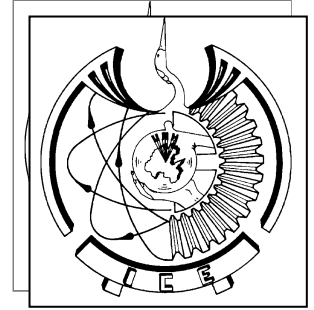




**UNIVERSIDAD AUTÓNOMA
DEL ESTADO DE HIDALGO**

**INSTITUTO DE CIENCIAS BÁSICAS
E INGENIERÍA**

**Centro de Investigación en Tecnologías
de Información y Sistemas (CITIS)**



Tesis Doctoral

**“Elementos básicos para diseño y desarrollo de sistemas de seguridad
basados en agentes para equipos de cómputo
con ambiente Windows-Tecnología NT”**

Presentada por:

María de Guadalupe Cota Ortiz

Director de tesis:

Dr. Joel Suárez Cansino

Pachuca de Soto, Hidalgo, México, Septiembre 2009

“Existe un mundo virtual, que paralelamente al mundo real, ha sido creado por el hombre para poder viajar a través del ciberespacio denominado Internet, donde las actividades del hombre trascienden fronteras en milésimas de segundos, y se requiere de sistemas de protección que aseguren la supervivencia de sus individuos en forma óptima y exitosa.”

Lupita Cota

Dedicatoria:

A mis padres les agradezco profundamente su amor y confianza.

A Pedro, mi esposo, a quien amo, y quien me ha brindado su confianza y apoyo para llegar a ser lo que soy.

A mis hijos y nietos, a quienes amo con todo mi corazón.

Agradecimientos:

A mis maestros y a todos aquellos que hicieron posible la realización de mis estudios de doctorado, y muy especialmente al cuerpo de doctores del CITIS por depositar su confianza en mí.

Al Dr. Joel Suárez, por su sabia dirección.

A mis sinodales, por ser guías en mi trabajo.

A los Doctores: Julio Waissman y Luís Enrique Ramos por su interés y atinada asesoría durante mis estudios de doctorado.

Contenido

Capítulo 1.- Presentación.

1.1 Introducción.....	1
1.2 Descripción de la problemática.....	2
1.3 Descripción del estado del arte.....	7

Capítulo 2.- Elementos básicos para sistemas de seguridad basados en agentes.

2.1 Aspectos de diseño.....	12
2.1.1 Jerarquía de agentes.....	13
2.1.2 Interacción de Agentes.....	14
2.1.3 <i>CLASS-W</i> : Gramática para desarrollo de sistemas de seguridad basados en agentes....	16
2.1.4 Aspectos generales gramaticales de <i>CLASS-W</i>	16
2.1.5 Estructura general de un programa de <i>CLASS-W</i>	18
2.1.6 Plataforma <i>JADE</i>	18
2.1.7 Componentes adicionales.....	18
2.2 Arquitectura de agentes.....	19
2.2.1 <i>BDI</i>	20
2.2.2 <i>GAIA</i>	21
2.2.3 <i>MESSAGE</i>	22
2.2.4 <i>PRS</i>	23
2.2.5 Diseño de la arquitectura <i>ARSEC-AMS</i>	23
2.3 Esquema funcional de agentes.....	23
2.3.1 Roles de agentes.....	24
2.3.2 Esquema interno de comunicación.....	25
2.4 Descripción del comportamiento general de agentes.....	28
2.5 Ejemplos de caso de uso.....	31
2.6 Resumen.....	33

Capítulo 3.- Módulo de Pizarra y Deliberativo-Cognitivo.

3.1 Módulo manejador de pizarra <i>BLACKBOARD-ECRE</i>	35
3.1.1 Aspectos gramaticales de <i>CLASS-W</i> para uso de la pizarra.....	37
3.2 Módulo deliberativo-cognitivo.....	38
3.2.1 Funcionamiento.....	39
3.2.2 Flujo de información.....	40
3.2.3 Representación del conocimiento.....	40
3.2.4 Aspectos gramaticales de <i>CLASS-W</i> para uso de <i>Mod-Logic</i> y reglas generadas con - el algoritmo 2G.....	41
3.2.5 Estructura general de <i>Mod-Logic</i>	42
3.2.6 Etapa de traducción.....	43
3.2.7 Flujo de datos.....	43
3.2.8 Estructura de la base de datos de <i>Mod-Logic</i>	43
3.2.9 Procedimiento de la construcción de plantillas <i>SQL</i>	45
3.3 Resumen.....	50

Capítulo 4.- Módulo de Seguridad y Reactivo.

4.1 Módulo de Seguridad.....	51
4.1.1 Descripción gramatical de <i>CLASS-W</i> para el módulo de seguridad.....	51
4.1.2 Esquema de protección <i>NetPS-ComSA</i>	54
4.1.3 Prueba del procedimiento <i>NetPS-ComSA</i>	64
4.2 Módulo Reactivo.....	66
4.2.1 Descubrimiento de nuevos conocimientos.....	66
4.2.2 Minería de datos.....	67
4.2.3 Reconocimiento de patrones.....	67
4.2.4 Modelo de clasificación del algoritmo <i>2G</i>	70
4.2.5 Implementación del algoritmo <i>2G</i>	75
4.3 Resumen.....	78

Capítulo 5.- Ejemplo de implementación de *SISAG-W*.

5.1 Diseño de implementación de <i>SISAG-W</i>	80
5.2 Diseño de la interfaz visual de <i>Mod-Logic</i> y su funcionamiento.....	80
5.3 Funcionamiento de <i>SISAG-W</i>	86
5.4. Resumen.....	90

Capítulo 6.- Conclusiones.

Conclusiones.....	91
Bibliografía.....	92

Anexo 1

Diagramas <i>AUML</i> correspondientes al ejemplo implementado en <i>SISAG-W</i>	96
--	----

Capítulo I.- Presentación.

1.1 Introducción.

Conforme avanza la tecnología computacional los problemas de seguridad aumentan y se vuelven más complejos de resolver, lo que propicia que tanto particulares como organizaciones públicas o privadas deban buscar nuevas alternativas que los protejan de usuarios malintencionados, que haciendo uso de ciertos recursos, ocasionan daños o tienen acceso a información sin previa autorización, lo que en un momento dado, puede reflejarse en pérdidas financieras que pueden ascender a montos valorados en millones de dólares.

En forma general, se puede decir que existen grandes avances en esta área, y que muchos de los problemas relacionados con virus, troyanos, objetos maliciosos, gusanos, etc., se pueden evitar con software antivirus, sistemas detectores de intrusos, etc., pero éstos hasta la fecha, no son capaces de reconocer patrones de comportamiento anormales que no tengan registrados en su bases de conocimientos, y por lo tanto, no pueden resolver problemas que se presentan en tiempo real (problema del día cero o zero day), que pueden ocasionar pérdidas cuantiosas mientras se encuentra la solución apropiada.

Para controlar este tipo de problemas con un enfoque distinto a los actuales, se propone utilizar elementos básicos para desarrollo de sistemas de seguridad como los que a continuación se mencionan:

- a) Contar con una arquitectura híbrida de agentes orientada al área de seguridad computacional como *ARSEC-AMS*, que toma este nombre por las siglas de ‘Security Architect – Multiagent System’, la cual es una de las aportaciones originales de este trabajo de tesis, en virtud de que entre las existentes no se encontró alguna que cubriera en su totalidad los requerimientos para el desarrollo de sistemas de seguridad basados en agentes. La arquitectura antes mencionada, en su propuesta, incluye entre otras cosas, módulos para implementar mecanismos de comunicación y de razonamiento que permitan detectar situaciones anormales, aun cuando éstas no estén registradas en las bases de conocimientos del sistema, y brinda la posibilidad de dotar a los agentes de capacidad para tomar decisiones e implementar acciones que se reflejen en su entorno.
- b) Utilizar como elementos de comunicación:
 - La gramática ‘*Content Language Security System Agent for Windows*’, que en adelante será referenciada a través de las siglas *CLASS-W*, la cual es un resultado original del trabajo de tesis, y ha sido diseñada para generar lenguajes de contenido, y ser utilizada en el desarrollo de sistemas de seguridad basados en agentes, presentándose en este trabajo una propuesta inicial orientada a equipos de cómputo con ambiente Windows-Tecnología NT.
 - Un módulo de pizarra denominado *BLACKBOARD-ECRE*, que ha sido diseñado sobre un esquema de bases de datos, que permite instanciar a un grupo de temas previamente definidos en la etapa de diseño de sistemas de seguridad basados en esta propuesta.
- c) Aplicar estrategias de control, basadas en un diseño jerárquico relacionado con los tipos de roles de agentes, que permiten especializar la coordinación del trabajo conjunto, la supervisión del comportamiento de una red, y la vigilancia del funcionamiento interno de los equipos que la componen.

- d) Estructurar esquemas de bases de datos que sirvan como apoyo para manipular información relacionada con la gramática *CLASS-W*, políticas, niveles de seguridad, temas, manejo de temas de pizarra, etc.
- e) Diseñar mecanismos de comunicación y razonamiento para asegurar la estabilidad del sistema, presentándose en este trabajo un ejemplo donde se aplican reglas generadas por un árbol de decisión construido con el Algoritmo *2G*, o las contenidas en la base de datos del módulo *Mod-Logic* (Módulo que se utiliza para traducir a una base de datos relacional predicados lógicos tipo Prolog e interpretarlos con plantillas generadas con Structured Query Language (*SQL*), y *NetPS-ComSA* (Network Protection Schema-Communication System of Agents), que se presenta como una propuesta de esquema de protección del sistema de comunicación de agentes, en lo relativo al intercambio de información por red.

Cabe mencionar que no se encontraron referencias bibliográficas sobre la existencia de una gramática como *CLASS-W*, ni sobre el diseño de *Mod-Logic*.

Por otra parte, el algoritmo *2G* se basa en la teoría de reconocimiento de patrones implementada a través de árboles de decisión, que incluye aportaciones originales que otros algoritmos de clasificación del mismo tipo no contemplan, encontrándose resultados muy favorables en comparación con los reportados en la bibliografía revisada para tal efecto, y que en este caso, se utiliza para generar reglas que se incorporan a la base de conocimientos del sistema de agentes para su aplicación.

1.2 Descripción de la problemática.

En la actualidad, los países desarrollados son los más susceptibles a los delitos cibernéticos que se cometen con el fin de corromper, destruir o modificar los datos que se almacenan en sistemas de cómputo o se transmiten a través de redes de computadoras; situación que obliga a adoptar medidas de seguridad que permitan proteger las operaciones que se realizan por este medio, y así, poder garantizar un nivel óptimo de confidencialidad, integridad y disponibilidad de la información [1].

Uno de los ataques típicos de red más graves que se conocen en materia de seguridad computacional, es el causado por gusanos, programas que cuentan con capacidad de reproducirse automáticamente a través de una red de computadoras, ya sea utilizando el servicio de correo electrónico, o a través de recursos compartidos, con el fin de vulnerar y comprometer los equipos donde se alojan. La propagación se realiza por conexiones de red o archivos adjuntos [1].

Entre las pérdidas registradas mundialmente en la última década, contabilizadas en miles de millones de dólares (MMD), está la del año 2002, cuando los costos de reparación por causa de gusanos y virus se estimó en 45 MMD, la de agosto del año 2003, que alcanzó un monto de 38 MMD con un costo anual entre 119 y 145 MMD [2], y las reveladas por analistas de Computer Economics, quienes aseguran que por ‘*malware*’ -palabra que se forma por ‘*malicious software*’ para identificar todo tipo de software dañino-, en el año 2004, las pérdidas causadas por ataques de programas tipo gusano son de 17.5 MMD, de los cuales, 4.5 MMD (21.71%) corresponde al gusano *MyDoom*, y entre 3.5 y 2.75 MMD (17.14%) a los gusanos de tipo *Sasser* o *NetSky* [3], presentando en total un porcentaje muy alto por pérdidas del 38.95%, lo cual provocó gran preocupación y aumentó la necesidad de encontrar otras alternativas de solución a este tipo de problemas (ver Figura 1).

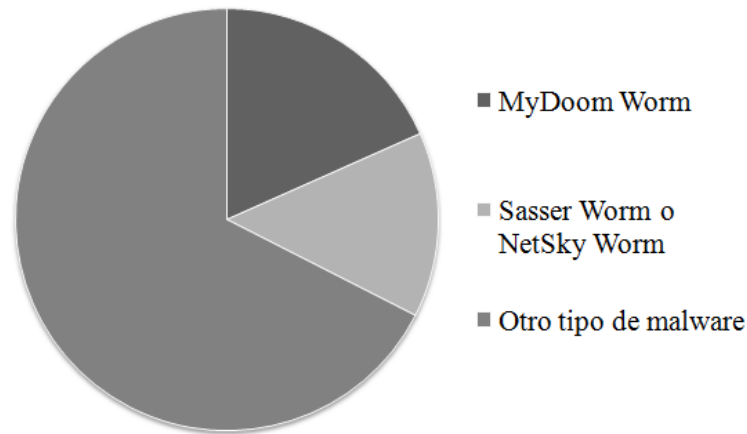


Fig. 1.- Pérdidas por 'malware' en el año 2004.

Una de las amenazas más graves en seguridad computacional, es llamada 'ataque del día cero (*zero day*)', la cual ha ocasionado considerables pérdidas de información y de dinero, y se identifica como el período de tiempo que inicia cuando aparecen virus, códigos maliciosos o gusanos, que al activarse, no pueden ser detectados por el software de protección instalado en los sistemas de cómputo, por no estar registrados en las bases de datos de dichos programas [4]. En el caso de los gusanos, generalmente, tienen como propósito comprometer los recursos de los equipos que han sido afectados y exponerlos a intrusiones o ataques que aprovechan esta situación hasta que el problema es detectado y controlado por personal de seguridad [3].

Cabe señalar, que los problemas relacionados con el 'ataque del día cero', son difíciles de resolver, y pueden ser ocasionados por distintos tipos de 'malware'. Por ejemplo, una de las últimas amenazas detectada el día 12 de diciembre del 2006, que aún no ha podido solucionarse por las distintas empresas que desarrollan software de protección, es el 'Microsoft Word 0-Day Vulnerability III', dirigido a equipos con Windows para explotar una vulnerabilidad de Microsoft Word, que puede permitir la ejecución de código en el equipo infectado por usuarios remotos [5].

El primer incidente de seguridad computacional tipo gusano que causó gran alarma, ocurrió en noviembre de 1988, cuando Robert Morris lanza el primer 'malware' de este tipo en internet para buscar debilidades en sistemas de cómputo, propagándose y contagiando automáticamente a otras computadoras [6]; situación que propició que el Organismo de Proyectos de Investigación Avanzada de Defensa, parte del Instituto de Ingeniería de Programas de la Universidad Carnegie Mellon, en Pittsburg, creara el Centro de Coordinación 'Computer Emergency Response Team', que en adelante se mencionará por las siglas *CERT*, para proporcionar una respuesta rápida a problemas de seguridad, reportándose en la actualidad, que dicho Centro ha manejado más de 260,000 incidentes, ha catalogado y trabajado en la solución de más de 11,000 vulnerabilidades en computadoras y ha publicado una gran cantidad de alertas, constituyéndose en un órgano muy importante en esta área [7].

Como una muestra del alcance de reproducción del 'malware' tipo gusano, en la Figura 2 se puede observar una comparación de la rapidez con que se propagaron en un día los gusanos *Blaster*, *Slammer* y *Code Red*, en 336,000, 55,000 y 265,000 computadoras, respectivamente.

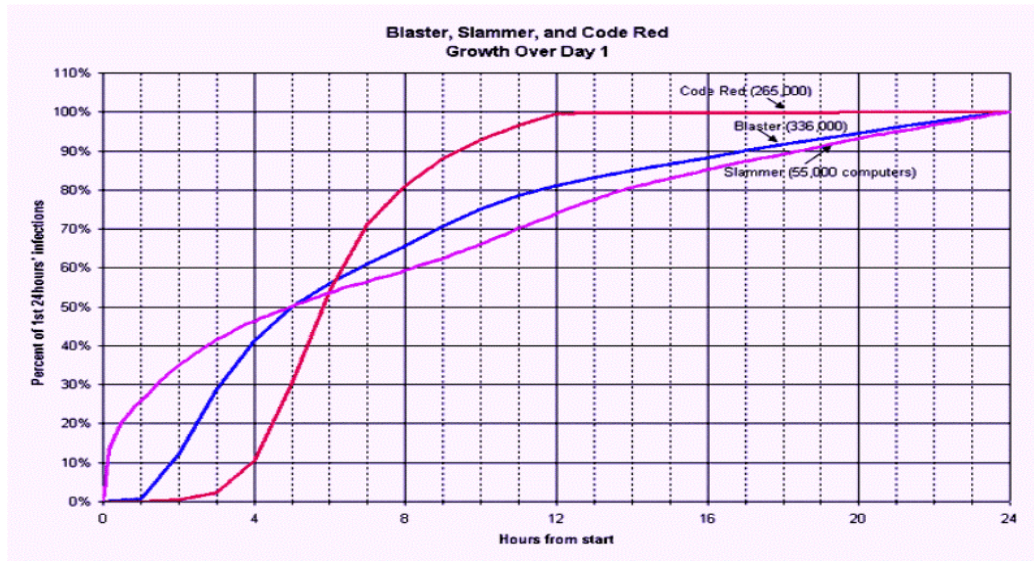


Fig. 2 Comparativo de propagación de gusanos *Blaster*, *Slammer* y *Code Red* en un día (tomada de [8]).

Tomando en cuenta lo anterior, se puede concluir que una de las principales características que convierte a los gusanos en una amenaza muy grave, es la rapidez que tienen para reproducirse a través de la red de computadoras. Por ejemplo, si se estima que el nivel más alto del dominio de la red es de $255.255.255.255 = 4,228,250,625$ nodos [9], y se considera como base el dato más alto del ejemplo anterior de 336,000 equipos afectados por día [8], y que por cada uno de ellos, se logra infiltrar el programa gusano en 100 equipos más diariamente en un período de 30 días, se tendría un total de 1'008,000,000 nodos contaminados, y se tendría a una cuarta parte del total de la red infectada (ver Figura 3).

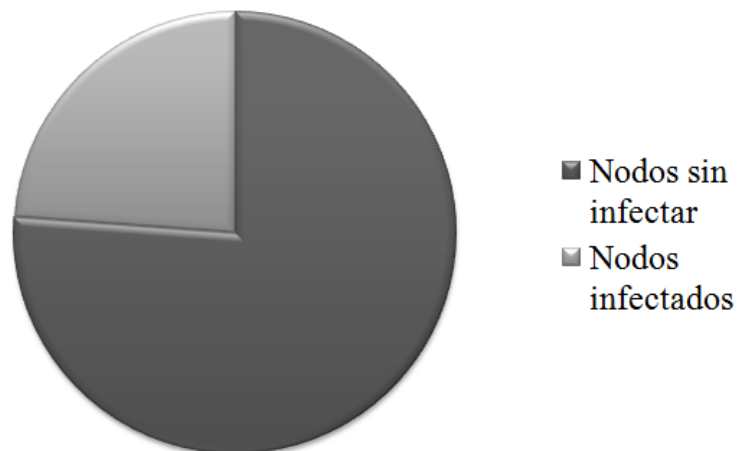


Fig. 3.- Simulación de actividad de *CodeRed* en 30 días.

Con el fin de presentar un ejemplo real de la vigencia, persistencia e importancia que sigue teniendo mundialmente el tema de contaminación de gusanos, a continuación se presenta información tomada en línea de la página de internet de la compañía NOD32 Antivirus System, sitio en el cual se puede revisar el porcentaje de propagación de códigos maliciosos detectados en más 10 millones de equipos que tienen instalado el programa de la compañía antes

mencionada, base sobre la cual se obtiene información estadística de los niveles de infección detectados en un período de 24 horas anterior al momento en que se solicita el reporte [10]:

Del sitio mencionado anteriormente [10], durante el desarrollo de este trabajo se han venido tomando reportes en distintas fechas, y comparando los resultados obtenidos en diciembre del 2006, enero del 2007 y junio del 2009, según se muestra en las Figuras 4, 5, y 6 respectivamente, encontrándose que el ‘malware’ más activo corresponde a gusanos tipo Win32, es decir, son ataques dirigidos a equipos con sistemas Windows, lo cual apoya la teoría, de que al igual que en años anteriores, los gusanos siguen representado una de las principales amenazas, especialmente para sistemas con ambiente Windows, y de que continúan propagándose a través de la red, como es el caso del *Win32/NetSky,Q Worm*, detectado por primera vez en el año 2004 y aún sigue vigente en distintas versiones. De igual forma, se puede observar en estos reportes que a la fecha surgen nuevos gusanos, tal como se comprueba con la aparición del gusano *Win32/TrojanDownloader:Small.EDN trojan*, detectado el día 25 de diciembre del 2007 y el gusano que aparece en el mes de octubre del 2008 llamado *Conficker Worm* [11], y reportado todavía en el mes de marzo del 2009, en la versión denominada *Win32/Conficker.D*.

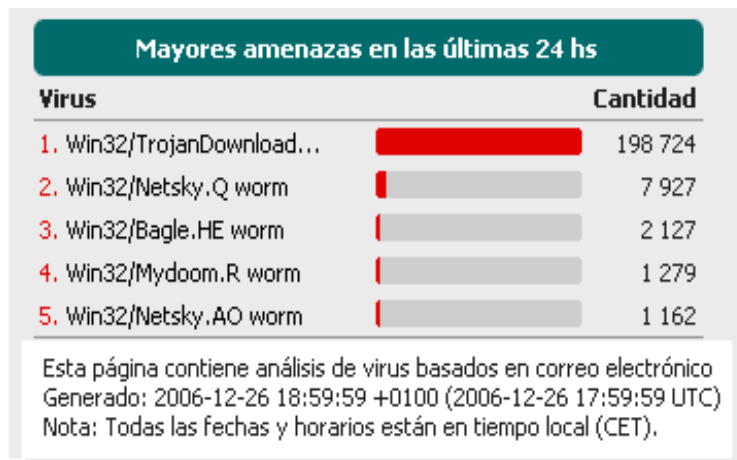


Fig. 4.- Estadísticas NOD32 (tomada de [10]).

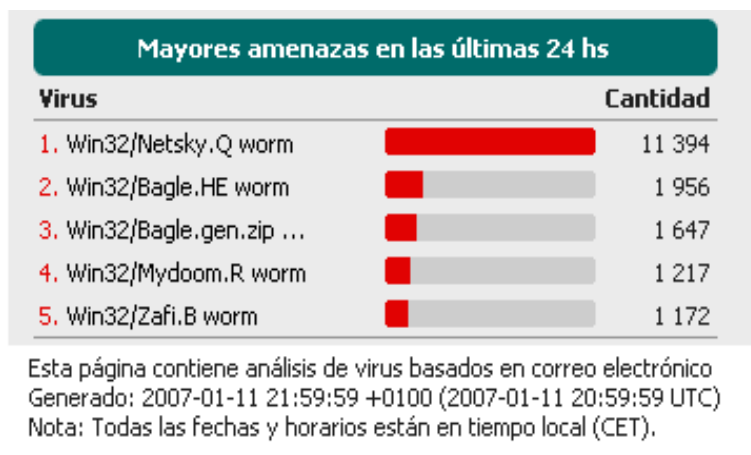


Fig. 5.- Estadísticas NOD32 (tomada de [10]).

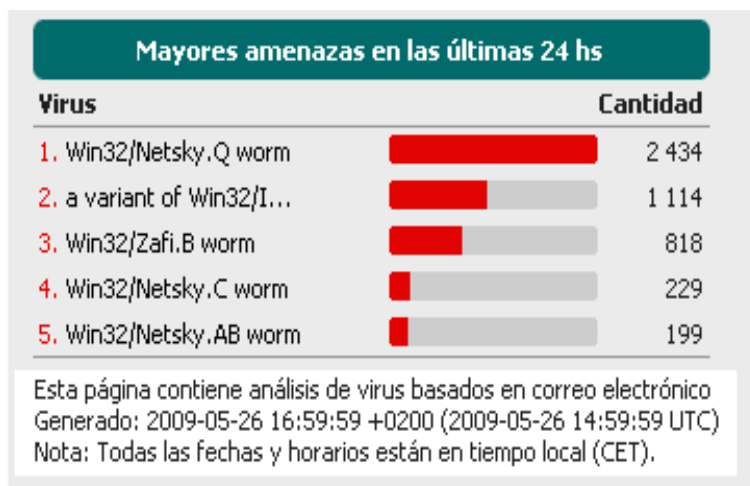


Fig. 6.- Estadísticas NOD32 (tomada de [10]).

Por otra parte, hay que tomar en cuenta que la evolución de los programas tipo ‘malware’ vienen presentando cambios en su comportamiento, confundiendo a los estudiosos del área, lo cual puede identificarse en los ejemplos que a continuación se mencionan:

- a) En ocasiones, el problema puede presentar características tipo gusano y a la vez tipo troyano, como es el caso que fue dado a conocer en diciembre del año 2006 por el Diario de Ciencia y Tecnología ‘La Flecha’, donde se indica que desde el mes de octubre se detectó una amenaza que NOD32 Antivirus llamó *Win32/Skyperise*, clasificándola como gusano, la cual afectaría a usuarios del popular software Skype, aplicación Voice Over IP (VoIP) que permite conexiones vía voz, webcam, mensajería instantánea, chat, etc. entre usuarios del propio programa. La conclusión a que se ha llegado, es que dicho ‘malware’, al tener la capacidad de propagarse a través del chat de dicha aplicación, lo enmarcaba dentro de los programas tipo gusano, pero también tenía la característica de que la descarga del mismo, se realizaba con la autorización del usuario receptor, lo cual, lo sacaba del esquema general del comportamiento de gusanos, y terminaba encuadrándolo en el de troyano [10].
- b) En el mes de junio del 2007, se presenta el caso del gusano *SpreadBanker.A*, que se propagó mediante un video de la página web del sitio Youtube, el cual fue programado para robar las contraseñas que se introducían en el sitio electrónico de varias entidades bancarias, y además podía obtener las claves de varios juegos como *Age of Mythology*, *GTA*, *Unreal*, *Tournament*, *WarCraft* y *Final Fantasy*. Dicho gusano se formó con dos componentes, de tal forma que cuando el usuario ejecutaba el primero de ellos, para conectarse al sitio de internet antes mencionado para ver videos, se descargaba la segunda parte del gusano y se instalaba en el equipo automáticamente, haciendo modificaciones al registro de Windows, y creando copias de sí mismo en carpetas de los programas del tipo punto a punto (P2P) [12].

Para tratar de resolver éste y otro tipo de problemas, desde hace tiempo se viene desarrollando software especializado, tanto de tipo comercial como de libre distribución, que se identifica como ‘Sistema Detector de Intrusos’, el cual permite registrar acciones que pueden comprometer la integridad, confidencialidad o disponibilidad de los sistemas de cómputo de una red de computadoras, y sirve como mecanismo de protección contra usuarios no autorizados que se introducen a través de internet, o contra usuarios autorizados que tratan de obtener privilegios adicionales para comprometer el buen funcionamiento de los equipos de cómputo [13].

Sin embargo, este tipo de herramientas presenta algunos inconvenientes, entre los cuales se puede mencionar, que la mayoría está diseñada para trabajar sobre uno o varios filtros de paquetes de red, lo que ocasiona un bajo rendimiento de la misma, que genera un gran número de alertas falsas, y que además para manejarlos, se requiere que el personal seleccionado para desarrollar esta tarea tenga conocimientos en materia de seguridad computacional, ya que para obtener buenos resultados en el funcionamiento del sistema de protección, es necesario analizar las bitácoras que se generan y definir si el registro corresponde solamente a una alerta o se trata de un ataque real, en cuyo caso, esto sólo servirá para llevar un control de lo que sucede en la red o para prevenir que se presente un nuevo ataque del mismo tipo, ya que no es posible evitar algo que ya sucedió [14].

1.3 Descripción del estado del arte.

La gran rapidez con que se propagan los gusanos a través de la red de computadoras y las cuantiosas pérdidas que ocasionan [15], ha obligado a las compañías que desarrollan software de protección a buscar soluciones que permitan resolver esta problemática, lo cual se ha logrado en parte, pues la mayoría de ellas los controlan a través de software de protección especializado, con la única salvedad de que, cada vez que surge un gusano nuevo, en el tiempo que transcurre mientras se estudia su comportamiento y se desarrolla la herramienta que pueda eliminarlo, los equipos infectados están expuestos a intrusiones no autorizadas, y a este período de tiempo, los expertos en seguridad le han llamado '*amenazas del día cero (zero day)*' [2].

Como un importante antecedente en la búsqueda de soluciones a esta problemática, se puede mencionar que Symantec Corporation [16], empresa líder en la infraestructura de software para protección de equipos de cómputo, el año 2004 publicó el artículo de seguridad de Richard Clarke, quien asegura que éste ha sido uno de los años más difíciles en materia de seguridad computacional, por los graves daños ocasionados, en su mayoría, por ataques de gusanos, lo cual provocó grandes pérdidas financieras, y plantea la posibilidad de poder prever un '*ataque del día cero*' tomando medidas que permitan afrontar, en un futuro, una posible producción de gusanos y virus con una carga explosiva más perjudicial a los que surgieron en ese tiempo, mencionando como una solución la de instalar sistemas de detección de intrusos basados en red y host, que buscaran comportamientos que pudieran representar una amenaza o violación a las políticas de seguridad, y emitieran alarmas cuando se presentaran violaciones o ataques de seguridad [2].

Por otra parte, en la versión de Windows 2000, se propuso implementar un equipo de respuesta a la intrusión llamado Intrusion Request Team (*IRT*) [12], como una normativa para detección y reacción frente a intrusiones en red, recabando pruebas de ataques que pudieran constituirse en un instrumento legal.

Y en la actualidad, para atender este tipo de exigencias en seguridad, se cuenta con una variedad de herramientas de software, entre los cuales han destacado los llamados '*Sistemas Detectores de Intrusos*', mismos que se dedican a monitorear el tráfico de paquetes de una red de computadoras, lo analizan, y generan '*alarmas*' que se registran y almacenan en archivos, los cuales deben ser revisados por personal de seguridad para diagnosticar si dichas '*alarmas*' son reales o no. Existen de tipo comercial y de libre distribución, y entre los más destacados se pueden mencionar los siguientes:

Snort [17], además de contar con las características propias de un sistema detector de intrusos, es una herramienta que permite generar bitácoras que son almacenadas en archivos individuales

o en una base de datos de MySQL [18]. Es un software muy popular, eficiente y de libre distribución, y puede utilizarse en plataformas como Windows, Unix y Linux [19].

SFlow [20], es un sistema que trabaja en un ambiente de red de ‘switches’ o ‘routers’ para proporcionar estadísticas sobre los paquetes de red que se filtran a través de este medio y se implementa a través de hardware especializado.

Kerf [21], utiliza una base de datos de MySQL [18], y ofrece una interfaz que permite revisar el comportamiento de una red de computadoras, mostrando datos específicos sobre equipos que la componen, usuarios, eventos, etc., integrando un lenguaje llamado *SawQL* que se basa en las especificaciones del lenguaje de consultas Structured Query Language [22], que en adelante será referenciado por las siglas *SQL*.

Por otro lado, Hicham Tout & William Hafner, han realizado una aportación con el proyecto ‘*An Agent-based, Application-Layer Security Framework*’ [4], por medio del cual simulan el funcionamiento de defensa del sistema inmunológico humano, enfocado a proteger los servicios tipo *WEB* (World Wide Web), utilizando una aplicación para detección con dos tipos de agentes para interceptar ataques antes de que logren su objetivo; uno de ellos simula el proceso de ejecución de servicios para interceptar posibles ataques, y el otro emplea un esquema para clasificar la respuesta que permita detectar posible robo de información. Implementan una estructura que se compone de siete componentes: generador de perfiles, interceptor, guardian, agente simulador, clasificador de respuesta, clasificador de amenazas, y consejero contra amenazas.

Kuper P., en el artículo sobre ‘*The state of security*’ [23], menciona que a pesar de que la tecnología en seguridad de información avanza rápidamente, se está caminando en reversa en ese sentido, ya que, en su opinión, el mercado de la seguridad, primero se enfoca a sistemas de protección cortafuegos o tecnología de antivirus, aplicando un enfoque para otorgar más protección a nivel de aplicación, y concluye que se debiera de dar más importancia al cuidado de la integridad de la información en la red, y buscar que la seguridad inicie con la protección del lugar, cerrando recursos tanto como sea posible.

En dicho artículo se menciona que en los cinco años anteriores se invirtieron cerca de 15 billones de dólares en software de protección antivirus, cortafuegos y software de redes privadas virtuales y en una de las tecnologías más utilizadas para protección, que es la criptografía, y se hace hincapié en que debe existir un cambio en los paradigmas de seguridad que permitan que Internet continúe desarrollándose en forma positiva, para lo cual deben implementarse nuevos modelos y realizarse más investigación, sobre todo en los temas relacionados con los siguientes aspectos:

- Integridad en los datos que se transmiten a través de la red.
- Inclusión/Exclusión.- Tener control de usuarios y privilegios.
- Seguridad incrustada.- Desarrolladores y operadores deberán conocer aspectos de seguridad y funcionalidad para tomarlos en cuenta.
- Perfeccionar enfoques.- Enfoques anticuados podrían reemplazarse por otros nuevos, por ejemplo, antivirus software podría reemplazarse por antispymware soluciones; el perímetro de corta fuegos debería tener capacidad para manejar dinámicamente Extensible Markup Language (*XML*), y arquitecturas de servicios *WEB* para incrementar el nivel de seguridad.

Bruce Schneier [24], un reconocido profesional en seguridad a nivel mundial, en su artículo ‘*SIMS (Security Information Management Systems): Solution, or Part of the Problem?*’ [25], concluye que la clave para resolver los problemas de seguridad en una red está en la gente y no en el uso de productos de seguridad, como sistemas detectores de intrusos, programas de protección tipo antivirus o ‘*cortafuegos*’ (firewall), ya que no han sido capaces de detener en su totalidad los ataques de intrusos, gusanos, códigos maliciosos, virus, etc.

Además, existen propuestas como la de Cyber-Threat Analytics (*Cyber-TA*) [26], proyecto que está integrando grupos de investigadores que puedan encontrar alternativas de solución para poder afrontar una próxima generación de amenazas de seguridad, que se prevé, irá en aumento. Para ello, plantean implementar una estructura conformada por grandes centros que trabajen en forma colaborativa y sean capaces de reconocer ataques, analizando los datos generados por sistemas detectores de intrusos, cortafuegos u otro tipo de herramientas disponibles.

Tomando como ejemplo lo mencionado en párrafos anteriores, se puede concluir que desde hace varios años profesionistas expertos del área, y las compañías que se dedican a desarrollar software de protección de distintos tipos, están dirigiendo sus esfuerzos a buscar alternativas de solución para afrontar problemas de seguridad computacional, entre los cuales se encuentran los relacionados con la problemática del día cero.

Por otra parte, en relación a los enfoques de los sistemas detectores de intrusos mencionados anteriormente, se puede observar lo siguiente:

La aplicación que utiliza agentes para seguridad en servicios *WEB* a través de simulación inmunológica humana, puede presentar un retardo en la respuesta, con posible pérdida de información por el tiempo que se invierte en la simulación de procesos, pero es novedoso, y faltaría valorar si el costo de estos retardos no es significativo, ya que en el artículo no se presentan resultados de prueba o implementación de este sistema.

El sistema *SFlow* tiene una arquitectura donde se implementa software de seguridad incrustado en hardware especializado como ‘*switches*’ o ‘*routers*’ y se maneja principalmente para obtener estadísticas de red.

Snort y *Kerf* requieren de la instalación del software correspondiente, funcionan bajo cierta configuración de reglas que deben ser estructuradas por un administrador, y presentan como inconvenientes el propiciar cuellos de botella en el tráfico de red, y un alto grado de generación de falsas alarmas.

De todo lo mencionado anteriormente, se puede resaltar el hecho de que en general todo este tipo de sistemas tienen como característica común que requieren de la intervención de personal con suficiente conocimiento en seguridad para distinguir entre las situaciones de ataques reales y las de falsas alertas que se generan [27], y las distintas opiniones de expertos en relación a que se está invirtiendo demasiado dinero en productos para seguridad, cuando debiera darse más importancia a la restricción de recursos y al cuidado del área física donde se encuentran los equipos, o las de quienes sugieren que se cuente con equipos de trabajo en seguridad y políticas para detección de intrusiones en la red para analizar la información filtrada, y las de otras personas que piensan que se deben implementar nuevos modelos y realizar más investigación en los temas relacionados con integridad de datos, control de recursos, aspectos de desarrollo y operación de software, y perfeccionamiento de enfoques orientados a seguridad.

