales se mueven en un entorno que no contiene ningún otro agente, en otras palabras no son capaces de contactarse con otros agentes solo se contactan con el usuario y con diferentes fuentes de información tal como una o varias bases de datos. En contraste los sistemas multiagente (conjunto de agentes inteligentes autónomos) cuentan con numerosos agentes que pueden comunicarse o cooperar con otros.

La capacidad de comunicación está presente tanto en agentes individuales como en los sistemas multiagente y se puede colocar dentro de la categoría de número de agentes. La capacidad de cooperación afecta tanto al criterio de Inteligencia como al del número de agentes. El comportamiento

autónomo afecta la Inteligencia y la movilidad de un agente. La operación de un agente móvil es útil solo cuando tiene un máximo grado de autonomía.

En general las distintas tareas realizadas por los agentes (información, cooperación, transacción), se pueden asignar a los campos de la matriz de clasificación. (Santacruz, 2005)

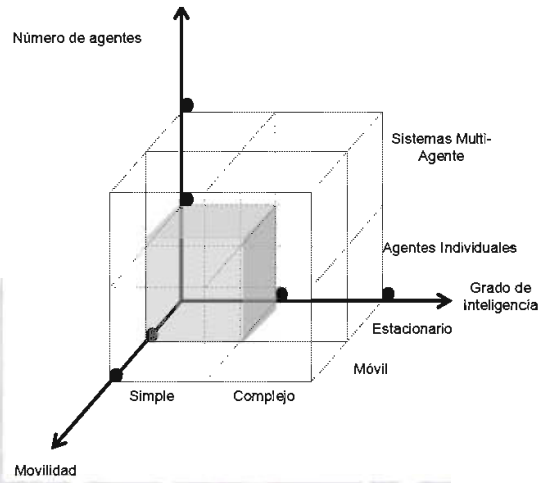

Figura 11: Ubicación de los Agentes de Información dentro de la Matriz de clasificación para los Sistemas de Agentes
Fuente: [Santacruz, 2005].

Los agentes de información tienen una cantidad relativamente limitada de inteligencia y pertenecen al área de agentes individuales, además son agentes que operan individualmente porque no necesitan una cooperación extensiva de varios agentes para desempañar las tareas de búsqueda de información. Los agentes que existen en la actualidad tienen una naturaleza estacionaria, sin embargo, se desea que los agentes de información se comporten como agentes móviles, de esta forma se incrementa la autonomía y se reduce la carga en la red. En consecuencia se espera un incremento en la proporción de agentes móviles cuando se den las condiciones de infraestructura adecuadas para ellos.

Los agentes cooperativos deben tener un alto grado de inteligencia y su ubicación debe corresponder al área de agentes complejos. El criterio de

movilidad es aplicado porque ellos deben ser activos en un entorno multiagente. La movilidad no es esencial porque los sistemas de agentes orientados a cooperación se concentran en el proceso actual de solución del problema, se desarrollan y conciben para operar dentro de un sistema de computación. Sin embargo, la movilidad puede ser un criterio deseable para los agentes cooperativos.

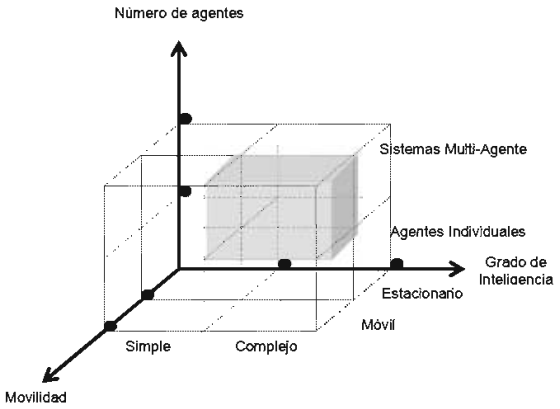

Figura 12: Ubicación de los Agentes de Transacción dentro de la Matriz de Clasificación de los Sistemas de Agentes
Fuente: [Santacruz, 2005].

Los agentes de transacción pueden ser utilizados como agentes individuales y en sistemas multiagente, por ejemplo un número de agentes que realizan transacciones con otros agentes involucrados en la compra y venta electrónica basada en agentes.

Por otra parte, los agentes se usan para supervisar transacciones dentro de una red de telecomunicaciones como un sistema individual. El uso de los agentes de transacción en sistemas multiagente puede ser una excepción. Si por ejemplo las negociaciones complejas tienen lugar entre varios agentes y un agente debe tener una estrategia de negociación clara, el agente se relaciona con las correspondientes demandas de su inteligencia. Los agentes de transacción se pueden realizar como agentes móviles o estacionarios. La decisión depende del escenario de aplicación asociado. (Santacruz, 2005)

2.2.4.3 TAXONOMÍA DE LOS AGENTES

Los agentes se pueden clasificar dentro de las diferentes tecnologías existentes. Se habla entonces de una clasificación dentro del contexto de sistemas de agentes individuales donde se encuentran los agentes locales y los agentes de red y sistemas multiagentes que trabajan con los agentes basados en DAI (Inteligencia Artificial Distribuida) y agentes móviles.

En un sistema de agentes Individuales un agente ejecuta tareas en nombre del usuario o de algún proceso. Mientras ejecuta esas tareas el agente puede comunicarse con el usuario así como también con otros sistemas de recursos remotos o locales. En contraste, los agentes en un sistema multiagente pueden cooperar extensivamente con cada uno de los demás para realizar sus propios objetivos, por supuesto en esos sistemas el agente puede interactuar con los recursos del sistema y con los usuarios.

La tabla siguiente muestra la clasificación de la tecnología de agentes y sus correspondientes áreas de aplicación: (Santacruz, 2005)

Sistemas de Agentes Individuales		Sistemas multiagente	
Agentes Locales	*Agentes de red*	*Agentes basados en DAI*	*Agentes Moviles*
Asistentes Personales Asistentes de Consejo (ej: Sistemas de Ayuda) Planificador de reuniones Asistentes Personales	Correos Inteligentes recuperadores de Información Automatización de Procesos	Resolución distribuida de Problemas	Telecomunicaciones Communicaciones personales Manejo de redes Servicios sobre demanda Mercado Electrónico

Tabla 3: Clases de Agentes Inteligentes y sus Aplicaciones.
Fuente: [Santacruz, 2005].

2.2.4.3.1 AGENTES LOCALES

Estos agentes acceden a recursos locales. Generalmente un agente local actúa como un agente de ayuda, o como asistentes personales que soportan a

usuarios humanos durante su trabajo diario. El objetivo de estos agentes es colaborar con el usuario por ello el principal énfasis en la investigación lo constituye el campo de interacción usuario/agente. Por otra parte este tipo de agente ha sido llamado también Agente de Interfaz o Interfaz Inteligente Los agentes locales pueden asistir al usuario de diferentes formas: ejecutan tareas en nombre del usuario, pueden ocultar la complejidad de tareas difíciles, actuar asíncronamente, aprender y enseñar al usuario. La cantidad de tareas que el usuario puede realizar es virtualmente ilimitada: recuperar y filtrar información local, manejar correo local, planear reuniones, etc. (Santacruz,2005)

2.2.4.3.2 AGENTES DE RED

En contraste con los agentes locales, los agentes de red pueden acceder no solo a recursos locales sino también a los remotos, y tienen un conocimiento de la infraestructura de la red y de los servicios disponibles dentro de la red. La principal diferencia entre esta clase de agentes y los sistemas multiagente es que los agentes de red no asumen la cooperación con otros agentes.

Muchos de los agentes de redes existentes actúan como asistentes personales. Los ejemplos más populares probablemente son los carteros inteligentes y motores de búsqueda. Ellos no solo proporcionan una interfaz inteligente al usuario, sino que también hacen uso extensible de varios servicios disponibles en la red. En contraste con los carteros inteligentes los cuales desarrollan un filtrado avanzado del correo basado en las preferencias del usuario, los motores de búsqueda agrupan el conocimiento disponible en la red en nombre del usuario. Este tipo de agente se conoce también como KnowBot o Softbot. Al acumular la información el agente no solamente oculta la complejidad de la red sino que accede a un amplio rango de información heterogénea y a protocolos invisibles para el usuario y finalmente presenta los resultados al usuario. Este tipo de agente es especialmente importante para Internet. El increíble crecimiento del WWW ha generado una creciente demanda de herramientas para el soporte y manejo de las grandes cantidades de información disponible en la red. Entre las muchas soluciones desarrolladas hay

varias implementaciones basadas en agentes conocidas como WebCrawlers, Spiders y Robots. (Santacruz, 2005)

2.2.4.3.3AGENTES BASADOS EN INTELIGENCIA ARTIFICIAL DISTRIBUIDA (DAI)

La principal preocupación en los sistemas multiagente basados en DAI es la coordinación entre un conjunto de agentes autónomos inteligentes, por ejemplo coordinan sus objetivos, habilidades, conocimiento, planifican la realización de tareas o resuelven problemas. Este tipo de agentes se utiliza en un amplio rango de aplicaciones por ejemplo en la supervisión distribuida de vehículos, la fabricación de computadoras integradas, planeación de transporte y en particular en la gestión y manejo de las telecomunicaciones.

Estos agentes se desarrollan mediante técnicas de Inteligencia Artificial (IA), como sistemas basados en reglas, ejemplos basados en razonamiento. Las representaciones aplicadas al conocimiento y los mecanismos de inferencia son básicamente iguales a los utilizados en los sistemas basados en conocimiento (no-distribuidos). En los sistemas multiagente basados en DAI un agente puede comunicarse con el usuario, con los recursos del sistema y con otros agentes, además cooperar en la realización de tareas, esta cooperación se establece por medio de un protocolo de contratación que permite la negociación entre agentes. La teoría del lenguaje de agentes, se utiliza para definir la semántica de los mensajes por ejemplo KQML (Knowledge Query and Manipulation Language) y KIF (Knowledge Interchange Format), que son lenguajes utilizados para la transmisión de conocimiento entre varios tipos de sistemas basados en conocimiento, modelado de reglas y contenidos. (Santacruz, 2005)

2.2.4.3.4 AGENTES MÓVILES

Los agentes móviles están orientados a ofrecer gran número de servicios sofisticados en grandes redes de computadoras, por ejemplo para el filtrado

avanzado en Internet y agentes de búsqueda, mensajeros inteligentes y conexión de redes.

Los agentes de este tipo se desarrollan por medio de lenguajes scripting por ejemplo Java, Telescript. Java se utiliza para realizar operaciones de sincronización y ejecución remota de aplicaciones dentro de Internet sin la cooperación de agentes. La metáfora utilizada para la tecnología Telescript es el mercado electrónico. Dentro de este mercado los agentes ejecutan tareas asíncronas en nombre del usuario. Ellos pueden comunicarse con el usuario, con otros servicios disponibles en la red y con otros agentes, sin embargo los mecanismos sofisticados de cooperación entre agentes están aún fuera del alcance de Telescript.

La tecnología existente soporta la movilidad de los agentes, mientras que Java solo permite la ejecución remota de agentes dentro de Internet, los agentes Telescript pueden migrar mientras están activos.

En la tabla siguiente se resumen las propiedades de las clases de agentes identificadas, se debe hacer énfasis en que la taxonomía de los conceptos de agentes inteligentes no es completa y debe ser considerado solamente como un medio para categorizar los conceptos de agentes existentes. (Santacruz, 2005)

Clases / Atributos	Agentes Locales	Agentes de Red	Agentes basados en DAI	Agentes Móviles
Inteligencia	□	□	□	□
Operación Asíncrona	□	□	□	□
Comunicación		□	□	□
Cooperación			□	□
Movilidad				□

Tabla 4: Clases de Agentes y sus Características.
Fuente: [Santacruz, 2005].

La idea de realizar operaciones cliente/servidor mediante la transmisión de programas ejecutables entre clientes y servidores por ejemplo el despacho de programas para ejecución remota es bastante vieja y se denomina procesos de trabajo remoto en los 70"s y función de expedición o evaluación remota en los 80"s. En el pasado la principal motivación para aplicar este principio fue la falta de capacidad para ejecutar programas localmente y el deseo de compartir recursos y mejorar el balance de carga en los sistemas distribuidos. Estos conceptos se diseñaron para entornos específicos o entornos cerrados, los nuevos conceptos de agentes apuntan a entornos abiertos (por ejemplo dentro de Internet).

2.2.5. SISTEMAS MULTIAGENTES

Un Sistema Multiagente (SMA) es aquel que contiene una colección de dos o más agentes. Dentro de esta colección de agentes, debe de existir al menos una relación entre dos agentes para cooperar. Este conjunto de agentes están organizados de alguna forma e interactúan en un ambiente común para alcanzar una meta (Iglesias,1998).

Las características de un Sistema Multiagente son:

- Los agentes tienen información y capacidad incompleta para resolver el problema, por lo tanto su punto de vista es limitado.
- No existe un sistema de control global.
- Los datos son descentralizados.
- El cómputo es asíncrono.

2.2.6. MAS-commonKads METODOLOGÍA ORIENTADA A SISTEMAS MULTIAGENTE

Esta metodología es una extensión de CommonKads para sistemas de agentes usando técnicas de metodologías orientadas a objetos.

MAS-CommonKADS propone los siguientes modelos para el desarrollo de sistemas Multiagente (Iglesias,1998).

• *Modelo de Agente (AM):* especifica las características de un agente: sus capacidades de razonamiento, habilidades, servicios, sensores, efectores, grupos de agentes a los que pertenece y clase de agente. Un agente puede ser un agente humano, software, o cualquier entidad capaz de emplear un lenguaje de comunicación de agentes.

• *Modelo de Organización (OM):* es una herramienta para analizar la organización humana en que el Sistema Multiagente va a ser introducido y para describir la organización de los agentes software y su relación con el entorno.

• *Modelo de Tareas (TM):* describe las tareas que los agentes pueden realizar: los objetivos de cada tarea, su descomposición, los ingredientes y los métodos de resolución de problemas para resolver cada objetivo.

• *Modelo de la Experiencia (EM):* describe el conocimiento necesitado por los agentes para alcanzar sus objetivos. Sigue la descomposición de CommonKADS y reutiliza las bibliotecas de tareas genéricas.

• *Modelo de Comunicación (CM):* describe las interacciones entre un agente humano y un agente software. Se centra en la consideración de factores humanos para dicha interacción.

• *Modelo de Coordinación (CoM):* describe las interacciones entre agentes software.

• *Modelo de Diseño (DM):* mientras que los otros cinco modelos tratan del análisis del sistema multiagente, este modelo se utiliza para describir la arquitectura y el diseño del sistema multiagente como paso previo a su implementación.

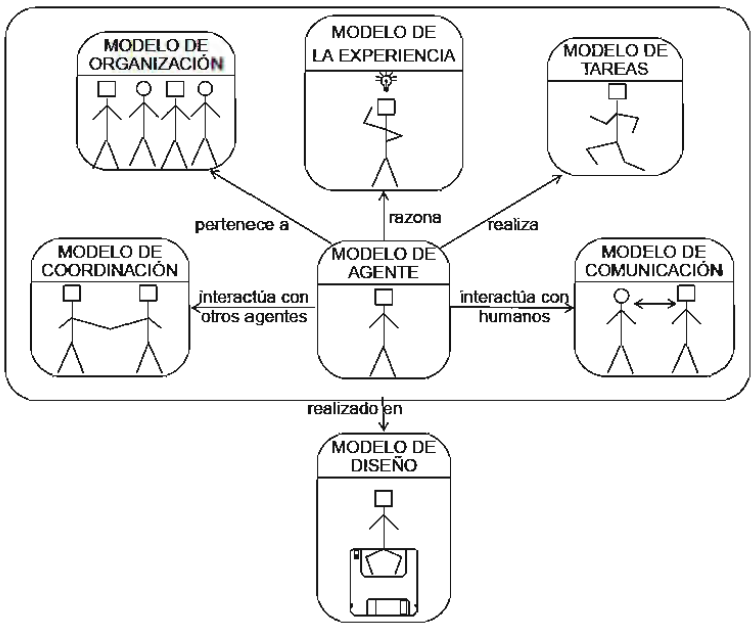

Figura 13: Los modelos de MAS-CommonKADS.
Fuente: [Iglesias,1998].