Respecto a la última idea planteada en el párrafo anterior, cabe mencionar que en México, como en otros países, existen instituciones que ya cuentan con líneas de investigación en materia de seguridad computacional, contándose entre ellas:

- El Instituto Tecnológico de Monterrey, que en sus cátedras incluye el tema ‘*Innovación e Investigación en Seguridad Computacional*’ con líneas de investigación en ‘*Detección de Intrusiones y Ataques*’ y ‘*Análisis de Protocolos de Seguridad*’ [28].
- El Departamento de Seguridad en Cómputo (*DSC*) de la Dirección General de Servicios de Cómputo Académico de la Universidad Nacional Autónoma de México (*DGSCA*) [29], a través del cual, entre otras cosas, se establecen políticas de seguridad adecuadas a fin de disminuir la cantidad y gravedad de los problemas de seguridad en cómputo y se mantiene una relación de vinculación con el *CERT*, publicando periódicamente estadísticas sobre problemas reportados como los que se muestran en la Figura 7, que corresponden al período 2004-2008, donde destacan en gran proporción los ataques que se orientan hacia los sistemas operativos con ambiente Windows. Y del origen de estos problemas que se muestran en la Figura 8, donde se puede observar que la mayoría se producen por Spam, gusanos como *Bagle*, *Slammer*, *MyDoom*, *Sasser*, *NetSky*, *Bots* utilizados en forma maliciosa (tipo de programas robots que se utilizan para realizar tareas rutinarias en Sitios *WEB*), y *Scanners*, que comúnmente son utilizados para tareas que son realizadas por gusanos para buscar vulnerabilidades en los equipos.

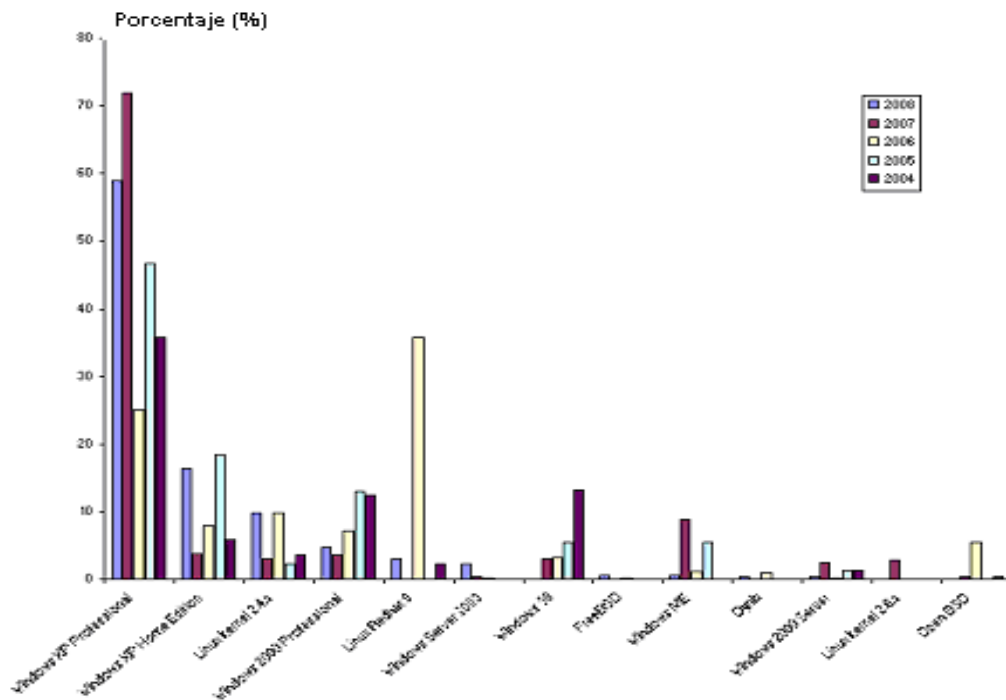


Fig. 7 Ataques a sistemas operativos detectados en el período 2004-2008 por el *DSC* (tomada de [29]).

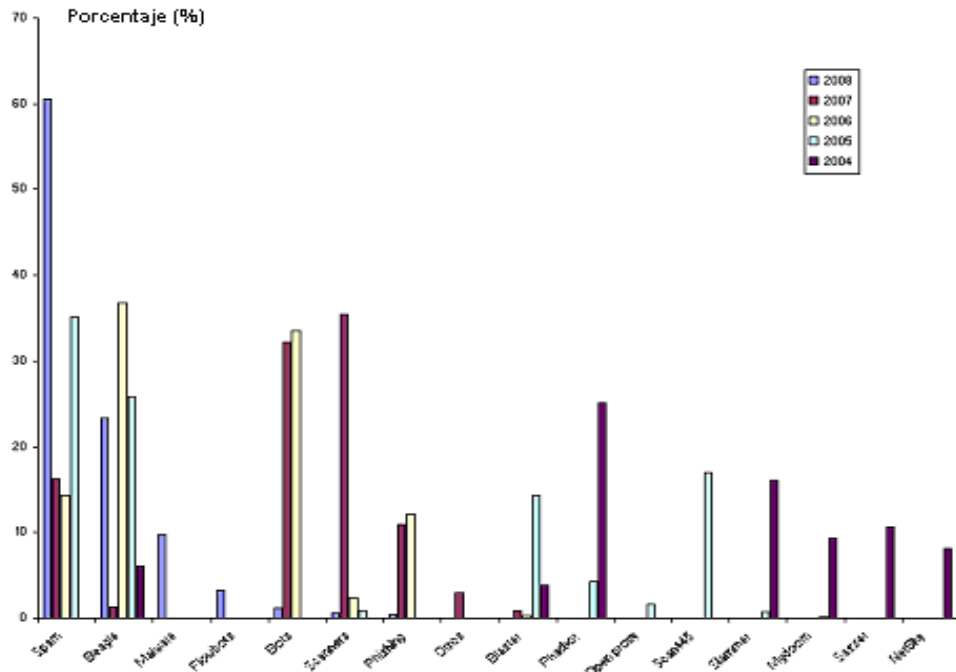


Fig. 8 Origen de los ataques a sistemas operativos detectados en el período 2004-2008 por el DSC (tomada de [29]).

Por último, cabe agregar que, buscando un enfoque distinto a los actuales, en este trabajo se presenta una propuesta que contempla un conjunto de elementos básicos que son necesarios para la administración de los recursos de los equipos con ambiente Windows-Tecnología NT y para el desarrollo de sistemas de seguridad basados en agentes. Dicha propuesta incluye el uso de un lenguaje orientado a seguridad que pueda ser implementado sobre una plataforma de software, que permita generar grupos de agentes que trabajen como un sistema integrado, tomándose como base una arquitectura de agentes que incluya componentes o módulos que permitan abordar el tema de seguridad computacional, aplicar mecanismos de razonamiento de agentes, que cuente con un módulo de seguridad que apoye a los desarrolladores y administradores de este tipo de sistemas, y que incorpore un esquema de protección de la información que se transmite a través de la red.

Capítulo 2.- Elementos básicos para sistemas de seguridad basados en agentes.

2.1 Aspectos de diseño.

Dado que los agentes pueden considerarse como entidades inteligentes que conforman un sistema, que pueden percibir su ambiente a través de sensores, y que tienen capacidad para evaluar sus percepciones, tomar decisiones por medio de mecanismos de razonamiento sencillos o complejos, y establecer comunicación con otros agentes para obtener información, actuando sobre el medio en el que se desenvuelven a través de ejecutores/actuadores/efectores [30] (ver Figura 9), a continuación, se presenta el esquema de funcionamiento de un sistema de agentes que tiene como objetivo controlar los eventos de seguridad en sistemas operativos Windows-Tecnología NT, mismo que en adelante será referenciado por las siglas *SISAG-W* (*Agent System for Windows-Technology NT*), desarrollado con la implementación de la Gramática *CLASS-W* e implementado sobre la plataforma *JADE* (*Java Agent Development Framework*) [31], y el Lenguaje de Comunicación de Agentes (*ACL*) [32], estandarizado por la Foundation for Intelligent Physical Agents (*FIPA*) [33].

Para representar gráficamente algunos de los componentes que se analizan en este documento, en adelante se aplicarán las siguientes especificaciones:

- Los rectángulos se utilizan para identificar procesos, exceptuando aquellos casos en que se usan como bordes externos de figuras o imágenes para simular los límites del ambiente donde se desenvuelven los agentes, o en diagramas de datos y jerárquicos para agrupar ciertos componentes o módulos.
- Las estructuras de control para datos secuenciales se representan con círculos colocados sobre una pequeña línea horizontal.
- Los módulos se identifican por un rectángulo con esquinas curvas.
- Los rectángulos con esquinas curvas y línea punteada se utilizan para delimitar áreas de gráficos.

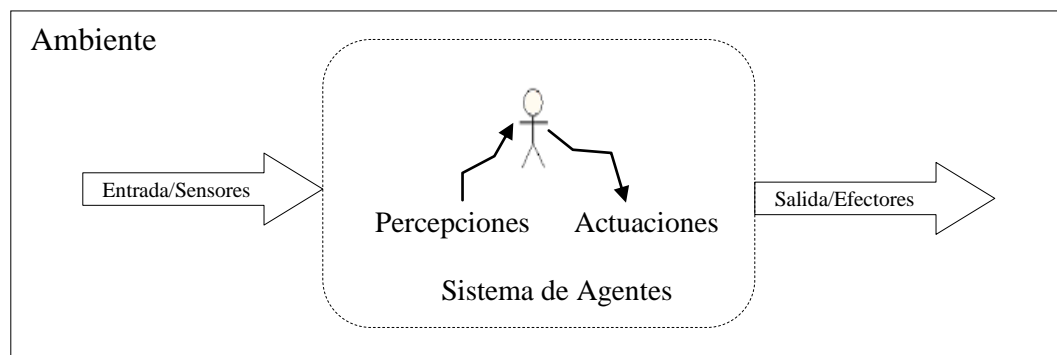


Fig. 9.- Interacción de Agentes con su medio ambiente a través de sensores/efectores.

Por otra parte, para estructurar una propuesta basada en un sistema de agentes que aborde la temática de seguridad computacional, es necesario contar con los siguientes elementos básicos:

- a) Una arquitectura híbrida de agentes orientada al tema de seguridad computacional. Una arquitectura híbrida según [34], permite integrar agentes de tipo reactivo y deliberativo, mismos que aunque difieren en su comportamiento, en un esquema de trabajo conjunto, pueden coadyuvar en el alcance de metas y objetivos comunes cuando así se requiere. Cabe mencionar que de las arquitecturas revisadas, no se encontró ninguna que en forma total aporte los recursos necesarios para abordar la temática de seguridad computacional, como lo es la integración de módulos que permitan implementar conceptos de seguridad, y la integración explícita de uno de los modelos de arquitecturas de pizarra que se constituya en un componente o módulo más que sirva como medio de comunicación a los agentes. También es necesario aclarar que de algunas de las arquitecturas que se mencionan en el apartado 2.2 de este documento, se seleccionaron aportaciones parciales que fueron integradas a la propuesta de una arquitectura híbrida de agentes para abordar el tema de seguridad computacional con tecnología de agentes, la cual se describe en el apartado 2.2.5, y se detalla en el desarrollo de los siguientes capítulos de este escrito.
- b) Niveles jerárquicos que permitan distinguir el grado de autoridad de los agentes para acceder a los recursos del sistema.
- c) Roles que permitan identificar las funciones que los agentes pueden realizar cuando asuman ciertos comportamientos.
- d) Un lenguaje de comunicación de agentes que permita la programación de transacciones específicas para el control de recursos relacionados con aspectos de seguridad.
- e) Programación del conocimiento a través de módulos que proporcionen formas de razonamiento al sistema de agentes.
- f) Una pizarra que facilite el proceso de interacción de agentes.
- g) Una estructura de bases de datos que entre otras cosas, permita organizar la información relacionada con políticas, niveles de seguridad, temas, e instancias de la pizarra relacionadas con los temas previamente definidos.
- h) Una plataforma de software que permita la implementación del sistema de agentes.
- i) Un esquema de protección de mensajes intercambiados por los agentes a través de la red.

Para efectos de organización de este documento, en este apartado sólo se hará una descripción general sobre algunos de los puntos antes mencionados, y en los siguientes capítulos se presentará una descripción más detallada de los elementos que así lo requieran.

2.1.1 Jerarquía de agentes.

En esta sección se presenta una clasificación de agentes por nivel (ver Diagrama 1), donde a cada nivel le corresponde un rol o papel acorde a la especialidad o actividad en la que el agente es experto [30]. Los tipos de agentes que se utilizarán en el sistema descrito en este trabajo son los que a continuación se describen:

1° nivel:

Agente Manejador del Sistema (AMS) – Es definido por *FIPA* para manejar el sistema de agentes y es implementado sobre la plataforma *JADE*.

2° nivel:

Agente Facilitador de Directorio (DF).- Es definido por *FIPA* e implementado sobre la plataforma *JADE* para facilitar información de directorio y de servicios.

3° nivel:

Agente Coordinador (AC) – Realiza una actividad especializada, orientada a la coordinación de transacciones que permiten aplicar niveles de seguridad en el sistema, y a la coordinación del trabajo de agentes locales y de red.

4° nivel:

Agente Local (AL) – Tiene asignadas como principales actividades, la de vigilar el cumplimiento de políticas de seguridad, revisar el comportamiento de ciertos recursos del sistema, y monitorear en forma local los eventos que se presentan en el equipo donde se crean.

5° nivel:

Agente de Red (AR) - Tiene como función monitorear los paquetes de red en el equipo donde se crea, y aplicar las reglas especificadas para este tipo de eventos.

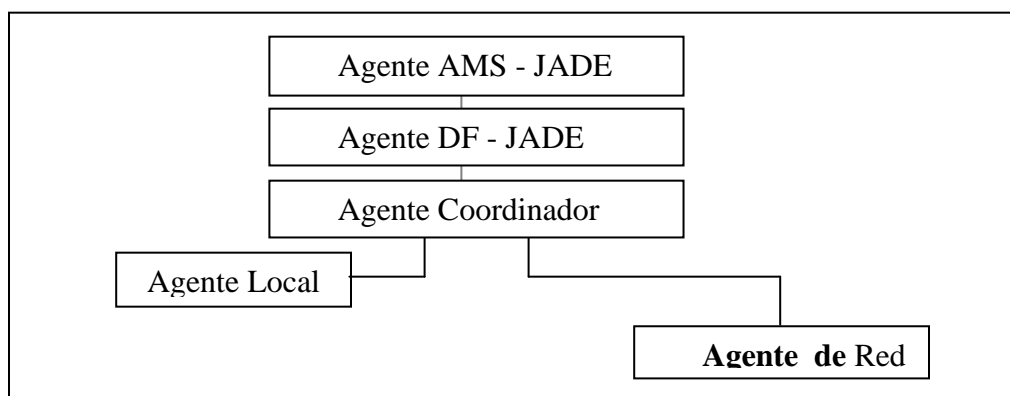


Diagrama 1. Jerarquía de agentes de *SISAG-W*.

Entre las principales ventajas que tiene el utilizar esquemas jerárquicos en sistemas de agentes, está la de poder identificar los permisos o privilegios que se tienen para el acceso a los recursos del sistema, definir el alcance funcional para realizar ciertas tareas, y poder establecer grupos de trabajo conformados por varios tipos de agentes, que en base a una conducta combinada, pueden producir resultados que permitan lograr el cumplimiento de los objetivos previamente especificados. Es por ello, que para el diseño de *SISAG-W*, en relación a la jerarquía que se muestra en el Diagrama 1, se toman como base los siguientes criterios:

- 1) Conforme disminuya el nivel jerárquico del agente, tendrá menor grado de autoridad para acceder a los recursos del sistema.
- 2) Cada rol se asocia con un nivel jerárquico, de tal forma que independientemente del grado de jerarquía que tenga un agente, no puede realizar las funciones de otro nivel.

Enseguida, con el fin de desarrollar el tema relacionado con el trabajo conjunto de agentes, que se lleva a cabo a través del intercambio de mensajes, y que tiene como base el esquema funcional y de organización descrito en los párrafos anteriores, se presentará la descripción de la forma de interacción de agentes, los aspectos básicos de una gramática libre de contexto para desarrollo de sistemas de seguridad basados en agentes y los detalles sobre la estructura física de *SISAG-W*.

2.1.2 Interacción de Agentes.

Los eventos que se generan en el medio ambiente donde coexisten los agentes, son registrados a través de sensores que producen percepciones en un agente, que les permite asumir una actitud o comportamiento y reaccionar o tomar decisiones sobre las acciones

que deben de realizar, las cuales a su vez, pueden producir cambios en el entorno y reflejar una nueva realidad contextual.

Por lo antes expuesto, y por la importancia que tiene reconocer las conductas que pueden estar relacionadas con cada nivel del esquema jerárquico expuesto en el punto 2.1.1 de este escrito, a continuación se describen algunos comportamientos que los agentes pueden tener para interactuar con su medio ambiente:

- ❖ **Reactivo o reflejo.**- El agente tiene capacidad de reaccionar automáticamente a los eventos que se presentan de acuerdo al rol asignado [34] (ver Figura 10).
- ❖ **Deliberativo.**- Cuando se presenta una situación que exige reflexión sobre un asunto antes de tomar una decisión, el agente tiene la oportunidad de utilizar su capacidad de razonamiento sobre las creencias e intenciones que hayan sido previamente definidas [34] (ver Figura 11).
- ❖ **Cognitivo.**- Este caso se presenta cuando el proceso implica abordar una tarea y una toma de decisión consciente e intencionada para realizar operaciones complejas, con experiencia y capacidad de razonamiento y posibilidades de aprendizaje de nuevos conocimientos [34] (ver Figura 12).

Puesto que los comportamientos mencionados anteriormente no se contraponen entre sí, para el desarrollo de este trabajo, estos comportamientos de utilizarán de acuerdo a los posibles escenarios que se puedan presentar en un momento dado.

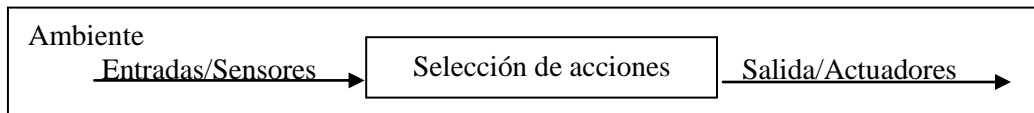


Fig. 10.- Funcionamiento agente reactivo o reflejo.

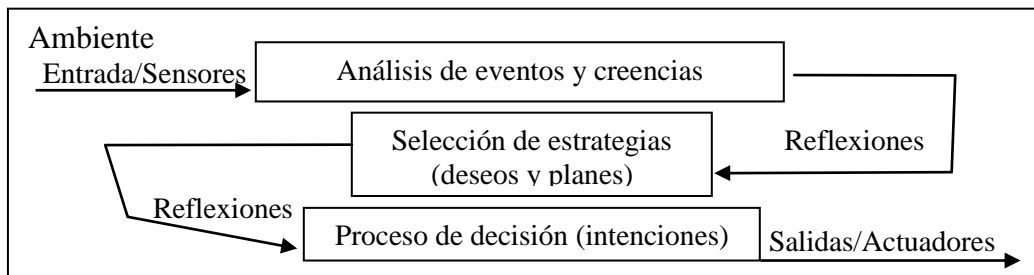


Fig. 11.- Funcionamiento de un agente deliberativo.

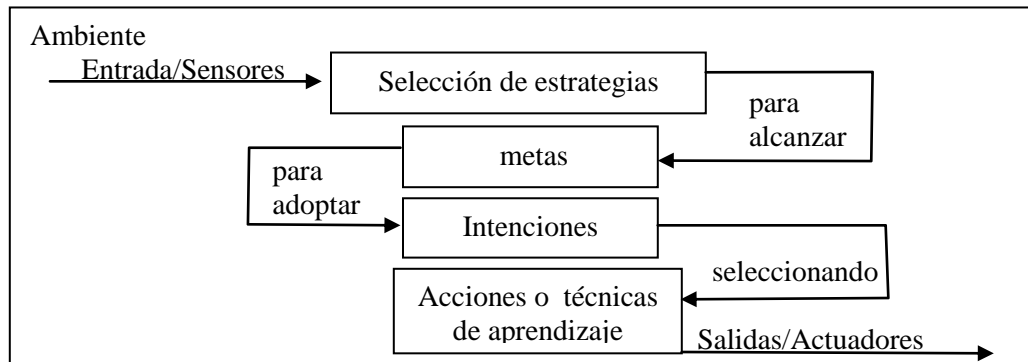


Fig.12.- Funcionamiento de un agente cognitivo.

Así como en el proceso de interacción es indispensable tener formas de conducta, también lo es el tener un protocolo de comunicación por medio del cual los agentes puedan intercambiar información. Por ello, enseguida se presenta la propuesta de una gramática que ha sido diseñada específicamente para desarrollar sistemas orientados a seguridad basados en la tecnología de agentes.

2.1.3 CLASS-W: Gramática para desarrollo de sistemas de seguridad basados en agentes.

Dado que los agentes, para interactuar entre sí, requieren de un lenguaje que les permita comunicarse e intercambiar mensajes, como una alternativa para resolver problemas de seguridad, en este trabajo se presenta la descripción general de *CLASS-W*, una gramática libre de contexto, que genera un lenguaje de contenido que sirve como base para el desarrollo de sistemas de seguridad basados en agentes para equipos de cómputo con ambiente Windows-Tecnología NT, el cual se utiliza con el Formato 1, especificado por *FIPA* [33] para el Lenguaje de Comunicación de Agentes (*ACL*) [32].

Esta idea surge, por la necesidad de contar con alternativas flexibles que permitan aplicar los conocimientos de expertos del área de seguridad en la administración de redes de computadoras, planteándose de inicio utilizar como forma de representación del conocimiento el uso de predicados tipo Prolog y reglas generadas por árboles de decisión. Sin embargo, al utilizar el enfoque planteado en este documento, queda a criterio de los diseñadores o desarrolladores la elección de aplicar alguno de los métodos o técnicas que existen para representar el conocimiento del sistema de agentes.

Por otra parte, dada la extensión de la totalidad de la gramática, en el desarrollo de este documento, y a partir de este punto, se presentarán las especificaciones de la misma que sean necesarias para aclarar el funcionamiento del sistema de agentes *SISAG-W*, que ha sido desarrollado con *CLASS-W*.

2.1.4 Aspectos generales gramaticales de CLASS-W.

Como una forma de mostrar la potencia expresiva del lenguaje para abordar determinado tipo de problemas, se presenta una gramática libre de contexto con la cual es posible desarrollar sistemas de seguridad basados en agentes, y controlar el comportamiento de cierto tipo de objetos y sus relaciones, utilizando para efectos de comunicación un flujo de mensajes.

Para implementar el lenguaje de contenido que se genera por *CLASS-W* en el Lenguaje de Comunicación de Agentes (*ACL*) [32], estandarizado por *FIPA* [33], se toman como base las siguientes consideraciones:

- a) Un lenguaje de contenido se usa para el intercambio de conocimientos entre un conjunto de agentes.
- b) Las especificaciones gramaticales de *CLASS-W*, se implementan en la estructura del Formato 1 de *ACL*, a través de la opción '*content*' o contenido.
- c) Las secciones del formato mencionado en el inciso anterior, que corresponden a '*sender*', '*receiver*' y '*content*', son elementos obligatorios definidos por *FIPA*, y el resto, pueden utilizarse en forma opcional.
- d) De los actos comunicativos que han sido definidos por *FIPA* para utilizarse en el formato de *ACL* antes mencionado, en *SISAG-W*, sólo se utiliza el de '*inform*', ya que el diseño de transacciones permite identificar la información

que se relaciona con sesiones de trabajo, manejo de pizarra, programación del conocimiento, y demás especificaciones necesarias para el funcionamiento del sistema de agentes.

```
(< communicative_act>
  :sender <identifier>
  :receiver <identifier>
  :reply-to <expresion>
  :content <message>
  :language <expresion>
  :encoding <expresion>
  :ontology <expresion>
  :protocol <expresion>
  :conversation-identifier <expresion>
  :reply-with <expresion>
  :in-reply-to <expresion>
  :reply-by <expresion>
  :envelope <expresion>)
```

Formato 1.- Estructura de mensajes ACL.

Por otro lado, el diseño de *CLASS-W*, se basa en la teoría de Gramáticas Libres de Contexto (*GFC*), con la cuádrupla denotada por $G = (N, T, P, S)$ [35] donde:

- N*** Conjunto finito de símbolos No Terminales.
- T*** Conjunto finito de símbolos Terminales.
- S*** Símbolo de *N*, que se constituye como símbolo inicial o raíz de la *GFC*.
- P*** Conjunto finito y no vacío de producciones del tipo: $A \rightarrow a$, donde *A* pertenece a *N* y *a* pertenece a $(N \cup T)^+$.

Para expresar la sintaxis de las *GFC* [36] se utiliza la Forma Normal de Backus-Naur Extendida (*EBNF*), con especificaciones básicas como las que se muestran en la Tabla 1, las cuales se utilizan para representar las unidades sintácticas de *CLASS-W* en el desarrollo de este documento.

Tabla 1. Notación básica en *EBNF* para *CLASS-W*.

Notación	Descripción
	'Or'
{ t }	0 o más elementos t
[t]	Opcionalidad sobre t
,	'And'
< t >	Símbolo No Terminal t
::=	Implicación
....	Valores por definir
n	Número máximo de elementos

Y, por último, para dar inicio a la descripción de *CLASS-W* en forma detallada, en los siguientes puntos se presenta la estructura general y las principales producciones de la gramática.

2.1.5 Estructura general de un programa de *CLASS-W*.

Para la especificación de sistemas desarrollados con *CLASS-W* e implementados sobre la plataforma *JADE*, es necesario identificar el símbolo inicial del programa que utiliza la gramática, el cual se denota por el símbolo no terminal <contenido> que se deriva con la siguiente producción:

```
<contenido> ::= '(' <mensaje> ')'  
<mensaje>  ::= '#<clave>' <agente_emisor> <agente_receptor> <tipo_agente> <tipo_conversación> <tipo_transacción>
```

En el contexto de la estructura anterior, el símbolo no terminal <tipo_transacción> es utilizado para canalizar los mensajes que se generan sobre los eventos que se presentan.

2.1.6 Plataforma de desarrollo *JADE*.

Para el desarrollo de *SISAG-W*, se utiliza la plataforma *JADE* (Java Agent DEvelopment Framework) [31], la cual es considerada como un middleware de código abierto y libre escrito en Java, que proporciona un entorno de desarrollo y ejecución para la realización y mantenimiento de sistemas de agentes en forma muy eficiente, lo cual ha hecho que sea una de las más populares. Cumple con las especificaciones definidas por *FIPA* [33], tanto a nivel de arquitectura como de mensajes de *ACL* [32], e incluye librerías que permiten implementar las características definidas por *FIPA*, incluyendo las de tipo obligatorio como es el caso de que cada plataforma debe contar con un agente facilitador de directorio (*DF*), y otro que administre el sistema de agentes (*AMS*). Además cuenta con un canal de comunicación de agentes (*ACC*), y para comunicación remota utiliza el método Remote Method Invocation (*RMI*) y el de Internet Inter-Orb Protocol (*IIOP*).

2.1.7 Componentes adicionales.

Para elevar la eficiencia en tiempo de ejecución del software desarrollado, y acceder a recursos del sistema operativo casi en forma directa, *SISAG-W* es apoyado por componentes adicionales codificados en lenguaje C/C++, organizados en librerías de enlace dinámico (DLL), en base a la estructura física que se muestra en la Figura 13 y Figura 14, incluyéndose entre estos elementos, los que proporcionan formas de acceso a la información contenida en bases de datos.

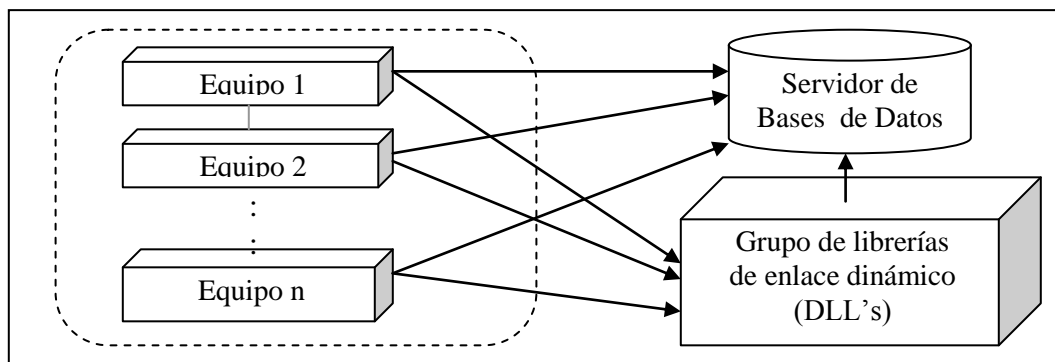


Fig. 13.- Estructura física general de *SISAG-W*.

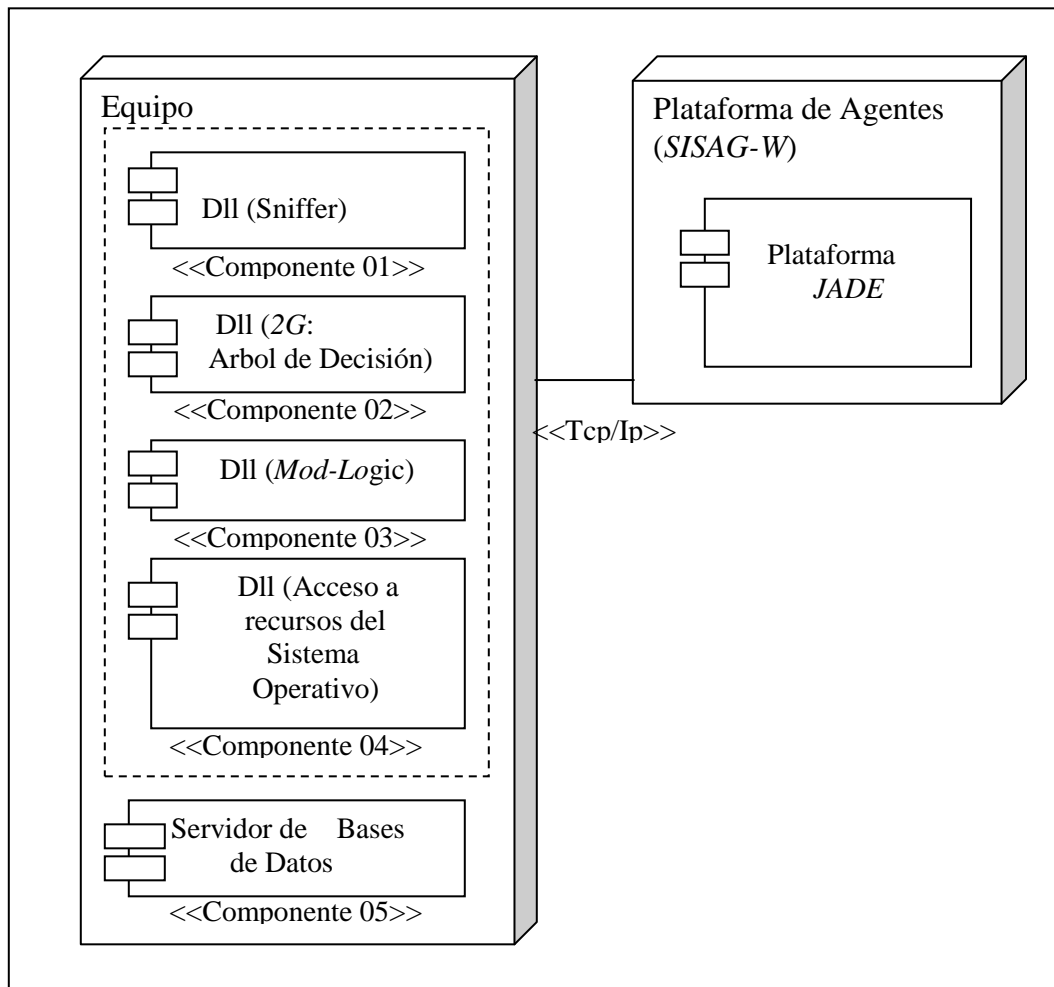


Fig. 14.- Estructura física de componentes de SISAG-W.

En los párrafos siguientes se describe la propuesta de una arquitectura híbrida que resulta del análisis de varios modelos existentes que fueron tomados como base para su diseño, que tiene una relación directa con las estructuras físicas presentadas en este punto.

2.2 Arquitectura de Agentes.

Para complementar el esquema básico que permita asignar roles a los agentes, definir el comportamiento de los mismos, y diseñar posibles escenarios de interacción de acuerdo a los sucesos que se presentan en su entorno, de las arquitecturas existentes para modelar sistemas de agentes, se revisó el funcionamiento de metodologías como MAS-CommonKADS [37], INGENIAS [38], PRODIGY [39], etc., tomándose como referencia básica para este trabajo, las especificaciones de *BDI* (Beliefs, Desires and Intentions) [40], *GAIA* [41] [42], *MESSAGE* [43], *PRS* (Procedural Reasoning System) [44], y de las arquitecturas de pizarra, se eligió como esquema básico la de Hearsay-II [45] (ver Figura 28). De lo anterior resultó el diseño de una arquitectura de pizarra específica para la implementación de *CLASS-W*, que ha sido denominada *BLACKBOARD-ECRE* (Blackboard-Element Control for Register of Events), y una arquitectura de agentes híbrida llamada *ARSEC-AMS* (Security Architect – Multiagent System), con

aplicación de inferencia lógica para el comportamiento de agentes, incluyéndose ideas como las que se describen en los siguientes puntos.

2.2.1 BDI, como modelo cognitivo, se utiliza para representar el razonamiento y comportamiento del ser humano en un sistema de agentes y se estructura de la siguiente forma:

Beliefs (Creencias).- Conjunto de *'ideas'* o *'concepciones'* del estado del *'mundo'* o del *'ambiente'*, que pueden representarse a través de variables, estructuras de base de datos relacionales o expresiones simbólicas.

Desires (Deseos).- Se conciben como *'metas'* y se representan a través del valor de una variable, el registro de una estructura, o una expresión simbólica lógica para expresar algún tipo de estado final o el objetivo que se desea alcanzar.

Intentions (Intenciones).- Conjunto de procesos para representar los *'mundos'* posibles.

Este modelo contempla el uso de un intérprete que interacciona con conjuntos de datos agrupados que se relación con creencias, objetivos, intenciones y planes de agentes (ver Figura 15), donde los eventos que ocurren en el ambiente de los agentes se toman como entradas al sistema, y se controlan a través de una cola de eventos.

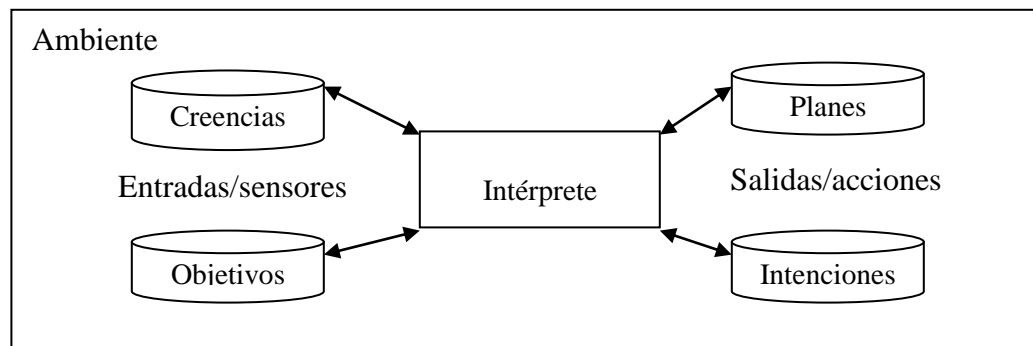


Fig. 15.- Modelo *BDI*.

Dadas las abstracciones implementadas en el modelo antes mencionado, en el diseño de la propuesta de la arquitectura que se plantea en este apartado, se ha decidido tomar en cuenta del nivel externo de *BDI*, los aspectos que se mencionan a continuación:

- ❖ Especificar una jerarquía de agentes (ver Diagrama 1).
- ❖ Incluir componentes externos en la estructura de agentes.
- ❖ Diseñar esquemas de relación del sistema de agentes con componentes externos.
- ❖ Diseñar funciones independientes para relacionar los componentes externos con el sistema de agentes.
- ❖ Contar con información requerida como archivos, bases de datos, etc.
- ❖ Para el modelo de interacción utilizar:
 - Lenguaje de comunicación de agentes (*ACL*).
 - Lenguaje de contenido (*CLASS-W*).
 - Especificaciones de alcance de actuación.
 - Diseño de mensajes.

- Definiciones de comportamiento de agentes (reactivo, deliberativo o cognitivo).
- ❖ Contar con un módulo lógico para generar reglas tipo Prolog y plantillas tipo *SQL* para su interpretación.
- ❖ Contar con un conjunto de reglas generadas por el algoritmo 2G basado en árboles de decisión, para ser utilizadas por el sistema de agentes, especialmente por perfil asignado al Agente de Red.