2.2.6.1 FASES DE DESARROLLO DE *MAS-CommonKADS*

El modelo de ciclo de vida para el desarrollo de sistemas multiagente con CommonKADS sigue las siguientes fases:

• *Conceptuación:* tarea de elicitación para obtener una primera descripción del problema y la determinación de los casos de uso que pueden ayudar a entender los requisitos informales y a probar el sistema.

• *Análisis:* determinación de los requisitos de nuestro sistema partiendo del enunciado del problema. Durante esta fase se desarrollan los siguientes modelos: organización, tareas, agente, comunicación, coordinación y experiencia.

• *Diseño:* determinación de cómo los requisitos de la fase de análisis pueden ser logrados mediante el desarrollo del modelo de diseño. Se determinan las arquitecturas tanto de la red multiagente como de cada agente.

El modelo de ciclo de vida propuesto para desarrollar estas tareas en la metodología es el modelo en espiral dirigido por riesgos, siguiendo la gestión de proyectos de CommonKADS. Tanto para proyectos pequeños como para el aprendizaje de la metodología, proponemos seguir otro modelo de ciclo de vida, basado en un modelo en cascada con reutilización.

Los proyectos pequeños, desarrollados por una única persona y con un número reducido de agentes, pueden emplear un ciclo de vida convencional en el que se incluye la utilización de componentes, que también se incluye en los proyectos grandes.

2.3 PROTOCOLO SNMP

2.3.1 PROTOCOLO SNMP

En esta parte se trata la situación actual de los dos protocolos[1] de gestión de red más importantes: el SNMP (Simple Network Management Protocol, protocolo simple de gestión de red) y el CMIP (Common Management Information Protocol, protocolo común de gestión de información). Se presentan las ventajas y desventajas de cada uno.

A finales de los años 70 las redes de computadoras experimentaron un espectacular crecimiento y empezaron a conectarse entre sí. A estas nuevas redes se les llamó inter-redes o internets. Pronto se hicieron muy difíciles de gestionar, y se hizo necesario el desarrollo de un protocolo de gestión.

El primer protocolo que se usó fue el SNMP (RFC 1157). Se diseñó como un recurso provisional hasta que se desarrollara otro protocolo más elaborado.

En los años 80 aparecieron dos nuevos protocolos: por un lado, la segunda versión del SNMP, que incorporaba muchas de las funciones del

[1] Protocolo de red, en informática, es un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red.

original (que sigue en uso) e incluía nuevas características que mejoraban sus deficiencias. Por otro, el CMIP, que estaba mejor organizado y contenía muchas más funciones que las dos versiones del SNMP.

Pronto el público general tendrá que elegir entre CMIP (RFC 1189) y SNMPv2 (RFC 1441), y esta decisión será de gran importancia: no en vano una empresa gasta alrededor del 15% del presupuesto asignado a sistemas de información en gestión de red.

Los criterios de elección deben basarse en las necesidades del usuario, esto es, un buen sistema de seguridad de red, una interfaz amigable, implementación relativamente barata y una reducción del tiempo empleado en gestión. Estos serán algunos de los criterios que se seguirán en la comparación de ambos protocolos.

2.3.2 INTRODUCCIÓN A SNMP

Para el desarrollo de la gestión de redes en inter-redes basadas en TCP/IP, el IAB (Internet Activities Board) decidió seguir la estrategia de usar a corto plazo el Simple Network Management Protocol (SNMP) para gestionar los nodos, proponiendo para largo plazo la estructura de gestión de redes OSI[1]. Se escribieron entonces dos documentos para definir la gestión de la información: RFC 1065 que definía la Estructura de la Información de Gestión (Structure of Management Information, SMI), y RFC 1066, que definía la Base de Información de Gestión (Management Information Base, MIB). Ambos documentos fueron diseñados para ser compatibles con la estructura SNMP y la de gestión de redes OSI.

Posteriormente se observó que los requerimientos de SNMP y los de gestión de redes OSI diferían más de lo esperado en un principio, por lo que los requerimientos de compatibilidad entre el SMI y el MIB fueron suspendidos.

[1] El modelo de interconexión de sistemas abiertos, también llamado OSI (open system interconnection). Marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones.

La IAB ha designado al SNMP, a la SMI, y a la Internet MIB inicial como "Protocolos Estándar", con status de "Recomendado". Por medio de esta acción, la IAB recomienda que todas las implementaciones de IP y TCP sean gestionables por red, y los adopten y apliquen.

Así pues, la actual estructura para gestión de redes basadas en TCP/IP consiste en:

• Estructura e Identificación de la Información de Gestión para redes basadas en TCP/IP, que describe cómo se definen los objetos gestionados contenidos en el MIB tal y como se especifica en la RFC[1] 1155.

• Protocolo de Gestión de Redes Simples, que define el protocolo usado para gestionar estos objetos, según se expone en la RFC 1157.

2.3.3 LA ARQUITECTURA DE SNMP

Implícita en el modelo de arquitectura del SNMP existe una colección de estaciones de gestión de red y de elementos de red. Las estaciones ejecutan aplicaciones de administración que monitorizan y controlan los elementos de red. Los elementos de red son dispositivos como hosts, gateways, servidores de terminal, y parecidos, que poseen agentes de gestión para realizar las funciones de administración de red solicitadas por las estaciones de gestión de red. El SNMP es usado para transmitir información de administración entre las estaciones de gestión y los agentes en los elementos de red.

2.3.4 PROPÓSITOS DE LA ARQUITECTURA

El SNMP explícitamente minimiza el número y complejidad de las funciones de gestión realizadas por el propio agente de gestión. Esta meta es atractiva al menos en cuatro aspectos:

- El coste de desarrollo del software del agente de gestión necesario para soportar el protocolo se reduce acordemente.

[1] RFC (Request For Comment - Petición de Comentarios). Documentos que se iniciaron en 1967 que describen los protocolos de internet.

- El grado de complejidad de funciones de gestión soportado remotamente se incrementa, posibilitando un uso completo de los recursos de Internet en la tarea de gestión imponiendo las mínimas restricciones posibles en la forma y sofisticación de herramientas de gestión.

- Los conjuntos simplificados de funciones de gestión son fácilmente entendibles y usados por los creadores de herramientas de gestión de red.

Un segundo objetivo del protocolo es que el paradigma funcional para monitorizar y controlar sea lo suficientemente flexible como para posibilitar aspectos de gestión y operación de la red adicionales y posiblemente no anticipados.

Un tercer propósito es que la arquitectura sea en lo posible independiente de los mecanismos de hosts o gateways particulares.

2.3.5 ELEMENTOS DE LA ARQUITECTURA

La arquitectura SNMP formula una solución al problema de gestión de redes en términos de los siguientes puntos:

- Alcance de la información de gestión comunicada por el protocolo.

- Representación de la información de gestión comunicada por el protocolo.

- Operaciones soportadas por el protocolo en la información de gestión.

- Forma y significado de los intercambios entre entidades de gestión.

- Forma y significado de las referencias a la información de gestión.

Alcance de la información de gestión

El alcance de la información de gestión transmitida por operaciones del SNMP es exactamente el representado por casos de todos los tipos de objetos no agregados, definidos en el estándar MIB (RFC 1156) de Internet, o definidos

en cualquier otro sitio de acuerdo a las convenciones expuestas en el estándar SMI (RFC 1155) de Internet.

Representación de la información de gestión

La información de gestión se representa según el lenguaje ASN.11, que es especificado para la definición de tipos no agregados en el SMI. El SNMP utiliza un subconjunto bien definido de dicho lenguaje, incluyendo un subconjunto más complejo para la descripción de objetos gestionados y para describir las unidades de datos de protocolo (PDU's) utilizadas para gestionar esos objetos. Así mismo solo se utiliza un subconjunto de las reglas básicas de codificación del ASN.1, esto es, todas las codificaciones utilizan la forma de longitud definida.

Con el deseo de facilitar una futura transición a protocolos de gestión de redes basados en OSI, se procedió a la definición en el lenguaje ASN.1 de un SMI standard de Internet y de un MIB.

Operaciones soportadas por la información de gestión

El SNMP modela las funciones del agente de gestión como lecturas (get) o escrituras (set) de variables. Esta estrategia posee al menos dos consecuencias positivas:

- Limita el número esencial de funciones de gestión realizadas por el agente de gestión a dos.

- Evita introducir el soporte de comandos de gestión imperativos en la definición del protocolo.

La estrategia plantea que la monitorización del estado de la red se puede basar a cualquier nivel de detalle en el sondeo (poll) de la información apropiada en la parte de los centros de monitorización. Un número limitado de mensajes no solicitados (traps) guían el objetivo y la secuencia del sondeo.

[1] ASN 1, Abstract Syntax Notation One (notación sintáctica abstracta 1)

Las funciones de los pocos comandos imperativos actualmente soportados pueden ser fácilmente implementadas en este modelo de modo asíncrono.

Forma y significado de los intercambios

La comunicación de la información de gestión entre entidades de gestión se realiza en el SNMP por medio del intercambio de mensajes de protocolo.

El intercambio de mensajes SNMP sólo requiere un servicio de datagramas poco fiable, y todo mensaje se representa por un único datagrama de transporte.

Forma y significado de las referencias a objetos gestionados

El SMI[1] (RFC 1155) requiere que la definición de un protocolo de gestión contemple:

- Resolución de referencias MIB ambiguas.
- Resolución de referencias MIB en presencia de múltiples versiones MIB.
- Identificación de casos particulares de tipos de objetos definidos en el MIB.

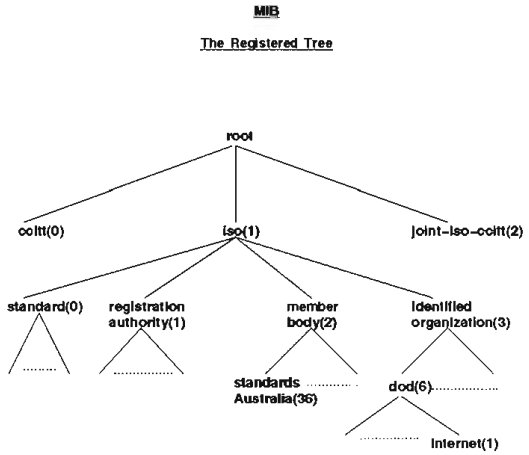

Figura 14: Estructura de un Árbol MIB (Management Information Base).
Fuente: [web2, 2010].

[1] SMI (Structure of Managment Information) estructura de la información de administración

Resolución de referencias MIB ambiguas: debido a que el alcance de cualquier operación SNMP está conceptualmente confinado a los objetos relevantes a un único elemento de red, y ya que todas las referencias SMI a objetos MIB son por medio de nombres de variables únicos, no hay posibilidad de que una referencia SNMP a cualquier tipo de objeto definido en el MIB se pueda resolver entre múltiples casos de ese tipo.

Resolución de referencias entre versiones MIB: el objeto referenciado por cualquier operación SNMP es exactamente el especificado como parte de la operación de petición, o en el caso de una operación get-next su sucesor en el conjunto de MIB. En particular, una referencia a un objeto como parte de una versión del MIB estándar de Internet, no se aplica a ningún objeto que no sea parte de dicha versión, excepto en el caso de que la operación sea get-next, y que el nombre del objeto especificado sea el último lexicográficamente entre los nombres de todos los objetos presentados como parte de dicha versión.

Identificación de los casos de objetos: cada caso de un tipo de objeto definido en el MIB se identifica en las operaciones SNMP por un nombre único llamado su "nombre de variable". En general, el nombre de una variable SNMP es un identificador de objeto de la forma x.y, donde x es el nombre del tipo de objeto no agregado definido en el MIB, e y es un fragmento de un identificador de objeto que de forma única para dicho tipo de objeto, identifica el caso deseado. Esta estrategia de denominación admite la completa explotación de la semántica de la PDU[1] GetNextRequest, dado que asigna nombres para variables relacionadas de forma que sean contiguas en el orden lexicográfico de todas las variables conocidas en el MIB.

2.3.6 SNMP: ESPECIFICACIONES DEL PROTOCOLO

El protocolo de administración de red es un protocolo de aplicación por el que las variables del MIB de un agente pueden ser inspeccionadas o alteradas.

[1] PDU (protocol data unit) : La información que es entregada como una unidad entre entidades de una red y que pueden contener información de control, información de direcciones o datos.

Las entidades de protocolo se comunican entre sí mediante mensajes, cada uno formado únicamente por un datagrama UDP. Cada mensaje está formado por un identificador de versión, un nombre de comunidad SNMP y una PDU (Protocol Data Unit - Unidad de datos de protocolo). Estos datagramas no necesitan ser mayores que 484 bytes, pero es recomendable que las implementaciones de este protocolo soporten longitudes mayores.

Todas las implementaciones del SNMP soportan 5 tipos de PDU:

- GetRequest-PDU
- GetNextRequest-PDU
- GetResponse-PDU
- SetRequest-PDU
- Trap-PDU

Figura 15: Operaciones de SNMP.
Fuente: [web2, 2010].

2.4 RESUMEN CAPITULO 2

La Programación Extrema

Es que a partir del año 2001 nace una nueva metodología llamada *eXtreme Programming* (XP) o ***Programación Extrema*** donde por su docilidad y teniendo como diferencias principales:

La Heurística proveniente de prácticas de código, o como se puede decir es una metodología que se encuentra más del lado del programador que del analista.

El trabajo preparado para cambios durante el proyecto, fácilmente es una metodología abierta para el trabajo con prototipos.

Los procesos menos controlados, lo que hace de esta metodología más flexible que otras metodologías más cerradas como es el caso de la metodología RUP.

Grupos pequeños y proyectos pequeños, así esta metodología es adecuada para proyectos de menos de 10 integrantes, donde hasta el cliente llega a ser parte del equipo de desarrollo.

Que hacen del trabajo de programación más libre y menos controlada, dando un motivo para su uso en el presente trabajo de investigación, orientada para el desarrollo de la parte web, más específicamente para el NMS (Network Management System) que es el sistema de administración de red.

Los agentes Inteligentes

Ya descrita la metodología usada, es que se debe dar explicación al motivo de la presente tesis que es el aporte de "**El *Agente Inteligente"*** que ya habiendo dado una definición de varios autores en este capítulo y de sus características principales que es la autonomía, lo sensible y oportunista, que nos dan en resumen un arma más para un buen sistema de monitoreo de host,

tomando como guía la reacción ante estímulos que nos ofrecen los ***"agentes reactivos".***

Los agentes reactivos representan una categoría especial de agentes que no poseen modelos simbólicos internos de sus entornos, en su lugar reaccionan o responden en modo de estímulo respuesta para representar el estado del entorno en el que están empotrados.

Los agentes reactivos son simples y fáciles de entender, su recurso cognitivo es bastante bajo, ya que tienen que recordar muy pocas cosas. Ellos no hacen planes futuros o revisan algunos modelos del mundo, sus acciones dependen de lo que pasa en el momento presente.

Este tipo de agente en dialogo con las pcs, usando como lenguaje el protocolo SNMP es que percibe todo lo que ocurre en la red a través de sus sensores.

El protocolo SNMP

El protocolo SNMP que para sus comienzos fue usado muy poco, hoy en día es una herramienta muy útil para los administradores de red, teniendo como componentes para su funcionamiento a:

NMS que es la interfaz para el uso del snmp que como tal es un simple protocolo.

MIB que es el árbol de información, guarda un parecido a la información resguardad en una Base de Datos pero que para sus consultas necesita de mensajes especiales.

MENSAJES El protocolo de administración de red es un protocolo de aplicación por el que las variables del MIB de un agente pueden ser inspeccionadas o alteradas.

Las entidades de protocolo se comunican entre sí mediante mensajes, cada uno formado únicamente por un datagrama UDP. Cada mensaje está formado por un identificador de versión, un nombre de comunidad SNMP y una PDU (Protocol Data Unit - Unidad de datos de protocolo). Estos datagramas no necesitan ser mayores que 484 bytes, pero es recomendable que las implementaciones de este protocolo soporten longitudes mayores.

Todas las implementaciones del SNMP soportan 5 tipos de PDU: GetRequest-PDU, GetNextRequest-PDU, GetResponse-PDU, SetRequest-PDU y Trap-PDU.