Y del nivel interno que utiliza *BDI*, se plantea como referencia:

- ❖ Un modelo para estructurar estados (de información, de intención, de deseos, etc.).
- ❖ Las estructuras de control para determinar la conducta de los agentes.
- ❖ Un modelo para la definición de:
 - Creencias, con información sobre el ambiente del sistema, el estado y acciones a realizar por los agentes.
 - Objetivos y respuestas a los eventos.
 - Planes que definan los pasos a seguir para el alcance de los objetivos y dar respuesta a los eventos que se presenten.

Más adelante se describe una propuesta de arquitectura que fue planteada tomando en cuenta las ideas planteadas anteriormente.

2.2.2 De *GAIA*, que es una metodología orientada al diseño de sistemas basados en agentes, que tiene como principal objetivo maximizar alguna medida de calidad global en el sistema, para que a partir de ciertos requisitos, inicialmente se logre contar con un diseño que esté lo suficientemente detallado como para ser implementado en forma directa (ver Figura 16), se incluyen en nuestra propuesta los siguientes lineamientos:

- ❖ Utilizar un diseño organizacional.
- ❖ Diseñar un conjunto de entidades abstractas para análisis y conceptualización, donde la entidad más abstracta es el concepto de ‘*organización*’ o de ‘*sociedad*’.
- ❖ Contar con roles que contemplen los siguientes atributos:
 - Responsabilidades relacionadas con funcionalidad.
 - Permisos o derechos asociados al rol.
 - Protocolos asociados.
- ❖ Implementar una etapa de análisis donde se identifiquen los roles y protocolos sociales, y otra para diseño de un modelo de agentes basado en roles, servicios, conocimientos, y formas de interacción.

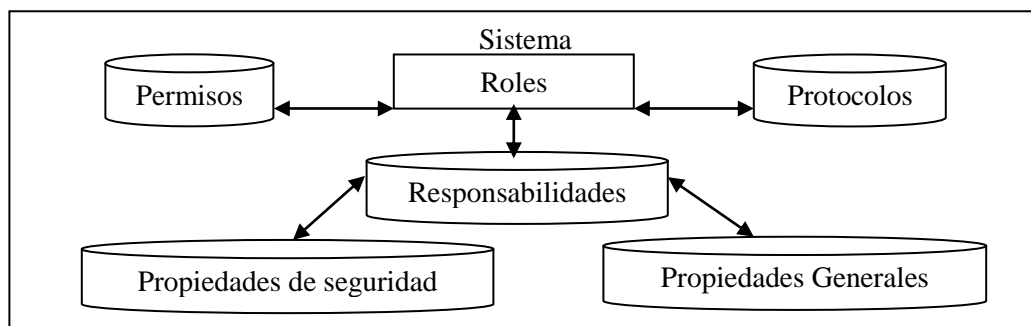


Fig. 16.- Modelo GAIA.

Dado que *GAIA* contempla la definición de permisos, roles, protocolos, responsabilidades, y propiedades generales y de seguridad, estos elementos se retoman para enriquecer la propuesta integral de una arquitectura híbrida que permita incluir conceptos relacionados con el tema de seguridad como los antes mencionados.

2.2.3 De *MESSAGE* (Methodology for Engineering Systems of Software Agents), donde se propuso utilizar meta-modelos para representar a los agentes, organizaciones, dominio, interacciones, tareas y objetivos, se toman las ideas relacionadas con:

- ❖ Utilizar los siguientes conceptos:
 - Goal (objetivo).
 - Role (funcionalidad).
 - Agent (Entidades de agentes).
 - Information Entity (Información de identidad de agentes).
 - Event (evento).
 - Action (acciones que tienen impacto en el ambiente (efectores)).
 - Task (tarea).
 - Resource (Bases de datos o programas externos).
 - Direct Action (efectores).
 - Communicative Act (actos comunicativos).
- ❖ Presentar notaciones UML [46] cuando sea posible para diagramas de clase y actividad e incorporar *AUML* [47] para diagramas de interacción.
- ❖ Utilizar como base el esquema de funcionamiento del sistema de agentes incluyendo las modificaciones a implementar por *SISAG-W* (ver Figura 17).

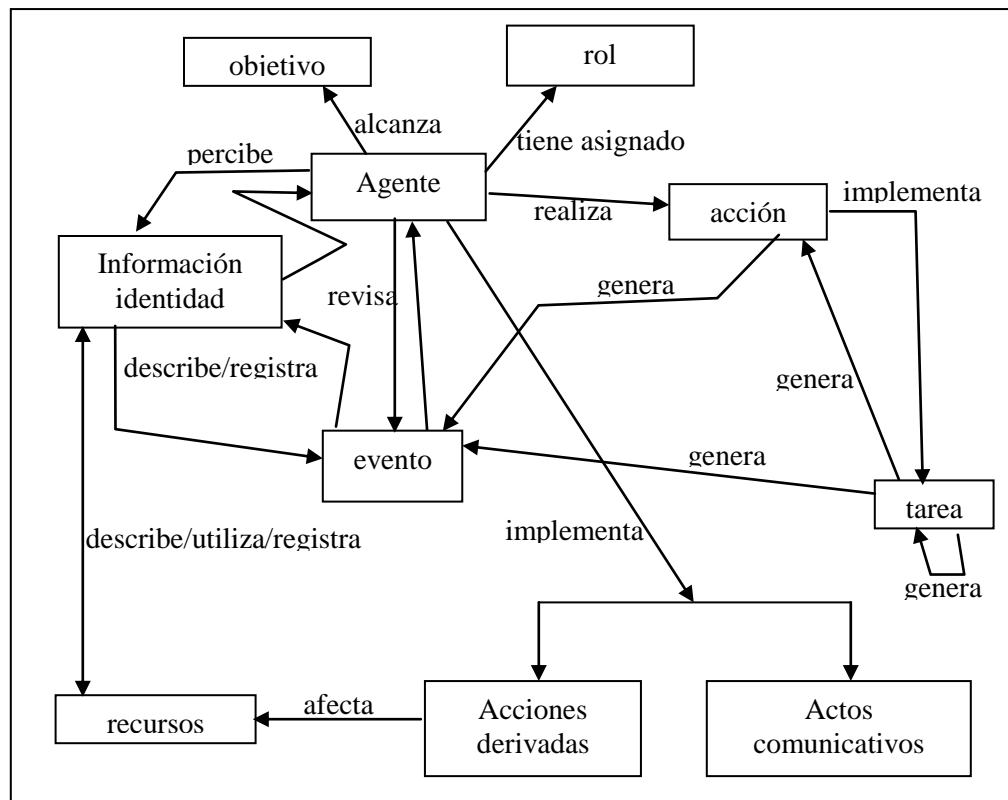


Fig. 17.- Modelo de agentes para *SISAG-W*.

2.2.4 De *PRS*, se toman ideas como la de permitir aplicar un razonamiento para sistemas de tiempo real, utilizando para ello elementos como una base de datos, creencias del agente, un conjunto de objetivos a alcanzar, una biblioteca de procedimientos que describen la forma de actuar del mismo, y una estructura de intenciones que se representa a través de un conjunto de planes a seguir para el alcance de los objetivos, además de un intérprete que funciona como mecanismo de inferencia con capacidad para seleccionar y modificar planes.

2.2.5 Diseño de la arquitectura de agentes *ARSEC-AMS*.

Una vez que se revisaron las especificaciones de las arquitecturas existentes, se llegó a la conclusión de que, en forma individual, ninguna cumple totalmente con los requerimientos necesarios para diseñar sistemas de seguridad, y que podrían retomarse algunas ideas de las arquitecturas como las mencionadas en los puntos anteriores, para elaborar la propuesta original de la arquitectura híbrida para abordar problemas de seguridad con tecnología de agentes *ARSEC-AMS* (ver Figura 18), en la cual se contempla el uso de cuatro módulos: reactivo, deliberativo-cognitivo, de seguridad y el de control de pizarra, de tal forma que las señales que se reciben en el sistema a través de sensores, son filtradas por un intérprete que se encarga de actuar en forma coordinada, y dar un seguimiento al conjunto de acciones a realizar para alcanzar los objetivos correspondientes.

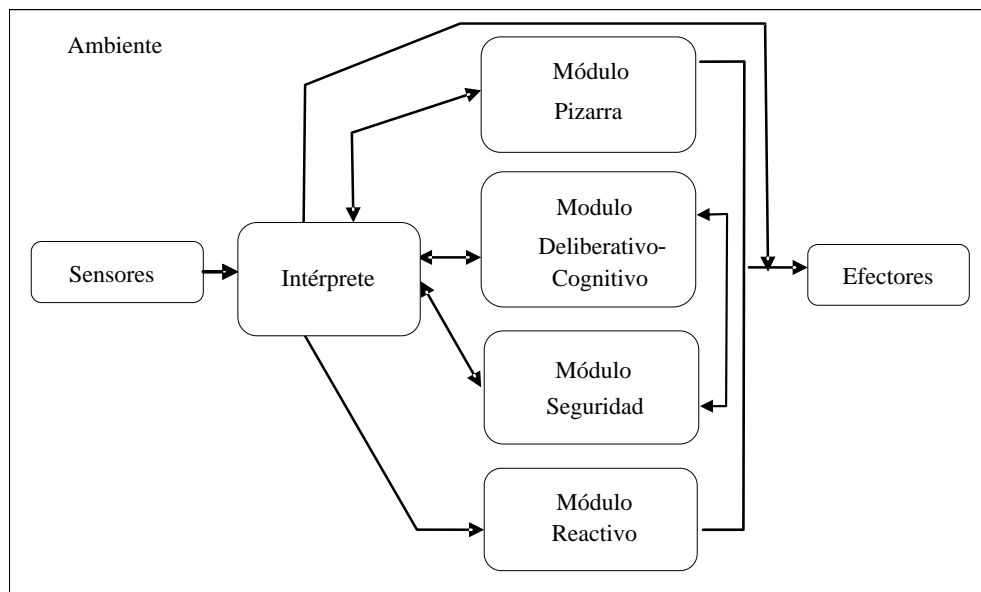


Fig. 18.- Arquitectura *ARSEC-AMS*.

Por último, cabe mencionar que la descripción de los módulos que componen la arquitectura *ARSEC-AMS*, se presentará en los siguientes capítulos, incluyendo las especificaciones de *CLASS-W* relacionadas con cada uno de ellos.

2.3 Esquema funcional de agentes.

Tomando en cuenta que para el buen desempeño de un sistema de seguridad, se debe de contar con procedimientos para registro de eventos que ocurren en el medio ambiente, que serán

tomados como base para el procedimiento de análisis y toma de decisiones razonadas, en este trabajo se propone utilizar una 'pizarra' que se constituya en uno de los elementos básicos del esquema funcional de agentes (ver Figura 19), la cual será descrita más adelante.

Además de lo anterior, en los siguientes puntos se definen aspectos de coordinación, como lo es la asignación de roles a los agentes para delimitar el alcance de sus funciones, y tener una forma de comunicación interna basada en protocolos de comunicación, que permitan establecer las reglas de interacción.

2.3.1 Roles de agentes.

Los roles que se utilizan en este trabajo, han sido diseñados para establecer las distintas funcionalidades de los tipos de agente del sistema, asociando a cada rol, entre otras cosas, conceptos como el de responsabilidad, permiso y privilegio, que a su vez tienen definiciones íntimamente ligadas con aspectos de seguridad, y además, están identificados con un nivel jerárquico que define su grado de autoridad, ventajas y limitaciones.

Es importante aclarar que de los casos que se describen en esta sección, **AMS** y **DF**, son perfiles que han sido introducidos por ser elementos obligatorios definidos en los estándares dictados por **FIPA**, e implementados automáticamente en la plataforma de **JADE**, que ha sido utilizada para probar el sistema de agentes **SISAG-W**.

Enseguida se presentan los tipos de roles que se aplican en el sistema, ordenándolos por nivel de jerarquía, de mayor a menor autoridad:

- a) **AMS**.- Administrador general del sistema de agentes.
- b) **DF**.- Administrador del registro de agentes y servicios.
- c) **AC**.- Administrador del trabajo especializado del conjunto de agentes.
- d) **AL**.- Administrador del funcionamiento interno del equipo donde se crea.
- e) **AR**.- Supervisor del comportamiento de la red del equipo donde se crea.

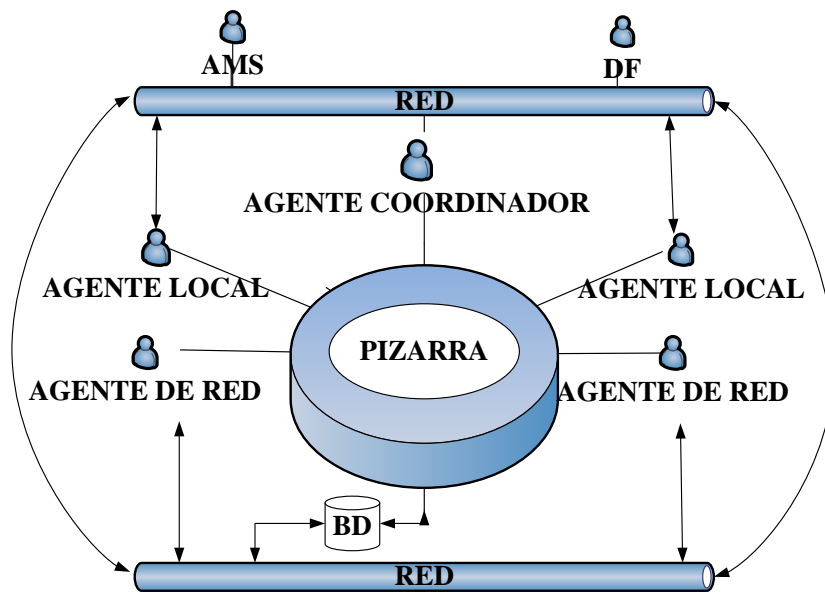


Fig. 19.- Esquema funcional de agentes de ARSEC-AMS.

Según puede observarse en la Figura 19, el esquema general interno del sistema de agentes se basa en el intercambio de mensajes a través de la red, estructurados de tal forma que pueden ser analizados y comprendidos por los agentes receptores, y que están relacionados con el cumplimiento de alguna de sus funciones.

Por otra parte, es importante tomar en cuenta que los sistemas que se desarrollen con los elementos planteados en este documento, tienen como principal objetivo alcanzar un alto nivel de seguridad en el ámbito en que se desenvuelven, por lo tanto, el intercambio de información entre sus miembros debe tener un buen grado de protección, de tal forma que los mensajes no puedan ser interceptados y/o analizados por entidades externas al sistema a través de intrusiones.

Para resolver el problema planteado anteriormente, queda a elección del desarrollador elegir e implementar alguno de los métodos criptográficos existentes que permitan enmascarar los datos en forma eficiente y segura. Sin embargo, con el afán de ejemplificar, más adelante se presentará una propuesta como esquema de protección de la información que se transmite a través de la red por el sistema de agentes.

2.3.2 Esquema interno de comunicación.

Tomando como base el protocolo '*request_protocol*' propuesto por *FIPA* [48], y con el fin de que el sistema de comunicación de agentes sea eficiente, en este trabajo se plantea utilizar un esquema básico (ver Figura 20), que permita llevar un seguimiento de las transacciones que se puedan realizar. El procedimiento para lograr este objetivo, consiste en establecer un protocolo de comunicación que sirva como base para implementar los mensajes relacionados con las siguientes opciones:

- a) Solicitud para realizar una acción.- Un agente puede solicitar a otro la realización de una acción.
- b) Aceptar comunicación.- Este mensaje, además de informar sobre la aceptación de realizar la acción solicitada, sirve como un aviso de que el mensaje transmitido ha llegado en forma correcta a su destino.
- c) Envío de información.- Consiste en enviar un mensaje que transmita cierta información sin requerir respuesta.
- d) Informe de rechazo.- Cuando por alguna razón, la solicitud de un agente no puede ser atendida, se envía un mensaje de rechazo, y opcionalmente, se agrega información adicional que indica el motivo por el cual no se puede realizar la acción solicitada o no puede aceptar establecer una sesión de trabajo.
- e) Error en secuencia de comunicación.- Cuando una comunicación ha sido interrumpida antes de llegar a una confirmación de finalización de la misma por alguna de las partes, pueden emitirse mensajes que informen sobre las fallas de comunicación, y opcionalmente, pueden incluirse reportes con indicios que permitan detectar el origen del problema. En este caso existe la posibilidad de reiniciar completamente la transacción, o continuarla desde el punto donde fue interrumpida.
- f) Error en el mensaje.- Si un mensaje no puede comprenderse pero se tienen los datos de referencia del emisor, puede enviarse un mensaje informado la situación para que se re programe el envío del mensaje original.
- g) Error en comunicación por red.- Además de los casos contemplados en el protocolo '*request_protocol*' de *FIPA* [48], en esta propuesta se incluye una opción adicional

que permite al agente emisor, en base a un tiempo de espera previamente definido, detectar si el mensaje enviado ha llegado a su destino. Cuando se presenta este tipo de error, se recomienda reprogramar la transacción para el envío del mensaje.

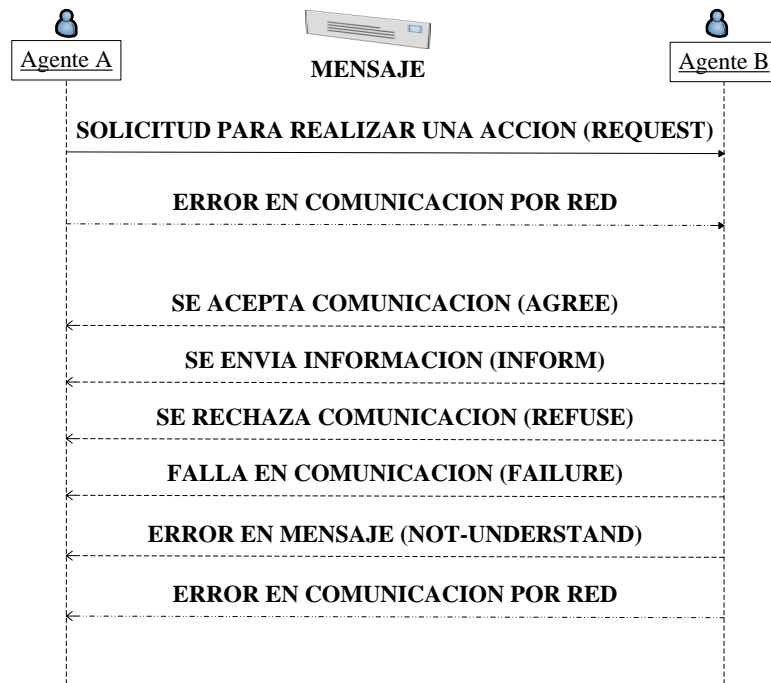


Fig. 20.- Esquema básico para envío/recepción de mensajes.

En base a lo descrito anteriormente, y para efectos de mantener un control interno de las sesiones de trabajo que establezcan los agentes, y el envío/recepción de mensajes interactivos, se ha diseñado una transacción específica identificada como 'sesión', a través de la cual se implementan los siguientes casos (ver Figura 21):

- a) *Iniciar sesión.*- Para que un agente pueda solicitar a otro que se realice una acción o se identifiquen como válidos los mensajes recibidos, entre otras cosas, es necesario que exista una sesión establecida, registrada en un contenedor de datos controlado en forma individual por cada agente, que básicamente contenga el nombre y el IP o dirección de red del equipo de cómputo donde fue creado el agente con el cual se intercambia información. En caso de que una sesión se finalice por cualquier causa, el registro de ésta debe ser eliminado de la lista en forma inmediata.
- b) *Sesión establecida.*- Cuando un agente recibe una solicitud para iniciar una sesión, puede responder con un mensaje de 'Sesión establecida', con lo cual acepta mantener un canal de comunicación para envío y recepción de mensajes. Una vez que se establece sesión, deben agregarse a la lista de control de sesiones de trabajo interna, los datos correspondientes.
- c) *Petición rechazada.*- Si el agente que recibe una solicitud para iniciar una sesión de trabajo, por alguna razón, en esos momentos no puede responder con el mensaje de 'Sesión establecida', debe rechazar la petición, y opcionalmente, puede agregar información que detalle o explique el motivo de este tipo de respuestas.
- d) *Cerrar sesión.*- Si una transacción ha llegado a su fin o la sesión de trabajo debe finalizarse, el mensaje que debe enviarse es el que corresponde a 'Cerrar sesión'.

- e) *Sin respuesta*.- Atendiendo a lo estipulado en el esquema básico, esta es una opción que permite verificar si los mensajes enviados han llegado a su destino o se han presentado errores en la transmisión de los datos.

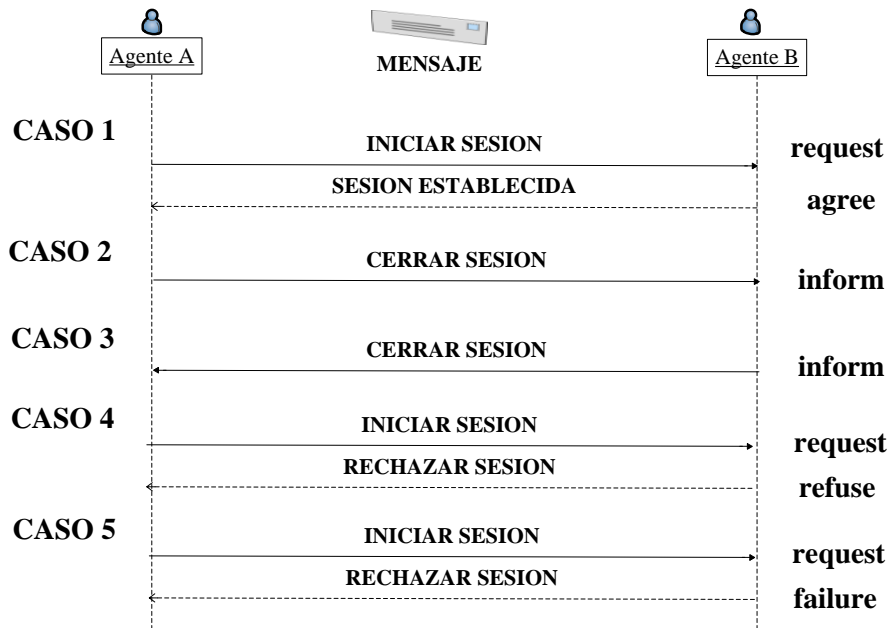


Fig. 21.- Esquema para control interno de Sesiones de Trabajo.

En el control interno planteado para manejo de sesiones, en los casos 1, 4 y 5 (ver Figura 21), el agente que envía el mensaje 'Iniciar sesión', debe establecer un tiempo límite para obtener respuesta del receptor, que indique si falla, acepta o rechaza su petición, con el fin de reprogramar las peticiones de sesión que no hayan sido atendidas pero que deben llevarse a cabo en un momento dado.

En base a lo mencionado anteriormente, y para que las sesiones de trabajo puedan implementarse en un sistema de agentes, se ha diseñado la parte gramatical de *CLASS-W*, que contiene las siguientes reglas o producciones para implementar el procedimiento que corresponde al intercambio de mensajes en la transacción identificada como 'sesion':

```

<sesion> ::= <inicio_sesion> | <cerrar_sesion>
           | <establecer_sesion> | <rechazar_sesion>
           | <error_sesion>
<inicio_sesion> ::= 'inicio_s' \/' <agente>
<cerrar_sesion> ::= 'cerrar_s' \/' <agente>
<rechazar_sesion> ::= 'rechazar_s' \/' <agente>
<establecer_sesion> ::= 'estab_s' \/' <agente>
                  \/' 'ref' \/' <agente>

```

Una vez que se han definidos los roles y el esquema interno de comunicación del sistema, el siguiente paso consiste en modelar los comportamientos que tendrán los distintos tipos de agentes, de tal forma que se adecuen a la arquitectura propuesta para sistemas de seguridad, para que su funcionamiento sea acorde al rol y nivel jerárquico

que le corresponda. A continuación se hará una descripción de los mecanismos que se han diseñado para modelar solamente el comportamiento de los agentes *AC*, *AL* y *AR*, ya que como antes se ha mencionado *AMS* y *DF* son elementos que se implementan para respetar los estándares fijados por *FIPA*, que son implementados en *JADE*, que es la plataforma utilizada para probar el sistema *SISAG-W*.

2.4 Descripción del comportamiento general de agentes.

El comportamiento de los agentes puede presentar variaciones, que dependerán de los mensajes entrantes relacionados con los eventos que ocurren en su medio ambiente, pero que también pueden ser motivadas por actualizaciones del estado mental, derivadas de los cambios contextuales del medio ambiente, relacionados con las especificaciones basadas, tanto en las creencias proporcionadas a través de estructuras de control, como en la identificación de funciones que corresponden al rol y nivel jerárquico al que pertenecen.

En la arquitectura *ARSEC-AMS*, en el módulo '*interprete*', se identifica el significado de los mensajes relacionados con los eventos filtrados por los sensores de entrada al sistema, y se modelan dos tipos de comportamiento que se pueden presentar, donde uno corresponde a los agentes de tipo coordinador (*AC*) (ver Figura 22), y el otro, a los de que se identifican como agentes locales (*AL*) y de red (*AR*) (ver Figura 23).

Para que los agentes lleven a cabo tareas como la de inicialización interna, interpretación de mensajes entrantes, canalización de transacciones hacia el resto de los módulos de la arquitectura y emisión de mensajes a otros agentes, a continuación se presenta una descripción de los pasos a seguir en cada uno de los comportamientos antes mencionados:

a. Comportamiento para el Agente Coordinador (*AC*):

Los agentes de tipo coordinador (ver Figura 22), desempeñan un rol que les permite coordinar el trabajo conjunto de un grupo de agentes conformado por agentes de tipo local y de red, y además son los que pueden ser programados para realizar entre otras cosas, tareas especializadas como las que a continuación se indican:

- a) Obtener los datos relacionados con los agentes del sistema, y crear una lista con los datos correspondientes.
- b) Crear una lista de sesiones de trabajo con el resto de los agentes y llevar un control del estado en que se encuentran.
- c) Crear instancias de la pizarra de acuerdo a la temática registrada.
- d) Clausurar instancias de la pizarra.
- e) Llevar a cabo el análisis de las instancias de la pizarra para detección de situaciones anormales y tomar las decisiones correspondientes para atender los sucesos que se hayan registrado.
- f) Tomar en cuenta tanto los lineamientos proporcionados por el administrador del sistema, como la definición de criterios para el período de tiempo de revisión, políticas, niveles de seguridad, alertas, acciones a tomar, etc.
- g) Emitir mensajes al administrador cuando se presenten casos en los cuales existen sospechas de un problema de seguridad, y tomar las medidas necesarias para resguardar los recursos involucrados, mientras se registran las alertas y acciones a tomar en adelante sobre el mismo tipo de eventos, o se desactiva la protección por indicaciones del mismo administrador.

- h) Participar como coordinador y entidad principal en el esquema de protección de la información que se transmite a través de la red.

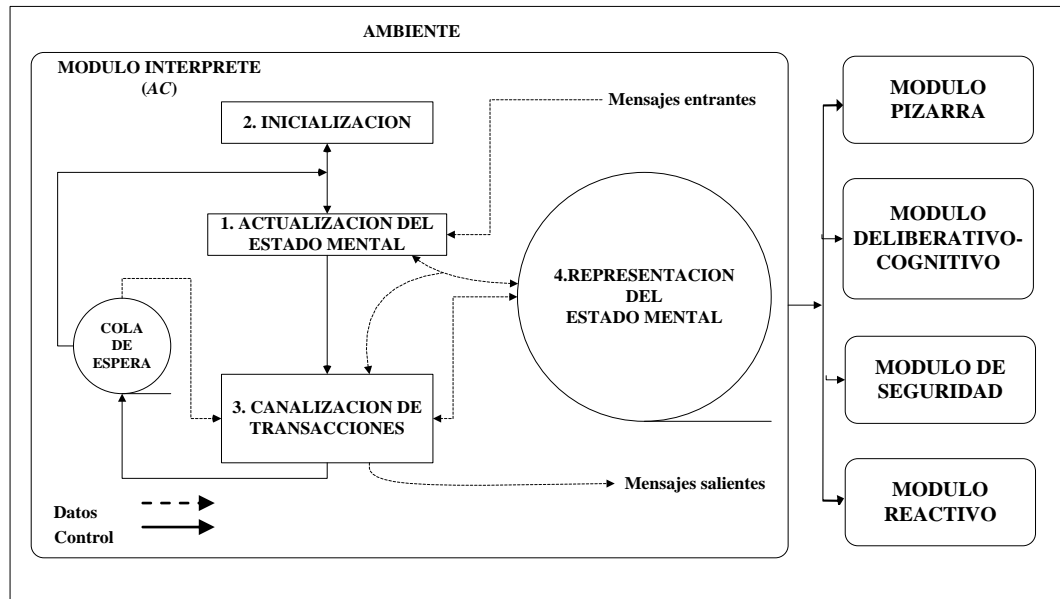


Fig. 22.- Flujo del intérprete del Agente Coordinador.

Enseguida, se presenta una descripción del flujo de control del intérprete para agentes coordinadores (AC), y la funcionalidad de los componentes especificados en la Figura 22 que le corresponden a este módulo:

Proceso 1.- Actualiza el estado mental, llevando a cabo las siguientes acciones:

- Inicializar estado mental y capacidades (Relación con el Proceso 2).
- Proceso de canalización de transacciones (Relación con el Proceso 3).
- Recepción de mensajes entrantes (entradas/sensores).

Proceso 2.- Inicializa el estado mental y las capacidades del agente que tienen relación con:

- Recolectar información relacionada con el entorno (creencias).
- Representación del estado mental (Relación con datos secuenciales 4).

Proceso 3. Proceso de canalización de transacciones, donde se pueden presentar casos relacionados con:

- La ejecución de acciones realizada en tiempo y forma, en cuyo caso se actualiza la información del estado del objetivo a alcanzar.
- La ejecución falla o no puede realizarse de momento, en cuyo caso, la transacción se pondrá en lista de espera para tratar de llevarse a cabo más tarde.
- Se envían mensajes salientes (salidas/acciones/efectores).

Datos Secuenciales 4. Corresponde a la representación del estado mental, que consiste en realizar acciones como las que a continuación se mencionan:

- Recolectar información relacionada con el evento o mensaje recibido (creencias).
- Recolectar información relacionada con respecto a planes a seguir (intenciones).
- Registrar la meta o el objetivo a alcanzar (deseos).
- Registrar el estado actual de los objetivos registrados, donde el estado indica si el objetivo se encuentra: iniciado, suspendido o finalizado (con error, correcto, inconcluso).

b. La Figura 23 corresponde al funcionamiento del intérprete de los agentes locales (**AL**) y de red (**AR**).

Los agentes locales son entidades que se incluyen como figuras de soporte básico, ya que sus funciones están orientadas al cuidado, vigilancia y protección del equipo donde se crean, y al igual que los agentes de red, están supeditados a obedecer las órdenes de los agentes de tipo coordinador que pueden estar relacionadas entre otras cosas, con aplicación de políticas, vigilancia y control de procesos internos del equipo, adición, eliminación o reconfiguración de claves del registro y tareas que se cargan cuando se inicia el sistema operativo, etc.

Por otra parte, los agentes de red son agentes de tipo reactivo, sólo se encargan de tareas relacionadas con la vigilancia del funcionamiento de red del equipo donde se crean, y de aplicar reglas previamente generadas por el algoritmo 2G, que más adelante se describe, y que se basa en la teoría de reconocimiento de patrones y árboles de decisión.

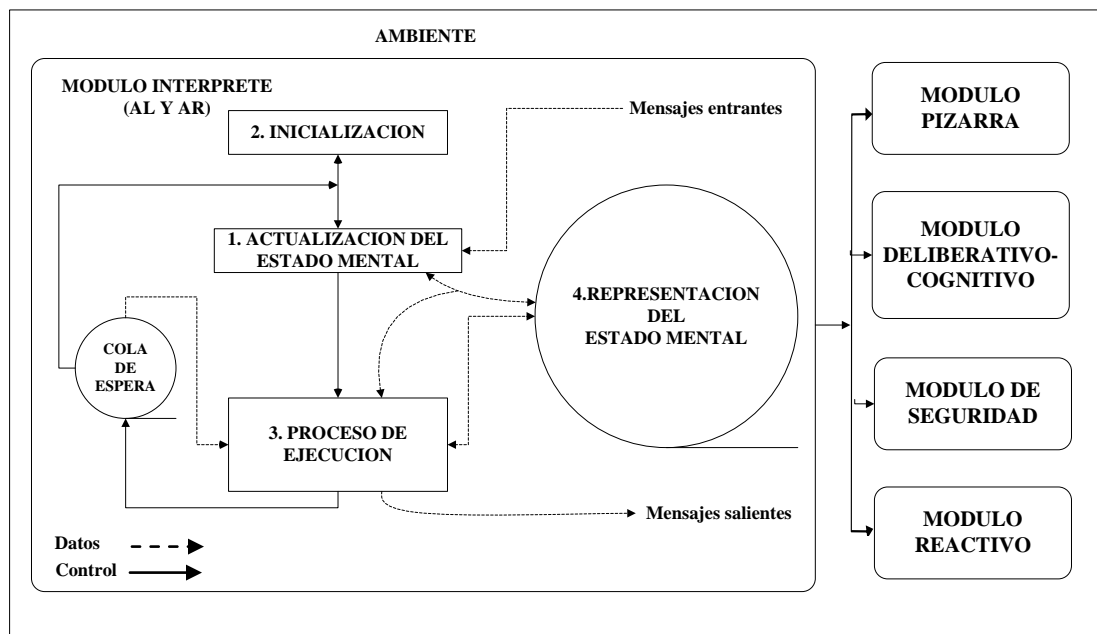


Fig. 23.- Flujo del intérprete del Agente Local y de Red.

La descripción del flujo de control del intérprete para agentes locales (**AL**) y agentes de red (**AR**), así como de la funcionalidad de los componentes especificados en la Figura 23 que le corresponden a este módulo, es la siguiente:

Proceso 1.- Actualiza el estado mental realizando las acciones como:

- Inicializar estado mental y capacidades (Relación con el Proceso 2).
- Proceso de canalización de transacciones (Relación con el Proceso 3).

- Recepción de mensajes entrantes (entradas/sensores).

Proceso 2.- Inicializa el estado mental y las capacidades del agente que consiste en:

- Recolectar información relacionada con el entorno (creencias).
- Representación del estado mental (Relación con el objeto de datos secuenciales 4).

Proceso 3.- Se encarga de ejecutar las acciones solicitadas por agentes de tipo coordinador, pudiendo presentarse los siguientes casos:

- La ejecución de las acciones se realizó en tiempo y forma, en cuyo caso se actualiza la información del estado del objetivo a alcanzar.
- La ejecución falló o no puede aplicarse de momento, en cuyo caso se pondrá en lista de espera para tratar de realizarse más tarde.
- Se encarga del envío de mensajes salientes (salidas/acciones/efectores).

Datos secuenciales 4.- Representación del estado mental, que se realiza a través de acciones como:

- Recolectar información relacionada con el evento o mensaje recibido (creencias).
- Recolectar información relacionada con respecto a planes a seguir (intenciones).
- Registrar la meta o el objetivo a alcanzar (deseos).
- Registrar el estado actual de los objetivos registrados, donde el estado indica si el objetivo se encuentra: iniciado, suspendido o finalizado (con error, correcto, inconcluso)

En el siguiente apartado, se presentan dos casos de uso que permiten ejemplificar comportamientos basados, entre otras cosas, en las especificaciones de roles y niveles jerárquicos, y se muestran procesos interactivos de agentes para la toma de decisiones razonadas, que pueden servir para resolver o prevenir un posible ataque de seguridad que pudiera estar gestando, y que si no se detiene, los daños que se pueden ocasionar en los equipos pueden ir desde leves hasta severos, y en algunos casos llegar a ocasionar la pérdida parcial o total de recursos, de información, o incluso de hardware.