2.5 MAPA CONCEPTUAL CAPITULO 2

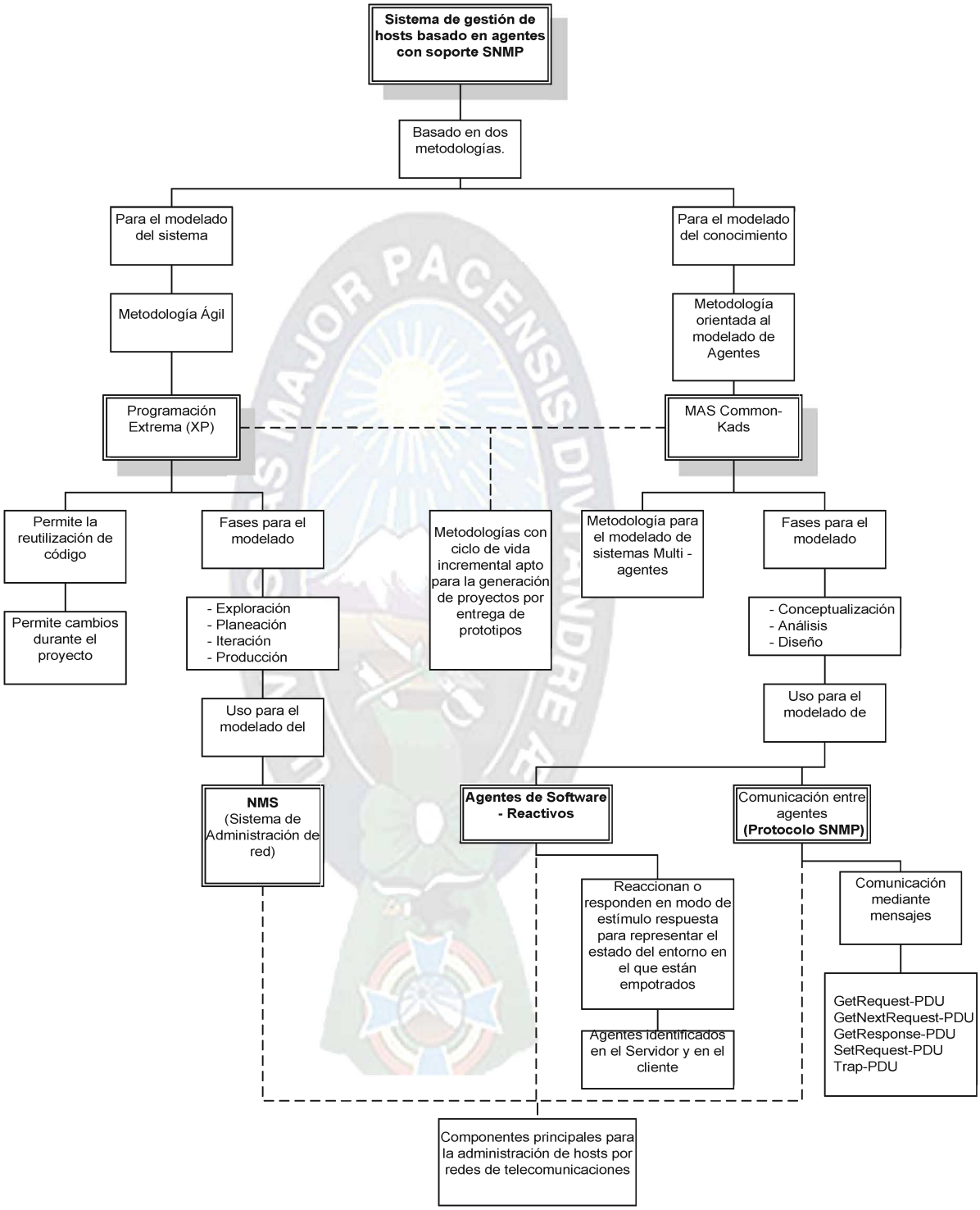

Figura 16: Mapa Conceptual de relación del marco teórico con el tema de tesis

Fuente: [Elaboración Propia].

2.5.1 DESCRIPCIÓN

El sistema de gestión de hosts basado en agentes con soporte SNMP planteado en la presente tesis hace uso de dos metodologías:

- **Para el modelado de los Agentes:** Se hace uso de la metodología ***MAS Common-KADS***, que es una metodología que brinda apoyo en la generación de ***Sistemas Multi-agentes***, cuenta con las fases de: ***Conceptualización, Análisis y Diseño***, dichas fases son útiles para el modelado de los agentes y su interacción, agentes que por su forma de participación en el sistema se los identifico como ***Agentes de software reactivos***.

- **Para el modelado del sistema:** donde se encontrará insertado uno de los Agentes ya mencionados, se hace uso de la Metodología Ágil enfocada en la ***Programación Extrema***.

Ambas metodologías poseen un ***ciclo de vida incremental (a base de iteraciones)***, motivo por el cual son aptas para el desarrollo de proyectos a base de prototipos, forma de trabajo de la presente tesis.

La Programación Extrema: ya que es una Metodología Ágil contempla la reutilización de código y los cambios durante el proyecto, dentro de sus fases están: la exploración, planeación, Iteración y producción, dichas fases que son útiles para la elaboración del ***NMS (Sistema de Administración de red)***, componente de la administración de red por SNMP.

Los Agentes de software reactivos: reaccionan o responden a base de estímulos, por lo cual son aptos para el trabajo en redes debido a su comunicación mediante mensajes momentáneos, de los cuales los mensajes implementados más importantes que son usados para la comunicación mediante el protocolo SNMP son: GetRequest-PDU, GetNextRequest-PDU, GetResponse-PDU, SetRequest-PDU, Trap-PDU.

Así de esta manera se cubre los tres componentes más importantes que son: el sistema administración de red (**NMS),** los agentes y el protocolo de comunicación SNMP, de la gestión de hosts por redes de telecomunicaciones.

Capítulo 3

Marco Aplicativo

3.1 INTRODUCCIÓN

La construcción de un Sistema de Gestión de Hosts basado en agentes integra dos tecnologías del área del conocimiento: por un lado están las técnicas de la inteligencia artificial para dotar a el sistema la capacidad de tratar situaciones imprevistas y tomar decisiones; y por el otro están las técnicas de la ingeniería de software, para estructurar el proceso de desarrollo del sistema en su conjunto de forma global.

De esta manera el presente capitulo se ve dividido en dos enfoques principales, donde cada enfoque implementa el uso de una metodología acorde al desarrollo del sistema.

3.2 DESCRIPCIÓN GENERAL

El prototipo que se propone en el siguiente trabajo como solución al problema en la gestión de hosts posee tres componentes principales:

- El primero y el más importante es el **"agente de software"** (Supervisor) que forman parte del SMA (Sistema Multiagente) e interactúa directamente con el "*NMS*" (Sistema de Gestión de Red); Este agente de software es el encargado de la interpretación de los mensajes enviados por los agentes SNMP y toma las decisiones correspondientes, pues forma parte del NMS dentro la lógica SNMP que se describió en el capítulo 2.

- **Los agentes SNMP** que forman parte del SMA como nodos instalados en los hosts clientes, son los encargados de notificar el estado de la Pc monitoreada, haciendo uso del protocolo SNMP para la comunicación con el agente de software ubicado dentro del servidor NMS. Para el caso de los sistemas operativos Windows y Linux se hará uso de agentes SNMP que se instalan como servicios [figuras 15 y 16].

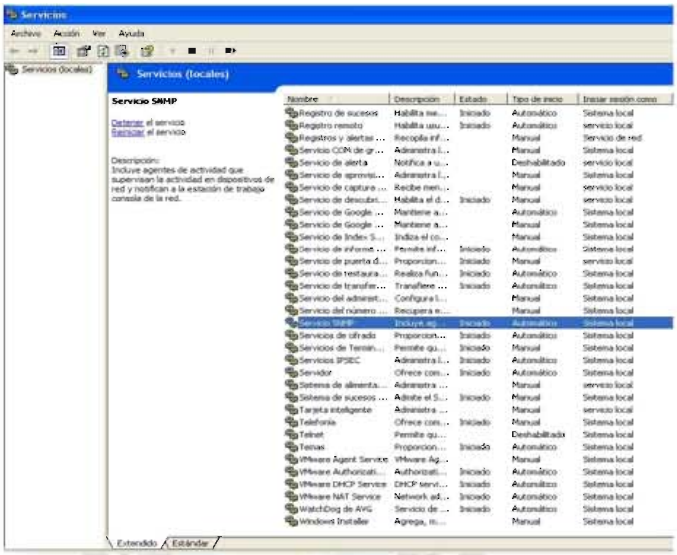

Figura 17: Vista del Agente SNMP corriendo en Windows XP.

Fuente: [Elaboración Propia].

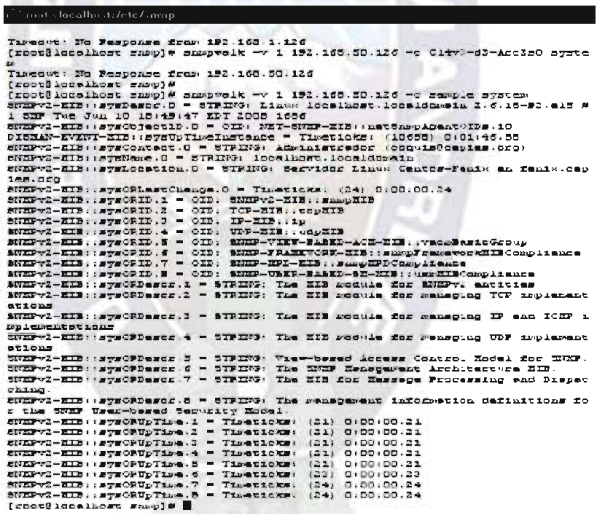

Figura 18: Vista del Agente SNMP corriendo en Linux Centos 5.

Fuente: [Elaboración Propia].

- **El NMS** (Network Management System – Sistema de Administracion de Red) es desarrollado con la colaboración de una herramienta B.P.M, para la interacción en forma de: casos, notificaciones por email y reportes, con los usuarios del sistema que por el momento son: el administrador de red y el personal de soporte.

La comunicación entre los tres componentes se lo realiza de la siguiente manera:

- **Agente de software <> Agente SNMP**: la comunicación es mediante los cinco tipos de PDU's propios del protocolo SNMP ya mencionados en el capítulo 2.

- **Agente de software > NMS**: la comunicación se la realiza mediante un puente, debido a que el Sistema Multiagente y el NMS son desarrollados bajo dos lenguajes de programación muy distintos que son JAVA y PHP.

Las metodologías son aplicadas de la siguiente manera:

Para el modelado del Sistema Multiagente (figura 16.1) se hace uso de la metodología MAS-CommonKads.

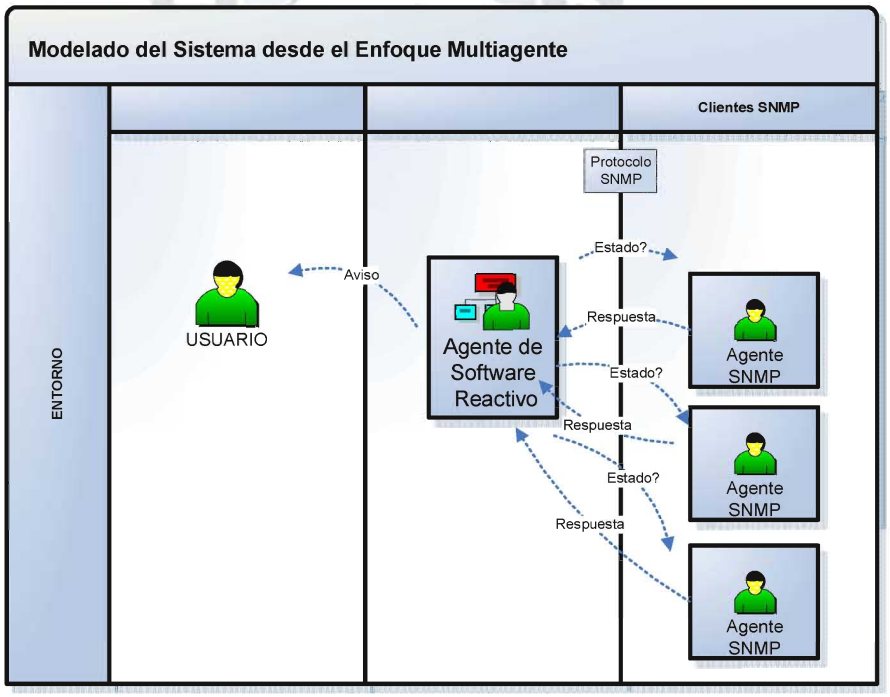

Figura 19: Vista del Sistema desde el enfoque Multiagente.
Fuente: [Elaboración Propia].

Para el modelado del Sistema basado en SNMP (figura 16.2) se hace uso de la metodología XP.

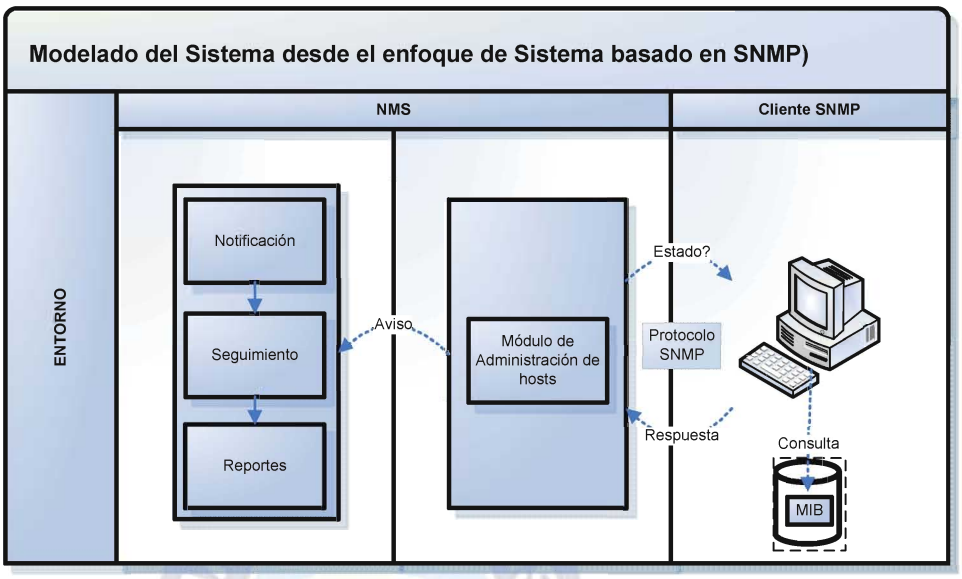

Figura 20: Vista del Sistema desde el enfoque SNMP

Fuente: [Elaboración Propia].

El resultado final de la implementación de ambas metodologías es el prototipo propuesto como solución al problema planteado:

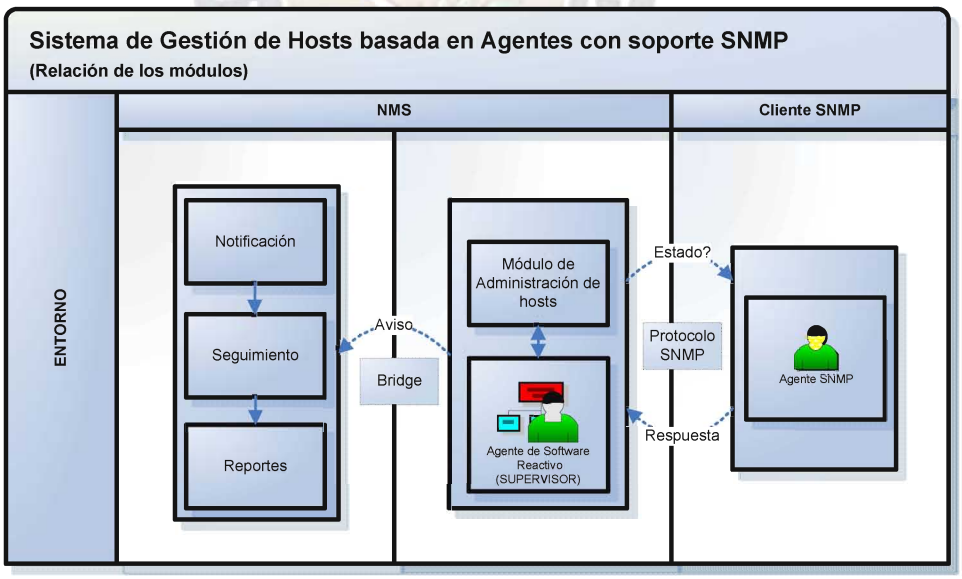

Figura 21: Esquema del prototipo planteado.