2.5 Ejemplos de casos de uso.

A continuación se presentan casos con los cuales se pretende mostrar la asignación de algunas de las tareas y funciones que están perfiladas en el rol que juegan los agentes de tipo coordinador (**AC**) y de red (**AR**):

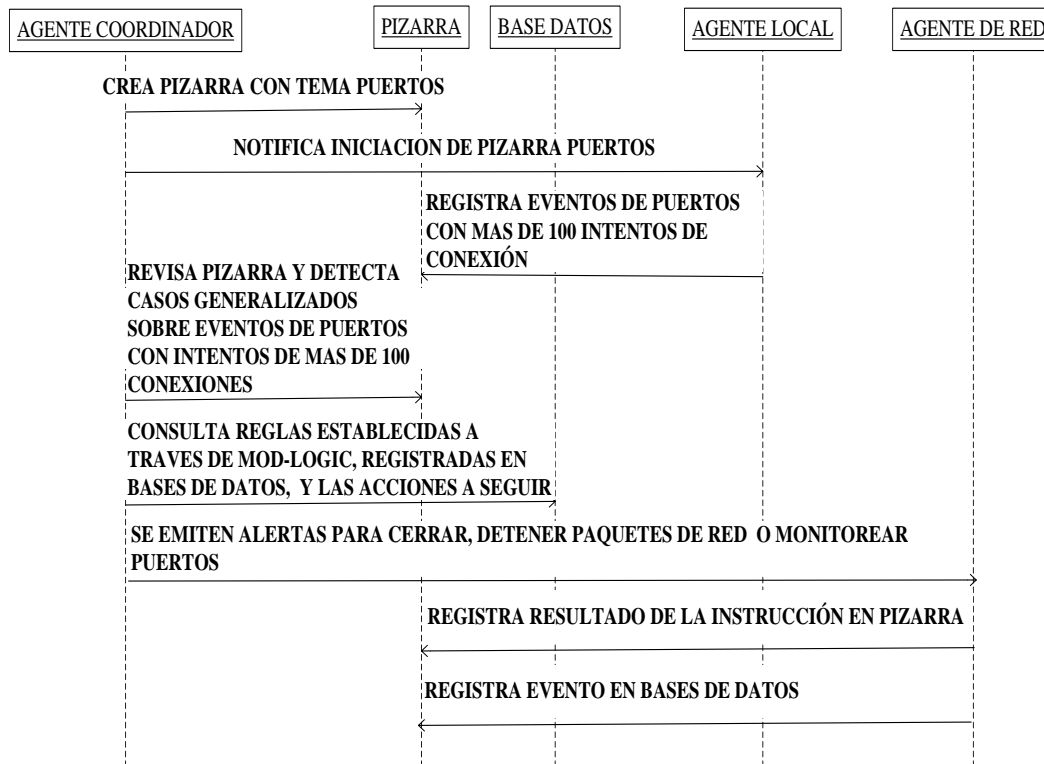
- Ejemplo 1** (ver Figura 24).- El agente coordinador (**AC**) analiza la pizarra relacionada con el tema 'puertos', y específicamente se revisa el 'caso de uso para detección de problema generalizado con un puerto que registra más de 1,000 intentos de conexión'. En este caso, se supone que el agente coordinador (**AC**) se encarga de registrar el evento en la base de datos correspondiente, de consultar el plan o grupo de intenciones a seguir, y de reconocer el objetivo o deseo que se debe alcanzar para resolver el problema. Para ello, se auxilia del módulo deliberativo-cognitivo, y posteriormente envía mensajes a los agentes de red (**AR**), mismos que accederán al módulo de seguridad para programar las acciones que se deben ejecutar, que en este caso pueden ser: revisar el estado del puerto, cerrarlo si está abierto, y

al módulo reactivo para registrar una alerta a fin de que se realice un monitoreo constante, y por último se registra el resultado obtenido.

CASO DE USO: ANALISIS DE PIZARRA.

TEMA: PUERTOS

OBJETIVO: DETECTAR PUERTOS ABIERTOS CON MÁS DE 100 INTENTOS DE CONEXIÓN POR PERIODOS DE 10 MINUTOS



Fi

g. 24.- Caso de uso: Análisis de Pizarra, Tema: 'Puertos'.

- b) **Ejemplo 2** (ver Figura 25).- El agente coordinador (AC) analiza la pizarra relacionada con el tema 'tareas de inicio', y en el 'caso de uso para detección de problema generalizado con cadenas que se introducen en el Registro de Windows para programar aplicaciones que se carguen como tareas de inicio – actividad usual que realizan los virus, especialmente los llamados 'gusanos' – que pueden ser dañinas para el buen funcionamiento de un equipo'. El agente coordinador (AC) se encarga de registrar el evento en la base de datos correspondiente, de consultar el plan o grupo de intenciones a seguir, y de reconocer el objetivo o deseo que se debe alcanzar para resolver el problema. Para ello, se auxilia del módulo deliberativo-cognitivo, y posteriormente envía mensajes a los agentes locales (AL), mismos que acceden al módulo de seguridad para programar las acciones que se deben ejecutar, que en este caso pueden ser: revisar si la cadena existe en el registro como tarea de inicio y eliminarla, revisar si el archivo ejecutable a que se hace referencia con esa cadena ya se encuentra en el sistema, y de ser así, proceder a su eliminación, el estado del puerto, cerrarlo si está abierto, registrar una alerta en el módulo reactivo para monitoreo constante, y por último registrar el resultado obtenido.

OBJETIVO: DETECTAR TAREAS DE INICIO EN EL REGISTRO, CONSIDERADAS COMO DAÑINAS

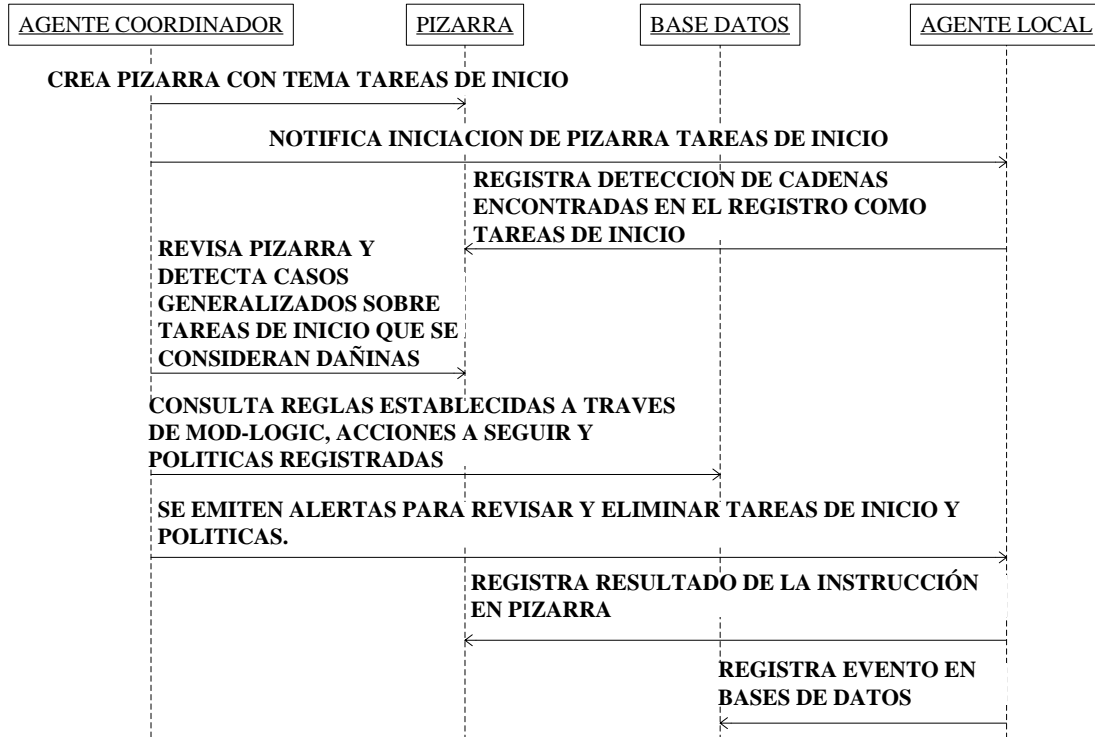


Fig. 25.- Caso de uso: Análisis de Pizarra, Tema: ‘Tareas de Inicio’.

2.6 Resumen.

Una vez realizado el análisis sobre los aspectos que se deben de tomar en cuenta para el desarrollo de sistemas de seguridad basados en agentes, en relación a los elementos básicos que se requieren para realizar esta tarea, a continuación se presentan las siguientes reflexiones:

En la etapa de diseño de este tipo de proyectos, es indispensable tener como soporte una arquitectura que permita fijar los módulos necesarios para abordar la temática de que se trate. En este caso, al no haber encontrado una que cubriera totalmente los requerimientos para el tema de seguridad, se presenta la propuesta de una arquitectura híbrida denominada *ARSEC-AMS*, conformada por el módulo *‘interprete’* que identifica el significado de los sucesos de entrada del sistema que son filtrados por los sensores, por el módulo *‘pizarra’*, a través del cual se registran los eventos en una pizarra, el módulo deliberativo-cognitivo, el módulo de seguridad y el módulo reactivo, que se retroalimentarán de la información que es organizada a través de estructuras de control internas.

Por otra parte, desde el punto de vista organizacional, es necesario identificar a un conjunto de entidades abstractas y definir su funcionalidad, de tal forma que se asignen roles que pueden o no estar asociados a una jerarquía, y puedan asumir formas de comportamiento que permitan el cumplimiento de los objetivos del sistema. En nuestra propuesta, se contempla el uso de agentes

coordinadores (*AC*), locales (*AL*) y de red (*AR*), y se incluye tanto al agente manejador del sistema (*AMS*), como al facilitador de directorio (*DF*), estandarizados por *FIPA* e implementados en la plataforma *JADE*, que se usa para efectos de prueba en este trabajo, y además, se establece una jerarquía, donde cada tipo de agente sólo puede desempeñar funciones de acuerdo al nivel que le corresponde, y los privilegios y responsabilidades otorgados para acceder a los recursos del sistema son restringidos conforme el nivel jerárquico disminuye de nivel.

Para especificar un esquema interno de comunicación se diseñó un protocolo para controlar las transacciones y sesiones de trabajo, y para efectos de organizar la información que será utilizada en los procesos de análisis, se usa como elemento de registro de eventos, una pizarra instanciada por temas.

Además, tomando en cuenta que se debe utilizar un lenguaje para el intercambio de mensajes de agentes, se ha decidido utilizar el formato del lenguaje de comunicación *ACL*, que ha sido especificado por *FIPA*, y que es el soporte de la plataforma *JADE*, a través del cual se implementa un lenguaje de contenido generado por la gramática *CLASS-W*, la cual ha sido diseñada para el desarrollo de sistemas de seguridad basados en agentes, por no haberse encontrado en la bibliografía revisada, un lenguaje que cumpliera fácil y eficientemente con este objetivo.

Para continuar con el desarrollo de este documento, los módulos de la arquitectura *ARSEC-AMS*, se irán detallando en los siguientes capítulos, incluyéndose las reglas gramaticales de *CLASS-W*, y los detalles de correspondencia con los componentes físicos del sistema *SISAG-W*, así como los aspectos relativos al esquema de protección de la información que se intercambia por los agentes a través de una red de equipos de cómputo.

Capítulo 3.- Módulo Pizarra y Deliberativo-Cognitivo.

3.1 Módulo manejador de pizarra *BLACKBOARD-ECRE*.

Si bien es cierto que uno de los recursos más importantes de un sistema de agentes, es la información que se define para efectos de organización y funcionamiento, o la que se genera en forma dinámica durante sus períodos de actividad, y que su significado e interpretación, proporciona un medio de identificación de patrones en los sucesos que se presentan en el medio ambiente, también lo es el elemento que se utilice para registrarla.

Es por ello que a partir del análisis realizado sobre las posibles alternativas existentes relacionadas con lo antes expuesto, se ha decidido implementar una arquitectura de pizarra como uno de los módulos de la arquitectura *ARSEC-AMS* denominada *BLACKBOARD-ECRE* (Blackboard-Element Control for Register of Events), y para ello se han tomado como base las especificaciones que se mencionan en los siguientes párrafos.

La arquitectura de pizarra sirve como medio de comunicación en un sistema de agentes a través de sus componentes: una pizarra, un conjunto de fuentes de conocimiento (KSs, Knowledge Sources), y un mecanismo de control [34].

En este proyecto, el elemento mencionado anteriormente se ha diseñado como se muestra en la Figura 26, y se representa a través de instancias de una variable de relación de una base de datos, las cuales son monitoreadas permanentemente por agentes *AC*, que tienen como función:

- ❖ Revisar continuamente la información que es contenida en cada una de ellas a través de un ciclo de control (Ver Figura 27).
- ❖ Llevar un seguimiento del comportamiento de las mismas.
- ❖ Realizar procesos de análisis para toma de decisiones.
- ❖ Registro de eventos relevantes.
- ❖ Selección de acciones que permiten buscar una solución a los problemas que se presentan.
- ❖ Llevar un control y seguimiento para alcanzar un objetivo.

Para ello se ha tomado como base la arquitectura de pizarra Hearsay-II (ver Figura 28), la cual permite el registro de eventos en una pizarra, utiliza un elemento de control (monitor) para identifica los disparadores (triggers) que deben realizarse, y permite llevar un registro a través de una agenda, tener un control sobre la base de datos, y usar un programador (scheduler) que actualiza las fuentes de conocimiento que retroalimentan a la pizarra [45].

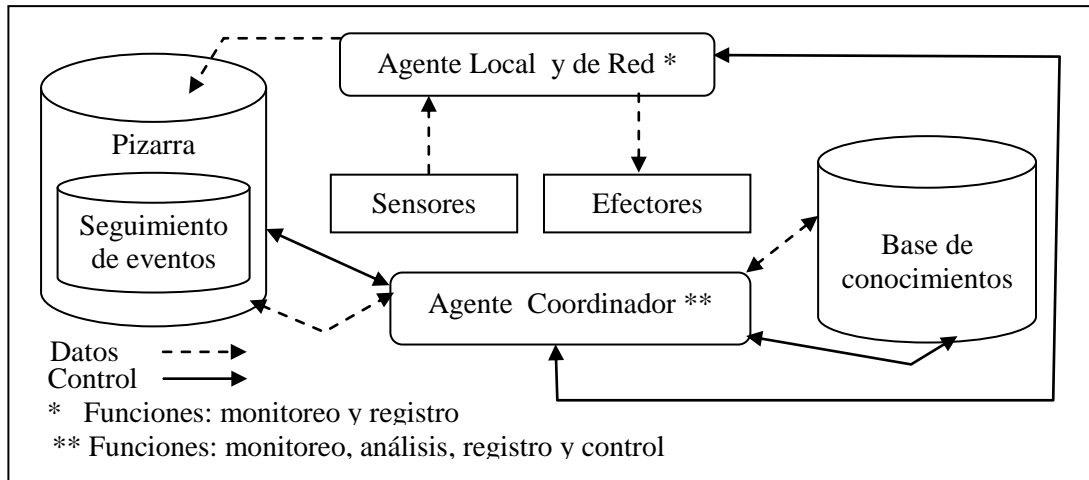


Fig. 26.- Arquitectura de pizarra BLACKBOARD BLACKBOARD-ECRE.

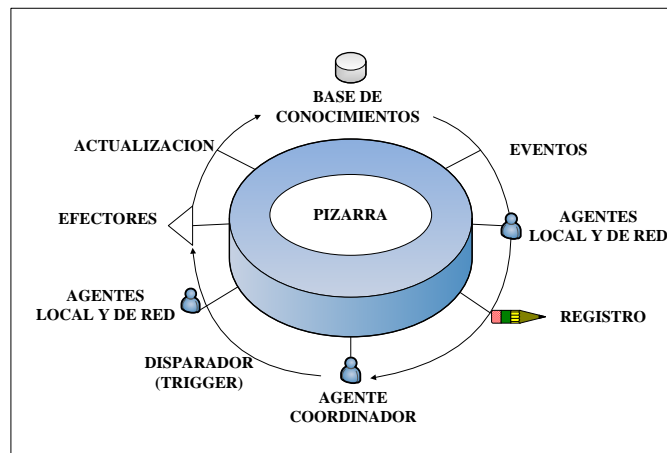


Fig. 27.- Ciclo de Control para el manejo de pizarra BLACKBOARD BLACKBOARD-ECRE.

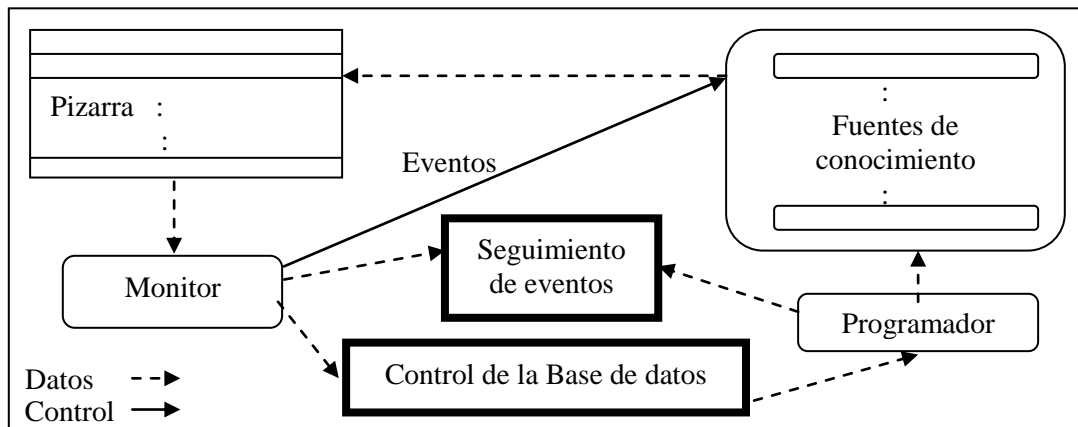


Fig. 28.- Arquitectura de pizarra Hearsy-II's.

Para el funcionamiento del módulo pizarra, en la gramática *CLASS-W* se han incluido las producciones gramaticales que se describen en el siguiente punto.

3.1.1 Aspectos gramaticales de CLASS-W para uso de la pizarra.

En la gramática CLASS-W se ha incluido la transacción para el uso del módulo pizarra, que incluye la implementación de reglas por medio de las cuales el grupo de agentes interactúa cuando se detectan situaciones consideradas como inusuales, anormales o dañinas, y mantiene control sobre la información que se registra a través de las variables de relación de la base de datos del sistema identificadas con los siguientes nombres (ver Figura 29): 'pizarra', 'aportacion' y 'temas'. Las unidades sintácticas que corresponden a esta transacción son las siguientes:

```

<pizarra> ::= <registro_pizarra> | <notif_pizarra>
           | <fin_pizarra> | 'espera_pizarra' | <operación_pizarra>
<registro_pizarra> ::= 'registro_pizarra' \/' <agente>
                    \/' <ontología_temas>
<notif_pizarra> ::= 'notif_pizarra' \/' <ontología_temas>
<fin_pizarra> ::= 'fin_pizarra' \/' <ontología_temas>
<operación_pizarra> ::= 'id_pizarra' \/' <numero> \/' <agente>
                    \/' <ontología_f> \/' <inf_adicional> \/'
                    <tipo_aportacion> | 'id_pizarra' \/' <numero> \/'
                    <agente> \/' <ontología_f> \/' <tipo_aportacion>
<tipo_aportacion> ::= 'pregunta' \/' <cadena>
                    | 'resp_afirmativa' \/' <cadena> | 'resp_negativa'
                    \/' <cadena> | 'resp_informativa' \/' <cadena>
                    | 'resp_conocimiento' \/' <cadena> | 'propuesta' \/'
                    <cadena> | 'intencion' \/' <cadena> | 'deseo' \/' <cadena>
                    | 'datos_adicionales' \/' <cadena>

```

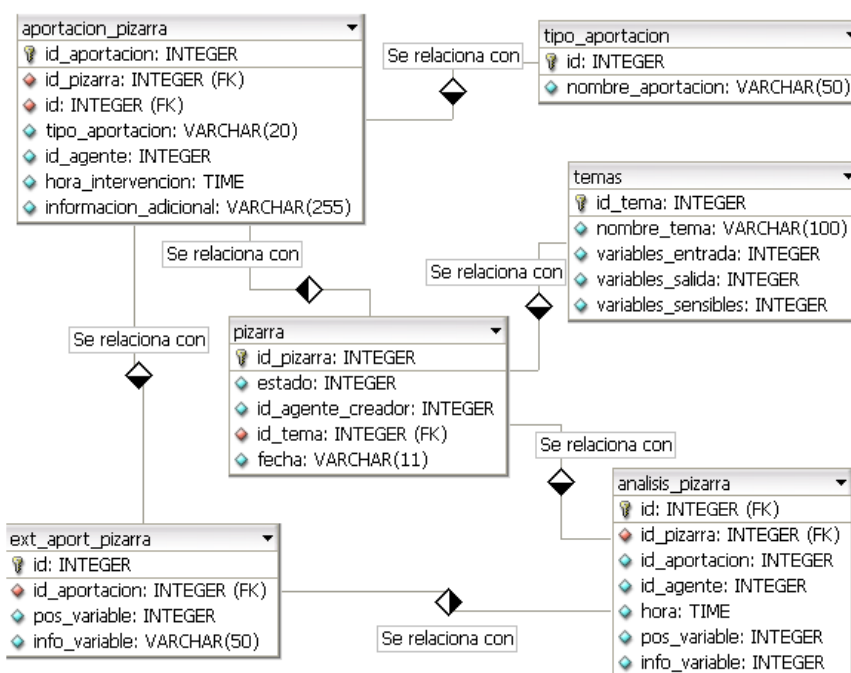


Fig. 29.- Variable de relación: 'pizarra' para SISAG-W.

Debido a que en la estructura de la base de datos que corresponde al módulo pizarra (ver Figura 29), hay atributos que pueden tomar valores de tipo enumerativo, o que incluyen valores que deben tomarse como base pero que pueden extenderse de acuerdo a los requerimientos del diseñador y administrador del sistema, a continuación se hará una descripción donde se aplica el siguiente formato de presentación ‘*tabla-atributo [rango de valores secuenciales]: {lista de valores}*’:

- pizarra-estado [1-3]: {iniciado, resuelto, inconcluso}.
- Pizarra.operación [1-2]: {lectura, escritura}.
- Aportación-tipo_aporación [1-9]: {pregunta, respuesta afirmativa, respuesta negativa, respuesta conocimiento, información, propuesta, intención, deseo, datos adicionales}.
- Temas-ontología_temas [0-*n*]: {procesos, puertos, servicios, registro de tareas de inicio, políticas, configuración, ..., *n*}.

3.2 Módulo deliberativo-cognitivo.

La programación lógica es muy importante en el desarrollo de sistemas del área de Inteligencia Artificial, donde uno de los lenguajes más eficientes y utilizado en este paradigma es Prolog (Programmation en Logique), ya que permite describir la solución de un problema utilizando mecanismos que, a través de la abstracción de conceptos, permiten evaluar distintas opciones para obtener una conclusión a partir de una premisa [49] [50]. Para efectos de contar con un módulo lógico en este proyecto, se ha diseñado un componente externo a *SISAG-W* llamado *Mod-Logic*, desarrollado con lenguaje C/C++, y con soporte para el Sistema Gestor de Base de Datos MySQL, con posibilidades de ser reutilizado en otro tipo de aplicaciones, y que a futuro, pueda extenderse esta funcionalidad para otro tipo de manejadores de bases de datos como SqlServer, Oracle, PostgreSQL, ODBC etc.

Para el diseño y desarrollo de este componente se tomaron en cuenta las siguientes consideraciones:

- a) Compiladores como SWI-Prolog, Visual Prolog, SICStusProlog, etc., son herramientas indispensables para la implementación de lenguajes lógicos, y con algunos de ellos se pueden crear librerías de enlace dinámico, pero éstas últimas no siempre son compatibles con los estándares que se manejan en otro tipo de lenguajes de programación. Para resolver este problema, algunas compañías ofrecen módulos de código que se utilizan como intérpretes de predicados tipo Prolog, que deben incrustarse en los proyectos que se desarrollan para implementar este tipo de funcionalidad, como por ejemplo JIProlog, GNUProlog o JPLProlog para Java, VisualProlog para C/C++, o Sicstus Prolog para Visual Basic, etc., y algunos soportan el uso de Datalog.
- b) Datalog es considerado como un lenguaje declarativo muy útil que permite representar el conocimiento con programación lógica, a través de la implementación de consultas recursivas sobre bases de datos relacionales, sin embargo, para implementarlo, se requiere del uso de compiladores que ofrezcan esta funcionalidad, y que el programador tenga conocimientos avanzados en el área.
- c) Entre los objetivos que se desean alcanzar con este trabajo, por un lado está el de contar con un módulo lógico que ofrezca conectividad con bases de datos relacionales, que permita guardar además de los datos, la estructura de los predicados, y por otra parte, que sea amigable y que los conocimientos de los desarrolladores de este tipo de sistemas puedan ser

básicos en este tipo de programación, lo cual no puede ser implementado con las alternativas que se mencionan en los párrafos anteriores.

Mod-Logic brinda la posibilidad de utilizar bases de datos relacionales para guardar, además de los datos, la estructura y relaciones de predicados lógicos, y cuenta con una interfaz amigable que permite introducir y recuperar información en forma transparente al usuario.

3.2.1 Representación del conocimiento.

Buscando utilizar una forma de razonamiento parecida a la del ser humano para el sistema de agentes, se ha diseñado un módulo lógico tomando en cuenta las siguientes consideraciones:

En computación, para representar el conocimiento se utilizan símbolos -un número o cadena de caracteres- que representan a un objeto o a una idea, por lo cual es necesario que los sistemas desarrollados para trabajar en este ámbito, traduzcan las expresiones a símbolos manejables, de tal forma que se pueda trabajar con las máquinas a nuestro nivel de representación [51].

Para que un sistema de software sea eficiente debe cumplir con las siguientes características [51]:

- Que sean fáciles de modificar por procedimientos manuales o a través de técnicas automáticas.
- Que permitan la incorporación de nuevos conocimientos de forma sencilla.
- Que faciliten la detección de incoherencias y faltas de consistencia.
- Que tengan la posibilidad de reutilizar las sentencias, procedimientos, etc.

En el área de ingeniería del conocimiento, algunos de los principales paradigmas que se usan para diseño de sistemas son [51]:

- Representación procedural.- Consiste en lograr la representación a través de un conjunto de procedimientos que se ejecutan en forma secuencial.
- Representación declarativa.- Permite expresar hechos, reglas y relaciones en forma independiente de su procesamiento. Prolog se ubica en este paradigma, y la construcción de sistemas expertos se realiza en base a este esquema.
- Representación relacional.- El conocimiento se representa a través de tuplas o registros de información en forma estructurada. Por ejemplo, las bases de datos relacionales utilizan esta funcionalidad.
- Representación jerárquica.- Consiste en agrupar en clases los objetos que tienen características comunes, estableciendo relaciones entre ellos a través de la propiedad de herencia que se aplica jerárquicamente estableciendo una clase base y las clases derivadas.

Con la lógica de predicados se puede enunciar o predecir algo sobre un objeto, y se pueden definir atributos y relaciones entre un conjunto de elementos utilizando dos tipos básicos de cuantificadores [51]:

- a. Existencial \exists (existe algún objeto que cumple con el atributo que se evalúa).
- b. Universal \forall (todos los objetos cumplen con el atributo que se evalúa).

Además, para probar o rechazar las afirmaciones hechas sobre un objeto, y a partir de los resultados inferir nuevos conocimientos, se pueden aplicar las siguientes técnicas [51]:

Resolución.- Algoritmo que prueba si los hechos son falsos en base a una contradicción.

Unificación.- Técnica aplicada en Prolog, que consiste en tomar dos sentencias y comprobar si sus argumentos se pueden unificar, encontrando sustituciones para las cuales los valores sean idénticos.

En el paradigma declarativo, en la programación lógica, la evaluación de las reglas se lleva a cabo a través de la validación de un conjunto de condiciones y se utiliza el formato si-entonces (if-then), de tal manera que cada regla es una unidad que forma parte de la base de conocimientos y puede enunciarse de la siguiente forma [51]:

Si antecedente entonces consecuente

Además, de los mecanismos de búsqueda existentes, Prolog aplica el llamado ‘*encadenamiento hacia atrás*’ (backtracking), a través del cual se realiza una evaluación que permite unificar un conjunto de antecedentes, de tal forma que si los valores obtenidos son verdaderos, se determina que el consecuente planteado también lo es [51].

3.2.2 Funcionamiento.

El funcionamiento general de *Mod-Logic* se basa en las ideas mencionadas anteriormente y ofrece las siguientes ventajas:

- a. Es una librería amigable y sencilla que puede utilizarse en aplicaciones que implementen la forma de representación del conocimiento a través de predicados tipo Prolog.
- b. Pueden crearse interfaces visuales que puedan ser utilizadas por usuarios que no tengan conocimientos avanzados en programación lógica.
- c. No requiere del uso de compiladores de programación lógica.
- d. El diseño de almacenamiento establecido evita el uso indiscriminado de tablas en la base de datos relacional.
- e. Está codificado con lenguaje C/C++, lo que valida su eficiencia, rapidez y portabilidad.
- f. Implementa conectividad a bases de datos relacionales utilizando el Sistema Gestor de Base de Datos MySQL, sin necesidad de utilizar librerías complejas, o hacer uso de la conectividad que tienen incluidas algunos compiladores, y a futuro extenderse esta funcionalidad a otros Sistemas Gestores de Bases de Datos cambiando solamente las librerías de conectividad correspondientes.

3.2.3 Flujo de información.

El flujo de información para interactuar con *Mod-Logic* se puede presentar de acuerdo a las siguientes opciones:

- a) El usuario puede consultar o insertar información en la base de datos a través de una interfaz visual, que a su vez tiene conexión directa con el módulo lógico, y éste último con la base de datos respectiva (Ver Figura 29).

- b) Las aplicaciones pueden interactuar directamente con el módulo lógico, utilizando la gramática que para tal efecto se ha diseñado (Ver Figura 30).

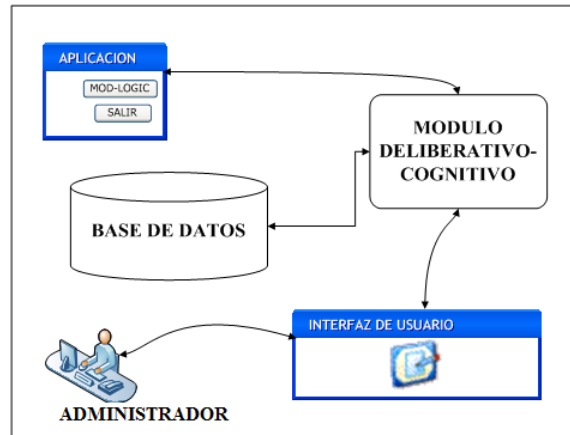


Fig. 30.- Flujo de información.

3.2.4 Aspectos gramaticales de *CLASS-W* para uso de *Mod-Logic* y reglas generadas con el algoritmo 2G.

La parte gramatical ha sido diseñada para trabajar en el contexto general de otras gramáticas, por lo cual puede utilizarse con otros lenguajes fácilmente, pero en especial, se implementaron en *CLASS-W* como una herramienta para programar el conocimiento de los agentes a través del formato que corresponde a un lenguaje de contenido para intercambio de mensajes, y las unidades sintácticas de *P* son:

```

<programacion_conocimiento> ::= 'insertar_regla' \/' <regla>
    | 'evaluar_regla' \/' <regla> | 'buscar_regla' \/' <regla>
    | 'insertar_condicion' \/' <condicion> | 'evaluar_condicion'
    \/' <condicion> | 'buscar_condicion' \/' <condicion>
<regla> ::= <cabeza> '(' {<argumentos>} ')' ':' '-' <lista_reglas>
    | <hechos>
<hechos> ::= <cabeza> '(' {<argumentos>} ')'
<cabeza> ::= <letra> {<letras>}
<lista_reglas> ::= <regla> [{ '\', ' | 'or' ] <regla> }
<argumentos> ::= <letra> {<letra>} | <letra> {<letra>}
    [ '\', ' {<letra>} ]
<evaluar_reglas> ::= 'evaluar' \/' 'si' \/' <regla> \/' '=' \/' 'true'
    \/' 'entonces' \/' <accion>
<condicion> ::= 'normal' \/' <cadena> \/' <operador_condicional>
    \/' <cadena> | 'condicion_arbdec' \/' <condicion_arbdec>
<condicion_arbdec> ::= 'real' \/' <real> | 'valor_discreto'
    \/' <valor_discreto>
<real> ::= 'si' \/' <cadena> \/' <operador_condicional> \/' <nodo>
    '\&' <cadena> \/' <operador_condicional> \/' <id_atributo>
    '\&' <cadena> \/' <operador_condicional> \/' <valor_atrib_min>
    '\&' <cadena> \/' <operador_condicional> \/' <valor_atrib_max>

```

```

<valor_discreto> ::= 'si' '/' <cadena> '/' <operador_condicional>
                '/' <nodo> '&' <cadena> '/' <operador_condicional> '/'
                <id_atributo> '&' <cadena> '/' <operador_condicional> '/'
                <valor_atrib>
<operador_condicional> ::= '<' | '>' | '<=' | '>=' | '=' | '<>' | '><'
                | '~'
<nodo> ::= <cadena>
<id_atributo> ::= <cadena>
<valor_atrib_min> ::= <cadena>
<valor_atrib_max> ::= <cadena>
<valor_atrib> ::= <cadena>

```

Implementándose la programación del conocimiento utilizando dos tipos de reglas:

- a) Las que se formulan con el formato de predicados tipo Prolog, y que se aplica a partir del símbolo no terminal <regla>.
- b) Y las que son generadas por el algoritmo 2G que se describe en el siguiente capítulo, que se aplican con el símbolo no terminal <condición>.

Por otra parte, para incrustar *Mod-Logic* en aplicaciones que no hacen uso de *CLASS-W* en su totalidad, se pueden utilizar las siguientes expresiones gramaticales:

```

<programa> ::= '(' <expresion> ')'
<expresion> ::= <tipo_transacción>
              | <especificaciones_otros_lenguajes>
              | <tipo_transacción>
              especificaciones_otros_lenguajes>
              | <especificaciones_otros_lenguajes>
              <tipo_transacción>
              <especificaciones_otros_lenguajes>
<tipo_transacción> ::= <programación_conocimiento>

```

Correspondiéndole al no terminal <especificaciones_otros_lenguajes>, las reglas sintácticas especificadas para gramáticas distintas a *CLASS-W*.

3.2.5 Estructura general de *Mod-Logic*.

La estructura general de *Mod-Logic* se compone de los siguientes elementos:

- a. Un traductor de programas y predicados, que se encarga de interpretar y traducir los datos a una base de datos relacional.
- b. Un generador de consultas *SQL* para interpretar la validez de la información que se filtra a través de la sintaxis del símbolo no terminal <regla>, y que proviene de la aplicación o de la interfaz visual.
- c. Una interfaz amigable que:
 - Sirve como base para el diseño de predicados en forma transparente al usuario.
 - Permite realizar consultas sobre la base de conocimientos.

3.2.6 Etapa de traducción.

En esta etapa, *Mod-Logic* lee un archivo que se divide en tres secciones:

- a) **Semántica.**- En esta sección se define el significado de los argumentos de cada predicado que se incluyen.
- b) **Reglas.**- Esta sección incluye las reglas.
- c) **Datos relacionados con los hechos.**- Este apartado contiene los hechos como parte de la base de conocimientos.

3.2.7 Flujo de datos.

Como primer paso, se introduce el código que será revisado por un analizador léxico [36]. Cuando en la etapa anterior no se encuentra ningún error, se verifica que todos los símbolos presentan un orden correcto, se realiza el análisis semántico, que consiste en revisar que cada regla y hecho está definido en la sección de la semántica y que exista coincidencia en el número de argumentos especificados. Por último, se pasa al proceso de traducción de la información hacia la base de datos (Ver Figura 31).

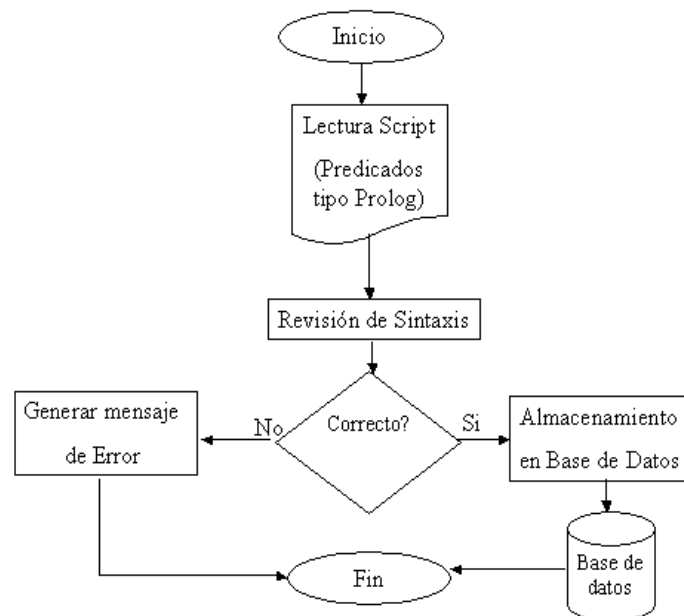


Fig. 31.- Flujo de datos.

Con este procedimiento y la construcción de plantillas *SQL*, se asegura que los resultados que se obtengan al evaluar la información, serán iguales a los que arrojan los compiladores de Prolog.

3.2.8 Estructura de la base de datos de *Mod-Logic*.

Para guardar la información, se diseñó una estructura de base de datos para *Mod-Logic*, la cual se compone de las siguientes variables de relación (Ver Figura 32):

- predicados.- Para identificación de la cabeza de las reglas y hechos registrados.
- lista_reglas.- Para identificar la lista de reglas que forman el cuerpo de la regla.
- hechos.- Guarda la información básica para conformar la base de conocimientos.
- semantica_argumentos.- Proporciona el sentido semántico o significado de los argumentos de los hechos registrados.
- semantica_reglas.- Proporciona información sobre la asociación de la identificación de posición de los argumentos de una regla con otra.
- asociacion_datos.- Permite representar las relaciones existentes entre los datos registrados en la tabla de hechos.
- asociación-reglas.- Se utiliza para identificar la asociación de identificación semántica de los argumentos de unas reglas con otras.