Fuente: [Elaboración Propia].

3.3 MODELADO DEL SISTEMA DESDE EL ENFOQUE MULTIAGENTE CON MAS-CommonKADS

3.3.1 CONCEPTUALIZACIÓN

Durante esta fase se lleva a cabo tareas de identificación y organización de la información, para obtener una descripción preliminar del problema. Estas tareas se las realizan determinando algunos casos del uso (escenarios) que ayuden a entender requisitos informales.

FASE DE CONCEPTUALIZACIÓN
(Ciclo de desarrollo de casos de uso)

► DESCRIPCIÓN DE ACTIVIDADES

- **Actividad: Identificación de los actores**

  El SMA orientado a la gestión de hosts está basado en tres tipos de agentes:

  - Agente Supervisor
  - Agente SNMP
  - Agente Usuario

- **Actividad: Descripción de los actores**

  Nombre ***Agente Supervisor***: encargado de recolectar información de los hosts gestionados para evaluarlos y así tomar una decisión ya sea con la ayuda de un experto o de manera autónoma.

  Nombre ***Agente SNMP:*** encargado de proporcionar información del host gestionado y enviarlo al NMS.

  Nombre ***Agente Usuario:*** encargado de colaborar con el agente supervisor para la realización de una tarea o en la toma de decisiones.

- **Identificación de los casos de uso**

El sistema cuenta con 4 casos de uso.

- **Reportar errores:**

  Se reporta problemas capturados en el host gestionado de forma inmediata al NMS.

- **Realizar diagnóstico:**

  Se realiza el diagnóstico del error reportado y se toma una decisión si la solución es conocida.

- **Notificar problema:**

  Se realiza la notificación al usuario para la toma de decisiones y para la generación de un historial.

- **Solicitar sugerencia:**

  Se solicita la sugerencia del usuario para tomar una decisión ante el problema de un host.

- **Notación de los casos de uso**

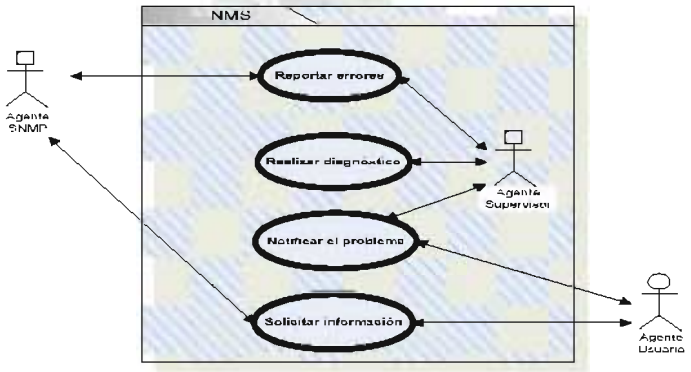

Figura 22: Diagramas de Casos de Uso del Sistema.

Fuente: [Elaboración Propia].

- **Descripción de los casos de uso**

  - **Caso de uso "Reportar errores"**

Actores	Agente SNMP, Agente Supervisor
Descripción	Se detecta un error en el host gestionado y se lo reporta de inmediato.
Precondición	Debe de existir por lo menos un agente SNMP habilitado.
Excepción	No se toma en cuenta errores para los que el agente SNMP no fue configurado.

Tabla 5.1 Descripción del Caso de Uso "Reportar errores"

Fuente: [(**Pérez,2007**),Elaboración Propia]

  - **Caso de uso "Realizar diagnóstico"**

Actores	Agente Supervisor
Descripción	Se identifica la presencia de un mensaje de error y se la evalúa para una posible solución.
Precondición	El mensaje debe de estar bajo el protocolo SNMP como forma de comunicación.
Excepción	No se evalúan mensajes no entendibles.

Tabla 5.2 Descripción del Caso de Uso "Realizar diagnóstico"

Fuente: [(**Pérez,2007**),Elaboración Propia]

  - **Caso de uso "Notificar el problema"**

Actores	Agente Supervisor, Agente Usuario
Descripción	Se reporta problema y se almacena al historial.
Precondición	El agente supervisor debe conocer al menos a un agente usuario.
Excepción	No se hace seguimiento de las notificaciones

Tabla 5.3 Descripción del Caso de Uso "Notificar el problema"

Fuente: [(**Pérez,2007**),Elaboración Propia]

  - **Caso de uso "Solicitar información"**

Actores	Agente Usuario, Agente SNMP
Descripción	El agente Usuario solicita información a un agente SNMP específico para saber el estado del host que supervisa.
Precondición	El agente SNMP debe estar habilitado para la consulta y

	debidamente configurado.
Excepción	El agente usuario no puede lanzar nuevas consultas que escapen a la configuración del agente SNMP.

Tabla 5.4 Descripción del Caso de Uso "Solicitar información"

Fuente: [(**Pérez,2007),**Elaboración Propia]

Tabla 5 Fase de Conceptualización

Fuente: [(**Pérez,2007),**Elaboración Propia]

3.3.2 ANÁLISIS

La etapa de análisis corresponde a la segunda fase de la metodología de desarrollo en la cual se desarrollan los modelos de agente, tareas, comunicación, coordinación e inteligencia, con el fin de obtener las especificaciones y requerimientos del SMA (**Pérez,2007**).

3.3.2.1 MODELO DE AGENTES

Ya identificados los agentes en la fase anterior tenemos:

- Agente Supervisor
- Agente SNMP
- Agente Usuario

MODELO DE AGENTES
(Fase de Análisis)

Definimos ahora el modelo de los agentes a través de las tarjetas CRC (Clase / Responsabilidad / Colaboración).

AGENTE: Supervisor			**CLASE:** Software	
Objetivo	**Planes**	**Conocimiento**	**Colaborador**	**Servicio**
Supervisar a todos los agentes	Oír todo lo reportado		Agente SNMP	

SNMP	por los agentes SNMP.			
Identificar qué tipo de solución se aplicara	Analizar el tipo de error reportado.	Qué solución es la más apropiada para el error.	Agente Usuario	Identificar tipo de solución para el error.
Notificar el problema	Se notifica según el grado de riesgo a un tipo de usuario especifico	Cuál es el tipo de usuario correcto según el grado de riesgo		Notificar el problema y decidir el tipo de usuario

Tabla 6.1 Tarjeta CRC para el Agente Supervisor

Fuente: [(**Pérez,2007**),Elaboración Propia]

AGENTE: SNMP			**CLASE:** Software	
Objetivo	**Planes**	**Conocimiento**	**Colaborador**	**Servicio**
Reportar errores en el host gestionado	El agente vive como un servicio dentro del Sistema operativo atento a los errores que	Cuáles son los servicios y/o componentes verificados		Reportar errores en el host gestionado

	se puedan generar			

Tabla 6.2 Tarjeta CRC para el Agente SNMP

Fuente: [(**Pérez,2007**),Elaboración Propia]

AGENTE: Usuario			**CLASE:** Humano	
Objetivo	**Planes**	**Conocimiento**	**Colaborador**	**Servicio**
Brindar información de errores comunes y sus posibles soluciones	Dar información de los errores más comunes y su posible tratamiento			Colaborar en el aprendizaje del agente supervisor
Solicitar el estado de un host especifico			Agente SNMP	

Tabla 6.3 Tarjeta CRC para el Agente Usuario

Fuente: [(**Pérez,2007**),Elaboración Propia]

- **DESCRIPCIÓN DE AGENTES**

Agente Supervisor

Tipo: Agente software
Capacidades-razonamiento

Experiencia:

Conocimiento de las soluciones a ciertos tipos de problemas y reporte de dicho problema a un tipo de usuario especifico.

Descripción:

Este agente al informarse del problema decide qué tipo de solución aplicará, en el caso de que no disponga de la información de la posible solución solicita la ayuda del usuario, en ambos casos se realiza la notificación para salvar la información como antecedente.

- ***Agente SNMP***

Tipo: Agente software

Experiencia:

Conocimiento de los servicios, componentes a gestionar y de la forma de comunicar la existencia de algún problema de estos.

Descripción:

Este agente basado en el radio de observación que posee pone en aviso del problema presentado al agente supervisor, también brinda una información global del estado del host gestionado a solicitud del agente usuario.

- ***Agente Usuario***

Tipo: Agente humano

Experiencia:

Conocimiento de las soluciones a ciertos tipos de problemas que el agente supervisor desconoce.

Descripción:

Este agente es el encargado de apoyar al agente supervisor en la en la búsqueda de soluciones, además es capaz de solicitar al agente SNMP un informe global del estado del host gestionado por este.

Tabla 6 Modelo Agentes
Fuente: [**(Pérez,2007),**Elaboración Propia]

3.3.2.2 MODELO DE TAREAS

El modelo de tareas permite describir las actividades relacionadas para alcanzar un objetivo. El objetivo del desarrollo del modelo de tareas es documentar la situación actual y futura de la organización, facilitar la gestión de cambios, y ayudar a estudiar el alcance y viabilidad del sistema inteligente que se desea desarrollar.

Para la descripción de una tarea se incluirá:

- Su nombre
- Objetivo
- Estado de control : entrada y salida de parámetros
- Condiciones: de activación y finalización
- Descripción

MODELO DE TAREAS
(Fase de Análisis)

Las tareas identificadas del SMA son:

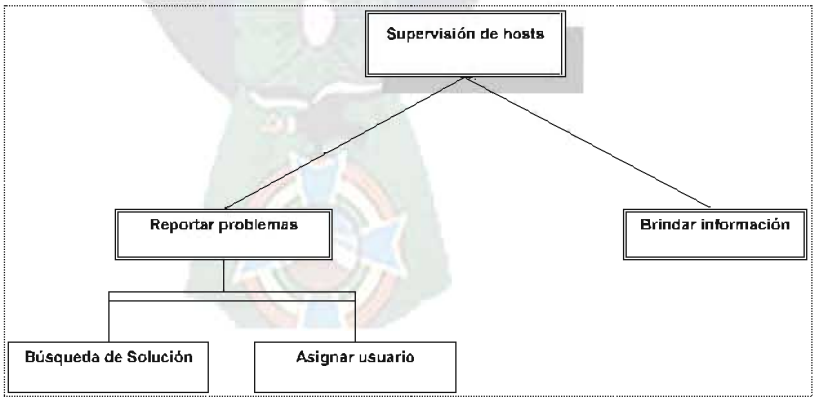

Figura 23: Tareas del Sistema de Gestión de Host.
Fuente: [Elaboración Propia].

- **DESCRIPCIÓN DE TAREAS**

Tarea	Reportar problemas
Objetivo	Decidir qué solución se dará y a que usuario se derivará el problema.
Parámetros de entrada	Problema reportado por el agente SNMP.
Parámetros de salida	Solución y tipo de usuario a notificar.
Condición de activación	Al nacer el agente y recibir un Trap (mensaje de problema).
Condición de finalización	Al dar solución al problema.
Descripción	En esta tarea se decide qué solución se dará y a quien se notificara un problema.

Tabla 7.1: Descripción de Tarea: Reportar problemas.
Fuente: [Elaboración Propia].

Tarea	Brindar información
Objetivo	Recabar información de los servicios y/o componentes gestionados y publicarlos.
Parámetros de entrada	Solicitud de información.
Parámetros de salida	Detalle del estado de lo supervisado.
Condición de activación	Al detectar una solicitud.
Condición de finalización	Al desplegar la información de los solicitado.

Descripción	En esta tarea se da respuesta a una solitud de información de un host específico.

Tabla 7.2: Descripción de Tarea: Brindar información.

Fuente: [Elaboración Propia].

Tarea	Búsqueda de solución
Objetivo	Decidir qué tipo de solución le conviene al problema reportado.
Parámetros de entrada	Trap (mensaje de problema).
Parámetros de salida	Respuesta al problema.
Condición de activación	Al recibir el Trap.
Condición de finalización	Al dar una solución al Trap.
Descripción	En esta tarea se busca dar una solución a un problema reportado ya sea de manera automática o asistida.

Tabla 7.3: Descripción de Tarea: Búsqueda de solución.

Fuente: [Elaboración Propia].

Tarea	Asignar usuario
Objetivo	Decidir a qué tipo de usuario se asignara el problema reportado.
Parámetros de entrada	Grado del problema.
Parámetros de salida	Tipo de usuario correspondiente.
Condición de activación	Al recibir la solicitud de un usuario.

Condición de finalización	Usuario asignado.
Descripción	En esta tarea se busca asignar un tipo de usuario según el grado de riesgo que el problema implique.

Tabla 7.4: Descripción de Tarea: Asignar usuario.
Fuente: [Elaboración Propia].

Tabla 7: Modelo de Tareas.
Fuente: [Elaboración Propia].

3.3.2.3 MODELO DE COORDINACIÓN

Con el modelo de coordinación se describe el esquema de comunicación del SMA; es decir, la conversación, los protocolos y los lenguajes asociados **(Pérez,2007)**.

MODELO DE COORDINACIÓN
(Fase de Análisis)

- **IDENTIFICACIÓN DE LAS CONVERSACIONES**

  Se lleva a cabo con la ayuda de las tarjetas CRC (3.3 de este capítulo) donde ya se identificaron algunas conversaciones.

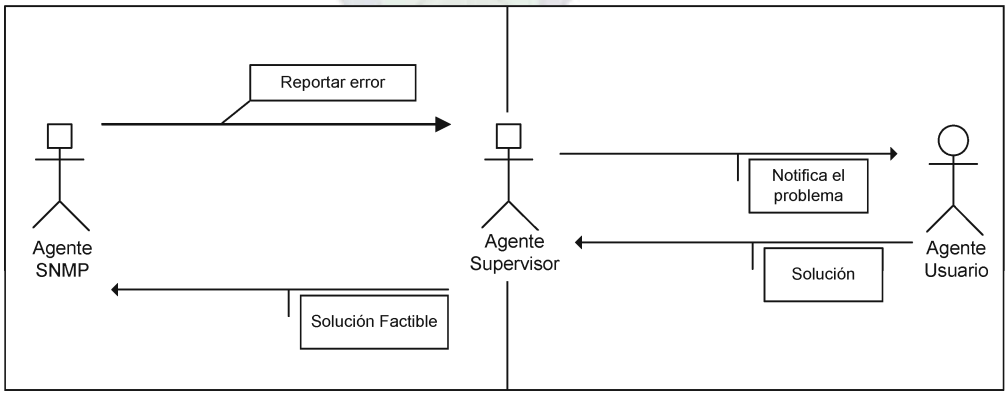

Figura 24: Conversación en el reporte de errores.
Fuente: [Elaboración Propia].

- **DESCRIPCIÓN DE LAS CONVERSACIONES**

La descripción de las conversaciones se realiza desde dos puntos de vista **(Pérez,2007)**:

• ***Externo:*** analizamos cuál es el objetivo de la conversación, su pre y pos condiciones, y qué participantes hay.

• ***Estructural***: qué fases tiene la conversación y qué intervenciones se dan en cada fase.
La conversación de los agentes es lineal, solo se interesa en hacer llegar el mensaje correspondiente a su destino.

CONVERSACIÓN	Reportar error
Tipo	Entrega de información
Objetivo	Informar de un error ocurrido en el host gestionado
Agentes	Agente SNMP, Agente Supervisor
Iniciador	Agente SNMP
Servicio	Error ocurrido
Precondición	Al menos un agente SNMP activo y configurado
Pos condición	Acuse de recibo
Condición - terminación	Devolución de una decisión
Tiempo ejecución	Lance inmediato
Descripción	El agente SNMP anuncia al agente supervisor de la presencia de un error en host gestionado

Tabla 8.1: Descripción de la conversación: Reportar error.

Fuente: [Elaboración Propia].

CONVERSACIÓN	Notifica el problema
Tipo	Entrega de información
Objetivo	Informar de un problema ocurrido en un host gestionado
Agentes	Agente Supervisor, Agente Usuario
Iniciador	Agente Supervisor
Servicio	Error ocurrido
Precondición	Tipo de usuario al que se notificara
Pos condición	Decisión tomada
Condición - terminación	Devolución de una decisión
Tiempo ejecución	Lance inmediato
Descripción	El agente supervisor anuncia al agente usuario de la presencia de un error en host gestionado y al cual ya se dio solución o en el caso de que no pueda hacerlo pide su colaboración en la toma de una decisión.

Tabla 8.2: Descripción de la conversación: Notifica el problema.

Fuente: [Elaboración Propia].

CONVERSACIÓN	Solución
Tipo	Entrega de información
Objetivo	Informar de la decisión tomada
Agentes	Agente Usuario, Agente Supervisor
Iniciador	Agente Usuario
Servicio	Decisión tomada
Precondición	Agente supervisor activo
Pos condición	Solución enviada al agente SNMP
Condición - terminación	Devolución de una decisión
Tiempo ejecución	Lance inmediato
Descripción	El agente usuario toma la mejor solución

	ante el problema reportado y devuelve dicha decisión al agente supervisor.

Tabla 8.3: Descripción de la conversación: Solución.

Fuente: [Elaboración Propia].

CONVERSACIÓN	Solución factible
Tipo	Entrega de información
Objetivo	Informar de la decisión tomada
Agentes	Agente Supervisor, Agente SNMP
Iniciador	Agente supervisor
Servicio	Decisión tomada
Precondición	Al menos un agente SNMP activo y configurado
Pos condición	Acuse de recibo
Condición - terminación	Devolución de una decisión
Tiempo ejecución	Lance inmediato
Descripción	El agente supervisor informa de la decisión tomada al agente SNMP.

Tabla 8.4: Descripción de la conversación: Solución factible.

Fuente: [Elaboración Propia].

- **DETERMINACIÓN DE GRUPOS DE AGENTES**

  Se determinan los grupos aparecidos en la identificación de las conversaciones:

Grupo	Agentes SNMP
Tipo	Individual, dinámico
Componentes	Agente SNMP
Servicio ofrecido	Traps (reporte de errores)

Descripción	Supervisan los servicios y/o componentes descritos en su configuración

Tabla 8.5: Descripción del grupo de agentes SNMP.

Fuente: [Elaboración Propia].

Tabla 8: Modelo de Coordinación.

Fuente: [Elaboración Propia].

3.3.2.4 MODELO DE EXPERIENCIA

MODELO DE EXPERIENCIA
(Fase de Análisis)

El agente propuesto como supervisor es sencillo, por lo que no es susceptible a la experiencia, pues estará guiado por grados de riesgo para la toma de decisiones, grados de riesgo que son configurados como reglas por el usuario.

En cuanto al agente SNMP solo reacciona ante eventos ya programados, por lo tanto la experiencia no aplica para este agente.

Ambos agentes tienen una ontología básica pues al ser el dominio compuesto por ontologías y nuestras ontologías se basan en reglas predefinidas, hacen que el dominio ya este establecido, los agentes SNMP ya conocen de la existencia del agente supervisor y del conocimiento de la aplicación **(Pérez,2007,pg 100).**

Representación	Grados de riesgo
Tipo	De acuerdo al problema
Grado de confiabilidad	Es preciso ya que las reglas ya están definidas
Esquema de procesamiento	No cambia con el tiempo

Tabla 9.1: Cuadro de análisis de la Experiencia.

Fuente: [Elaboración Propia].

Mecanismo de Aprendizaje: "No aplica a este ejemplo pues no lo utiliza."

Mecanismo de Razonamiento: "Racionamiento estrictamente basado en reglas predefinidas"

Tabla 9: Modelo de Experiencia.
Fuente: [Elaboración Propia].

3.3.2.5 MODELO DE ORGANIZACIÓN

MODELO DE ORGANIZACIÓN
(Fase de Análisis)

El modelo de organización de MAS-CommonKADS tiene como objetivo analizar desde una perspectiva de grupo las relaciones entre los agentes (tanto software como humanos) que interactúan con el sistema **(Pérez,2007,pg 119).**

Condiciones iniciales:

El diseño del agente Supervisor, toma como ambiente de interacción un entorno de red cuya comunicación está basada en el protocolo SNMP con el agente SNMP, agente SNMP que ya está diseñado y solo se hace su reutilización con distinto enfoque. Como tal el agente supervisor se diseña para ser utilizado por un Usuario especialista en la administración de redes.

Es así que el agente supervisor no pertenece a ningún otro contexto organizacional de operación o de comunicación entre agentes, lo cual se convierte en precondición para el diseño del mismo.

- **DIAGRAMAS DE RELACIÓN DE CASOS REACTIVOS**

  - **DETECCIÓN DE ERRORES EN HOST GESTIONADO**

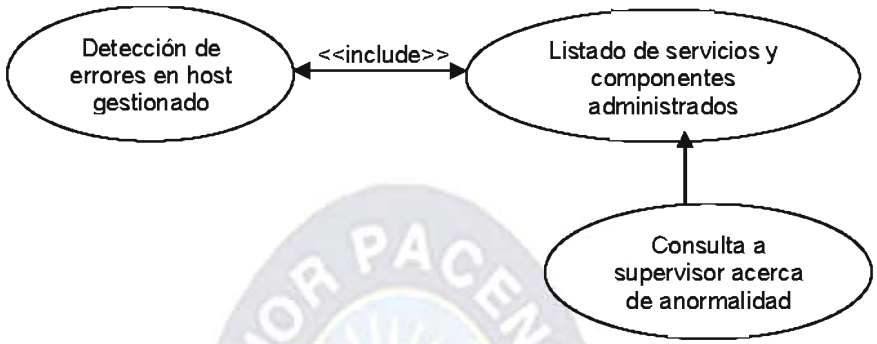

Figura 25: Caso reactivo para la detección de errores en host gestionado.
Fuente: [Elaboración Propia].

  - **NOTIFICACIÓN DE PROBLEMA**

Figura 26: Caso reactivo para la notificación de problema.
Fuente: [Elaboración Propia].

Tabla 10: Modelo de Organización.
Fuente: [Elaboración Propia].

3.3.2.6 MODELO DE COMUNICACIÓN

MODELO DE COMUNICACIÓN (Fase de Análisis)
En este modelo se reflejan las transacciones de comunicación entre los diferentes agentes involucrados en el sistema. Esto se hace en una forma

conceptual independiente de la implementación, como con el modelo de conocimientos. Su objetivo es especificar los procedimientos de intercambio de información para realizar la transferencia del conocimiento entre los agentes. Es decir, que en este modelo se especifican las necesidades y deseos en relación con las interfaces que se tiene con los agentes, así es posible tener una interfaz con el usuario y una interfaz con un sistema software **(Iglesias,1998)**.

Acto de habla:	**Reporte de error**
Objetivo	Reportar un error producido en el host gestionado
Tipo	Asertivo **(Iglesias,1998)**.
Comunicación	Directa
Agentes participantes	Agente SNMP, Agente supervisor
Emisor	Agente SNMP
Receptor	Agente supervisor
Conversación	Gestión de host
Servicio	Reporte de error
Datos de intercambio	
Descripción	El agente SNMP detecta un error en el host gestionado y reporta dicho error al agente supervisor
Precondición	Ambos agentes deben estar en servicio
Condición de terminación	Acuse de recibo
Per formativa	Reportar
Medio de comunicación	Protocolo SNMP

Tabla 11.1 Acto de habla reporte de error

Fuente: [Elaboración Propia]

Acto de habla:	Notificación de problema
Objetivo	Notificar de un error a un tipo de usuario especifico
Tipo	Directivo **(Iglesias,1998)**.
Comunicación	Directa
Agentes participantes	Agente supervisor, agente usuario
Emisor	Agente supervisor
Receptor	Agente usuario
Conversación	Notificacion de problema
Servicio	Asignar un problema para su solución
Datos de intercambio	
Descripción	El agente supervisor notifica al usuario mediante el inicio de un caso.
Precondición	El tipo usuario debe de existir
Condición de terminación	Asignación del caso al usuario
Per formativa	Notificar
Medio de comunicación	Información anexada en caso

Tabla 11.2 Acto de habla reporte de error

Fuente: [Elaboración Propia]

Tabla 11: Modelo de Comunicación.

Fuente: [Elaboración Propia].

3.3.3 DISEÑO

Para la utilización se deberá compilar los componentes del agente supervisor, el proceso se realiza de la siguiente manera:

1) C:\> javac Nms.java

2) C:\> java jade.Boot –agents supervisor:Nms

El agente corriendo se muestra en el GUI de JADE:

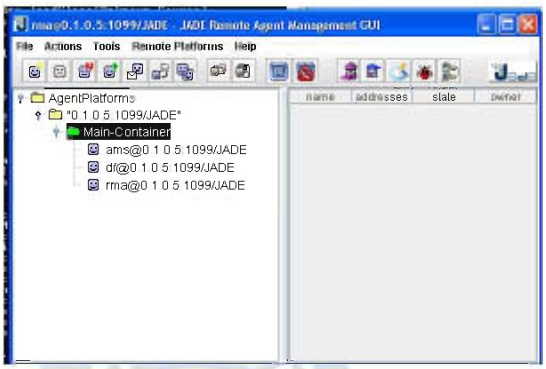

Figura 20 Inicialización del agente supervisor

Fuente: [Elaboración Propia]

La interfaz de interacción con el usuario donde se indica la dirección IP del host gestionado:

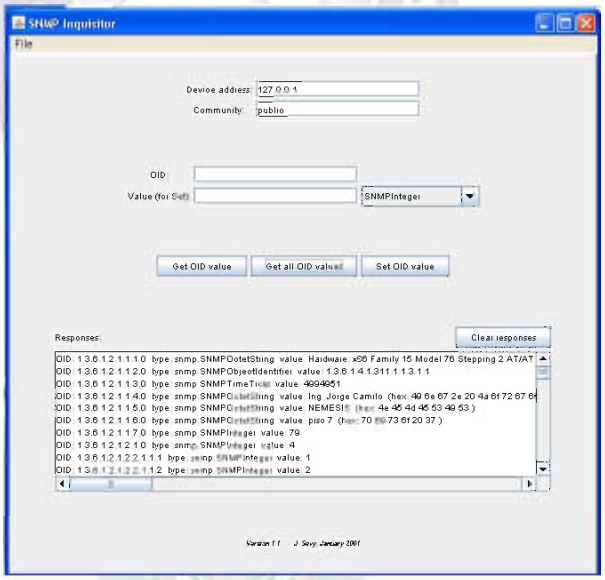

Figura 21 interfaz de usuario

Fuente: [Elaboración Propia]

El esquema de red que se posee para el agente supervisor y el NMS es el siguiente:

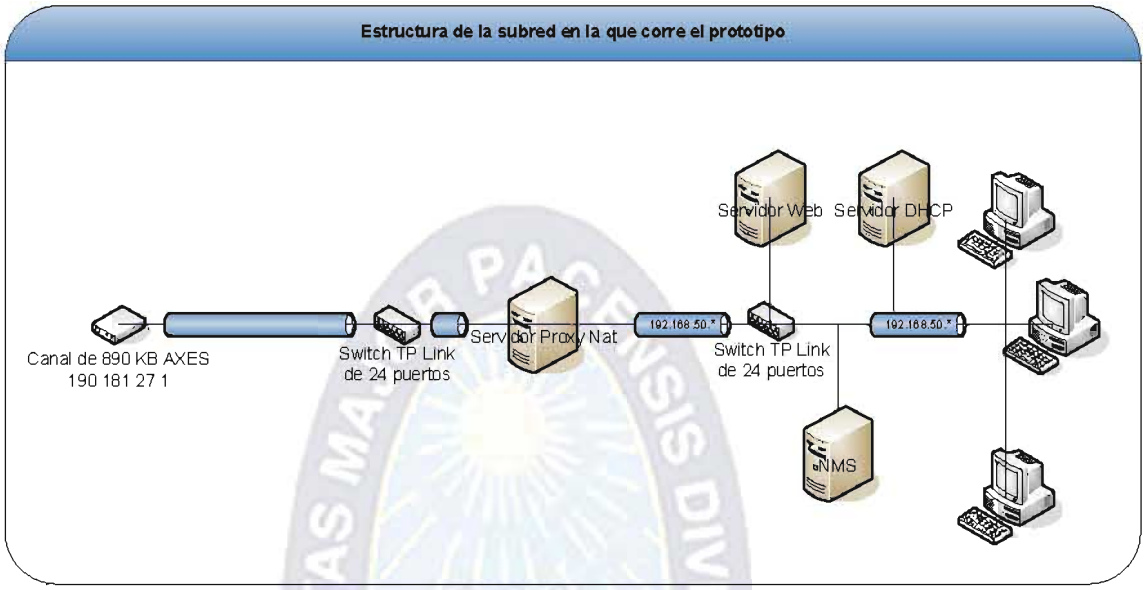

Figura 22 Estructura de Red en la que corre el prototipo.

Fuente: [Elaboración Propia]

Dicho esquema pertenece a una subred con la que se estructura la red del CEPIES (Centro de postgrado en educación superior de la Universidad Mayor de San Andrés).

3.4 MODELADO DEL SISTEMA DESDE EL ENFOQUE DE LA INGENIERÍA DEL SOFTWARE CON LA METODOLOGÍA DE LA PROGRAMACIÓN EXTREMA (XP)

3.4.1 FASE DE EXPLORACIÓN

Esta primera fase es equivalente a una fase de análisis en una metodología tradicional. En esta fase se identifican los problemas objetivamente para elaborar un diseño general de forma simple y ágil, se define el equipo de trabajo con el que se desarrolla el sistema de administración de red, se definen las historias de usuario que esbozaran de gran manera la funcionalidad inicial del sistema, se elabora la metáfora con la que el sistema será conocido por todo el equipo para una fácil conceptualización del mismo, después se escoge y diseña la arquitectura que el sistema tendrá, también se elige el conjunto de herramientas tecnológicas que se utilizaran para el desarrollo del sistema.

FASE DE EXPLORACIÓN
(Programación Extrema)

► DEFINICIÓN DEL EQUIPO

En lo que respecta al equipo de desarrollo no aplica por tratarse de una tesis de grado, ya que todo el trabajo es desarrollado por el programador.

► DECLARACIÓN DE HISTORIAS DE USUARIO

Las historias de usuario que se muestran a continuación, son el fruto de experiencia adquirida en el trabajo con usuarios dentro del área de redes.

Historia de Usuario	**Nº 1**	**CO2**
Nombre: ver estado de cada PC		**Prioridad**
Descripción: Cada cierto tiempo se realiza el censo de todos los equipos para saber su ubicación, usuario asignado y estado.		
Estimación:	**Dependencia:**	

Historia de Usuario	Nº 2	CO2
Nombre: Notificación por email de errores		**Prioridad**
Descripción: Se notificación por email de errores que necesiten de la intervención del administrador de red guardando el numero de notificación para su seguimiento, además se debe obtener la fecha y hora de la notificación.		
Estimación:	**Dependencia:**	

Historia de Usuario	Nº 3	CO2
Nombre: resumen y estadísticas		**Prioridad**
Descripción: Se realizan resúmenes y estadísticas de forma manual como reporte mensual para saber el número de casos atendidos.		
Estimación:	**Dependencia:**	

Historia de Usuario	Nº 4	CO2
Nombre: Seguimiento de casos		**Prioridad**
Descripción: Se hace el seguimiento de los casos mediante formularios simples, que sirven para respaldo de los usuarios por nivel de usuario.		
Estimación:	**Dependencia:**	

Historia de Usuario	Nº 5	CO2
Nombre: Atención a casos de bajo riesgo		**Prioridad**
Descripción: La notificación a problemas menores es lo más común como falta de espacio en disco y falta de espacio en RAM que distraen de problemas más críticos que se presentan en la administración de red.		
Estimación:	**Dependencia:**	

Tabla 11.1 Descripción de las Historias de usuarios

Fuente: [Elaboración Propia]

► METÁFORA

La metáfora es el nexo de comprensión entre el programador y el cliente, es usada para la descripción conceptual del sistema, se la define como se muestra en la tabla11.2.

Metáfora del sistema
"El funcionamiento de un equipo de soporte de red y técnico, donde cada miembro necesite atender y seguir un caso reportado, además de obtener el estado de las PCs que se administran, también poder proponer soluciones a problemas de bajo riesgo".