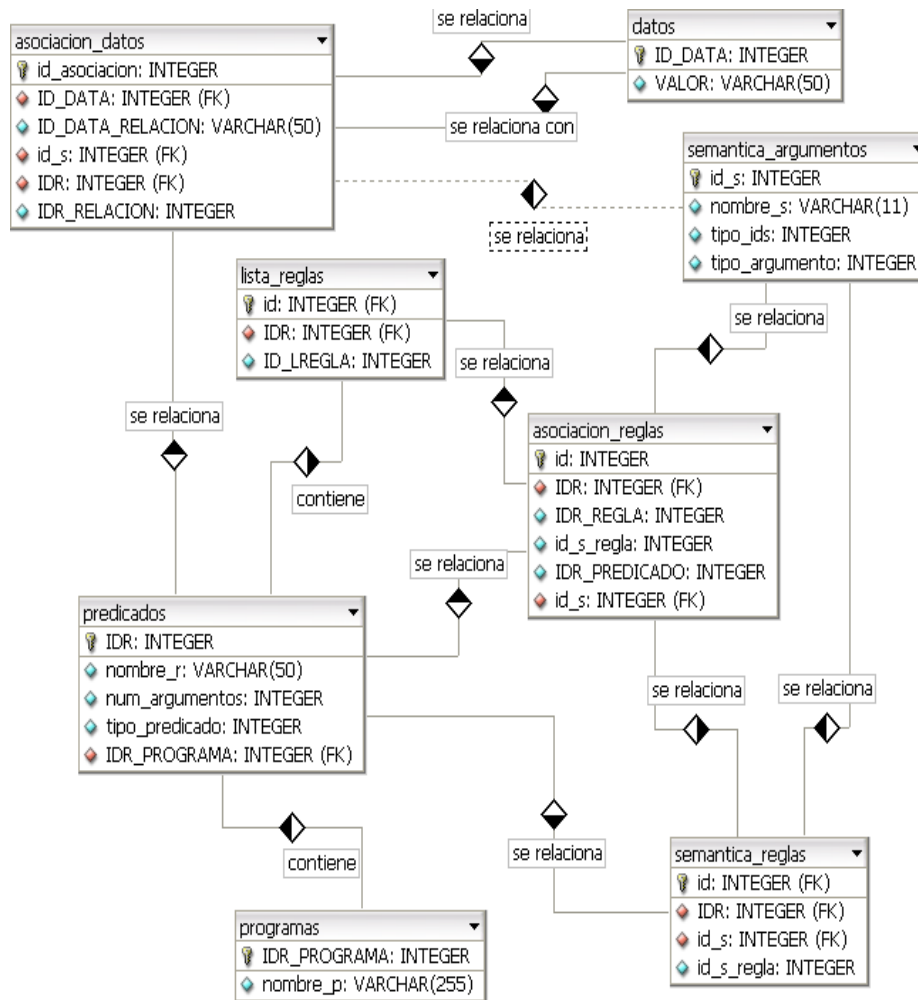


Fig. 32.- Estructura de la base de datos de *Mod-Logic*.

En el siguiente punto se presentará el procedimiento que se utiliza para la interpretación de la información, que consiste en ir concatenando las cadenas necesarias para construir una plantilla *SQL* que resolverá una consulta, y que se estructuran de tal forma que se obtienen resultados iguales a los que arrojan los compiladores para lenguajes lógicos como Prolog.

3.2.9 Procedimiento para la construcción de plantillas SQL.

Cuando se realiza una consulta relacionada con la evaluación de una regla que previamente fue registrada a través de *Mod-Logic*, se aplica un procedimiento para construir una plantilla estructurada con lenguaje *SQL*, siguiéndose el siguiente procedimiento:

1. Primero se localiza el número de identificación, tipo y número de argumentos que le corresponde al predicado que se desea evaluar, así como la identificación semántica de sus argumentos.
2. Después, se obtiene el tipo de predicado de que se trata, tomando como base las siguientes consideraciones:
 - Si el valor que tiene asignado en el campo '*tipo_predicado*' es 1, entonces se asume que el predicado solamente se utiliza para introducir datos al sistema.
 - En cambio, si el valor es 2, entonces se infiere que es una regla y que posee una cabeza y un cuerpo compuesto por uno o más predicados, en cuyo caso, se construye una lista de los mismos, y por cada uno se registra la semántica de sus argumentos. En este caso, la semántica de los argumentos tiene dos interpretaciones:
 - ❖ Una que corresponde a la relación de la sección '*SEMANTICA*', a través de la cual se identifica la posición que tienen los argumentos de la cabeza de la regla en relación a la posición de los argumentos del predicado que forma parte del cuerpo de la misma. En este contexto, el primer argumento se considera como '*argumento base*', y el resto como '*argumento de relación*'.
 - ❖ Otra forma que se usa para unificar parámetros, es la que se realiza a través de la identificación semántica de los argumentos de un predicado que forma parte del cuerpo de la regla con respecto a los argumentos de otros predicados que también forman parte del cuerpo de la misma.
3. Una vez recabada la información necesaria, se localiza el identificador de la semántica de los argumentos de los predicados, ordenándolos en forma ascendente, y se realiza la consulta de la información de los argumentos de la cabeza de la regla con la de los predicados del cuerpo de la misma. En caso de que existan argumentos relacionados entre predicados del cuerpo de la regla, se lleva a cabo un cruzamiento de información para obtener los valores correspondientes.
4. La relación de correspondencia que se utiliza para buscar los datos se aplica de acuerdo a las siguientes especificaciones:
 - ❖ Al primer argumento del predicado le corresponde la relación que existe entre el campo '*id_data*' de la tabla '*datos*' y el campo '*id-data*' de la tabla '*asociacion_datos*'.
 - ❖ Y al resto de los argumentos se les relaciona con los datos a través del campo '*id_data*' de la tabla '*datos*' y del campo '*id_data_relacion*' de la tabla '*asociacion_datos*'.

Para efectos de aclarar el procedimiento antes mencionado, a continuación se presenta la plantilla de un programa lógico generado con *Mod-Logic* que permite representar las relaciones de parentesco básicas de una familia (padre, madre, hijo, progenitores, hermanos), tomando como base los datos que se muestran en la Figura 33:

```

semantica:
    hijo(hijo, padre, madre)
    padre(hijo, padre)
    madre(hijo, madre)
    hermanos(hijo, hijo)
    progenitores(hijo, padre, madre)
reglas:
    hermanos (A,B):- padre (P,A),madre (M,A),
                    padre (P,B),madre (M,B) .
    progenitores (H,P,M):-madre (M,H),padre (P,H).
    madre (M,H):-hijo (H,_,M) .
    padre (P,H):- hijo (H,P,_) .
hechos:
    hijo('Maria', 'Rogelio', 'Veronica').
    hijo('Arturo', 'Rogelio', 'Veronica').
    hijo('Elisa', 'Rogelio', 'Veronica').
    hijo('Sara', 'Fernando', 'Maria').
    hijo('Monica', '', 'Maria').
    hijo('Andrea', '', 'Maria').
    hijo('Antonio', '', 'Monica').
    hijo('Ruben', '', 'Andrea').
    hijo('Manuel', 'Arturo', '').
    hijo('Alejandro', '', 'Elisa').

```

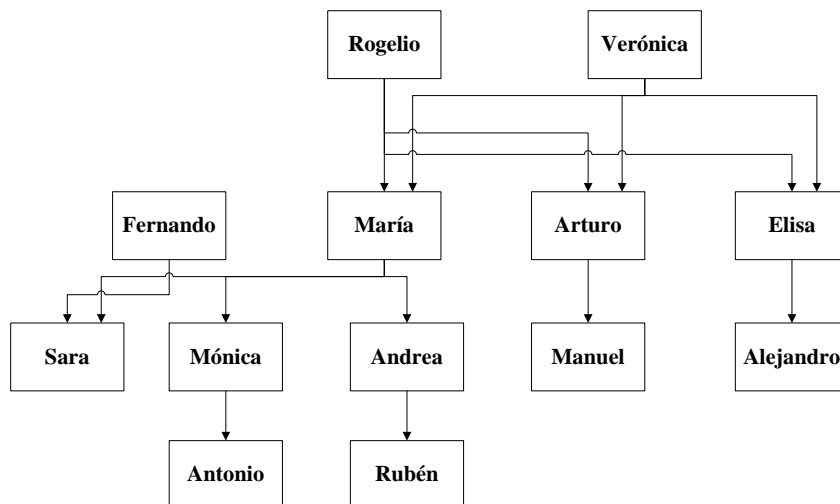


Fig. 33.- Representación gráfica de datos sobre relaciones familiares básicas.

Y para ejemplificar cómo se lleva a cabo el procedimiento de construcción de consultas *SQL*, se toma la regla '*progenitores*' para identificar la semántica de posición, se asigna un número secuencial a los argumentos del predicado que funge como *cabeza de regla*, se establece la correspondencia que tienen con respecto a los argumentos de los predicados que forman parte del *cuerpo* de dicha *regla*, y después, se continúa con la parte que implica la identificación de validación de parámetros que permitirá unificar los datos correspondientes (ver Figura 34).

Por otra parte, en caso de que tenga que establecerse correspondencia de posición y validación entre algunos de los argumentos de los predicados que forman parte del cuerpo de una regla, primero deben evaluarse los parámetros que se reciben a través de los argumentos de la cabeza de regla, y después, con los datos que se obtengan, debe llevarse a cabo un cruzamiento de información que permita unificar resultados. Para efectos de aplicar el procedimiento descrito anteriormente, se ha construido el ejemplo que se muestra en la Figura 35.

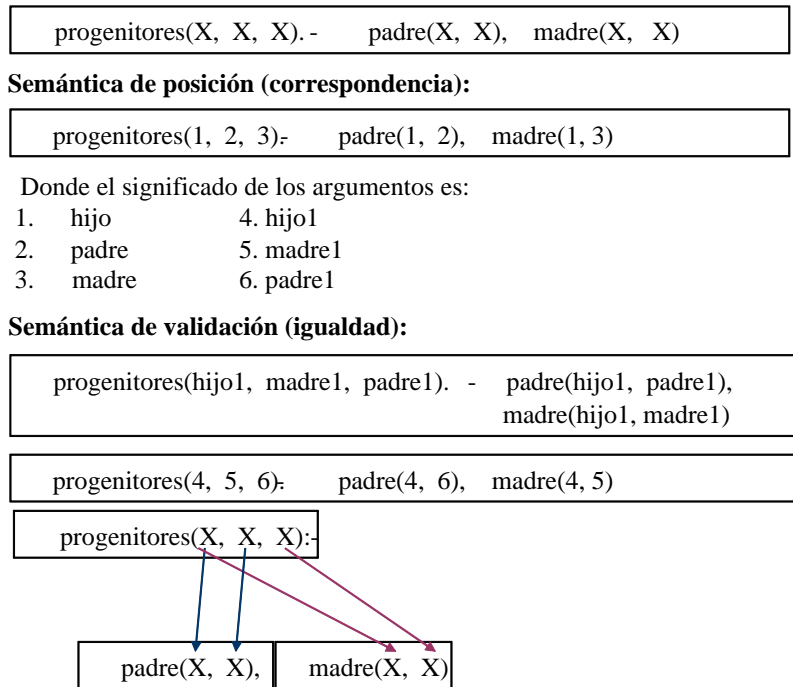


Fig. 34.- Relación de semántica de argumentos.

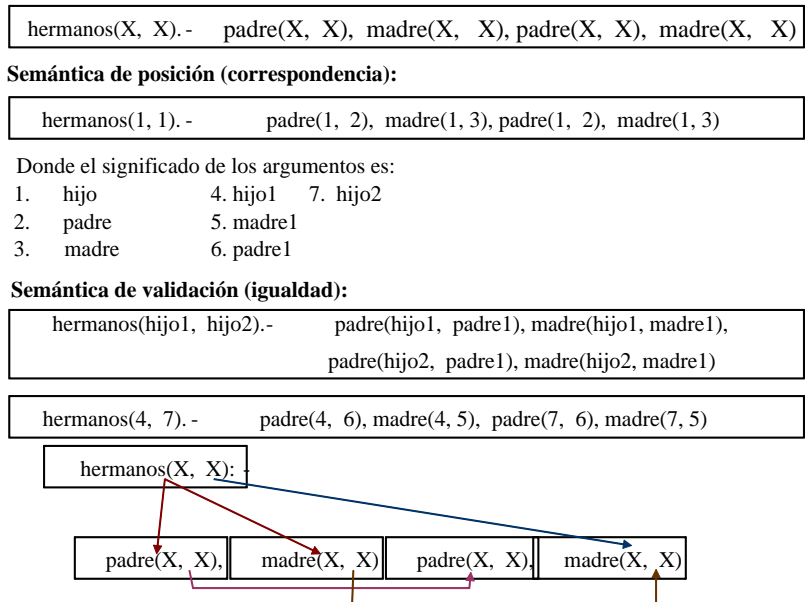


Fig. 35.- Relación de semántica de argumentos.

A continuación se presenta un posible escenario donde se aplica el esquema antes expuesto, tomando como base el tema 'puertos', y los siguientes supuestos:

- a. Se ha detectado que existe un 'gusano', que para efectos de ejemplificar hemos denominado 'scanworm', que se compone de la parte 'cliente' y la parte 'servidora', siendo ésta última la primera que se descarga para infectar los equipos.
- b. La parte 'cliente' es manejada por el autor del 'gusano', y se dedica a escanear la red buscando equipos que tengan abierto el puerto '2127', el cual es abierto en 'listening' por la parte 'servidora' del gusano, y está en espera de que la parte 'cliente' se conecte.
- c. El objetivo de este 'gusano' es establecer conexión entre ambas partes ('cliente' y 'servidor'), con el fin de obtener información confidencial que está guardada en el equipo infectado.
- d. Se detectan anomalías en los equipos infectados provocados por este 'gusano', pero aún no se tiene la herramienta para detectarlo, detenerlo y eliminarlo del sistema, pero se ha propagado una alerta administrativa que permite registrar una regla en el módulo lógico 'Mod-Logic', con el fin de cerrar el nodo de conexión, que es el puerto, y evitar la pérdida o robo de información hasta que se solucione el problema.
- e. Se registra la regla 'scanworm' y se obtiene la siguiente plantilla:

semantica:

```
scanworm(n_puerto,t_operador,n_conexion,t_accion).
num_puerto(n_puerto).
num_conexion(n_conexion).
tipo_accion(t_accion).
```

reglas:

```
scanworm(n_puerto,t_operador,n_conexion,t_accion):-
    num_puerto(n_puerto),
    num_conexion(n_conexion),
    tipo_accion(t_accion).
```

hechos:

```
num_puerto('2127').
num_conexion('0').
tipo_accion('cerrar').
tipo_accion('revisar').
```

Interpretación:

Si el puerto 2127 está abierto, se debe cerrar en forma definitiva y revisarse cada cierto período de tiempo para mantenerlo en ese estado hasta que el problema se haya corregido.

Plantilla construida para obtener los valores del predicado 'scanworm' a través de una consulta SQL:

```
Select distinct concat('Si número de puerto = ',
    h1.valor, ' y el número de conexiones es = ',
    h2.valor, h4.valor, ' entonces se trata de ',
    p1.nombre_r, ' y se debe tomar la accion de
    ', h3.valor, ' el puerto') as
'Regla resultante: ' from predicados as p1
inner join data_h as h1 inner join data_h as
```

```

h2 inner join data_h as h3 inner join data_h
as h4 inner join asociacion as a1 inner join
asociacion as a2 inner join asociacion as a3
inner join asociacion as a4 where p1.idr = 1
and h1.id_data = a1.id_data and h2.id_data =
a2.id_data and h3.id_data = a3.id_data and
h4.id_data = a4.id_data and h1.valor <>
h2.valor and h1.valor <> h3.valor and
h2.valor <> h3.valor and h1.valor <>
h4.valor and h2.valor <> h4.valor and
h3.valor <> h4.valor group by h1.valor
order by a1.id_asociacion asc limit 1

```

Resultados de la consulta SQL:

Si numero de puerto = '2127' y el número de conexiones es = a '0'
entonces se trata de 'scanworm'
y se debe tomar la acción de 'cerrar' y 'revisar' el puerto.

Y la información que se obtiene a través de *Mod-Logic* origina las siguientes consideraciones para el sistema de agentes:

- 1) Objetivo que debe alcanzar el Agente:
 - Cerrar puerto 2127.
- 2) La funcionalidad asignada a través de un conjunto de funciones o tareas que el agente debe realizar para interactuar con su entorno:
 - Revisar si el puerto 2127 está abierto.
 - Si está abierto, cerrarlo.
- 3) Las creencias o conocimientos subjetivos que el agente tiene de sí mismo o de otros agentes:
 - Que tiene facultad para revisar y cerrar puertos en el equipo.
 - Que debe atender instrucciones canalizadas a través del Agente Coordinador.
 - Que debe registrar en la pizarra el resultado de la acción.
 - Que puede consultar al módulo deliberativo-cognitivo y al de seguridad, para medir el nivel de seguridad del evento y reconocer si amerita inicio de monitoreo continuo.
 - Conoce el mecanismo para interactuar con el resto de los agentes en este caso.
- 4) Mecanismo para interactuar con el resto de los agentes:
 - Que tiene facultad para revisar y cerrar puertos.
 - Recibe mensaje alertando sobre posible ataque de gusano a través del puerto 2127.
 - Reconoce el plan a seguir y el objetivo a alcanzar.
 - Instrumenta el plan de acción.
 - Registra los resultados obtenidos en la pizarra correspondiente.
- 5) Procedimiento para manejar un conjunto de metas y planes:
 - El agente tiene facultad para revisar el estado de los puertos.
 - Se registran en bitácora los siguientes datos:
 - Fecha y hora de registro.

- Identificador del agente que registra.
 - Tema: Puertos.
 - Puerto: 2127.
 - Conexiones: 0.
 - Acción: ‘cerrar’ y ‘revisar’.
 - Estado: ‘inicio’.
 - Objetivo: Cerrar puerto 2127.
- Se revisa continuamente el alcance de los objetivos propuestos.

Cabe mencionar, que para mostrar cómo se puede aplicar el funcionamiento de *Mod-Logic* y el uso de la pizarra en programas que incluyan definiciones relacionadas con el área de seguridad, en el capítulo 5 se desarrolla un ejemplo basado en uno de los casos de uso planteados anteriormente.

3.3 Resumen.

Los sistemas o herramientas que se construyen para resolver o controlar problemas del área de seguridad, deben incluir en su diseño componentes que permitan revisar continuamente los eventos que ocurren, registrar en contenedores la información que se considere relevante de acuerdo a los criterios que previamente hayan sido establecidos, implementar técnicas de análisis para clasificar o reconocer patrones que permitan identificar ciertos comportamientos, y aplicar los procedimientos correctivos o preventivos correspondientes, cuando así se requiera.

Atendiendo a los requerimientos antes mencionados, en este apartado se ha presentado una propuesta que incluye como elemento de registro de información una pizarra denominada *BLACKBOARD-ECRE* instanciada por temas, estableciéndose por defecto los que se relacionan con el manejo y control de recursos de sistemas basados en ambiente Windows-Tecnología NT, como lo son los procesos, puertos, servicios, el registro, tareas de inicio, configuración de red, y archivos de configuración del sistema operativo, y que será controlada a través de un esquema de bases de datos relacional e implementada sobre las reglas gramaticales diseñadas para tal efecto en la gramática *CLASS-W*.

Por otra parte, dado que los sistemas basados en agentes, entre otras cosas, deben implementar formas de razonamiento que les permitan interactuar y asumir comportamientos que definan el nivel y el alcance de sus funciones, y siendo la programación lógica una de las formas más eficientes y parecidas a las que el ser humano usa para representar el conocimiento, en nuestra propuesta se ha planteado el uso de agentes que tendrán a su disposición el módulo deliberativo-cognitivo denominado *Mod-Logic*, que ha sido diseñado para traducir predicados tipo Prolog a una base de datos relacional y generar consultas con *SQL* para recuperar información tal como lo hacen los compiladores de este tipo de programación. Además, para la definición de reglas, criterios y acciones a seguir sobre el tema que se trate, el administrador cuenta con una interfaz amigable que permite generar una plantilla con el programa lógico que incluya las especificaciones de definición de semántica, reglas e identificación de datos correspondientes.

Para finalizar este capítulo, cabe aclarar que para tener una idea completa del funcionamiento de la arquitectura *ARSEC-AMS*, es necesario contar con la descripción de los módulos ‘*seguridad*’ y ‘*reactivo*’, y estos temas se desarrollan en el siguiente capítulo.

Capítulo 4.- Módulo de Seguridad y Reactivo.

4.1 Módulo de seguridad.

La seguridad es un concepto que puede aplicarse en distintas áreas, y que tiene como principal objetivo identificar, resolver o evitar problemas, que pueden requerir del establecimiento o aplicación de medidas preventivas y/o correctivas, y que en temas relacionados con el funcionamiento de equipos de cómputo con ambiente Windows-Tecnología NT, entre otras cosas, involucra problemas de seguridad relacionados con:

- a) El estado físico de los equipos.
- b) El acceso físico a los equipos.
- c) La funcionalidad del sistema operativo.
- d) Los privilegios establecidos en base a perfiles de hardware y de usuario.
- e) Los procedimientos para control de sesiones.
- d) Las políticas internas de seguridad para restringir el acceso a los recursos del equipo.
- e) Los procedimientos de auditoría o registro de eventos del sistema.
- f) Protección de la información que se intercambia entre dos o más estaciones remotas a través de la red.

Por otra parte, es importante destacar que para mantener un buen nivel de seguridad en un equipo de cómputo, es necesario regular y controlar el uso y acceso a los recursos del sistema, lo cual se puede lograr estableciendo medidas basadas en conceptos como autenticación, autorización, confidencialidad, disponibilidad e integridad de información [13].

Tomando en cuenta las ideas antes mencionadas, en este trabajo se han definido elementos básicos para el desarrollo de sistemas de seguridad basados en agentes, y en la propuesta de la arquitectura de agentes *ARSEC-AMS*, se incluye un módulo de seguridad, a través del cual se pueden proporcionar medios a los agentes, para que entre otras cosas, intercambien mensajes relacionados con niveles de seguridad previamente definidos por el administrador, apliquen políticas internas de seguridad, realicen operaciones sobre el registro de Windows, revisen alertas, controlen el flujo de transacciones relacionadas con instrucciones, lleven a cabo el registro de eventos, y cuenten con un esquema de protección de la información que se transmite a través de la red.

Además, de igual forma que se ha hecho para el resto de los módulos de la arquitectura mencionada en el párrafo anterior, para el funcionamiento del módulo '*seguridad*', también se han creado inicialmente unidades sintácticas de *CLASS-W* que serán detalladas en el siguiente punto, que permiten implementar las transacciones de control interno de sesiones de trabajo de agentes, de instrucciones, de manejo de alertas, y de registro de eventos de seguridad, y que a futuro pueden extenderse en base a los requerimientos del sistema que se desarrolle con este enfoque.

4.1.1 Descripción gramatical de *CLASS-W* para el módulo de seguridad.

Para que un sistema de comunicación de agentes sea eficiente, debe contar con especificaciones gramaticales que permitan un proceso donde se intercambie información, los mensajes sean interpretados correctamente, y las transacciones se lleven

a cabo de acuerdo a los planes previstos para el cumplimiento de los objetivos del sistema. En este caso, las producciones de *P* de *CLASS-W* que corresponden al módulo de seguridad, son las que a continuación se indican:

A) Transacción sesión.

Para mantener un control interno sobre las sesiones de trabajo de los agentes, las cuales son independientes de las sesiones de usuario de los sistemas Windows-Tecnología NT donde se crean los agentes, se utilizará como acto comunicativo del formato *ACL*, el mensaje '*inform*', de tal forma que las transacciones de '*sesión*' se basarán en las siguientes reglas gramaticales:

```

<sesion> ::= 'inicio_s' \/' <inicio_sesion>
          | 'cerrar_s' \/' <cerrar_sesion>
          | 'estab_s' \/' <establecer_sesion>
          | 'rechazar_s' \/' <rechazar_sesion>
          | <error_sesion>
<inicio_sesion> ::= <agente>
<cerrar_sesion> ::= <agente>
<rechazar_sesion> ::= <agente>
<establecer_sesion> ::= <agente> \/' 'ref' \/' <agente>

```

Cabe aclarar que los fundamentos de esta transacción han sido detallados en el capítulo 2, en el punto 2.3.2, donde se describe la propuesta del esquema interno de comunicación del sistema de agentes.

B) Transacción instrucción.

Para el manejo del conocimiento y toma de decisiones, las instrucciones que definirán las acciones a realizar relacionadas con puertos, servicios, aplicaciones, procesos, políticas, y los temas que sean registrados para el sistema de agentes, serán implementadas a través de las siguientes unidades sintácticas de *CLASS-W*:

```

<instruccion> ::= 'detener' \/' <detener>
                | 'activar' \/' <activar> | 'revisar' \/' <revisar>
                | 'habilitar' \/' <habilitar>
                | 'inhabilitar' \/' <inhabilitar>
                | 'crear' \/' <crear> | 'eliminar' \/' <eliminar>
                | 'modificar' \/' <modificar>
                | 'actualizar' \/' <actualizar>
                | 'buscar' \/' <buscar>
                | 'clasificar' \/' <clasificar>

```

Por ejemplo, una instrucción que introduzca el símbolo no terminal <detener>, puede referirse a detener un proceso, una aplicación o un servicio que pudiera estar activo, y que represente un posible daño para el equipo o para los recursos del sistema operativo.

C) Transacción alerta.

Una alerta puede generarse a partir de un suceso detectado a través de la pizarra, o puede registrarse a través de *Mod-Logic*, y solo pueden emitirse por agentes coordinadores (AC). La sintaxis que le corresponde es:

```
<alerta>::='alerta' '/' <id_alerta> '/' <agente> '/'  
    <nivel_seguridad> '/' <ontologia_temas> '/'  
    <inf_ontologia> '/' <accion>  
<inf_ontologia>::=<lista_procesos> | <lista_puertos>  
    | <lista_servicios> | <lista_t_inicio>  
    | <lista_politicas> | <lista_cambios_red>  
<resultado_alerta>::=<agente> '/' <id_alerta> '/'  
    <fecha> '/' <hora>
```

En cuanto a las alertas, cabe mencionar que las acciones que se deban seguir en cada caso, deberán ser definidas por el administrador.

D) Transacción registro de seguridad.

El registro de seguridad sirve para conservar un historial sobre eventos ocurridos considerados como graves, para efecto de servir como referencia en la adquisición de nuevos conocimientos. La sintaxis que le corresponde es:

```
<registro_seguridad>::=<id_mensaje> '/' <agente>  
    '/' <alcance> '/' <ontologia_temas> '/' <fecha>  
    '/' <hora> '/' <nivel_seguridad>  
    '/' <datos_part_externo> '/' { ',' <accion> }  
<nivel_seguridad>::=0|1|2|3|...|n
```

Y 'nivel_seguridad' puede tomar valores de [1- n]: {bajo, medio, medio alto, alto, ..., n} (ver figura 36).

Para mantener un control sobre los eventos que se presentan en el entorno de los agentes, es necesario que el administrador fije criterios de seguridad, lo que puede realizarse a través de una interfaz que permita guardar la información correspondiente en la tabla de la base de datos '*niveles_seguridad*' (ver Figura 36).

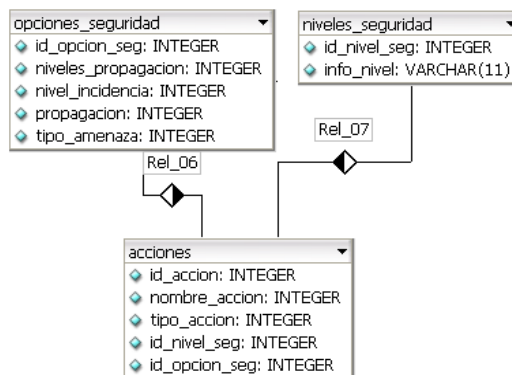


Fig. 36.- Variable de Relación: '*niveles_seguridad*' de SISAG-W.

Los valores que pueden tomar algunos atributos de la tabla ‘*niveles_seguridad*’ que son de tipo enumerativo son los que a continuación se indican:

- opciones_seguridad-niveles_propagación [0-2]: {individual, general, selectivo}.
- opciones_seguridad-nivel_incidencia [0-3]: {0-1, 2-10, 10-*n*}.
- opciones_seguridad-propagación [0-1]: {sí, no}.
- opciones_seguridad-tipo_amenaza [0-1]: {interna, externa}.

4.1.2 Esquema de protección *NetPS-ComSA*.

En el sistema de agentes que se propone en este trabajo, donde se intercambia información a través de una red de computadoras, se espera que el canal que se utiliza para transmisión de datos, sea lo suficientemente confiable, como para poder asumir que el contenido de los mensajes no será fácilmente descifrado por entidades no autorizadas para ello.

Para cumplir con el planteamiento expuesto en el párrafo anterior, se considera necesario utilizar recursos que permitan aplicar medidas de protección, de tal forma que básicamente se cuente con un protocolo de comunicación que tenga las siguientes características:

1. Que el canal de transmisión sea confiable.
2. Que permita utilizar métodos para controlar el acceso al sistema y a sus recursos.
3. Que se conserve el orden de los datos y se evite su duplicidad.
4. Que se valide que los mensajes enviados lleguen correctamente a su destino.
5. Que utilicen técnicas criptográficas eficientes para ocultar o enmascarar la información.

Por otra parte, para construir una propuesta que presente un esquema de protección que permita cumplir con los puntos descritos anteriormente, se han tomado en cuenta las siguientes consideraciones:

La disciplina científica que estudia la seguridad en las comunicaciones es la Criptografía [52]; es un área muy especializada, compleja, y cuenta con grandes aportaciones, por tal razón, para comprender a detalle los fundamentos teóricos que se aplican, se requiere de conocimientos avanzados en matemáticas.

Tomando en cuenta lo anterior, y dado que entre los objetivos planteados en este trabajo no está incluido el desarrollar aportaciones originales sobre el tema de criptografía, pero si se contempla para el sistema de agentes, la necesidad de contar con el uso de librerías o algoritmos existentes, que sean considerados eficientes y confiables, en el desarrollo de este apartado, se describe un esquema de protección de la información que se transmite por red, basado en el uso de los protocolos de comunicación Secure Shell (SSH) [53] en la versión de OpenSSH [54], y Secure Sockets Layer (SSL) [55] [56], en la versión de OpenSSL [57].

Cabe mencionar que los protocolos antes mencionados, son ampliamente conocidos y utilizados, ya que según [55], en el caso de SSL, compañías como Visa, MasterCard, American Express y otras instituciones financieras lo utilizan en las operaciones comerciales que realizan a través de Internet, así como algunas autoridades certificadoras

como Verisign, Thawte, GeoTrust, HispaSSL, CERTICAMARA-Colombia y E-CERTCHILE-CHILE. En cuanto al protocolo SSH, en [53] se menciona que es utilizado para establecer seguridad en túneles de comunicación por red, y en la protección de la información que se transmite a través de servicios como el de correo electrónico, servidores web y conexiones ftp.

Según [53], las principales características de SSH y su desarrollo son las siguientes:

- SSH es una herramienta de administración remota que se implementa en equipos conectados en red, utiliza el puerto 22, y sigue el modelo cliente-servidor.
- Es creado en 1995 por el finlandés Tatu Ylonen con el fin de sustituir comandos como rlogin, telnet, rsh, rcp o ftp, y proporcionar medidas de seguridad en sistemas operativos Unix, Linux, y sistemas con ambiente Windows y MacOS [53]; por otra parte la compañía SSH Communications Security, desarrolla y promueve soluciones comerciales en materia de seguridad en comunicaciones punto a punto basadas en este protocolo [58]; en 1997 es registrado ante la IETF (Internet Engineering Task Force) [59], organización internacional abierta de normalización creada en Estados Unidos en el año 1986 para contribuir a la ingeniería de Internet, actuando en diversas áreas como transporte, encaminamiento y seguridad de paquetes de red; OpenSSH [54], es la versión gratuita de este protocolo bajo licencia BSD [60].
- Proporciona métodos de autenticación, y servicios de cifrado con algoritmos asimétricos basados en llave pública y privada como Diffie-Hellman y RSA, simétricos como DES y AES, y algoritmos que implementan técnicas ‘hash’ como MD5, SHA y SHA1, etc.

Por otro lado, las características básicas de OpenSSH, según [54] son:

- Que es un proyecto desarrollado principalmente por el proyecto OpenBSD [61].
- Que cifra todo el tráfico (incluidas las contraseñas) para eliminar de un modo efectivo las ‘escuchas’, los secuestros de conexiones y otros ataques a nivel de red.
- Que ofrece posibilidades para crear túneles de comunicación seguros.
- Que ofrece métodos de autenticación utilizando algoritmos como RSA o DSA, y para el cifrado de datos 3DES, AES, IDEA, Blowfish, etc.
- Que existen versiones para múltiples plataformas como Linux, Windows, etc.

Y según [58], SSL:

- Es un protocolo de conexión segura que fue desarrollado por Netscape que se dio a conocer en 1996, y más tarde se desarrolló una nueva versión llamada TLS (Security Layer Transport), protocolo que es estandarizado por IETF (Grupo de Trabajo en Ingeniería de Internet).
- Puede utilizarse para crear un túnel sobre una red y crear redes privadas virtuales (VPN), como en el caso de OpenVPN [62].

Las características básicas de OpenSSL, según [57] son:

- Que ofrece una capa cifrada de transporte sobre la capa normal de comunicación, permitiendo la combinación con muchas aplicaciones y servicios de red.
- Que se hace validación cifrada de clientes de correo.
- Que permite generar pares de claves pública y privada.
- Que proporciona el uso de funciones para aplicar cifrado simétrico con algoritmos como AES, DES, 3DES, etc.

- Que en las transacciones basadas en web como pagos con tarjetas de crédito, etc., muchos organismos como apache ofrecen servicios para incluir módulos de OpenSSL como `www/apache13-ssl` o `mail/sylpheed-claws`, que ofrecen soporte de compilación para módulos OpenSSL.

Por otro lado, para el esquema de intercambio seguro de información por red del sistema de agentes *SISAG-W*, se han tomado como base las especificaciones que a continuación se describen y que aparecen en [63]:

Cuando es necesario garantizar el secreto de la información, es decir, mantener la confidencialidad de los datos, se utiliza el método criptográfico conocido como ‘*cifrado*’, de tal forma que el mensaje que se desea proteger o ‘*texto en claro*’, se cifra con un ‘*algoritmo de cifrado*’, el cual lo transforma en otro mensaje llamado ‘*texto cifrado*’. Para que el cifrado sea útil, debe existir otra transformación aplicando un ‘*algoritmo de descifrado*’ que permita recuperar el mensaje original a partir del ‘*texto cifrado*’.

Cifrado simétrico.- Consiste en utilizar para cifrar y descifrar información, una clave secreta que es conocida solamente por el emisor y el receptor de los mensajes, y la seguridad del sistema recae en mantener en secreto esa clave, la cual en adelante se identificará como ‘*csK*’.

Cifrado asimétrico.- A diferencia del cifrado simétrico, las técnicas criptográficas que hacen uso de claves asimétricas, no exigen que se comparta ningún tipo de secreto, y cada participante en la comunicación posee un par de claves, que tienen la particularidad, de que lo que con una de ellas se cifra, es descifrado por la otra, y viceversa. En este esquema, cada una de las partes hace pública una de las claves (‘*clave pública*’), y mantiene en secreto la otra (‘*clave privada*’).

Por otra parte, en un sistema de claves compartidas donde ‘*csK*’ debe ser distribuida por un canal no protegido, para ser aplicada como clave secreta en algoritmos simétricos, puede presentarse el problema de que ‘*csK*’ pueda ser interceptada por terceros no autorizados, situación que según [63] puede evitarse, complementando el uso de un algoritmo simétrico con un asimétrico de clave pública.

Para aplicar una forma de protección a la información que se transmite a través de la red, se han tomado en cuenta las ideas mencionadas en los párrafos anteriores, y para efectos de referencia en este trabajo, el procedimiento ha sido denominado *NetPS-ComSA* (Network Protection Schema-Communication System of Agents), y consiste en aplicar las siguientes especificaciones:

- a. Un emisor **A** genera una clave simétrica ‘*csK*’.
- b. **A** cifra ‘*csK*’ con la clave pública de **B**.
- c. **A** envía el resultado a **B**.
- d. **B** descifra el mensaje recibido con su clave privada, y obtiene la clave ‘*csK*’, que tanto **A** como **B** utilizarán posteriormente para cifrar las comunicaciones subsecuentes con el valor de ‘*csK*’ y un algoritmo simétrico.