Tabla 11.2 Metáfora del sistema
Fuente: [Elaboración Propia]

El glosario de términos es determinado para una comprensión adecuada.

- Caso reportado: notificación realizado por el NMS al usuario experto para su atención y solución.
- Administrar: llamado así a la supervisión de cada equipo de forma remota.
- Proponer soluciones: almacenamiento de posibles acciones ante problemas comunes y de bajo riesgo.
- Equipo: personal (usuario) experto en la atención de problemas técnicos y de red.

Tabla 11 Fase de Exploración
Fuente: [Elaboración Propia]

3.4.2 FASE DE PLANIFICACIÓN

En esta fase se ordenan las historias de usuario según el grado de importancia para el usuario, de acuerdo a esto se establecen tiempos para desarrollarlas, y obtener un producto en el cual el cliente ve pequeñas funcionalidades ya cumplidas.

FASE DE PLANIFICACIÓN
(Programación Extrema)

► ESTIMACIÓN DEL TIEMPO DE DESARROLLO

Esta fase se irá repitiendo a lo largo del proyecto retroalimentándose del final de cada iteración, debido a esto existirá una planificación para cada iteración en la evolución del sistema, se mostrara el proceso de la primera iteración.

Historias de Usuario – 1er Plan de Entregas	
Historias de Usuario	**Estimación de semanas**
Administración remota	2 semanas
Notificación de errores	1semanas
Reportes	2semanas
Proceso del seguimiento de casos	1semanas
Respuesta automática a problemas	3semanas

Tabla 12.1 Estimación tiempo desarrollo para historias de usuario
1º plan de entregas
Fuente: [Elaboración Propia]

El tiempo estimado para el desarrollo del prototipo es de 3 meses atendiendo la totalidad de las historias de usuarios debido a que es solo un prototipo y no un producto final.

Tabla 12 Fase de Planificación
Fuente: [Elaboración Propia]

3.4.3 FASE DE ITERACIÓN

Esta fase es iterativa, las entradas de esta fase son el plan de iteración y los test de aceptación fallidos, que a través de las reuniones diarias se asignan a un grupo de programadores, estos empiezan a desarrollar los módulos a partir de la historia de usuario y las tarjetas CRC asignadas. Para el caso de la presente tesis solo existe en el grupo un programador.

FASE DE ITERACIÓN
(Programación Extrema)

A continuación se describe las iteraciones que contemplo el sistema de administración de hosts.

Nro de iteración	Historias de usuario involucradas	Test Aceptados	Test Fallidos	Retroalimentación	Número de semanas
1	HU1 HU2 HU3	HU1	 HU2 HU3	 HU2 HU3	2-semanas 1-semanas 2-semanas
2	HU2 HU3 HU4	HU2	 HU3	 HU3 HU4	1-semanas 1-semanas 1-semanas
3	HU3 HU4 HU5	HU3 HU4 HU5			1-semanas 1-semanas 3-semanas
Total semanas					13-semas

Tabla 13.1 Iteraciones en la elaboración del sistema
Fuente: [Elaboración Propia]

Tabla 13 Fase de Iteración
Fuente: [Elaboración Propia]

3.4.4 FASE DE PRODUCCIÓN

La base de datos del Sistema de administración de red se ve enfocado en cuatro tablas, en las cuales se almacena la información de cada host, las incidencias que estos tuvieron y en el caso de reportar algún problema se almacena también por quien fue atendido.

Estas tablas son muy independientes de lo que representa el seguimiento de casos y formularios usados por la herramienta BPM usada.

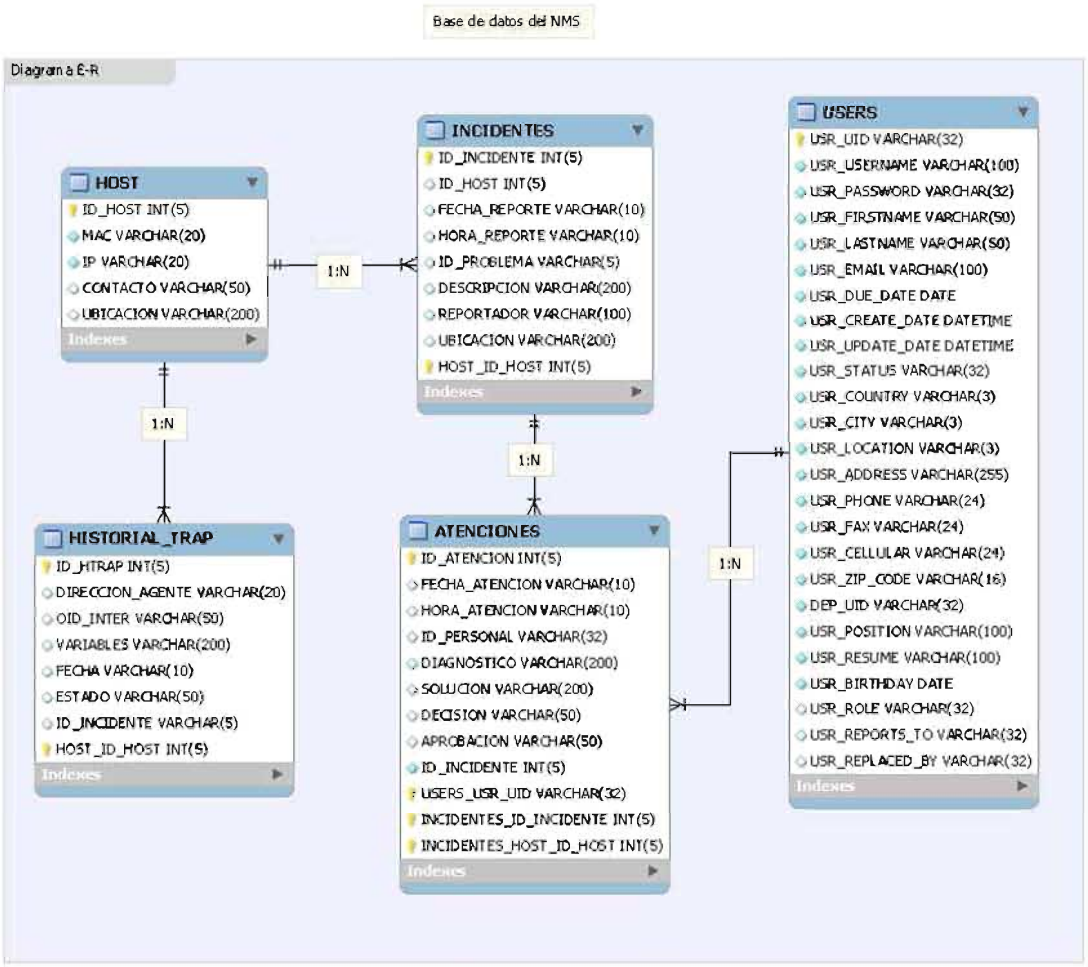

Figura 23: Diagrama E-R de la base de datos del NMS

Fuente: [Elaboración Propia].

En cuanto a la estructura del NMS podemos ver que cuenta con cuatro módulos cada uno con una función específica que ayudan a la interacción con el usuario y el agente, para que de esta manera se vea reflejada los requerimientos planteados en las historias de usuarios.

En la figura 24 se puede ver el orden de los módulos y la descripción de cada uno de ellos que son:

- Casos
- Procesos
- Administración y
- Reportes

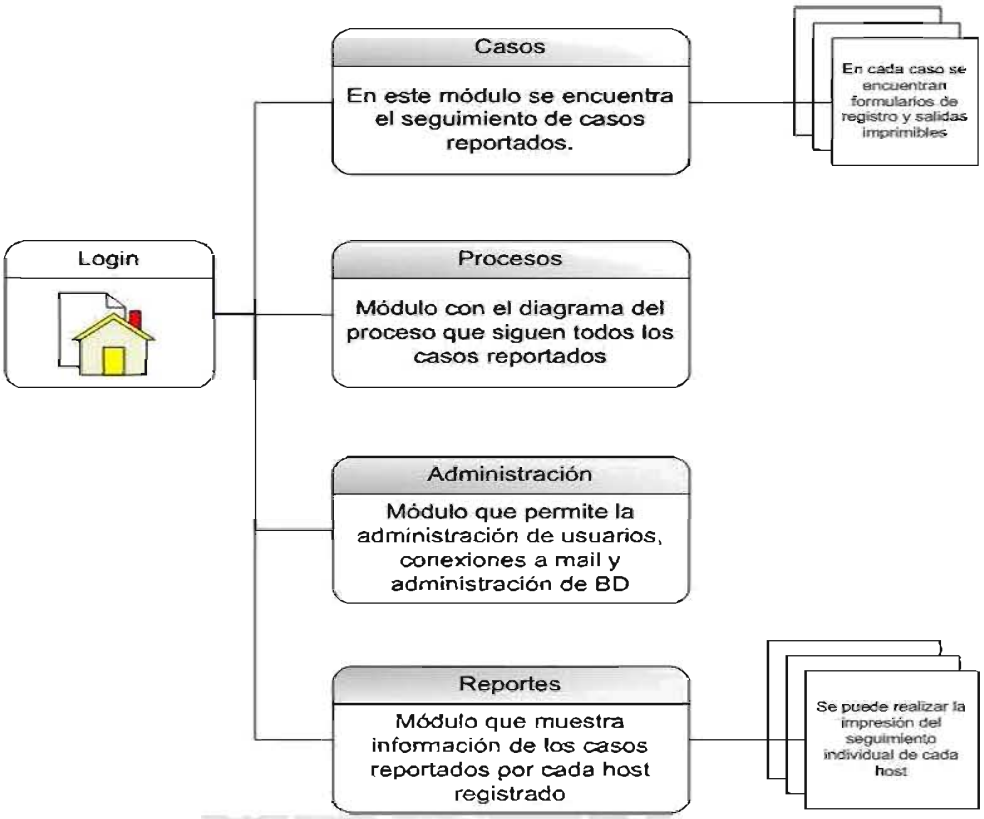

Figura 24: Diagrama web conceptual

Fuente: [Elaboración Propia].

El proceso que sigue cada caso se lo ve en la figura 25 donde se pueden identificar cuatro grupos de usuarios participantes que son:

- Usuarios/agente supervisor.
- Soporte técnico.
- Administración de red.
- Jefe de departamento.

Estos usuarios participan en seis tareas que llegan a ser el proceso que se sigue ante la notificación de cada problema.

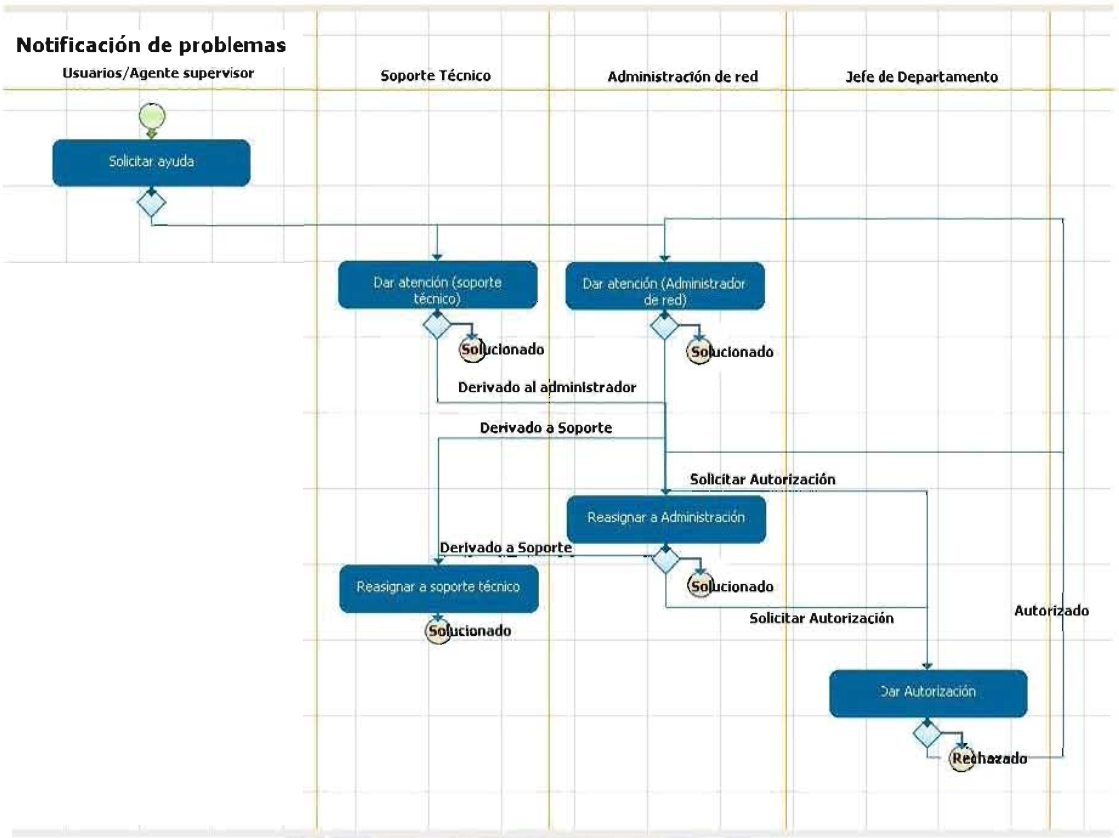

Figura 25: Proceso de la Notificación de Problemas

Fuente: [Elaboración Propia].

Los resultados obtenidos en el desarrollo del sistema se los muestran en la figura 26, estos resultados son fruto de 3 ciclos de iteraciones y aproximadamente 13 semanas, el área clara indica el porcentaje de fallas de test de los módulos y el área oscura de las aprobadas.

Se observa en la figura 26 que a partir de la séptima semana no existen fallos en las pruebas de aceptación, esto indica que a partir de la 2 iteración el desarrollo del prototipo se vuelve más estable.

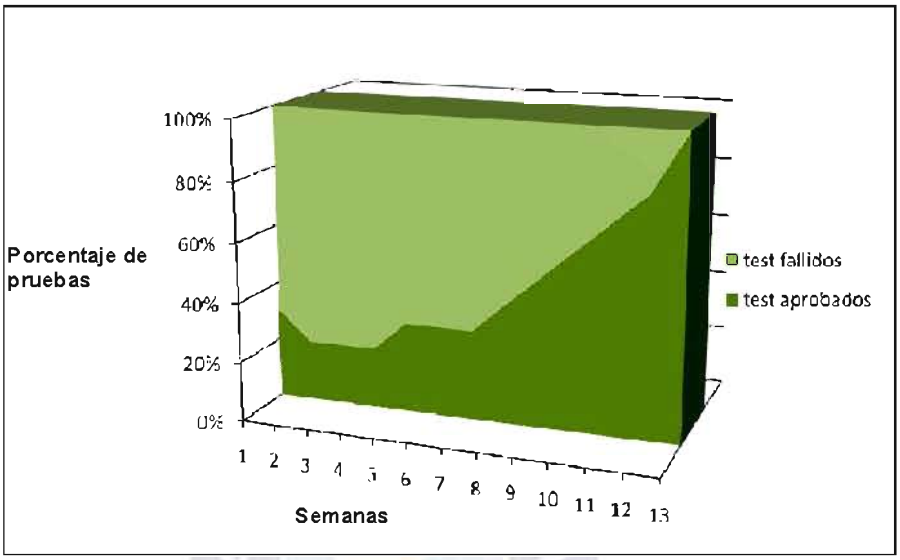

Figura 26: Resultado de unidades de Test.

Fuente: [Elaboración Propia].

3.5 MODELO FORMAL Y CONFRONTACIÓN DE HIPÓTESIS

3.5.1 MODELO FORMAL PARA LA VERIFICACIÓN DE HIPÓTESIS

Siendo la Hipótesis planteada:

"La automatización de la supervisión del comportamiento de los hosts mediante agentes optimizará la administración de PCS en una red LAN y reducirá la caída de servicios".

Se pueden identificar las siguientes variables:

t_n = Tiempo de Notificación de un problema.

t_s =Tiempo de solución a un problema.

t_a = Tiempo de notificación automatizado por agentes.

Pc_x = Hosts del dominio de red x=1,2,3,….n;

K = en una constante de la suma de Pc_x atendidos tomados al azar.

Donde:

T_{NA} = Tiempo total en la atención de problemas en host atendidos de forma no automatizada.

T_A = Tiempo total en la atención de problemas en host atendidos de forma automatizada.

C_{s1} = Número de caídas de servicio sin la automatización de la administración

C_{s2}= Número de caídas de servicio con la automatización de la administración

$$T_{NA} = K \sum_{i=1}^{i=k} (tn_i + ts_i) \quad ;$$

$$T_A = K \sum_{i=1}^{i=k} (ta_i + ts_i) \quad ;$$

Tenemos:

$$(T_{NA} > T_A) \rightarrow (C_{S2} < C_{S1}).$$

3.5.2 CONFRONTACIÓN DE HIPÓTESIS

Las pruebas se realizaron en ambientes del Centro Psicopedagógico de Investigación en Estudios Superiores (CE.P.I.E.S.), de la Universidad Mayor de San Andrés, trabajando bajo el esquema de red mostrado en la figura 22.

La gestión de hosts tomó como grupo de prueba a los siguientes nodos:

Descripción	Cantidad
PCs de escritorio	27
Servidor Web	1
Servidor Proxy Nat	1
Servidor DHCP	1

Tabla 14: Equipos Gestionados

Fuente: [Elaboración Propia]

Las pruebas se realizaron por un periodo de 6 meses, 3 de los cuales se trabajo en la obtención de datos del grupo supervisado en carencia de un sistema de gestión de hosts, obteniéndose los siguientes resultados:

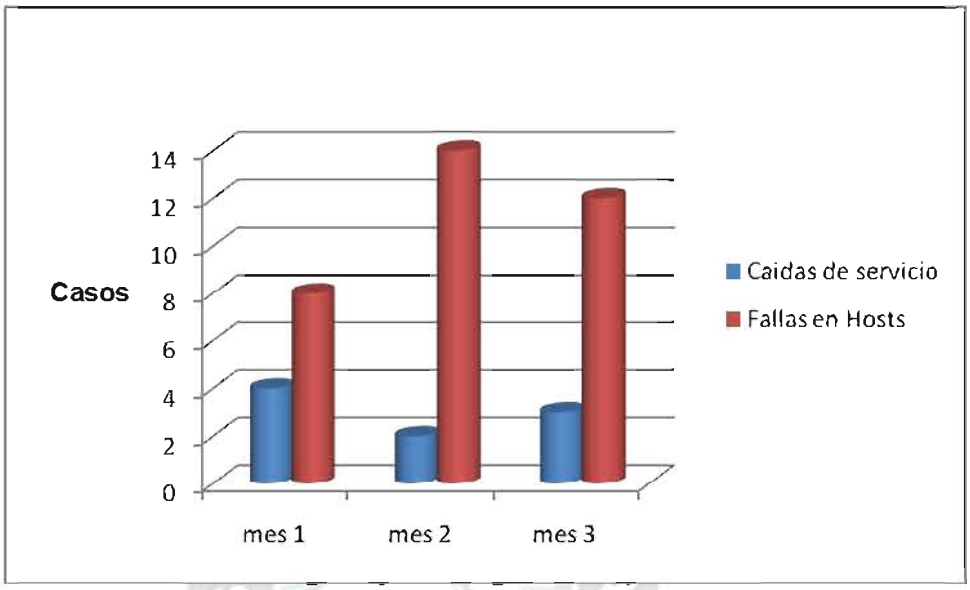

Figura 27: Problemas reportados.
Fuente: [Elaboración Propia].

Se obtuvo la siguiente información:

	mes 1	mes 2	mes 3	Totales
Caídas de servicio	Casos: 7	Casos: 6	Casos: 3	Casos: 16
Fallas en Hosts	5 horas	10 horas	12 horas	27 horas

Tabla 15: Datos obtenidos sin el prototipo
Fuente: [Elaboración Propia]

Podemos observar:

$T_{NA=}$ 30 * 27 Horas = 810 Horas

C_{s1} = 16

En el testeo del prototipo durante los 3 meses siguientes, se notó una reducción notable de problemas, debido a la antelación con que se atienden los

mismos, ya que la asignación de casos es rápida y los problemas pequeños son solucionados con prontitud, dichos resultados se los muestra en la figura 24.

Se obtuvo la siguiente información:

	mes 1	mes 2	mes 3	Totales
Caídas de servicio	Casos: 2	Casos: 1	Casos: 0	Casos: 3
Fallas en Hosts	5 horas	3 horas	3 horas	11 horas

Tabla 16: Datos obtenidos con el prototipo

Fuente: [Elaboración Propia]

Podemos observar un total de 11 horas.

$T_{A=}$ 30 * 11 Horas = 330 Horas

$C_{s2} = 3$

De tal manera podemos probar que:

$$(T_{NA} > T_A) \rightarrow (C_{S2} < C_{S1})$$

La reducción de problemas en los servicios y hosts alcanza aproximadamente a un 70 %.

La atencion de casos reportados se lo realiza de manera mas rápida debido al seguimiento via web de cada caso.

Se mostró que el acoplamiento de un agente en un Sistema de Gestion de hosts es funcional y que el protocolo SNMP es una herramienta de gran ayuda para cualquier tarea de administracion de redes.

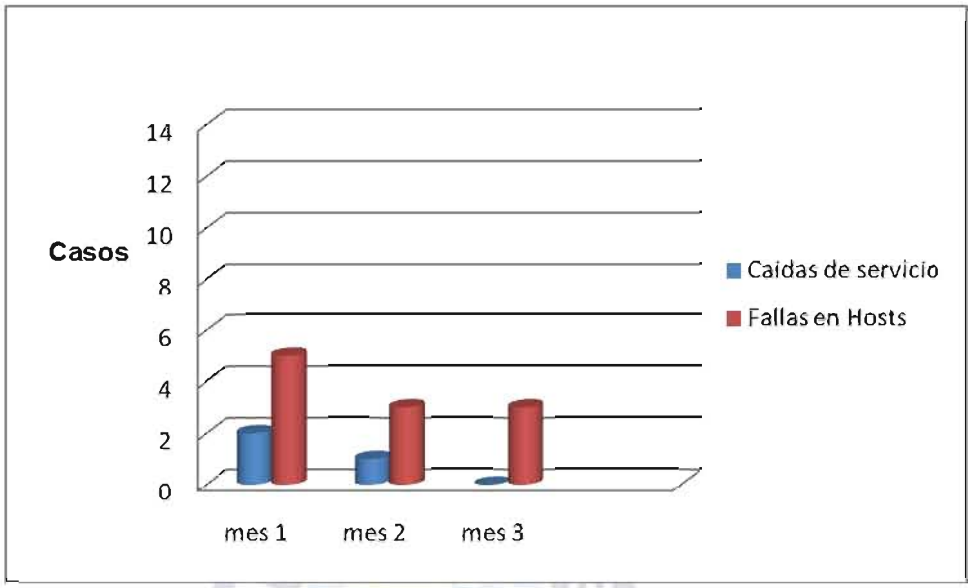

Figura 28: Problemas reportados.
Fuente: [Elaboración Propia].

El objetivo planteado juega un papel importante pues son los pasos que influyeron en la reducción del tiempo en la atención a problemas y la posterior verificación de la hipótesis.

Capítulo 4

Conclusiones y Recomendaciones

4.1 CONCLUSIONES

A la finalización del trabajo de investigación emprendido, se puede mencionar como conclusiones generales las siguientes:

La administración de redes es un área muy compleja, debido a su intervención en el hardware y software de la arquitectura de una red, lo que hace que el administrador de red deba tener amplios conocimientos en estas dos ramas o contar con el personal adecuado para realizar las tareas que se presentan dentro la administración.

La prevención y atención inmediata a los problemas originados en una red hace que los riesgos de problemas más serios de tratar se reduzcan.

El uso de agentes para la detección de problemas en hosts hace que la gestión de redes se trabaje de forma distribuida, produciendo de esta manera información exacta y alertas tempranas a errores presentados.

Los administradores de red no solo deben gestionar una infraestructura cada vez más compleja, sino también cumplir con los objetivos de equipamiento de red, que deben ser monitorizados para detectar fallos, analizar los cuellos de botella.

La organización en los diferentes equipos que intervienen en una administración de red juega un papel muy importante para mejorar los tiempos de atención a problemas.

El protocolo SNMP es una herramienta muy útil, la cual puede ser usada para distintos propósitos, como es el caso de la presente tesis se lo uso para la comunicación entre agente.

4.2 RECOMENDACIONES

Un tema muy importante es el uso de agentes móviles para la administración de redes, que podría ser propuesto en tesis futuras ya que el uso de dos agentes aumenta el uso de recursos en el servidor y el cliente y el uso de un agente móvil disminuiría el uso de recursos.

El modelo propuesto puede ser extendido a la supervisión de switchs, impresoras y otros dispositivos administrables para escalar lo propuesto en la presente tesis.

El uso de herramientas tales como Jess para JAVA son el paso siguiente para dotar de conocimiento a los agentes.

Para tener una base y continuar con este tema de investigación que es demasiado interesante dejo la dirección de mi blog que es:

http://iniciandoconocimiento.blogspot.com/2011/10/sistema-de-gestion-de-hosts-basado-en.html

En el podrán encontrar información de este tema de investigación, las herramientas y la bibliografía usada para la elaboración de la presente tesis.

Anexos

ANEXO 1

Código del agente supervisor hecho en jade

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package nmsagent;

/**
 *
 * @author Fenix
 */
import java.util.Properties;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.mail.Message;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import java.awt.Desktop;
import java.net.URI;
import jade.core.Agent;
import jade.core.behaviours.CyclicBehaviour;
import java.io.*;
import snmp.*;


public class NMSAgent extends Agent{


   protected void setup(){
       try
      {
         SNMPInquisitor theApp = new SNMPInquisitor();
         theApp.pack();
         theApp.setSize(600,500);
         theApp.show();
      }
      catch (Exception e)
      {}
      MyCyclicBehaviour c = new MyCyclicBehaviour();
      addBehaviour(c);
   }

   protected void takeDown(){
      System.out.println("Ejecucion finalizada-El Agente supervisor desconectado");
   }
```

```
   private class MyCyclicBehaviour extends CyclicBehaviour
                           implements SNMPv1TrapListener, SNMPv2TrapListener,
SNMPv2InformRequestListener, Runnable{
      //Variables SNMP
      SNMPTrapReceiverInterface trapReceiverInterface;
      PipedReader errorReader;
      PipedWriter errorWriter;
      Thread readerThread;
      String msg="Agente Supervisor Iniciado -Listo para la supervision!";

      //Comportamiento ciclico
      public MyCyclicBehaviour()
      {
         try
         {
            errorReader = new PipedReader();
            errorWriter = new PipedWriter(errorReader);

            readerThread = new Thread(this);
            readerThread.start();

            trapReceiverInterface = new SNMPTrapReceiverInterface(new
PrintWriter(errorWriter));
            trapReceiverInterface.addv1TrapListener(this);
            trapReceiverInterface.addv2TrapListener(this);
            trapReceiverInterface.addv2InformRequestListener(this);
            trapReceiverInterface.startReceiving();

         }
         catch(Exception e)
         {
           System.out.println("Problemas al iniciar el testeo de Traps: " + e.toString() +
"\n");
         }
      }

      //Accion a realizar en el comportamiento
      public void action() {
            if(msg != null)
            {
                  System.out.println(msg);
                     if(msg!="Agente Supervisor Iniciado -Listo para la supervision!")
                     {
                        sendMail();

                        try {
                           iniciaCaso();
                        } catch (Exception ex) {

Logger.getLogger(NMSAgent.class.getName()).log(Level.SEVERE, null, ex);
```

```
                    }
                }
                msg=null;
            }
            else{
                block();
            }
        }

    public void iniciaCaso()throws Exception{
        Desktop.getDesktop().browse(new
URI("http://192.168.1.11/sysco2/es/afore/7028955204ec998cbbc5ad0074548205/inicia
CasoAgente.php"));
    }

    public void sendMail()
    {
        try {
            Properties props = new Properties();
            props.setProperty("mail.smtp.host", "smtp.gmail.com");
            props.setProperty("mail.smtp.starttls.enable", "true");
            props.setProperty("mail.smtp.port", "587");
            props.setProperty("mail.smtp.user", "coquis39@gmail.com");
            props.setProperty("mail.smtp.auth", "true");

            System.out.println("Enviando...");

            // Preparamos la sesion
            Session session = Session.getDefaultInstance(props);
            //session.setDebug(true);

            // Construimos el mensaje
            MimeMessage message = new MimeMessage(session);
            message.setFrom(new InternetAddress("coquis39@gmail.com"));
            message.addRecipient(Message.RecipientType.TO,new
InternetAddress("coquis39@hotmail.com"));
            message.setSubject("Reporte del agente supervisor");
            //message.setText("Se ha iniciado un caso debido a un problema
detectado...");
            message.setText(
            "Hola soy el agente " + "<b>supervisor</b> <br> He detectado un problema
en un host con IP:<b>192.168.1.1</b>" + " tuve que iniciar un caso.<br> Le solicito
revisarlo saludos.",
            "ISO-8859-1",
            "html");
            // Lo enviamos.
            Transport t = session.getTransport("smtp");
            t.connect("coquis39@gmail.com", "6090578lp");
            t.sendMessage(message, message.getAllRecipients());

            // Cierre.
```

```
            t.close();

            System.out.println("\n\nEmail Enviado!");
            }
            catch (Exception e)
            {
                  //e.printStackTrace();
               System.out.println("Servicio de correo no detectado, no se envio el mail");
            }
      }
      //Esta funcion recive los traps de version 1 de SNMP
      public void processv1Trap(SNMPv1TrapPDU pdu, String communityName)
      {
         msg="Got v1 trap:\n";
         msg=msg+"  nombre de comunidad:     " + communityName + "\n";
         msg=msg+"  enterprise OID:     " + pdu.getEnterpriseOID().toString() + "\n";
         msg=msg+"  dirección de agent:      " + pdu.getAgentAddress().toString() + "\n";
         msg=msg+"  trap generic:       " + pdu.getGenericTrap() + "\n";
         msg=msg+"  trap específico:      " + pdu.getSpecificTrap() + "\n";
         msg=msg+"  timestamp:          " + pdu.getTimestamp() + "\n";
         msg=msg+"  variables suplementarias: " + pdu.getVarBindList().toString() +
"\n"+"\n";
         restart();
      }


      //Esta funcion recive los traps de version 2 de SNMP
      public void processv2Trap(SNMPv2TrapPDU pdu, String communityName)
      {
        msg="Got v2 trap:\n";

        msg=msg+"  nombre de comunidad:     " + communityName + "\n";
        msg=msg+"  system uptime:      " + pdu.getSysUptime().toString() + "\n";
        msg=msg+"  trap OID:           " + pdu.getSNMPTrapOID().toString() + "\n";
        msg=msg+"  var bind list:      " + pdu.getVarBindList().toString() + "\n";

        msg=msg+"\n";
        restart();
      }

       //Esta funcion recive los traps de version 2 de SNMP de tipo informe q fueron
solicitados
      public void processv2InformRequest(SNMPv2InformRequestPDU pdu, String
communityName)
      {
        msg=msg+"Got v2 inform request:\n";

        msg=msg+"  nombre de comunidad:     " + communityName + "\n";
        msg=msg+"  system uptime:      " + pdu.getSysUptime().toString() + "\n";
        msg=msg+"  trap OID:           " + pdu.getSNMPTrapOID().toString() + "\n";
        msg=msg+"  var bind list:      " + pdu.getVarBindList().toString() + "\n";
```

```
            msg=msg+"\n";
            restart();
        }

        @Override
        public void run() {
            //throw new UnsupportedOperationException("Not supported yet.");
        }
    }