Bajo los fundamentos antes mencionados, y con el fin de alcanzar el objetivo de proteger los mensajes que los agentes intercambian a través de la red a través de la implementación del procedimiento *NetPS-ComSA* (ver Figura 37), utilizando funciones

que proporcionan las librerías OpenSSH [54] y OpenSSL [57], se ha diseñado el siguiente esquema:

- Cuando el sistema de agentes es inicializado, los coordinadores (AC) activos deben actualizar sus creencias, y entre ellas está la de identificar si uno de ellos debe constituirse como responsable de habilitar el sistema de comunicación en forma segura, de acuerdo al esquema planteado en la Figura 37. Para ello se debe de aplicar el plan de actividad que se muestra en la Figura 38, y realizar las tareas correspondientes que consisten en revisar ciertos valores de la base de datos que les apoyen en la toma de esta decisión.
- Una vez que se han llevado a cabo las acciones descritas en el inciso anterior, en la primera etapa del procedimiento, el agente coordinador (AC) que haya quedado como responsable de inicializar el sistema de comunicación, de acuerdo a las especificaciones del esquema *NetPS-ComSA*, deberá aplicar el plan de actividad que se describe en la Figura 39, para generar y distribuir la clave secreta que se utilizará en las subsecuentes comunicaciones del sistema de agentes. En nuestro caso se aplica el algoritmo RSA [64] [65] [66] (Nombre que adquiere por las siglas iniciales de los apellidos de sus creadores: Ron Rivest, Adi Shamir y Len Adleman), por ser uno de los métodos de llave pública más populares por su eficiencia y que es implementado en las librerías de OpenSSH.
- Después, se pasa a la segunda etapa de *NetPS-ComSA*, donde todos los agentes del sistema cuentan con la clave secreta 'csK', y están listos para aplicarla con un algoritmo simétrico que permita cifrar los mensajes que serán intercambiados por la red. Para este fin se aplica el plan de actividad que se muestra en la Figura 40, y se utilizan las funciones de OpenSSL que permiten cifrar y descifrar información, seleccionándose de los métodos simétricos que proporciona el protocolo antes mencionado, el algoritmo AES (Advanced Encryption Standard) [67], el cual surge en el año 1997, cuando el NIST (*National Institute of Standards and Technology*) [68], realiza un concurso para elegir un nuevo algoritmo de cifrado para tomarlo como un estandar para proteger la información en forma eficiente, sustituyendo al algoritmo DES (Data Encryption Standard).

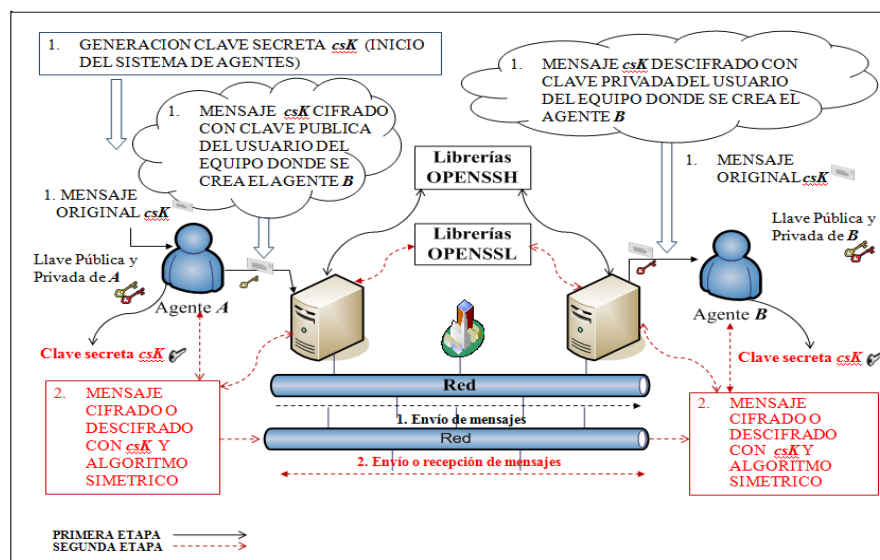


Fig. 37.- *NetPS-ComSA*, esquema de protección de red del sistema de comunicación de agentes).

OBJETIVO: Constitución de un Agente Coordinador como responsable de inicializar el sistema de comunicación segura por red, y actualización de creencias a este respecto.

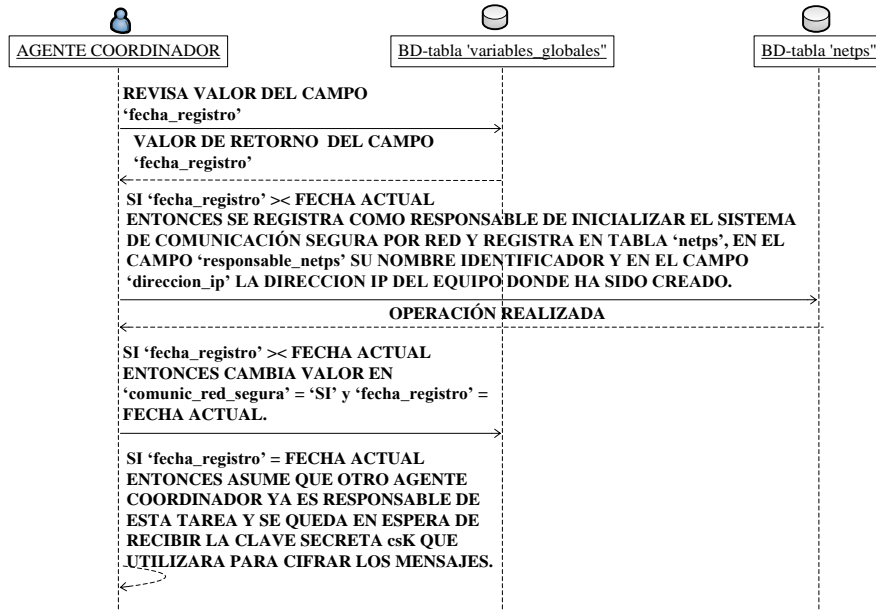


Fig. 38.- Inicialización del sistema de comunicación segura por red.

OBJETIVO: GENERACION Y DISTRIBUCION DE CLAVE SECRETA.

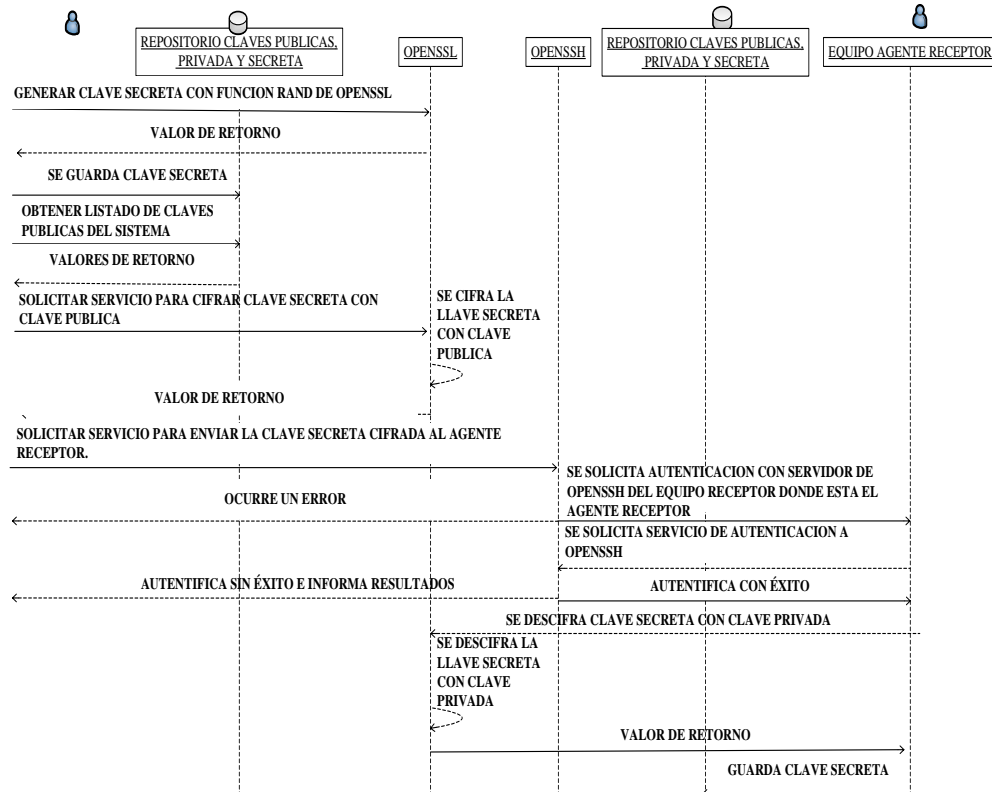
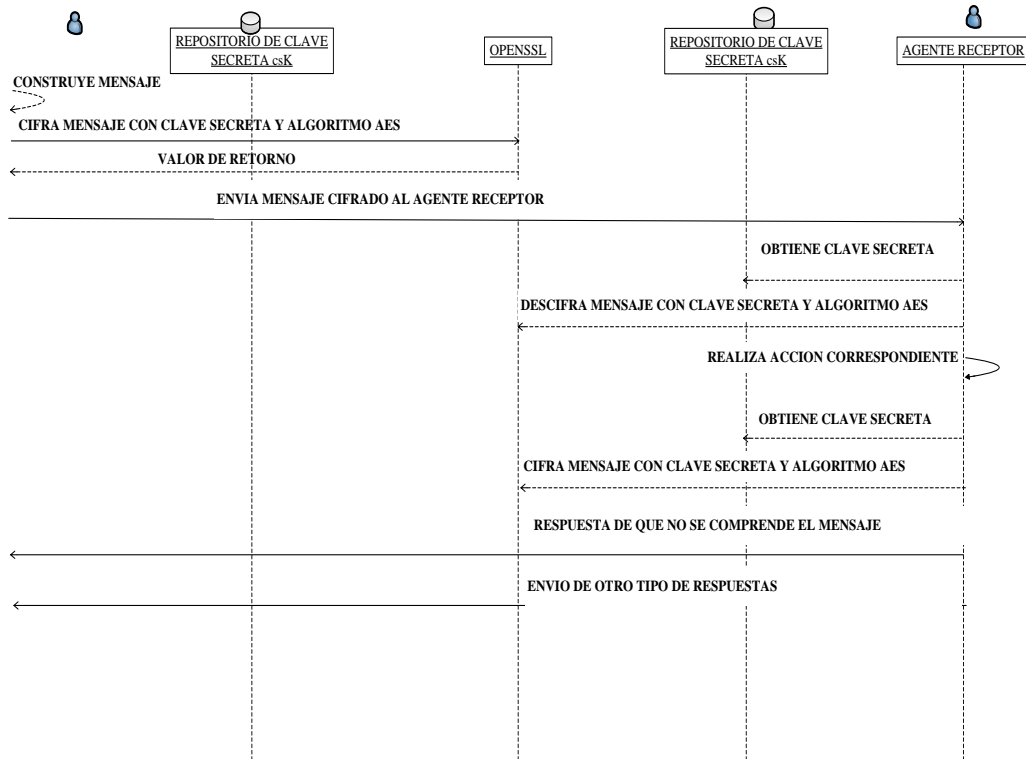


Fig. 39.- Generación y distribución de clave secreta en la primera de *NetPS-ComS*.

OBJETIVO: CIFRADO Y DESCIFRADO DE MENSAJES CON CLAVE SECRETA Y ALGORITMO SIMETRICO AES.



F

ig. 40.- Procedimiento de comunicación de *NetPS-ComS* en la segunda etapa.

Por otro lado, las principales especificaciones que deben tomarse en cuenta para utilizar OpenSSH [54] y OpenSSL [57], y que son requeridas para implementar el procedimiento *NetPS-ComS*, son las siguientes:

- 1) Para realizar conexiones a través de las librerías de OpenSSH, en los archivos 'ssh_config' y 'sshd_config', que se encuentran ubicados en la ruta donde se instala este software, en la carpeta 'etc', deben configurarse algunas opciones básicas con los valores que se indican en la Tabla 2 y en la Tabla 3.

Tabla 2. Opciones de configuración de 'ssh_config'.

Opción	Descripción
PasswordAuthentication yes	El valor 'yes' habilita la opción para que se realicen conexiones con los datos de un usuario y su contraseña, y se deshabilita con el valor 'no'. Esta variable también se configura en 'sshd_config' con los mismos valores.

Tabla 3. Opciones de configuración de ‘sshd_config’.

Opción	Descripción
PasswordAuthentication yes	Cuando esta variable tiene el valor ‘yes’ se asume que se permiten conexiones utilizando una contraseña, de lo contrario el valor debe ser ‘no’.
PermitEmptyPasswords no	El valor de esta variable indica con ‘yes’ que se permiten conexiones con contraseñas con valores vacíos (o nulos), y con el valor de ‘no’, se exige que el usuario que realiza la conexión proporcione una contraseña válida y no nula.

- 2) El protocolo se inicializa con el comando ‘*net start opensshd*’ que activa en memoria ‘*OpenSSH server*’ a través de los procesos ‘*cygrunsrv.exe*’, ‘*sh.exe*’ y ‘*sshd.exe*’ (daemon ssh). Para detener este servicio se usa ‘*net stop opensshd*’. La versión para sistemas con ambiente Windows registra un servicio específico que puede iniciarse automáticamente cuando se carga el sistema operativo. Los servicios que proporciona este protocolo y que se utilizan en este trabajo se muestran en la Tabla 4.

Tabla 4 Servicios que proporciona OpenSSH

Servicio	Descripción
sshd	Servidor de ssh.
ssh	Permite realizar la conexión con el servidor sshd.
scp	Permite copiar archivos entre equipos.

- 3) Para que los comandos de OpenSSH identifiquen la información de los usuarios de Windows que se registran para uso del protocolo, es necesario crear el archivo de grupo llamado ‘*group*’, que contiene los permisos y privilegios asignados, y el archivo ‘*passwd*’ que contiene la información relacionada con los identificadores de usuario (‘*login*’) y sus contraseñas (‘*password*’). Los comandos y los valores que deben aplicarse para contar con la información antes mencionada, que es necesaria para utilizar el protocolo OpenSSH, son los que se muestran en la Tabla 5.

Tabla 5 Opciones de configuración de OpenSSH.

Servicio	Descripción
<code>mkgroup -l >> ..\etc\group</code>	<p><code>'mkgroup'</code> es el comando que se usa para crear un archivo de grupo donde se registra la información de permisos y privilegios asignados a los usuarios de Windows.</p> <p>La opción que le precede puede ser:</p> <ul style="list-style-type: none"><code>'-l'</code> para registro de usuarios locales.<code>'-d'</code> para registro de usuarios de dominio. <p>Los símbolos <code>'>>'</code> se usan para redireccionar la información que se obtiene hacia el archivo <code>'group'</code> localizado en la ruta <code>'..\etc\'</code>, y los dos puntos indican que la carpeta <code>'etc'</code> está ubicada en la raíz de la carpeta donde se instaló OpenSSH.</p>
<code>mkpasswd -l [-u <username>] >> ..\etc\passwd</code>	<p><code>'mkpasswd'</code> es el comando que se usa para crear un archivo que contiene los datos de identificación de usuarios autorizados para utilizar el protocolo OpenSSH y sus contraseñas.</p> <p>La opción que le precede puede ser:</p> <ul style="list-style-type: none"><code>'-l'</code> para registro de usuarios locales.<code>'-d'</code> para registro de usuarios de dominio. <p>La opción <code>'-u'</code> indica que se introducirán datos relacionados con el usuario especificado en <code>'<username>'</code>; si se omite esta opción se asume que se tomarán los datos de todos los usuarios registrados en el equipo.</p> <p>Los símbolos <code>'>>'</code> se usan para redireccionar la información que se obtiene hacia el archivo <code>'passwd'</code> localizado en la ruta <code>'..\etc\'</code>, y los dos puntos indican que la carpeta <code>'etc'</code> está ubicada en la raíz de la carpeta donde se instaló OpenSSH.</p>

- 4) Para la infraestructura de llave pública PKI (Public Key Infrastructure), se utilizan las funciones de OpenSSL que permiten generar pares de claves pública y privada de RSA, siguiendo los siguientes pasos:

1. Para generar la llave privada se usa el comando `'genrsa'` con una longitud de 1024 bits.

'Openssl genrsa -out privada.key 1024'

Nota: En la opción `'-out'` se indica el nombre que se asigna al archivo que contiene la llave privada y enseguida se especifica el tamaño en `'bits'` con que debe crearse.

2. Para generar la llave pública se usa el comando `'rsa'` utilizando como base la llave privada que se obtuvo en el punto anterior.

```
'Openssl rsa -in privada.key -pubout -out publica.key'
```

Nota: En la opción `'-in'` se indica el nombre del archivo que contiene la llave privada que fue generada previamente, y en la opción `'-pubout -out'` se proporciona el nombre del archivo donde se guarda la llave pública que se obtiene.

Los archivos generados son:

'privada.key', que contiene la llave privada.
'publica.key', que contiene la llave pública.

La llave pública se guarda en la carpeta del proyecto, y la llave privada se guarda en una ruta estratégica del directorio que sea difícil de identificar.

- 5) Una vez que se cuenta con los archivos necesarios para realizar la conexión que se utiliza en el procedimiento *NetPS-Com*, para llevar a cabo las tareas de la primera etapa, y habiéndose llevado a cabo las actividades que se muestran en la Figura 38, que corresponde a la descripción del diagrama de actividad donde se constituye un agente coordinador (AC) como responsable de inicializar el sistema de comunicación en forma segura, se procede a generar una llave secreta `'csK'` que será distribuida entre los agentes para cifrar los mensajes que se intercambian a través de la red con la siguiente instrucción de OpenSSL:

```
'openssl rand -out C:\\ProyectoAgentes\\clave.txt 50'
```

Nota: En la opción `'-out'` se indica la ruta y el nombre del archivo donde se guarda la llave generada en forma aleatoria, recibiendo como semilla el valor del número que se especifica en el último parámetro.

- 6) Para cifrar `'csK'` con la llave pública de los agentes receptores, y descifrarla con la llave privada correspondiente, se aplican los comandos de OpenSSL que a continuación se indican:

1. Para cifrar la llave:

```
'Openssl rsautl -encrypt -pubin -inkey publica.key  
-in clave.txt -out cifrado.txt'
```

Nota: La opción `'-encrypt'` indica que se cifra el contenido del archivo que se especifica con `'-in clave.txt'`, utilizando para ello la llave pública indicada

en ‘-pubin -inkey publica.key’, y guardando el resultado en el archivo especificado con ‘-out cifrado.txt’.

2. Para descifrar la clave:

```
‘Openssl rsautl -decrypt -inkey privada.key  
-in cifrado.txt -out clave.txt’
```

Nota: La opción ‘-decrypt’ indica al comando ‘rsautl’ que el archivo especificado en ‘-in cifrado.txt’, se descifra utilizando la clave privada indicada en ‘-inkey privada.key’, y en la opción ‘-out clave.txt’ se proporciona el nombre del archivo donde se guarda el resultado.

7) Para enviar el archivo ‘cifrado.txt’ al equipo remoto, se utiliza el comando ‘pscp’, que es proporcionado libremente con las herramientas de SSH y se ejecuta a través de OpenSSH. Su sintaxis es:

```
pscp [pw password] [ruta y nombre de archivo  
local][usuario@hostServidor: [ruta y nombre de archivo remoto]
```

Ejemplo:

```
‘pscp -pw password C:\...\cifrado.txt  
Administrador@148.225.78.234:C:/ProyectoAgentes/cifrado.txt’
```

8) Para cifrar la información de los agentes con la clave secreta ‘csK’ se aplican las siguientes funciones de OpenSSL, asumiéndose que el agente ya tiene construido el mensaje que enviará o que ha recibido, y que se ha guardado en el archivo ‘msg.txt’:

1. Para cifrar el mensaje:

```
‘Openssl enc -aes-256-cbc -kfile clave.txt -in msg.txt  
-out encrypt.txt’
```

Nota: ‘-enc’, es un comando que se utiliza para cifrar el archivo especificado en ‘-in msg.txt’, con la clave secreta ‘csK’, contenida en el fichero ‘-kfile clave.txt’, guardándose los resultados en el archivo que se indica con la opción ‘-out encrypt.txt’.

2. Para descifrar el mensaje:

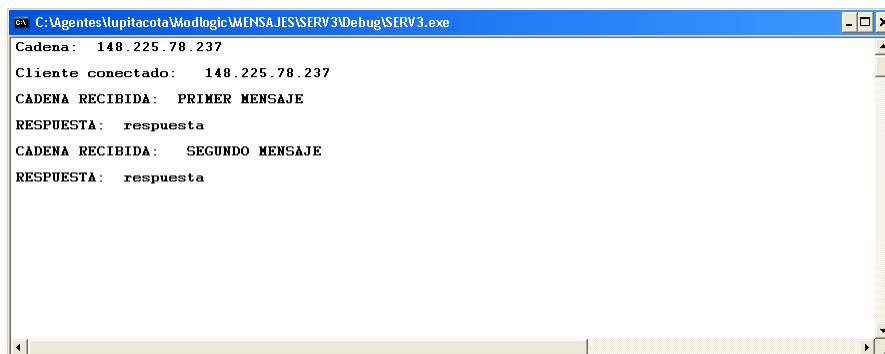
```
'Openssl enc -d -aes-256-cbc -kfile clave.txt -in msg.txt  
-out desencrypt.txt'
```

Nota: '-kfile clave.txt', indica el fichero que contiene la clave secreta 'csK', con la cual se cifrara el contenido del archivo indicado en la opción '-in msg.txt', guardándose los resultados en el archivo proporcionado con la opción '-out desencrypt.txt'.

4.1.3 Prueba del procedimiento NetPS-ComSA.

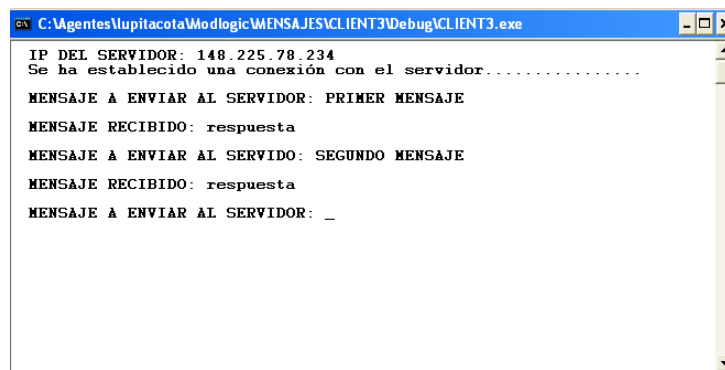
Para la implementación del procedimiento *NetPS-ComSA*, se programaron funciones con Visual C++ para probar los comandos descritos anteriormente, las cuales, dentro de una librería de enlace dinámico, se constituyen en puntos de entrada o de acceso, que pueden ser invocados por aplicaciones externas, como las de agentes. Para la implementación de las instrucciones se utilizó como apoyo la función 'system' y las librerías de OpenSSH y OpenSSL, obteniéndose los resultados que a continuación se detallan:

- 1) La Figura 41 muestra los mensajes de prueba del Servidor, y la Figura 42 los que corresponden al Cliente.



```
C:\Agentes\lupitacota\Modlogic\MENSAJES\SERVER3\Debug\SERVER3.exe  
Cadena: 148.225.78.237  
Cliente conectado: 148.225.78.237  
CADENA RECIBIDA: PRIMER MENSAJE  
RESPUESTA: respuesta  
CADENA RECIBIDA: SEGUNDO MENSAJE  
RESPUESTA: respuesta
```

Fig. 41. Mensajes del Servidor.



```
C:\Agentes\lupitacota\Modlogic\MENSAJES\CLIENT3\Debug\CLIENT3.exe  
IP DEL SERVIDOR: 148.225.78.234  
Se ha establecido una conexión con el servidor.....  
MENSAJE A ENVIAR AL SERVIDOR: PRIMER MENSAJE  
MENSAJE RECIBIDO: respuesta  
MENSAJE A ENVIAR AL SERVIDOR: SEGUNDO MENSAJE  
MENSAJE RECIBIDO: respuesta  
MENSAJE A ENVIAR AL SERVIDOR: _
```

Fig. 42. Mensajes del Cliente.

- 2) En la información capturada por la aplicación ‘Etherdetect’ [69] (ver Figura 43), que permite capturar los paquetes de red, y visualizar los detalles de los paquetes de red de las conexiones que se realizan entre los equipos de un área local, se pueden identificar las direcciones de red (IP) que fueron referenciadas en la Figura 41 y Figura 42.

#	Start Time	Client (IP:Port)	Server (IP:Port)	Protocol	Pac...
0	18:19:46.171	148.225.78.220:137	148.225.78.255:137	UDP.ne...	57
1	18:19:51.203	148.225.78.238:137	148.225.78.255:137	UDP.ne...	57
2	18:19:56.265	148.225.78.237:1731	148.225.78.234:10001	TCP	7
3	18:20:01.296	148.225.78.237:1732	148.225.78.234:22	TCP	54
4	18:20:13.406	148.225.78.237:1733	148.225.78.234:10001	TCP	7
5	18:20:26.515	148.225.78.237:1734	148.225.78.234:22	TCP	57
6	18:20:37.578	148.225.78.237:1735	148.225.78.234:10001	TCP	3
7	18:21:05.937	148.225.78.169:138	148.225.78.255:138	UDP	3
8	18:21:05.937	148.225.78.169:137	148.225.78.255:137	UDP.ne...	12
9	18:21:12.000	148.225.78.220:138	148.225.78.255:138	UDP	1
10	18:21:23.078	148.225.78.123:138	148.225.78.255:138	UDP	1

Fig. 43.- Evidencia de conexiones entre los equipos Cliente y Servidor.

- 3) Y en la Figura 44 se puede visualizar la información cuando es cifrada con los comandos como los que se utilizan con OpenSSH, utilizando la misma herramienta ‘Etherdetect’ [69].

#	Start Time	Client (IP:Port)	Server (IP:Port)	Protocol	Pack...	Time Offset	Pac...	Data...	Data
0	18:19:46.171	148.225.78.220:137	148.225.78.255:137	UDP.ne...	132				18.20.01.296 78 24 SSH-2.0-OpenSSH_3.7.1p1
1	18:19:51.203	148.225.78.238:137	148.225.78.255:137	UDP.ne...	131				18.20.01.296 82 28 SSH-2.0-PuTTY_Release_0.60m
2	18:19:56.265	148.225.78.237:1731	148.225.78.234:10001	TCP	7				18.20.01.296 654 600
3	18:20:01.296	148.225.78.237:1732	148.225.78.234:22	TCP	54				18.20.01.296 670 616
4	18:20:13.406	148.225.78.237:1733	148.225.78.234:10001	TCP	7				18.20.01.296 54 0
5	18:20:26.515	148.225.78.237:1734	148.225.78.234:22	TCP	57				18.20.01.296 70 16
6	18:20:37.578	148.225.78.237:1735	148.225.78.234:10001	TCP	3				18.20.01.296 478 424
7	18:21:05.937	148.225.78.169:138	148.225.78.255:138	UDP	3				18.20.01.296 60 0
8	18:21:05.937	148.225.78.169:137	148.225.78.255:137	UDP.ne...	12				18.20.01.296 470 416
9	18:21:12.000	148.225.78.220:138	148.225.78.255:138	UDP	1				18.20.01.296 790 736
10	18:21:23.078	148.225.78.123:138	148.225.78.255:138	UDP	1				18.20.01.296 70 16
11	18:21:26.093	148.225.78.214:138	148.225.78.255:138	UDP	1				18.20.01.296 54 0
12	18:21:38.218	148.225.78.117:138	148.225.78.255:138	UDP	1				18.20.01.296 106 52
13	18:21:58.421	148.225.78.111:138	148.225.78.255:138	UDP	4				18.20.01.296 106 52
14	18:22:22.671	148.225.78.118:138	148.225.78.255:138	UDP	1				18.20.01.296 138 84
15	18:22:48.921	148.225.78.127:138	148.225.78.255:138	UDP	1				18.20.01.296 1390 1336
16	18:23:17.328	148.225.78.129:138	148.225.78.255:138	UDP	1				18.20.01.296 154 100
17	18:23:56.734	148.225.78.111:137	148.225.78.255:137	UDP.ne...	24				18.20.01.312 138 84
18	18:24:02.781	148.225.78.114:138	148.225.78.255:138	UDP	1				18.20.01.312 350 296
19	18:24:28.015	148.225.78.120:138	148.225.78.255:138	UDP	1				18.20.01.312 90 36
20	18:24:33.062	148.225.78.128:138	148.225.78.255:138	UDP	1				18.20.01.312 122 68
21	18:24:50.203	148.225.78.126:138	148.225.78.255:138	UDP	1				18.20.01.312 106 52
22	18:25:17.515	148.225.78.124:138	148.225.78.255:138	UDP	1				18.20.01.312 122 68
23	18:25:25.593	148.225.78.127:137	148.225.78.255:137	UDP.ne...	1				18.20.01.312 142 88
24	18:25:26.593	148.225.78.113:138	148.225.78.255:138	UDP	1				18.20.01.312 106 52
25	18:25:40.750	148.225.78.122:138	148.225.78.255:138	UDP	1				18.20.01.312 54 0
26	18:25:55.321	148.225.78.125:138	148.225.78.255:138	UDP	1				18.20.01.312 106 52
27	18:26:00.984	148.225.78.125:137	148.225.78.255:137	UDP.ne...	1				18.20.01.312 106 52
28	18:26:04.015	148.225.78.130:138	148.225.78.255:138	UDP	1				18.20.01.312 106 52
29	18:26:05.015	148.225.78.128:137	148.225.78.255:137	UDP.ne...	1				18.20.01.312 106 52
30	18:26:10.078	148.225.78.120:137	148.225.78.255:137	UDP.ne...	1				18.20.01.312 106 52
31	18:26:26.218	148.225.78.124:137	148.225.78.255:137	UDP.ne...	1				18.20.01.312 106 52
32	18:26:40.359	148.225.78.126:137	148.225.78.255:137	UDP.ne...	1				18.20.01.312 106 52
33	18:27:16.703	148.225.78.117:137	148.225.78.255:137	UDP.ne...	1				18.20.01.312 106 52
34	18:27:22.765	148.225.78.113:137	148.225.78.255:137	UDP.ne...	1				18.20.01.312 106 52

Fig. 44.- Información transmitida por red cifrada con OpenSSH.

Para que este esquema sea aplicado al sistema de agentes, se requiere que las funciones que implementan las instrucciones que fueron detalladas anteriormente, a través del uso de librerías de OpenSSH y OpenSSL, sean compiladas como librerías de enlace dinámico, y cargadas por los agentes cuando se requiera.

Por último, cabe aclarar que el módulo de seguridad, inicialmente está planteado como un elemento que permite al administrador fijar criterios y establecer ciertos valores que serán utilizados por los agentes, y para establecer un esquema que permita proteger los mensajes que se transmiten por la red, pero a futuro, su funcionalidad puede crecer para el tratamiento de problemas de seguridad que se originan a través de los ataques que se realizan a través de la red, y que se relacionan con algún tipo de servicio, en cuyo caso podrían tomarse en cuenta las especificaciones que William Stalling plantea en [64], y que se muestran en la Tabla 6.

Tabla 6. Relación entre servicios y mecanismos de seguridad.

Service	Security Mechanism							
	Cipher	Digital Signature	Access Control	Data Integrity	Authentication Exchange	Traffic Padding	Routing Control	Exchange Data
Peer Entity Authentication	yes				yes			
Data Origin Authentication	yes							
Access Control			yes					
Confidentiality	yes						yes	
Traffic for Confidentiality	yes					yes	yes	
Data Integrity	yes	yes		yes				
No repudiation		yes		yes				yes
Availability				yes	yes			

En el siguiente punto se detalla el funcionamiento del módulo reactivo de la arquitectura *ARSEC-AMS*, a través del cual pueden activarse agentes que tienen capacidad de reaccionar ante ciertos eventos que se presentan en su entorno, y llevar a cabo las acciones que hayan sido programadas para tal efecto.

4.2 Módulo reactivo.

La necesidad de resolver problemas complejos donde se manejan datos masivos, ha obligado a automatizar los procesos donde se genera información que es imposible que sea analizada en forma manual por el ser humano. En la actualidad se le da mucha importancia a las técnicas automatizadas para extraer conocimientos de bases de datos de gran tamaño, sobre todo, porque la información que se obtiene, sirve como apoyo en la toma de decisiones, y es un elemento representativo de situaciones que han ocurrido en el pasado y en el presente, que se constituyen en la base para la aplicación de técnicas de reconocimiento de patrones que permitan descubrir nuevos conocimientos.

4.2.1 Descubrimiento de nuevos conocimientos.

El descubrimiento del conocimiento es un campo multidisciplinario, donde las principales áreas son el Aprendizaje Automático, las Bases de Datos y la Estadística. Las etapas que se aplican (ver Figura 45), consisten en comprender el problema que se

estudia, y preparar los datos para la extracción de patrones que posteriormente serán analizados e interpretados, y que se tomarán como base para los resultados que permitan encontrar soluciones a problemas complejos [70].

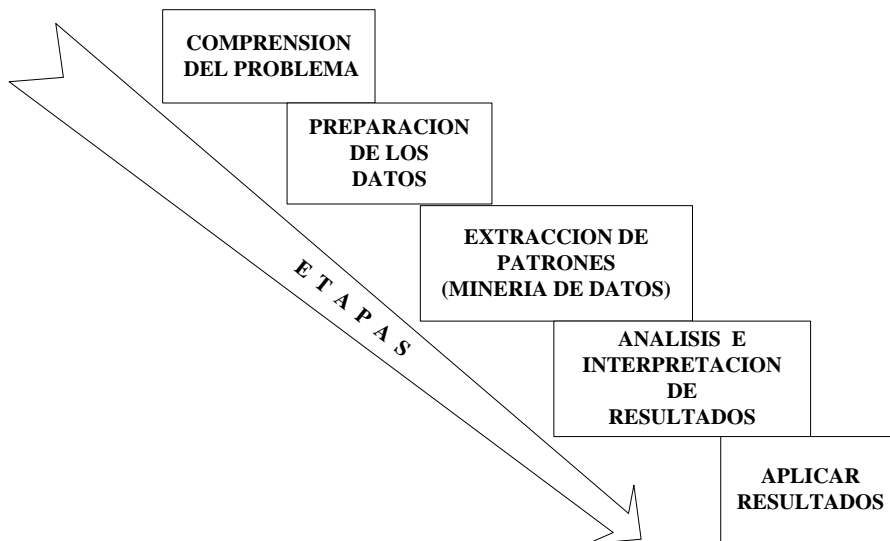


Fig. 45.- Etapas de KDD (Descubrimiento de nuevos conocimientos).

4.2.2 Minería de Datos.

La Minería de Datos es considerada como un proceso iterativo, que a través de la aplicación de técnicas inteligentes, y de la implementación de métodos automatizados o manuales en el análisis exploratorio de grandes volúmenes de datos, se convierte en un medio que permite descubrir nuevos conocimientos, y en una alternativa de trabajo muy útil para entender o describir eventos que se producen en el pasado y en el presente, de tal forma que los datos que se obtienen sirven para predecir información a futuro. Además, de lo anterior, es una opción más para que los expertos puedan modelar soluciones sobre problemas reales con distinto nivel de complejidad [70] [71].

4.2.3 Reconocimiento de patrones.

Los conceptos relacionados con el proceso de reconocimiento de patrones y árboles de decisión, que se han tomado en cuenta para el desarrollo del algoritmo 2G, a través del cual se analiza y se clasifica un conjunto de valores tomando como base la semejanza que existe entre ellos, son los que a continuación se presentan:

Concepto de ‘patrón’:

Aún cuando no existe una definición exacta para describir que es un ‘patrón’, se puede decir que cada autor maneja su propia versión, tratando de apegarse a la idea de contar con ‘algo’ que tiene características propias, y que se compara con un conjunto de valores que deben ser agrupados por semejanzas o criterios de vecindad, que es el principal elemento que sirve como base para desarrollar aplicaciones en el área de reconocimiento de patrones. En este trabajo, un ‘patrón’ se identifica como un conjunto de propiedades asociadas a un tipo de objeto, que pueden ser predefinidas para clasificar un conjunto de valores en un momento dado, o pueden inferirse por similitudes encontradas en un conjunto de valores de un espacio muestral [72].

Concepto de 'objeto':

Según se menciona en [71], la realidad se constituye por objetos y fenómenos, y la creación de la realidad coincide con la creación del tiempo, de tal forma que la transformación de la realidad es inherente a la expansión del espacio y despliegue del tiempo, donde los objetos ocupan el espacio y corresponden a distribuciones espaciales de materia, energía e información, y los fenómenos ocupan el tiempo, y corresponden a cambios temporales de los objetos [71].

Tomando en cuenta lo anterior, se puede definir como un objeto, a la representación detallada, concreta y particular de 'algo' que determina su identidad, estado y comportamiento, en un momento dado (ver Figura 46).

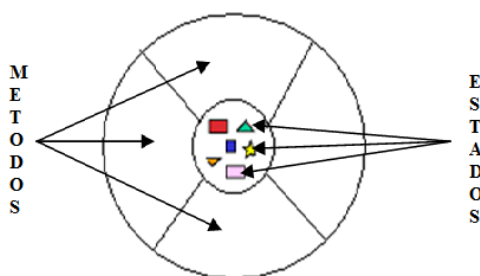


Fig. 46.- Descripción de un objeto.

Elementos relacionados con un objeto:

- ❖ *NOMBRE* (Una de las propiedades del objeto).
- ❖ *ESTADO* que se constituye por un conjunto de valores que lo caracterizan (Propiedades adicionales del objeto).
- ❖ *COMPORTAMIENTO* que se define por un conjunto de funciones propias del objeto (Métodos).

Caracterización y Discriminación:

Para clasificar o agrupar elementos en base a sus similitudes y/o disimilitudes, o aplicando criterios de vecindad [72], es necesario:

- ❖ Resumir las características generales o propiedades de una clase de datos objetivo, y utilizar algún método que permita diseñar la salida de la caracterización de los datos, siendo una de las posibilidades, la de representarlos por medio de gráficas.
- ❖ Hacer una distinción o discriminación, al momento de realizar una comparación entre las características generales de los objetos de la clase, con las características de un conjunto de objetos contrastantes.

Clasificación:

Es el procedimiento por medio del cual se agrupan objetos en clases previamente definidas, pero también es el proceso que se lleva a cabo para encontrar un conjunto de modelos (o funciones) que describan y distingan clases de datos o conceptos en base a

criterios de semejanza o de vecindad, cuando es necesario predecir la clase a la que pertenecen aquellos objetos para los cuales su clasificación es desconocida [72].

El modelo derivado de la clasificación, puede representarse de varias formas, siendo algunas de ellas las que enseguida se mencionan:

- a. Árboles de decisión.
- b. Reglas de clasificación.
- c. Fórmulas matemáticas.
- d. Redes neuronales.

Entrenamiento, aprendizaje y adaptación:

Para iniciar el proceso de clasificación, el sistema computacional debe pasar por un proceso de entrenamiento, que puede ser a través de un método supervisado o no supervisado. El entrenamiento es un procedimiento que se realiza para definir los criterios que se utilizarán para reconocimiento de patrones en el análisis de datos [73].

Al construir clasificadores es importante imponer algún tipo de generalidad o medida que permita resolver las incógnitas que se presenten en los patrones del modelo. Este aprendizaje se refiere a algún tipo de algoritmo que ayude a reducir la cantidad de errores en la información para el entrenamiento [73].

El proceso de adaptación se refiere a clasificar en dos o más clases de categorías al grupo de patrones conocido como conjunto de prueba. Esto se logra al calcular las categorías del conjunto de prueba comparándolo con un conjunto de entrenamiento (previo), donde un clasificador dado, mide la distancia entre varios puntos dados, para saber cuáles son los puntos más cercanos a la meta [73].

Proceso de clasificación:

En el proceso de clasificación se llevan a cabo tareas como:

- ❖ Reconocimiento de un conjunto de clases.
- ❖ Construcción del clasificador:
 - Elección del modelo.
 - Aprendizaje (entrenamiento del clasificador):
 - a) Supervisado.
 - b) No supervisado.
- ❖ Verificación de resultados.

En el reconocimiento supervisado se conocen a priori las clases que se utilizan, y en el no supervisado se conoce el número máximo de clases que se pueden encontrar, pero no existe información exacta sobre ellas, y los algoritmos de este último tipo deben ser capaces de descubrir una estructura que permita el agrupamiento de los datos [72].

La base del procedimiento de clasificación es establecer una medida de la semejanza entre dos objetos.

Reconocimiento de patrones y árboles de decisión:

Entre las alternativas que existen en el ámbito de la inteligencia artificial para reconocimiento de patrones, se tiene la familia de árboles TDIDT (Top Down Induction Trees), donde, para facilitar las cosas, se manejan el criterio ‘*divide y vencerás*’, que consiste en particionar el conjunto de ejemplos en subconjuntos más pequeños [73].

Un árbol de decisión es un modelo de predicción donde se aplica reconocimiento supervisado, y dada una base de datos, se construyen diagramas que sirven para categorizar una serie de condiciones, para lo cual se toma como base la información que se desprende del análisis del problema que se pretende resolver, que se proporciona a través de ejemplos de entrenamiento [73]. Es una de las formas más fáciles para representar el conocimiento, y es utilizado para estructurar el proceso de toma de decisiones bajo incertidumbre, generándose reglas que se desprenden de los recorridos que parten del nodo raíz hasta las hojas del árbol [73].

Estructura del árbol de decisión:

Nodos.- Representan a los atributos del problema.

Aristas.- Se crea una por cada valor del atributo que corresponda.

Hojas.- Identifican la clase, y es donde finalizan los recorridos del árbol.

Debido a que los árboles pueden tener un crecimiento tan grande que sea difícil de analizar, una vez que se construye, se aplican métodos de ‘poda’, que consisten en eliminar nodos que no clasifiquen totalmente un número de ejemplos de una misma clase, o que presenten un mínimo de error en el proceso de clasificación. Estos métodos se definen como:

Poda.- Consiste en aplicar criterios para ‘*podar*’ o cortar ramas del árbol de decisión al irse construyendo [73].

Post-poda.- Consiste en ‘*podar*’ o cortar ramas una vez construido el árbol, basándose en criterios previamente definidos, como es el caso de la estimación mínima de errores al construir las reglas, que actualmente es el método más utilizado [73].

4.2.4 Modelo de clasificación del algoritmo 2G.

Para el tratamiento de la información y la construcción del modelo que se aplica en la implementación del algoritmo 2G basado en la teoría de árboles de decisión, se han tomado como referencia los conceptos que a continuación se describen:

Dominio.- Nombre que permite identificar a un conjunto de valores con propiedades homogéneas [74]. En este trabajo, para efectos de identificación, se toman como base dos tipos de valores, que son:

- a. Los valores continuos, que se identifican con la palabra ‘*real*’.
- b. Los valores discretos, que se identifican con la palabra ‘*cadena*’.

Universo del discurso.- Entorno donde se define un problema representado por el producto cartesiano de un conjunto finito de dominios.

Atributo o variable del problema.- Objeto con características propias que puede tomar valores de atributo existentes en el universo del discurso, asociados a un determinado dominio.

Ejemplo de entrenamiento.- Se representa a través de un conjunto de valores contenidos en una fila o registro de una base de datos, que tiene correspondencia con un grupo de atributos relacionados entre sí, que identifican las propiedades del objeto que se utiliza para efectos de clasificación.

Clase.- Define la generalización de un objeto en particular, es decir, una clase representa un patrón o prototipo de una familia de objetos concretos. Una instancia de la clase es un objeto en particular, que para este caso se denomina ejemplo de entrenamiento [71].

Conjunto de clases.- Se utilizan en el proceso de clasificación y se compone de un grupo de valores registrados.

Dado que un conjunto de patrones se consideran un grupo si forman un conjunto homogéneo y diferenciado de los demás en el espacio de representación, como criterio para agrupar instancias de una misma clase en el modelo de clasificación, en este trabajo, para la selección de atributos, se utiliza la entropía o medida de incertidumbre que existe en un conjunto de ejemplos de entrenamiento [75] [76] [77] [78], con las especificaciones que a continuación se detallan:

S	conjunto de ejemplos de entrenamiento original.
S_r	conjunto reducido de ejemplos de entrenamiento.
p	número total de ejemplos de S o de S_r .
e	instancia de p .
c	clase a la que corresponde e .
k	número total de clases.
z	índice del valor de la clase c asociado con e y $z = \{1, \dots, c\}$.
n	número total de ejemplos de una clase de e_i .
l	número de atributos del conjunto S o de S_r .
A	conjunto de atributos.
Ar	conjunto de atributos seleccionados en $G(Ar)$.
t	es el número total de ejemplos de A o de Ar .
v	conjunto de posibles valores que puede tomar A o Ar en el universo del discurso.
m	número de ejemplos de v .
T	número de posibles valores v en A o en Ar .
$prob_i$	probabilidad de que e_i de Ar pertenezca a la clase c con $i = \{1, \dots, t\}$.

Las variables definidas anteriormente, se toman como base para aplicarse en las funciones: $H(S_r)$ para obtener la entropía conjunta (Fórmulas 1 y 2), $H(v)$ para obtener la entropía por atributo (Fórmula 3), y $G(Ar)$ para calcular la ganancia de información de Ar (Fórmula 4), que se aplican a través de las siguientes expresiones:

$$H(Sr) = \sum_{i=1}^p - prob_i * \log_2(prob_i) \quad 1)$$

$$H(Ar) = \sum_{r=1}^T H(Sr) - prob_r * H(v) \quad 2)$$

$$H(v) = \sum_{j=1}^l - prob_j * \log_2(prob_j) \quad 3)$$

$$G(Ar) = \sum_{x=1}^l H(S) - H(Ar) \quad 4)$$

Bajo este esquema (ver Figura 47), se toma como entrada un conjunto de ejemplos de entrenamiento, que son procesados para generar un árbol de decisión, que a su vez, proporciona las reglas que se utilizan para validar nuevas instancias del problema que se analiza.

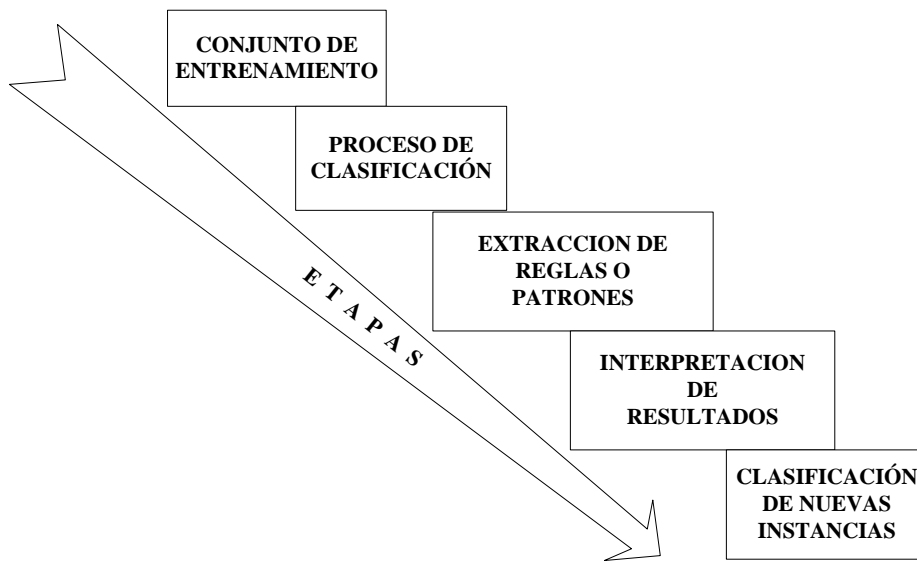


Fig. 47.- Procedimiento de clasificación en el Algoritmo 2G.

Estructura básica de la base de datos del algoritmo 2G:

La base de datos utilizada para el algoritmo 2G (ver Figura 48), ha sido diseñada de tal forma que se puede almacenar, tanto el esquema y valores que se utilizan en el conjunto de entrenamiento que se representa a través de las tablas 'atributos', 'valores de atributos' y 'clases', como la información que se va generando conforme se implementan las distintas etapas del proceso de clasificación, conformada por el árbol de decisión correspondiente que se identifica a través de la tabla 'arbol', los datos relacionados con nuevas instancias del problema que se analiza y que se introducen a través de la tabla 'dataentrenam', las reglas generadas en la tabla 'n_reglas', los resultados de evaluación que se guardan en la tabla 'resultados_evaluacion' y que permiten medir el margen de error que se obtiene, y las tablas auxiliares nombradas como 'auxiliar' y 'auxiliar2'.

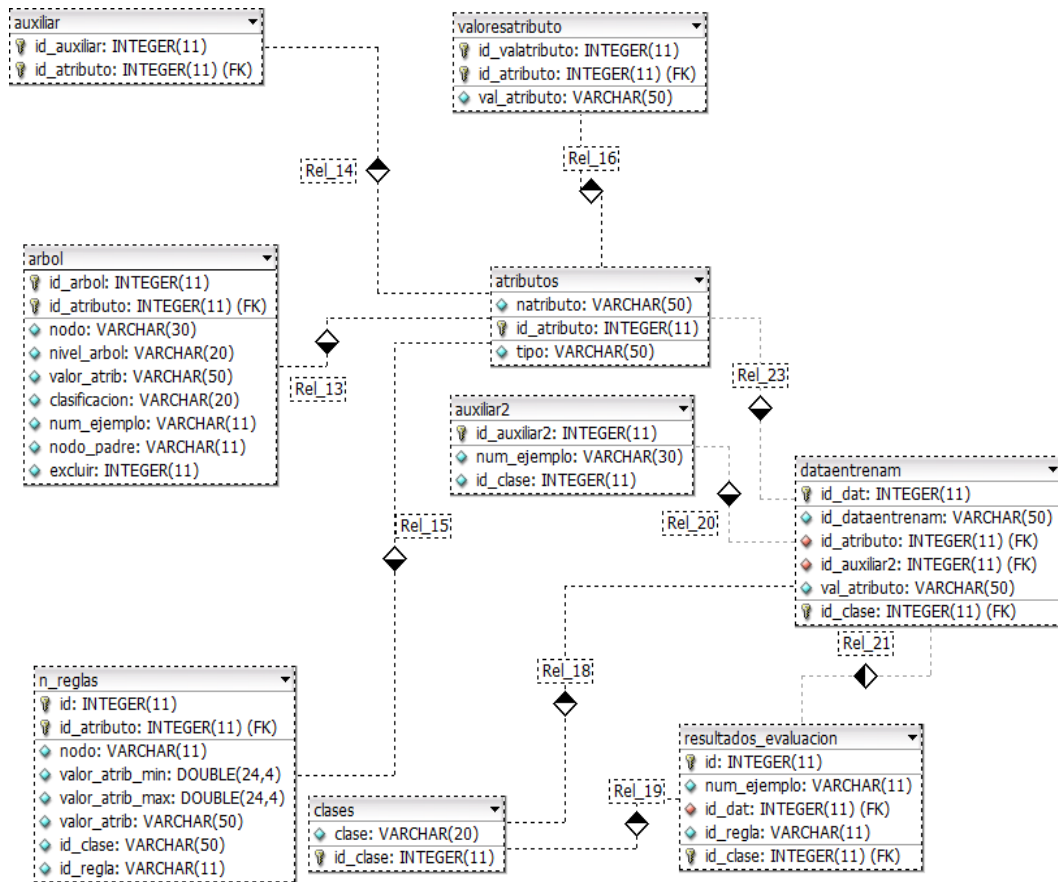


Fig. 48.- Estructura de la base de datos del algoritmo 2G.

A continuación se presentan los detalles del funcionamiento del algoritmo 2G y la correspondencia que existe entre sus elementos y las variables especificadas en el modelo planteado anteriormente:

- a) Se registra en la base de datos que se muestra en la Figura 44, la información relacionada con:
 - a. Clases (c).
 - b. Atributos (A).
 - c. Valores de atributos (v).
 - d. Ejemplos de entrenamiento (e).
- b) Se calculan los valores de las variables:
 - a. p, k, n, l, t, m, T.
 - b. Entropía Conjunta, aplicando la fórmula H(S).
 - c. Por cada atributo, se aplica la fórmula H(A).

- d. Se aplica la fórmula $G(A)$ que permite seleccionar el atributo que presenta mayor ganancia, para ser agregado como un nuevo nodo en el árbol y dejar de concursar en los siguientes ciclos. Este procedimiento se realiza hasta en tanto se concursen todos los atributos, o hasta que se cumpla un criterio de paro que puede ser designado por el usuario.
- e. En lugar de los métodos de ‘*poda*’ que se utilizan actualmente por otros algoritmos de clasificación, se optó por aplicar otro tipo de criterios como el de tomar en cuenta sólo el número de atributos que se indiquen, lo que hace que los árboles generados no sean muy grandes, y tengan igual o mejor eficiencia. Para ello, se manejan ciertos valores de inicio que definen distintos comportamientos.
- f. Una vez construido el árbol, se procede a generar las reglas

Una vez finalizado este procedimiento, la información queda a disposición de la aplicación que ha sido diseñada, para tomar decisiones basadas en la evaluación de las reglas que se generan.

Por otro lado, es importante mencionar que para la construcción de este algoritmo, y aplicarlo en materia de seguridad, fueron de mucha utilidad, entre otros, los artículos que a continuación se mencionan:

Pavel Laskov, Patrick Patrick Düssel, Christin Schäfer and Konrad Rieck [79], comentan que para detección de intrusiones, el algoritmo supervisado c4.5 presenta un excelente desempeño, con un porcentaje del 95% de eficiencia, en comparación con otros como el Perceptron Multicapa (MLP), Support Vector Machine (SVMI, y el algoritmo k-Nearest Neighbor.

Igualmente, Yacine Bouzida, Frederic Cuppens [80], reportan que en resultados experimentales se demostró que mientras las redes neuronales son altamente exitosas en detección de ataques conocidos, los árboles de decisión son más eficientes para detectar nuevos ataques. Mencionándose además en dicho artículo, que se utilizó el algoritmo c4.5 en la fase de aprendizaje, para generar reglas que permitieran evaluar casos como: ‘*IF (protocol type = icmp) and (count > 87) THEN class = smurf*’, siendo el significado de smurf: ‘*es un ataque de denegación de servicio que utiliza mensajes de ping al broadcast con spoofing para inundar (flood) un objetivo o sistema atacado*’.

En los artículos antes mencionados, se trabajó sobre la base de datos KDDCup’99, que es proporcionada como ‘*dataset público*’ para fines de investigación, y que fue utilizada en el Tercer Concurso de la Competencia sobre Descubrimiento del Conocimiento y Minería de Datos [81]. Esta base de datos reporta los nombres de algunos ataques que pueden ser detectados a través de la información recolectada para ese fin [82], entre los cuales, no se encuentran registros sobre problemas relacionados con detección de gusanos.

Por otra parte, el algoritmo 2G ha sido programado con lenguaje C/C++, y compilado con Microsoft Visual Studio C++. Es una versión que ha sido diseñada con aportaciones originales de la autora de este trabajo, para las cuales, no se encontraron referencias en la bibliografía revisada. Las reglas que se generan son utilizadas por agentes de tipo reactivo, pero a futuro podrán relacionarse con la programación de predicados lógicos a través del módulo *Mod-Logic* que ha sido descrito en capítulos anteriores, y a través de las producciones gramaticales que corresponden a la transacción identificada como ‘*programacion_del_conocimiento*’.

4.2.5 Implementación del algoritmo 2G.

Tomando en cuenta las especificaciones del algoritmo 2G que se presentaron en el punto anterior y las que corresponden al módulo reactivo de la arquitectura ARSEC-AMS, en este apartado se presenta el diseño de un agente de red, que tiene como tareas básicas utilizar técnicas de monitoreo y realizar el análisis del tráfico local de paquetes de red de un equipo, con el fin de responder a los eventos que se ajustan al contexto donde se deben de aplicar las reglas que previamente han sido generadas por el árbol de decisión correspondiente [51].

El agente de red tipificado como ‘*sniffer*’, tiene como función filtrar los paquetes de red en el equipo local, en uno de los más bajos niveles de las capas del Modelo para protocolos de red estándar ISO/OSI (International Standard Organization's Open System Interconnect), con el fin de que la información que se reciba por este medio, sea analizada a través del módulo reactivo aplicando técnicas de reconocimiento de patrones sobre ciertos comportamientos que pueden constituir, entre otros, un ataque de ‘gusano’, de tal forma que en el momento en que se detecten situaciones donde existan coincidencias con las especificaciones de las reglas registradas en la bases de datos, se implementen las acciones programadas que permitan reducir el riesgo de una contaminación o intrusión en el equipo.

Para efectos de ejemplificar el funcionamiento del agente que realiza monitoreo sobre los paquetes de red, se aplicó el algoritmo 2G para generar reglas a través del árbol de decisión que se muestra en la Figura 49, tomándose como base las definiciones que se especifican en la Tabla 7 y el conjunto de entrenamiento que se incluye en la Tabla 8, el cual se formó con información, que entre otras cosas, se utiliza para identificar problemas relacionados con el comportamiento de gusanos, a través de intentos de conexiones con el puerto 135 y 445, que comúnmente han sido utilizados por este tipo de ‘*malware*’ para infiltrarse en equipos sin ser detectados [13].

Cabe aclarar que la propuesta de utilizar técnicas de reconocimiento de patrones y árboles de decisión en seguridad computacional, es porque existe una gran variedad de problemas que están relacionados con el uso de puertos y servicios, y es ahí donde seguramente la aplicación de algoritmos como 2G pueden proporcionar soluciones eficientes en la generación de reglas, que permitan clasificar y evaluar eventos de distintos tipos en forma masiva.

Por ejemplo, el puerto 135 es utilizado por los sistemas con ambiente Windows para el servicio Remote Procedure Call (RPC), o llamadas a procedimientos remotos, que permite ejecutar código en una máquina en forma remota, y no es posible cerrar este puerto porque se utiliza por el sistema operativo para el funcionamiento interno del equipo [13].

Por otra parte, el puerto 445, también presenta problemas de seguridad, pues es reportado por las compañías de seguridad como vulnerable, ya que al igual que otros puertos, como el 139, se utiliza por Windows para compartir recursos, y es utilizado por programas tipo gusano para ejecutar herramientas que si bien son confiables, permiten accesos remotos y la ejecución de comandos que abren la posibilidad de que este tipo de ‘*malware*’ pueda infiltrarse en el sistema [13].

Tabla 7 Definiciones para construir el conjunto de entrenamiento para 2G.

	Posibles valores	Tipo
Clases	Normal, ataque_gusano, ataque_ftp, ataque_http, ataque_snmp.	Cadena
Atributos	Protocolo: (TCP, UPD, ICMP).	Cadena
	Servicio: (HTTP, POP3, SNMP, SMTP, ARP).	Cadena
	Puerto (80,21,20,25,445,135,1010,1011,8080,9,8088,110,2354).	Cadena
	Conexiones.	Real

Tabla 8 Conjunto de entrenamiento como ejemplo de prueba para 2G.

Clase	Protocolo	Servicio	Puerto	Número mínimo de conexiones remotas
1	TCP	http	80	1000
3	TCP	http	21	10
1	TCP	http	21	2
3	TCP	POP3	20	10
1	TCP	POP3	20	3
1	TCP	http	25	4
4	TCP	http	25	10
2	UDP	SNMP	445	1
2	UDP	SNMP	135	1
1	UDP	SNMP	1010	5
5	UDP	SNMP	1010	10
5	UDP	SNMP	1011	10
1	UDP	SNMP	1011	6
1	TCP	http	8080	1000
1	UDP	SNMP	9	3
5	UDP	SNMP	9	10
1	TCP	http	8088	1000
1	TCP	SMTP	110	9
1	TCP	SMTP	110	20
1	ICMP	ARP	2354	1000

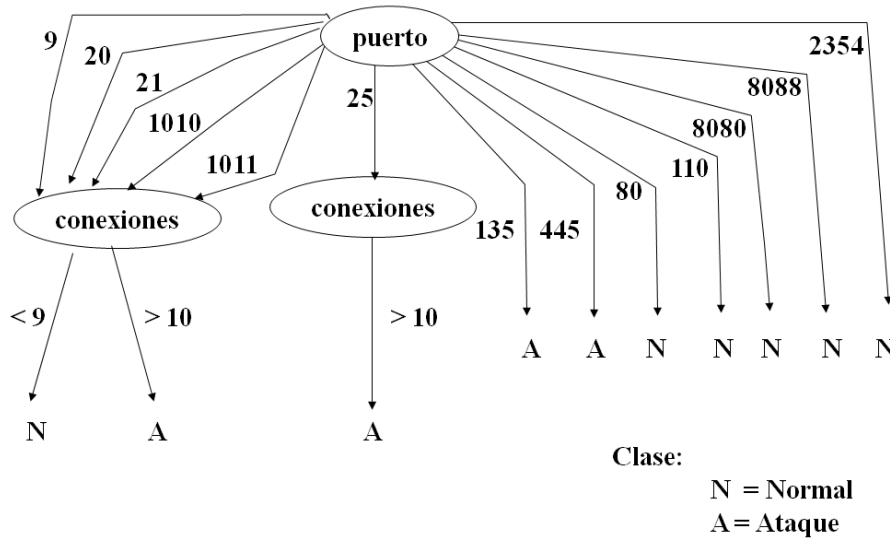


Fig. 49.- Arbol de decisión generado con el algoritmo 2G.

Obteniéndose las siguientes reglas:

- R1: Si puerto = 80 o puerto = 8080 o puerto = 8088 o puerto = 110 o puerto = 2354
→ clase = normal (Ej. 1, 14, 17, 18, 19, 20)
- R2: Si puerto = 445 o puerto = 135
→ clase = ataque_gusano (Ej. 8, 9)
- R3: Si puerto = 21 o puerto = 20 o puerto = 9 o puerto = 1010 o puerto = 1011 y conexiones < 9
→ clase = normal (Ej. 3.5.6.10.13.15)
- R4: Si puerto = 21 o puerto = 20 y conexiones > 10
→ clase = ataque_ftp (Ej. 2, 4, 16)
- R5: Si puerto = 25 y conexiones > 10
→ clase = ataque_http (Ej. 7)
- R6: Si puerto = 1010 o puerto = 1011 o puerto = 9 y conexiones > 10
→ clase = ataque_snmp (Ej. 11, 12)

Con los resultados de clasificación que a continuación se presentan:

Número de ejemplos de entrenamiento:	20
Ejemplos clasificados:	20
Errores:	0

Cabe mencionar que el agente de red tipificado como ‘*sniffer*’, ha sido probado en varias ocasiones para detectar paquetes de red que puedan ser reconocidos a través de los patrones obtenidos con el algoritmo 2G, siendo uno de los ejemplos el que a continuación se detalla, para el cual se realizó una simulación de ataque entre dos equipos:

Al equipo donde se aloja el agente le corresponde el IP 148.225.78.234, y al equipo desde donde se lanzaron paquetes a la dirección de red antes mencionada para simular intentos de ataque a través del puerto 135 y 445, le corresponde el IP 148.225.78.216. A través de esta prueba se obtuvieron los registros que se muestran en la Tabla 9, que fueron

realizados por el agente identificado como ‘sniffer’ cuando se detectaron eventos que se identificaban con las reglas generadas por el algoritmo 2G que fueron mencionadas anteriormente.

Tabla 9 Registro de eventos detectados.

Id	IPOrigen	IPDestino	Puerto	Clase	Fecha y hora
1	148.225.78.216	148.225.78.234	135	2	2007-12-05 19:00:25
2	148.225.78.216	148.225.78.234	135	2	2007-12-05 19:00:53
3	148.225.78.216	148.225.78.234	145	2	2007-12-05 19:00:54
4	148.225.78.216	148.225.78.234	445	2	2007-12-05 19:00:58
5	148.225.78.216	148.225.78.234	445	2	2007-12-05 19:00:59
6	148.225.78.216	148.225.78.234	445	2	2007-12-05 19:01:00
7	148.225.78.216	148.225.78.234	445	2	2007-12-05 19:01:01

En los datos que se muestran en la Tabla 9 se puede observar que los paquetes de red utilizados en la prueba antes indicada se clasifican con la regla ‘R2’, la cual identifica la clase ‘2’ que tiene correspondencia en este ejemplo con la letra ‘A’ del árbol de decisión que se muestra en la Figura 49, y que para efectos de simplificar, se usa para agrupar las reglas que incluyen alguno de los tipos de ataques incluidos en las definiciones de la Tabla 7.

4.3 Resumen.

Dado que del análisis realizado sobre las amenazas más importantes a la seguridad de un sistema operativo, se encontró que los problemas más graves están relacionados con accesos no autorizados, y la existencia de ‘malware’ que permite la intrusión de virus informáticos y programas troyanos o la propagación de gusanos en forma intencional, o por ignorancia de los usuarios, es importante mencionar, que en este contexto, entre las responsabilidades de un administrador, está la de mantener un nivel de seguridad que a través de procedimientos y mecanismos brinde protección a los recursos y desempeño del sistema, de tal forma que se realicen análisis periódicos que permitan monitorear el comportamiento de los equipos en una red, o se establezcan medidas preventivas o correctivas relacionadas con vulnerabilidades internas o externas [13].

Tomando en cuenta lo anterior, la gramática *CLASS-W* incluye expresiones gramaticales relacionadas con especificaciones para el módulo de seguridad que ha sido diseñado dentro de la arquitectura *ARSEC-AMS*, con el fin de alcanzar los siguientes objetivos:

- a. Llevar un control interno de las sesiones de trabajo de los agentes.
- b. Realizar registro de alertas y definir las acciones a implementar.
- c. Mantener un registro de seguridad que proporcione información sobre los eventos que ocurran en el medio ambiente de los agentes, que sean considerados como graves.
- d. Contar con un medio para que el administrador fije criterios y proporcione información sobre niveles de seguridad, defina tanto las acciones que deben de implementarse en ciertos casos, como el tipo de propagación que debe aplicarse en un momento dado en el proceso de comunicación (privada o pública), y además, conformar reglas que incluyan las instrucciones que se deben aplicar, etc.

Por otra parte, tomando en cuenta que el sistema de agentes intercambia mensajes a través de la red por un canal inseguro, ha sido necesario diseñar un esquema de protección, que para efectos de este trabajo ha sido denominado como *NetPS-ComSA*, y que se implementa en dos etapas:

1. En la primer etapa, se genera la clave secreta '*csK*', que se utiliza para cifrar los datos a través de la aplicación de un algoritmo simétrico (AES), donde uno de los agentes coordinadores (AC) se constituye como responsable para dar inicio a un proceso de comunicación segura, encargándose de distribuir '*csK*' a través de una infraestructura de llave pública con un algoritmo asimétrico (RSA), protegiendo a '*csK*' a través de un proceso de cifrado con las llaves públicas de los agentes receptores, la cual será descifrada con la llave privada que corresponde a cada uno de ellos.
2. En la segunda etapa, los agentes que cuentan con la información de la clave secreta '*csK*', la utilizan para cifrar y descifrar los mensajes que intercambian.

Para implementar el procedimiento *NetPS-ComSA*, se hace uso de OpenSSH y OpenSSL, ya que es código libre, y son librerías que se consideran muy eficientes, ya que son utilizadas por todo tipo de organizaciones y empresas que realizan transacciones a través de internet, y que por lo mismo requieren que las aplicaciones que utilizan protejan la información con esquemas de seguridad, que entre otras cosas, permitan hacer uso de algoritmos simétricos o asimétricos, en cuyo caso, el software antes mencionado proporciona varias alternativas al respecto.

Por otra parte, podemos agregar que mientras los algoritmos criptográficos proporcionados por OpenSSH y OpenSSL, como *RSA*, *AES*, etc., no sean vulnerados, se puede tener la confianza de que al usar sus librerías, la información del sistema de agentes puede ser considerada como confidencial y segura.

Cabe aclarar que en este trabajo no se ha planteado la realización de aportaciones originales sobre el tema de criptografía por varias razones: la primera es por considerar que se trata de un área compleja que requiere de conocimientos avanzados y especializados; la segunda se deriva de la reflexión de que es una parte de la ciencia que cuenta con grandes aportaciones realizadas por expertos, que durante muchos años han desarrollado técnicas y librerías comerciales o de libre distribución que pueden aplicarse con buenos resultados; y la tercera es que se ha llegado a la conclusión de que esta tarea llevaría demasiado tiempo y extendería demasiado este trabajo, que consiste en proponer un conjunto de elementos básicos para el diseño y desarrollo de sistemas de seguridad basados en agentes. Sin embargo, no se descarta a futuro el poder incursionar más sobre este tema, y adquirir nuevos conocimientos que coadyuven en el perfeccionamiento de alguno de los planteamientos que se presentan en este documento.

Por último, es importante mencionar que el módulo de seguridad ha sido diseñado inicialmente para cumplir con los objetivos mencionados en este trabajo, pero a futuro podría extenderse, y su crecimiento dependerá de los requerimientos de los sistemas de seguridad que se diseñen utilizando los elementos básicos aquí propuestos.

Para concluir la propuesta que se ha descrito en este documento, en el siguiente capítulo se presenta la programación de un sistema de agentes que ha sido diseñado de acuerdo a la arquitectura *ARSEC-AMS*, para efectos de probar la implementación de algunas producciones gramaticales de *CLASS-W* sobre la plataforma *JADE*, con la funcionalidad de *Mod-Logic* y un agente de tipo reactivo, que revisa los paquetes de red que se reciben en el equipo que se administra y aplica las reglas generadas con el algoritmo *2G*, que permiten identificar patrones que formen parte de la caracterización de alguno de los tipos de ataque registrados a través de este elemento.

Capítulo 5.- Ejemplo de implementación *SISAG-W*.

Una vez que se definieron los elementos básicos para desarrollo de sistemas de seguridad basados en agentes, ha sido necesario diseñar el sistema que hemos denominado como *SISAG-W*, con el fin de probar la implementación de la arquitectura *ARSEC-AMS*, así como el funcionamiento de algunas transacciones planteadas en la gramática *CLASS-W*, el uso de la pizarra *BLACKBOARD-ECRE*, el agente de red tipificado como 'sniffer', y la librería de *Mod-Logic*.

5.1 Diseño de implementación de *SISAG-W*.

Dado que en la implementación de *JADE* se utiliza el lenguaje de programación *JAVA*, en el anexo 1 se incluyen las Figuras 66-72, donde se muestran los diagramas *AUML* correspondientes al esquema aplicado en el ejemplo, relacionados con la organización de paquetes, el contenido de las clases que corresponden a cada paquete, y algunos componentes del sistema que se utilizan.

5.2 Diseño de la interfaz visual de *Mod-Logic* y su funcionamiento.

La interfaz visual de *Mod-Logic*, ha sido programada con Visual Basic (ver Figura 50), introduciéndose en el menú algunas opciones para efectos de ejemplificar el funcionamiento de los componentes *Mod-Logic*, el módulo de pizarra *BLACKBOARD-ECRE*, y la implementación de *CLASS-W* sobre *JADE*. A través de esta interfaz se pueden agregar y evaluar reglas relacionadas con el tema 'puertos', y reglas de tipo general, así como definir criterios y configurar acciones. El resto de las opciones especificadas en el menú, tal como puede observarse en la Figura 51, de principio no están disponibles y aparecen difuminadas.

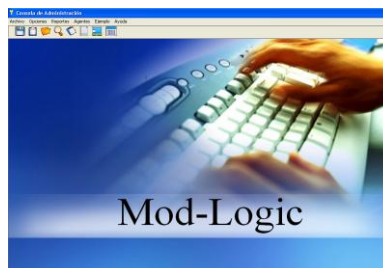


Fig. 50.- Interfaz visual de *Mod-Logic*.

Para explicar el funcionamiento de la interfaz visual de *Mod-Logic*, inicialmente se puede indicar que en 'agregar reglas', en relación al tema 'puertos', utilizando la interfaz que se muestra en la Figura 51, se implementa un procedimiento sencillo para conformar una plantilla con predicados tipo Prolog, que es registrado en la base de datos correspondiente. Dicho procedimiento se lleva a cabo a través de las siguientes acciones:

- El usuario proporciona los valores relativos al número de puerto, número de conexiones mínimas que se toman como base para analizar los eventos en un intervalo de tiempo previamente fijado por el administrador y/o desarrollador del sistema, el nombre de la regla, y selecciona la acción y nivel de seguridad que se implementarán.
- Se genera una plantilla a través del botón de comando correspondiente.
- Por último, si no se desea hacer modificaciones a la plantilla, se guarda la información automáticamente en la base de datos con el botón de comando que está disponible para ello.



Fig. 51.- Menú 'Opciones' -> 'Puertos' o 'Generales'.

Tomando como ejemplo los valores que se muestran en la Figura 52, que son:

Número de puerto: 135
 Número de conexiones: 20
 Nombre de la regla: ataquegusano1
 Acción: detener (Desechar paquetes de red que concuerden con este patrón)
 Nivel de seguridad: Alto

Las variables que definen la semántica para este tema son las siguientes:

Tema: puerto
 Número de puerto: n_puerto
 Número de conexiones: n_conexión
 Acción: t_accion

Y las especificaciones de la plantilla generada a través de la interfaz que se muestra en la Figura 52, son las que a continuación se presentan:

```

semantica:
    ataquegusano1(n_tema,n_puerto,n_conexion,t_accion).
    tipo_tema(n_tema). num_puerto(n_puerto).
    num_conexion(n_conexion).
    tipo_accion(t_accion).
reglas:
    ataquegusano1(n_tema,n_puerto,n_conexion,t_accion):-
        tipo_tema(n_tema),
        num_puerto(n_puerto),
        num_conexion(n_conexion),
        tipo_accion(t_accion).
datos:
    tipo_tema('puertos').
    num_puerto('135').
    num_conexion('20').
    tipo_accion('detener').
  