}
```

ANEXO 2

Pantallas del agente ejecutándose

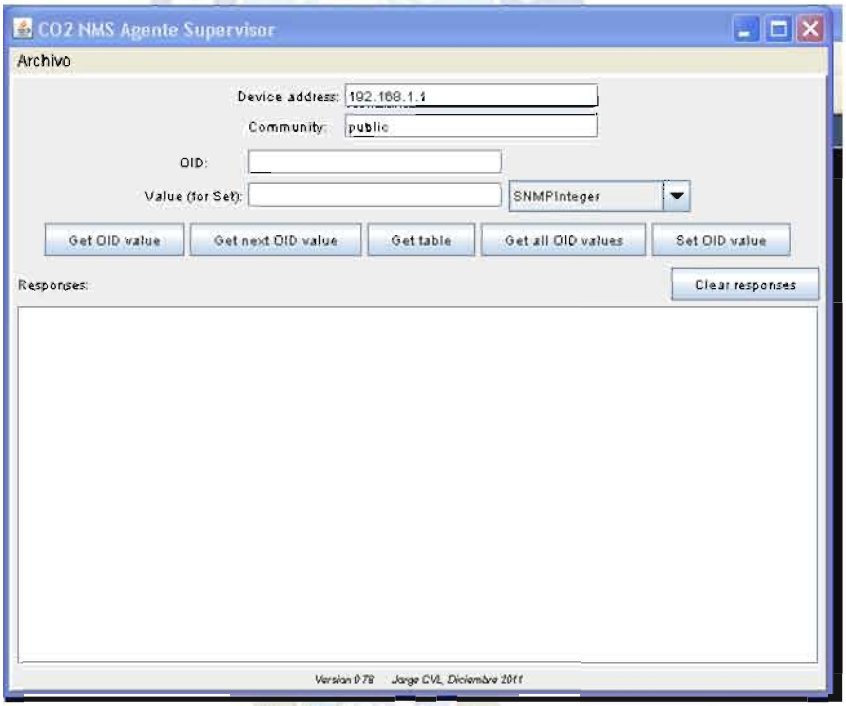

Figura 29: Pantalla del agente supervisor.

Fuente: [Elaboración Propia].

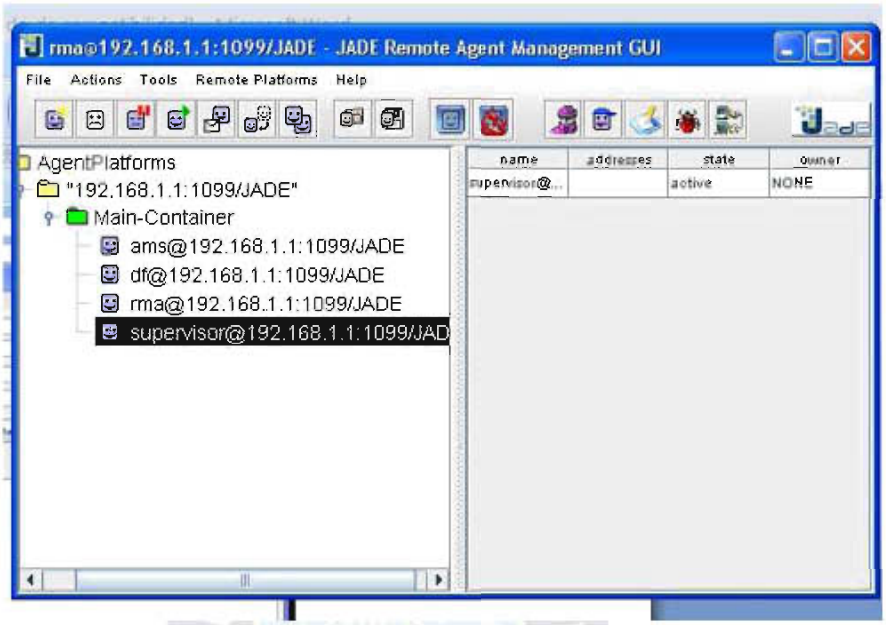

Figura 30: Agente supervisor en Jade.

Fuente: [Elaboración Propia].

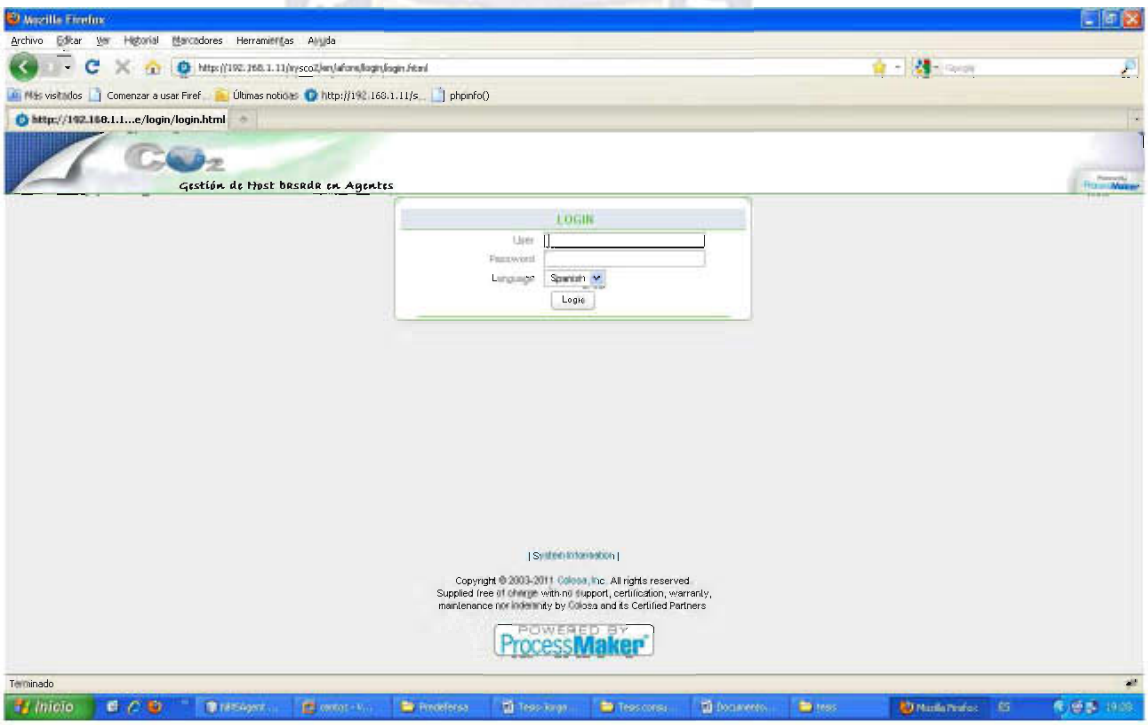

Figura 31: Login del NMS.

Fuente: [Elaboración Propia].

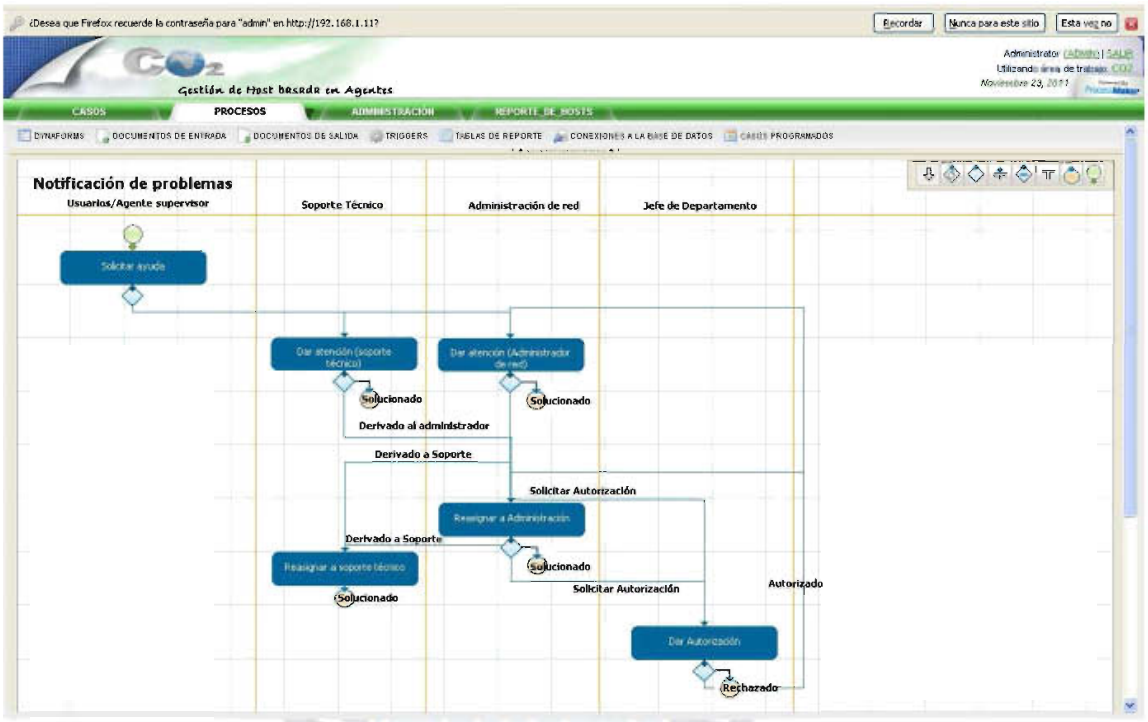

Figura 32: Proceso de reporte de errores.

Fuente: [Elaboración Propia].

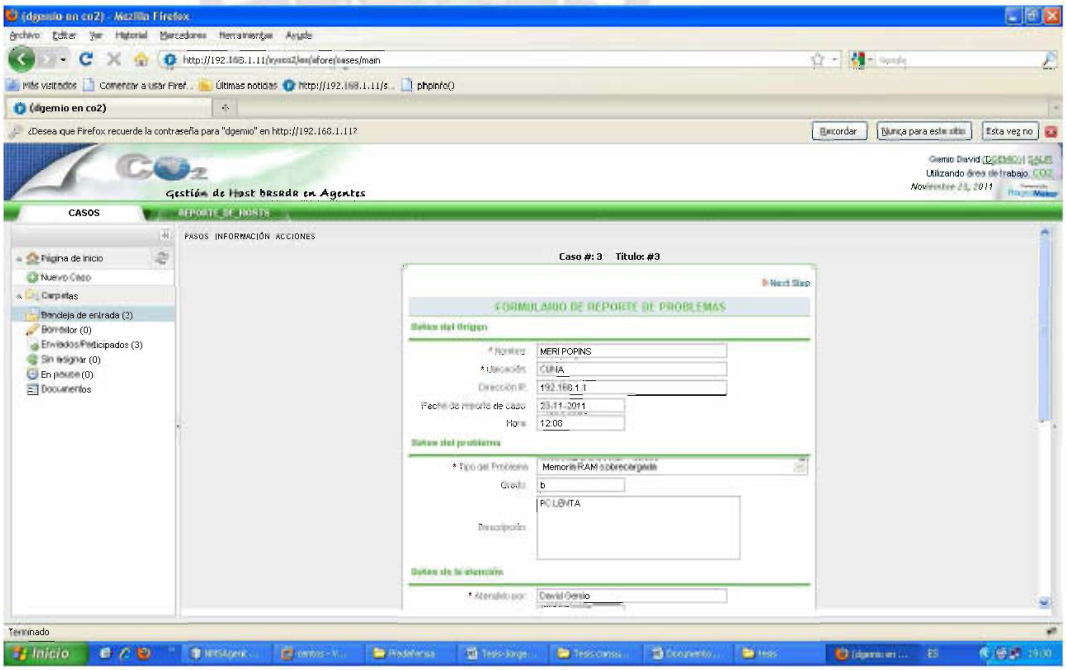

Figura 33: Formulario de reporte.

Fuente: [Elaboración Propia].

Bibliografía

REFERENCIA BIBLIOGRÁFICA

(Ambler ,2002) Ambler S. "Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process".

(Beck ,2000) Kent Beck, "Extreme Programming Explained", Addison Wesley, 2000,
ISBN: 201-61641-6

(Canos J,2003) Canos J., Letelier P., Penadés C. Metodologías Ágiles en el Desarrollo de Software.[12/11/03].
Disponible en:
http://issi.dsic.upv.es/publications/archives/f-1069167248521/actas.pdf
(Ultimo acceso 05/05/2011).

(Choque G.,2005) Choque G. Diseño ágil y programación extrema. Universidad Mayor de San Andrés,
2005.

(Enrich M,2003) Enrich M. Crystal Metodologies. [04/02/03].
Disponible en:
www.dsic.upv.es/asignaturas/facultad/lsi/trabajos/282002.ppt
(Ultimo acceso 18/04/2011).

(Palacio J,2007) Palacio J. Gestión de proyectos ágil: conceptos Básicos. [sf]. Disponible en:
http://www.nevegapolis.net/files/s/NST-003 01.pdf
(Ultimo acceso 01/07/2011).

(Nwana,1996) Nwana H. S., (1996), Software Agents: An Overview, AA&T, BT Laboratories.

(Santacruz,2005) Santacruz V.L..P., (2005) Agentes Taxonomia y Aplicaciones, Universidad Carlos III de Madrid.
Disponible en:
www.lite.etsii.urjc.es/liliana/agentes.pdf
(Ultimo acceso 21-09-2011).

(Molina,Garcia, Bernardo, 2004) Molina, L. J. M., García H. J., Bernardos B. A. M. , (2004), Agentes y Sistemas Multiagente, CEDITEC.
Disponible en:
http://www.ceditec.etsit.upm.es/index.php?option=com_docman&task=ddo_details&qid=3&Itemid=78&lang=es
(Ultimo acceso 21-09-2011).

(Wooldridge, 2002) Wooldridge M., (2002), An Introduction to Multiagent Systems, John Wiley & Sons Ltd

(Russell,Norvig, 1995) Russell, S. J. and Norvig P., "Artificial Intelligence: A Modern Approach", Englewood Cliffs, NJ: Prentice Hall, 1995

(Mauro,Schidt,2005)	DOUGLAS Mauro, Kevin Schmidt, 2005: "Essential SNMP, 2nd Edition".
(Bellifemine,Caire, Greenwood,2007)	FABIO Bellifemine, Giovanni Caire, Dominic Greenwood, 2007: "Developing Multi-Agent Systems with JADE".
(Weiss,1999)	GERHARD Weiss , 1999: "Multiagent Systems A Modern Approach to Distributed Modern Approach to Artificial Intelligence".
(Caire,2009)	GIOVANNI Caire, 2009: "Jade Programming for Beginners".
(Pan,1998)	HENG Pan, 1998: "SNMP-Based ATM Network Management".
(Deguchi, Yokohama, 2005)	Hiroshi Deguchi, Yokohama, Japan, 2005: "Agent-Based Social SystemsVolume 2".
(Bigus,2004)	Dr. JOSEPH P. Bigus, 2004: "Constructing Intelligent Agents With Java".
(Chli,Wilde,2009)	MARIA Chli, Philippe de Wilde, 2009: "Convergence and Knowledge Processing in Multi-Agent Systems".
(Fariba, Thielscher,2008)	MICHAEL Fisher Fariba Sadri, Michael Thielscher (Eds.), 2008: "Computational Logic in Multi-Agent Systems".
(Bordini, Dix, Dastani,2009)	RAFAEL H. Bordini, Jiirgen Dix, Mehdi Dastani, Amal El Fallah Seghrouchni, 2005: "Multi-Agent Programming (*Languages, Platforms and Applications)*".
(Bordini, Dix, Dastani,2005)	RAFAEL H. Bordini, Mehdi Dastani, Jurgen Dix, Amal El Fallah Seghrouchni, 2009: "Multi-Agent Programming".
(Pérez,2007)	Pérez Ardila Yanis Stanley, 2007: "Aplicación de metodologías ingenias, zeus, masina al desarrollo de sistemas multi-agente, partiendo de sma de subastas para la identificación de mejores prácticas".
(Iglesias,1998)	Carlos Ángel Iglesias Fernández, 1998: "Definición de una metodología para el desarrollo de sistemas multiagente".
(Web1)	JOEL Barrios Dueñas, 2006: "Cómo configurar SNMP", Disponible en: http://www.linuxparatodos.net/ (Ultimo acceso 01/2011).

(Web2) Introduccion a SNMP
Disponible en:
http://gicl.cs.drexel.edu/people/sevy/snmp/snmp_package_introduction.html
(Ultimo acceso 12/08/2009).

(Web3) Configuracion de SNMP
Disponible en:
http://web.madritel.es/personales3/edcollado/openview/OV_Introduccion.htm
(Ultimo acceso 10/2010).

UNIVERSITAS MAJOR PACENSIS DIVI ANDRE Æ


UNIVERSITAS MAJOR PACENSIS DIVI ANDRE Æ

Documentos