```

Tema: puertos

Número de puerto: 135
 Número de conexiones: 20
 Regla: ataqueusano1

Acción: detener

Agregar acciones a la base de conocimiento

Alerta:
 Tema: puerto
 Nivel Seguridad: alto

Script

SEMANTICA DE PREDICADOS:
 ataqueusano1(n_tema,n_puerto,n_conexion,t_accion).
 tipo_tema(n_tema).
 num_puerto(n_puerto).
 num_conexion(n_conexion).
 tipo_accion(t_accion).

REGLAS:
 ataqueusano1(n_tema,n_puerto,n_conexion,t_accion):-
 tipo_tema(n_tema),
 num_puerto(n_puerto).

Variable de Entrada	Activar que valor sea > = cualquier valor registrado
puerto	<input checked="" type="checkbox"/>
n_puerto	<input checked="" type="checkbox"/>
n_conexion	<input checked="" type="checkbox"/>
t_accion	<input checked="" type="checkbox"/>

Nuevo
 Reporte reglas existentes
 Ayuda sobre formato
 Cambiar nombre variables de semántica
 Generar Script
 Guardar
 Prueba
 Cerrar

Fig. 52.- Formulario para agregar predicados (tema: 'puertos').

Como una alternativa más, para agregar reglas de tipo más general, es decir, que no están relacionadas con los temas que se manejan en el sistema de agentes, se puede utilizar la forma que se muestra en la Figura 53.

Plantilla genérica

Predicados:
 Predicado:
 Número de argumentos:
 Identificador de argumentos:
 Número de argumentos agregados: 0

Reglas:
 Construcción de Reglas:
 Número de reglas: Número de reglas agregadas: 0

Introducción de Datos:
 Lista de Predicados: Num Argumentos:
 Dato:
 Número de datos agregados: 0

Script generado:

Nuevo Predicado
 Reiniciar Predicado
 Agregar identificador de argumentos
 Agregar
 Nuevos Datos
 Abrir plantilla
 Guardar
 Cerrar

Fig. 53.- Formulario para agregar plantillas genéricas tipo Prolog.

En la figura 53, se introduce una sección que cuenta con tres apartados:

- a. *Predicados*.- Se utiliza para introducir la '*semantica*' de los predicados, proporcionando los siguientes valores:
 - Nombre del predicado.
 - Número de argumentos del predicado.
 - Identificador de argumentos (variable semántica para identificar argumento), y una etiqueta que muestra el número de la identificación del argumento proporcionado.
 - Botones de comando para agregar nuevos predicados, reiniciar los valores proporcionados (limpiar cajas de texto), y agregar argumentos al predicado que se esté estructurando en un momento dado.
- b. *Reglas*.- Con los predicados que se han agregado, se forman las reglas, llevándose un control a través del número de predicados que son parte del cuerpo de la regla.
- c. *Introducción de datos*.- Permite agregar datos para ser utilizados en las reglas establecidas.
- d. El apartado de *resultados o del Script generado*, permite producir una plantilla o abrir alguna existente, y guardar la información correspondiente en la base de datos.

Para ejemplificar la evaluación de reglas sobre el tema '*puertos*' (ver Figura 54), es necesario contar con una cadena que puede construirse a través del botón de comando '*construir cadena*' (ver Figura 55), donde se requiere seleccionar el predicado que se evaluará y proporcionar los valores correspondientes. El valor que de retorno corresponderá a la acción que debe realizarse cuando se cumplen las condiciones especificadas, y que se tomarán en cuenta para generar la producción gramatical especificada en el símbolo no terminal que se describe en el apartado 4.1.1 de este documento, donde se hace una descripción gramatical de *CLASS-W* para utilizar el módulo de seguridad, en lo que corresponde al formato de instrucciones; en caso contrario, el '*Mensaje de la instrucción*' mostrará que '*no se registran valores para este predicado*'.

Por otra parte, para evaluar predicados declarados fuera del ámbito de temas de agentes, se utiliza la forma que se muestra en la Figura 55, donde se selecciona la regla a evaluar y se proporcionan los valores correspondientes, regresando un valor de verdad (true) o falso (false) que se muestra en '*Valor de retorno*', según sea el resultado que se obtiene.

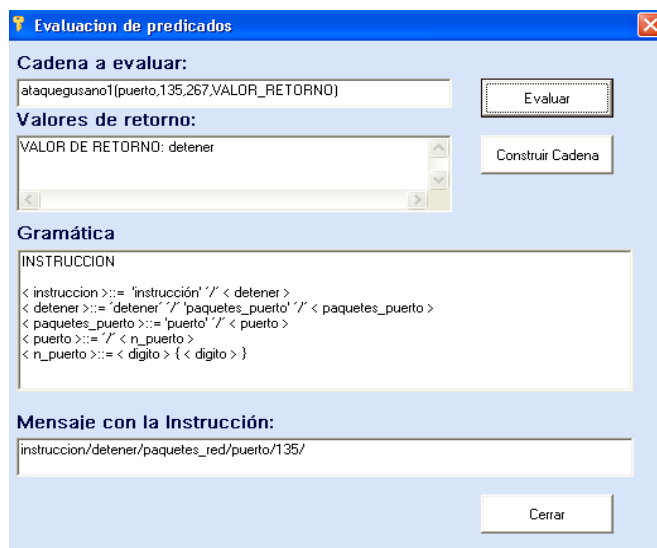


Fig. 54.- Interfaz para evaluar reglas (tema: puerto).

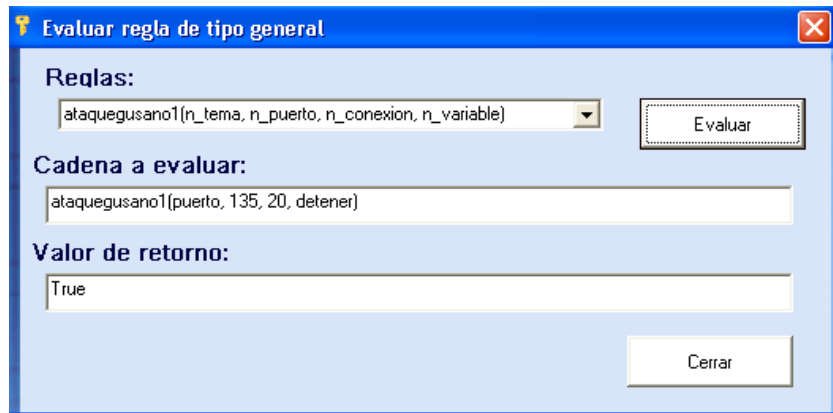


Fig. 55.- Interfaz para evaluar reglas en forma general.

Para '*configurar acciones*' se requiere utilizar la forma que corresponde a la Figura 56, donde se requiere proporcionar los valores: acción, nivel y opciones de seguridad.



Fig. 56.- Interfaz para configurar acciones.

Por otra parte, los criterios que aplicará el Agente Coordinador cuando realice evaluaciones sobre los eventos registrados en la pizarra, deberán configurarse tomando en cuenta: un intervalo de tiempo, la acción que debe aplicarse, y el nivel y opciones de seguridad adecuados, lo cual queda bajo la responsabilidad del administrador del sistema (ver Figura 57).

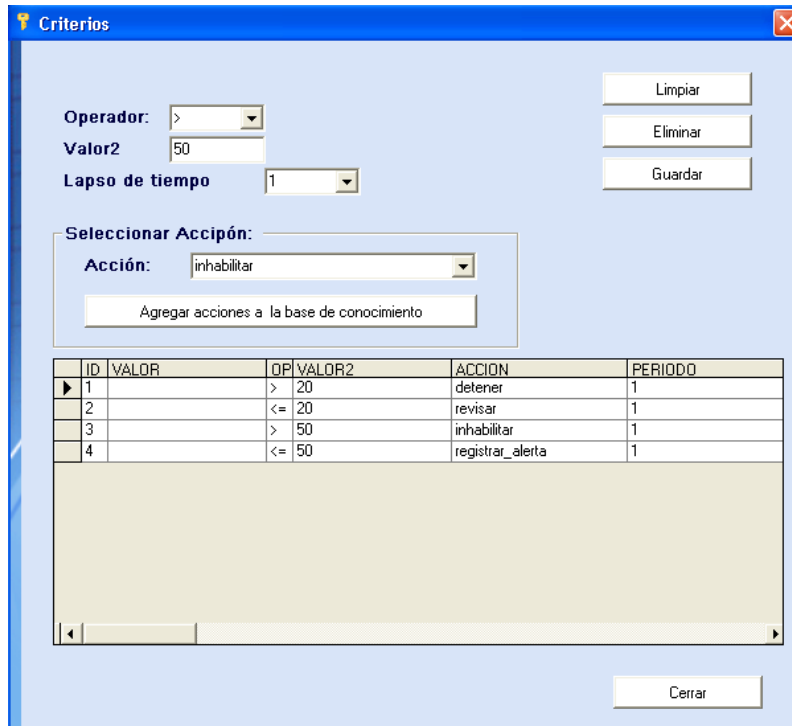


Fig. 57.- Interfaz para agregar criterios.

Además, la interfaz cuenta con la posibilidad de utilizar los formularios de reportes para consultar los predicados que existen en la base de datos, tal como se muestra en la Figura 58.

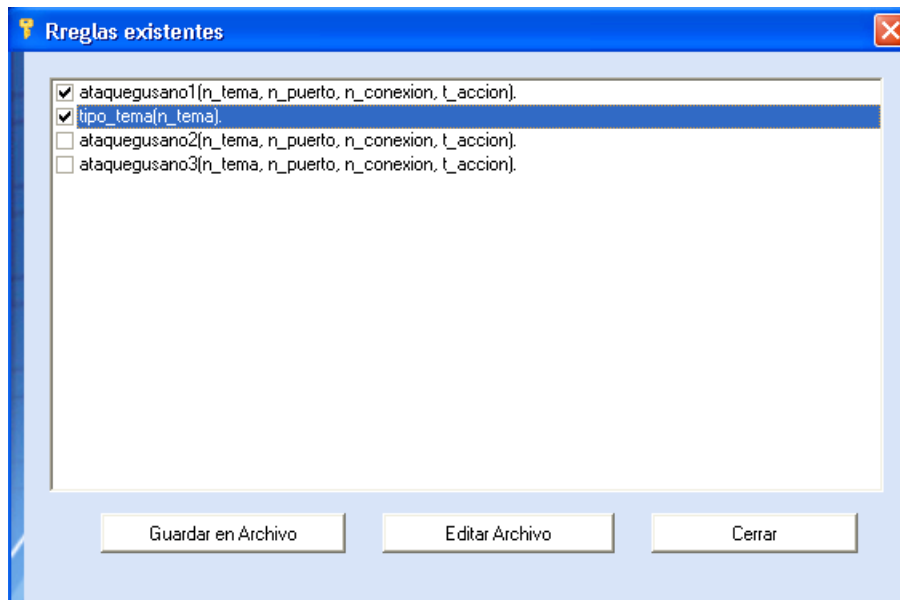


Fig. 58.- Reporte de predicados existentes.

5.3 Funcionamiento de SISAG-W.

Para probar el ejemplo de SISAG-W implementado en la plataforma de agentes JADE, en la interfaz de Mod-logic se ha incluido la opción del menú 'Agentes → Levantar aplicación de agentes', a través de la cual se ejecuta el archivo 'SisAgentes.jar' (ver Figura 59).



Fig. 59.- Interfaz de SISAG-W.

Entre las transacciones seleccionadas para describir el funcionamiento de SISAG-W aplicando CLASS-W, utilizando producciones gramaticales para el módulo de seguridad, módulo deliberativo-cognitivo y de pizarra, se eligieron las siguientes:

- ❖ La de '<sesion>' (ver Figura 60), que se aplica para mantener un control de las sesiones entre agentes -descrita en la sección 2.3.2 de este documento, que muestra el procedimiento para control interno de sesiones de agentes y las producciones gramaticales que le corresponden-, con botones de comando que permiten: que el Agente Coordinador inicie sesión con Agentes Locales (ver Figura 61), de tal forma que pueden verificarse cuales sesiones están establecidas (ver Figura 62), o que se cierren las sesiones que se encuentren abiertas.

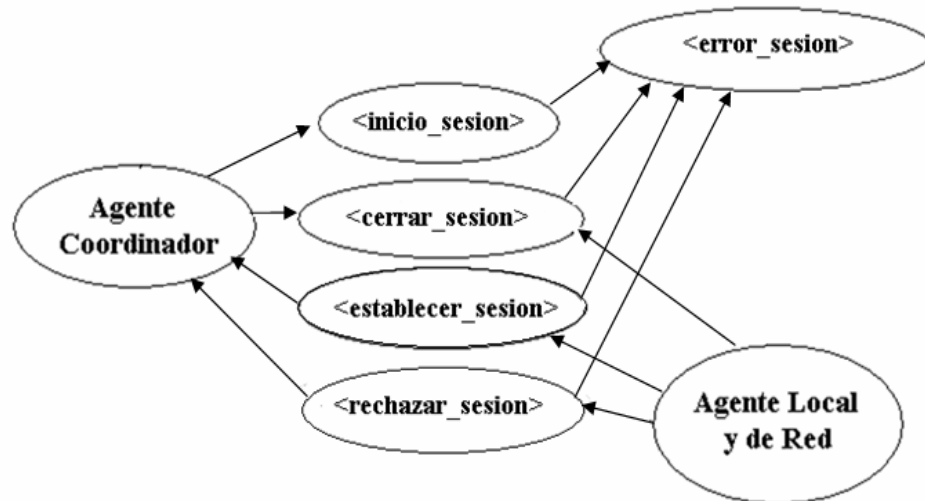


Fig. 60.- Flujo de mensajes de transacción <sesion>.

```

General Output
-----Configuration: SisAgentes - Java5b <Default> - <Default>-----
30/06/2009 12:41:52 PM jade.core.Runtime beginContainer
INFO: -----
This is JADE 3.5 - revision 5988 of 2007/06/21 11:02:30
downloaded in Open Source, under LGPL restrictions,
at http://jade.tilab.com/
-----
INFO: -----
Agent container Main-Container@CALIPSO is ready.
-----
NOMBRE AGENTE RECEPTOR: AgenteLocal2@CALIPSO:8888/JADE//////////
NOMBRE AGENTE RECEPTOR: AgenteLocal1@CALIPSO:8888/JADE//////////
NOMBRE AGENTE RECEPTOR: AgenteLocal3@CALIPSO:8888/JADE//////////
NOMBRE AGENTE RECEPTOR: AgenteLocal4@CALIPSO:8888/JADE//////////
NOMBRE AGENTE RECEPTOR: AgenteLocal5@CALIPSO:8888/JADE//////////
NOMBRE AGENTE RECEPTOR: AgenteLocal6@CALIPSO:8888/JADE//////////
NOMBRE AGENTE RECEPTOR: AgenteLocal7@CALIPSO:8888/JADE//////////
NOMBRE AGENTE RECEPTOR: AgenteLocal8@CALIPSO:8888/JADE//////////
NOMBRE AGENTE RECEPTOR: AgenteLocal9@CALIPSO:8888/JADE//////////
INSTRUCCION RECIBIDA: inicio_s ---
INSTRUCCION RECIBIDA: inicio_s ---
NOMBRE DEL AGENTE LOCAL RECEPTOR: AgenteLocal1
NOMBRE DEL AGENTE LOCAL RECEPTOR: AgenteLocal2
INSTRUCCION RECIBIDA: inicio_s ---
NOMBRE DEL AGENTE LOCAL RECEPTOR: AgenteLocal3
ENVIA MENSAJE SESION ESTABLECIDA AGENTE LOCAL: AgenteLocal3
INSTRUCCION RECIBIDA: inicio_s ---
NOMBRE DEL AGENTE LOCAL RECEPTOR: AgenteLocal5
ENVIA MENSAJE SESION ESTABLECIDA AGENTE LOCAL: AgenteLocal5
ENVIA MENSAJE SESION ESTABLECIDA AGENTE LOCAL: AgenteLocal1
INSTRUCCION RECIBIDA: inicio_s ---
NOMBRE DEL AGENTE LOCAL RECEPTOR: AgenteLocal9
ENVIA MENSAJE SESION ESTABLECIDA AGENTE LOCAL: AgenteLocal9
INSTRUCCION RECIBIDA: inicio_s ---
NOMBRE DEL AGENTE LOCAL RECEPTOR: AgenteLocal3
ENVIA MENSAJE SESION ESTABLECIDA AGENTE LOCAL: AgenteLocal3

```

Fig. 61.- Inicios de sesión con la transacción <sesion>.

```

General Output
ENVIA MENSAJE SESION ESTABLECIDA AGENTE LOCAL: AgenteLocal2
INSTRUCCION RECIBIDA: inicio_s ---
NOMBRE DEL AGENTE LOCAL RECEPTOR: AgenteLocal2
ENVIA MENSAJE SESION ESTABLECIDA AGENTE LOCAL: AgenteLocal2
INSTRUCCION RECIBIDA: inicio_s ---
NOMBRE DEL AGENTE LOCAL RECEPTOR: AgenteLocal1
ENVIA MENSAJE SESION ESTABLECIDA AGENTE LOCAL: AgenteLocal1
INSTRUCCION RECIBIDA: inicio_s ---
NOMBRE DEL AGENTE LOCAL RECEPTOR: AgenteLocal4
ENVIA MENSAJE SESION ESTABLECIDA AGENTE LOCAL: AgenteLocal4
INSTRUCCION RECIBIDA: inicio_s ---
NOMBRE DEL AGENTE LOCAL RECEPTOR: AgenteLocal2
ENVIA MENSAJE SESION ESTABLECIDA AGENTE LOCAL: AgenteLocal2
SESION ESTABLECIDA CON EL AGENTE LOCAL: AgenteLocal1
SESION ESTABLECIDA CON EL AGENTE LOCAL: AgenteLocal2
SESION ESTABLECIDA CON EL AGENTE LOCAL: AgenteLocal3
SESION ESTABLECIDA CON EL AGENTE LOCAL: AgenteLocal4
SESION ESTABLECIDA CON EL AGENTE LOCAL: AgenteLocal5
SESION ESTABLECIDA CON EL AGENTE LOCAL: AgenteLocal6
SESION ESTABLECIDA CON EL AGENTE LOCAL: AgenteLocal7
SESION ESTABLECIDA CON EL AGENTE LOCAL: AgenteLocal8
SESION ESTABLECIDA CON EL AGENTE LOCAL: AgenteLocal9

```

Fig. 62.- Verificación de sesiones establecidas.

- ❖ La de ‘<politica>’ (ver Figura 63), donde se muestra el flujo de mensajes que se intercambian entre el Agente Coordinador y los Agentes Locales, a fin de que se revise si existe una tarea específica que inicia cuando se carga el sistema operativo, y de ser así, se aplique la instrucción de eliminar la cadena del registro de Windows, para lo cual se deberá implementar la producción gramatical ‘<instruccion>’, referenciada en el punto 4.1.1 de este documento.

CASO DE USO: INSTRUCCIÓN PARA ELIMINAR TAREA DE INICIO ‘WORM_X.EXE’
TEMA: TAREAS DE INICIO

OBJETIVO: REVISAR Y ELIMINAR TAREA DE INICIO ‘WORM_X.EXE’ EN EL REGISTRO DE WINDOWS CONSIDERADA COMO DAÑINA.

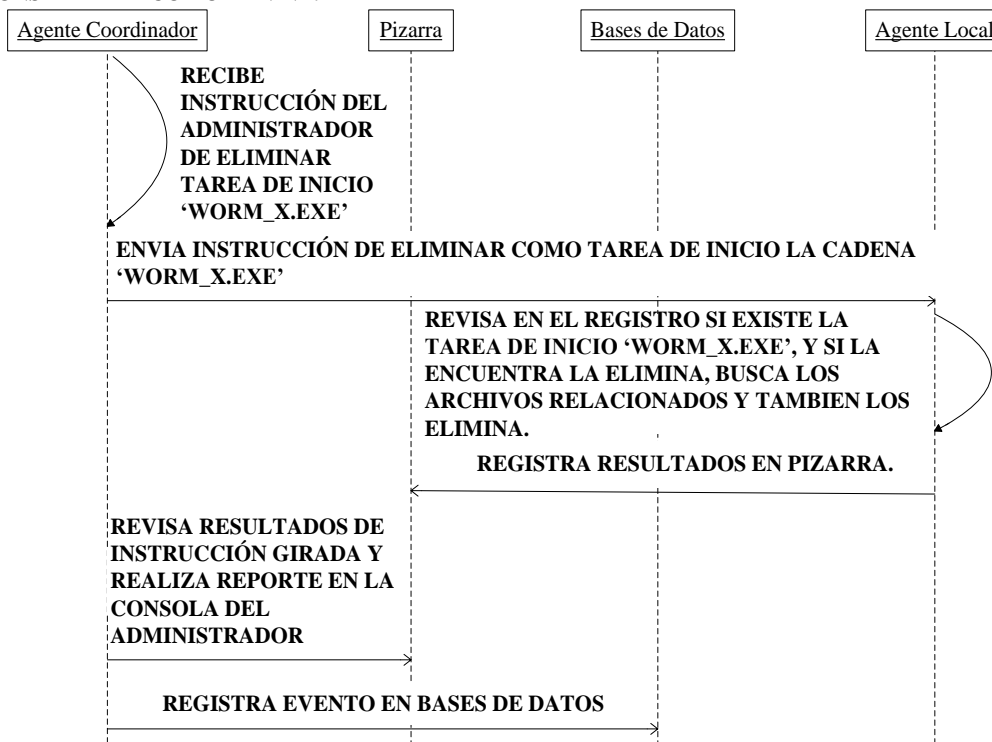


Fig. 63.- Flujo de mensajes para aplicar Política.

Para el caso que se muestra en la Figura 63, se presenta como posible escenario, el hecho de que el administrador ha tenido conocimiento de una tarea de inicio llamada ‘Worm_X.exe’, que es de tipo ‘malware’, y para la cual todavía no existe una herramienta que la inhabilite o elimine del sistema. Como medida preventiva, el administrador registra la instrucción correspondiente para los agentes de tipo coordinador, a fin de que se encarguen de programar el envío de mensajes a los agentes locales, de tal forma que éstos últimos revisen si en el equipo local existen claves del registro que hagan referencia a ‘Worm_X.exe’, y de ser así, se encarguen de eliminar las cadenas de inicio respectivas y los archivos relacionados.

- ❖ La que corresponde al análisis de pizarra, que es una tarea que automáticamente debe realizar el Agente Coordinador, a fin de revisar si existen eventos registrados en relación al

tema 'puertos', y decidir a cuáles Agentes Locales se les girará una instrucción para que implementen acciones sobre los puertos correspondientes. (ver Figura 64 y 65).

```

General Output
NUMERO DE PIZARRAS: 1
TEMA ACTUAL: puerto
Puerto: 135
NUM TOTAL EVENTOS: 175
PORCENTAJE[0]= 20
PORCENTAJE[1]= 45
PORCENTAJE[2]= 34
Puerto: 3005
NUM TOTAL EVENTOS: 60
PORCENTAJE[0]= 100
Puerto: 2005
NUM TOTAL EVENTOS: 100
PORCENTAJE[0]= 100
5
Número total de puertos: 3
INSTRUCCION: detener
MENSAJE ENVIADO - INSTRUCCION A: AgenteLocal2
INSTRUCCION: detener
MENSAJE ENVIADO - INSTRUCCION A: AgenteLocal3
INSTRUCCION: detener
MENSAJE ENVIADO - INSTRUCCION A: AgenteLocal5
INSTRUCCION: detener
INSTRUCCION RECIBIDA: detener --- puerto - 135 -
NOMBRE DEL AGENTE LOCAL RECEPTOR: AgenteLocal2
ENTRE A CERRAR SESION
RECIBI MENSAJE PARA DETENER - PROCESO
MENSAJE ENVIADO - INSTRUCCION A: AgenteLocal6
INSTRUCCION: detener

MENSAJE ENVIADO - INSTRUCCION A: AgenteLocal4
CADENA RECIBIDA: 135
CADENA RECIBIDA3: 135

```

Fig. 64.- Análisis de pizarra y envío de Mensajes.

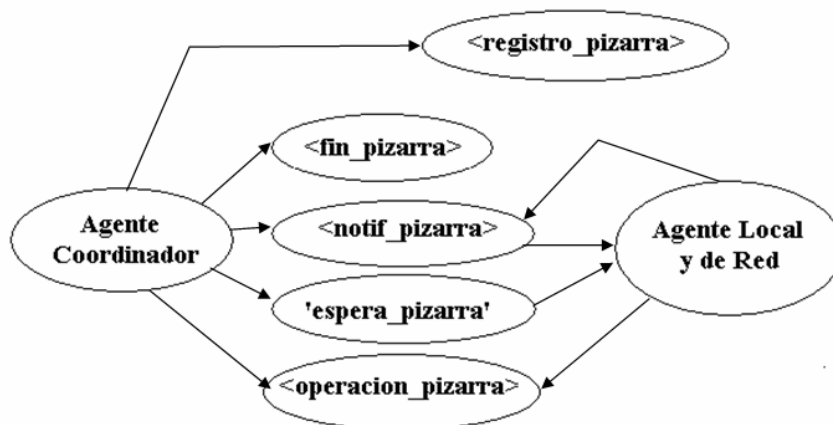


Fig. 65.- Flujo de mensajes de transacción <pizarra>.

5.4 Resumen.

Cabe mencionar que al ir realizando las pruebas para aplicar algunas producciones gramaticales de *CLASS-W*, tomando como base el diseño arquitectónico mencionado en el capítulo 2, así como los elementos básicos planteados en la propuesta general de este documento, se observó que es relativamente fácil desarrollar sistemas de seguridad con este enfoque, y se encontró que su implementación respalda las expectativas de poder obtener excelentes resultados con rapidez y eficiencia. Además, se observó que la plataforma *JADE* es una excelente alternativa para implementar los sistemas que se desarrollen con los elementos básicos descritos en este documento, ya que cumple, entre otras cosas, con los estándares definidos por *FIPA* para *ACL*.

Por otro lado, hay que mencionar que existen grandes ventajas al poder contar con un enfoque distinto a los actuales para abordar la problemática de seguridad computacional, pues al utilizar una gramática como *CLASS-W* se abren posibilidades para que los expertos, administradores o desarrolladores de sistemas en materia de seguridad, administren y/o desarrollen sistemas basados en agentes que permitan aprovechar su experiencia en el área, apoyándose básicamente en la información que se obtiene del sistema operativo relacionada con recursos de red, opciones de configuración, aplicaciones que se cargan al inicio del sistema, la existencia o ausencia de procesos, el uso de puertos, etc.

Aunado a lo anterior, el hecho de poder diseñar un sistema con componentes basados en librerías dinámicas con lenguajes de programación como C o C++, que pueden ser ligadas con módulos de JAVA fácilmente, flexibiliza el uso de lenguajes, y abre todo un mundo de posibilidades para programar módulos con lenguajes de medio o bajo nivel que permiten acceder a recursos del sistema casi en forma directa en sistemas con ambiente Windows-Tecnología NT, lo cual no puede hacerse con JAVA, por ser un lenguaje interpretado y de alto nivel.

Capítulo 6.- Conclusiones

La seguridad es un concepto que conforme pasa el tiempo cobra mayor fuerza en cualquier ámbito en que se maneje, motivo por el cual, hoy en día, los expertos de seguridad en redes de computadoras tratan de encontrar alternativas basadas en políticas de seguridad, que por un lado permitan detectar problemas de este tipo, y por otro, éstos se resuelvan en forma rápida y satisfactoria.

En relación a esta problemática, y debido a la gravedad de sus consecuencias, el tema relacionado con las 'amenazas del día cero' es el más importante, ya que las alternativas de solución, aún las más destacadas, no han logrado resolverla y/o controlarla eficientemente.

Tomando como base las afirmaciones planteadas en los párrafos anteriores, se puede concluir que es necesario utilizar enfoques distintos a los actuales y seguir trabajando en el estudio de elementos que faciliten el desarrollo de sistemas basados en agentes orientados a seguridad, ya que además de ser una tecnología relativamente novedosa, los resultados que se obtengan con ella, permitirán marcar el rumbo a seguir en un futuro próximo, en este tipo de sistemas de protección.

Por otra parte, y dado que la tecnología de agentes inteligentes presenta buenos resultados para encontrar soluciones precisas a problemas complejos que con otro tipo de técnicas o metodologías no pueden resolverse, en este trabajo se concluye que para desarrollar sistemas de seguridad en plataformas de agentes, se requiere conjuntar al menos con los siguiente elementos básicos:

1. Un lenguaje de comunicación diseñado especialmente para el área de seguridad computacional.
2. Un diseño de jerarquía y roles a desempeñar.
3. Mecanismos de razonamiento de agentes que permitan modelar su comportamiento.
4. Una plataforma de agentes que permita implementar el diseño del sistema.
5. Un elemento que sirva para registrar o analizar los eventos que se presentan en el medio ambiente.
6. Una arquitectura que integre componentes que permitan aplicar políticas y niveles de seguridad adecuados.
7. Un canal de comunicación confiable para la transmisión de información a través de la red.

En relación al punto cuatro mencionado anteriormente, se concluye que la plataforma de agentes *JADE* es una buena opción para implementar sistemas de agentes ya que soporta los estándares definidos por *FIPA*, y en cuanto al punto número siete, tomando en cuenta que en el desarrollo de sistemas orientados a seguridad es muy importante proteger la información que se transmite por red, se concluye que es indispensable utilizar algún protocolo donde se apliquen técnicas criptográficas como *OpenSSH* y *OpenSSL*, de tal forma que se asegure que los datos llegan íntegros a su destino, y que el contenido de los mismos está a buen resguardo de usuarios no autorizados.

Por último, cabe mencionar que la propuesta planteada se basa de principio para sistemas con ambiente Windows-Tecnología NT, pero a futuro puede extenderse fácilmente a otro tipo de sistemas operativos, que entre otras cosas, soporten la implementación de la plataforma de agentes *JADE*.

Bibliografía

- [1] Randall K. Nichols, Pano C. Lekkas, *Seguridad para comunicaciones inalámbricas (Serie de Telecomunicaciones)*. McGraw-Hill, 2003. pp. xx, xxi, 14, 51-53, 73, 188-190, 195, 287.
- [2] Richard Clarke, *Cómo afrontar los cambios radicales de la seguridad en el ciberespacio*. Julio 6, 2004, ID: 4198.
http://www.symantec.com/region/mx/enterprisesecurity/content/expert/LAM_4198.html.
- [3] La Flecha, Diario de Ciencia y Tecnología, *El coste de los virus informáticos asciende a 92,000 millones de euros en 10 años (110,000 millones de dólares)*. Fuente: Agencia EFE, Enero 30, 2006. <http://laflecha.net/canales/seguridad/200601301>.
- [4] Hicham Tout, William Hafner, *An Agent-based, Application-Layer Security Framework*. The 2006 World Congress in Computer Science Computer Engineering, and Applied Computing, Las Vegas, Nevada, USA, Junio 26-29, 2006.
<http://ww1.ucmss.cm/books/LFS/CSREA2006/SWW7269.pdf>.
- [5] McAfee Proven Security, *Microsoft Word 0-Day Vulnerability III*. Diciembre 27, 2006.
http://vil.nai.com/vil/Content/v_vul27264.htm.
- [6] Erika Tinajeros A., *Nuevas formas de delinquir en la Era Tecnológica: Primeras observaciones sobre Espionaje, Fraude y Sabotaje Informático*. AR: Revista de Derecho Informático, Alfa-Redi, Septiembre 8, 2006. ISSN: 1681-5726, <http://www.alfa-redi.org/rdi-articulo.shtml?x=7182>.
- [7] Richard D. Pethia, *Ataques a la Internet en 2003, La internet en evolución, cuestiones mundiales, usinfo.state.gov (International information programs)*. Septiembre 10, 2003.
<http://usinfo.state.gov/journals/itgic/1103/ijgs/gj11a.htm>.
- [8] Richard D. Pethia, *Viruses and Worms: What Can We Do About Them?*. Software Engineering Institute, Carnegie Mellon University, CERT® Coordination Center, Pittsburgh, Pennsylvania, USA, Septiembre 10, 2003.
http://www.cert.org/congressional_testimony/Pethia-Testimony-9-10-2003/.
- [9] Douglas E. Comer, *Redes Globales de Información con Internet y TCP/IP (Principios básicos, protocolos y arquitectura)*. Prentice-Hall, 1996. pp. 93-94, 146-158, 168, 217, 229.
- [10] *Virus Reporte*, ESET Virus Radar On-Line NOD32 Antivirus System, Diciembre 23 y 26, 2006. http://www.virusradar.com/index_esn.html.
- [11] Conficker. <http://conficker.com/>.
- [12] La Flecha, Diario de Ciencia y Tecnología, *Skype, gusanos, troyanos, y el enemigo de siempre, Fuente: VS Antivirus*. Diciembre 22, 2006.
<http://www.laflecha.net/canales/seguridad/noticias/skype-gusanos-troyanos-y-el-enemigo-de-siempre>.
- [13] Cox, P. & Sheldon, T. *Windows 2000, Manual de Seguridad*. McGraw-Hill, 2003, ISBN: 970-10-3720-0 pp. 18-24, 22-34, 44, 47, 54-81, 177, 410-424, 475, 675, 712-717.
- [14] El Universal.com.mx (Computación), *Se propaga Gusano en YouTube*. Junio 18, 2007.
<http://www.el-universal.com.mx/articulos/40642.html>.

- [15] Min Cai, Kai Hwang, Yu-Kwong Kwok, Shanshan Song, Yu Chen, *Collaborative Internet Worm Containment*, *IEEE Security and Privacy*. Vol.03, No. 3, pp. 25-33, Mayo-Junio, 2005. ISSN: 1540-7993, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1439499.
- [16] Symantec Corporation, *About Symantec*. <http://www.symantec.com/about/index.jsp>.
- [17] Snort (Sistema Detector de Intrusos). http://www.snort.org/about_snort/.
- [18] MySQL. <http://www.mysql.com>.
- [19] Frederic Raynal, Yann Berthier, Philippe Biondi, Danielle Kaminsky, *Honeypot Forensics Part I: Analyzing the Network*. IEEE Security and Privacy, Vol. 02, no. 4, pp. 72-78, Julio-Agosto, 2004. ISSN: 1540-7993.
<http://csdl2.computer.org/persagen/DLabsToc.jsp?resourcePath=/dl/mags/sp/&toc=comp/mags/sp/2004/04/j4toc.xml&DOI=10.1109/MSP.2004.47>.
- [20] InMon (Internet Monitoring), *Using sFlow and InMon Traffic Server for Intrusion Detection and Other Security Applications*. <http://www.inmon.com/pdf/sFlowSecurity.pdf>.
- [21] Javed Aslam, Sergei Bratus, David Kotz, Ron Peterson, Brett Toferl Daniels Rus, *The Kerf toolkit for intrusion analysis*, *IEEE Security and Privacy*. Vol. 02, No. 6, pp. 42-52, Noviembre-Diciembre, 2004. ISSN: 1540-7993.
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1366118.
- [22] Groff James R, Weinbert Paul N. *SQL Manual de Referencia*. McGrawHill, 2002. pp. 95-276.
- [23] Kuper, P., *The state of security*, *IEEE Security and Privacy*. Vol. 03, No.5, pp. 51-53, Septiembre-October, 2005. ISSN: 1540-7993.
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1514402.
- [24] Bruce Schneier. <http://www.schneier.com/>.
- [25] Bruce Schneier, *SIMS: Solution, or Part of the Problem?*, *IEEE Security and Privacy*. Vol. 02, No. 5, pp. 88, 2004. ISSN: 1540-7993.
<http://doi.ieeeecomputersociety.org/10.1109/MSP.2004.83>.
- [26] Phillip A. Porras, *Privacy-Enabled Global Threat Monitoring*, *IEEE Security and Privacy*. Vol. 04, No. 6, pp. 60-63, Noviembre-Diciembre, 2006. ISSN: 1540-7993.
http://ieeexplore.ieee.org/xpls/abs_all.jsp?isnumber=4020218&arnumber=4020236&count=25&index=14.
- [27] Gerald A. Marin, *Network Security Basics*, *IEEE Security and Privacy*. Vol. 03, No.6, pp. 68-72, Noviembre-Diciembre, 2005. ISSN: 1540-7993.
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1556540.
- [28] Instituto Tecnológico de Monterrey, Campus Estado de México.
http://www.itesm.edu/wps/portal!/ut/p/kcxml/04_Sj9SPykssy0xPLMnMz0vM0Y_QjzKLN4j3DADJgFjGpvqRqCKOcAFfj_zcVP0goESkOVAkwFQ_Kic1PTG5Uj9Y31s_QL8gNzSi3NvREQD_pMIM/delta/base64xml/L0lJSk03dWIDU1EhIS9JRGpBQU15QUJFUkVSRU1nLzRGR2dkWW5LSjBGUm9YZmcvN18wX1A1?WCM_PORTLET=PC_7_0_P5_WCM&WCM_GLOBAL_CONTEXT=/wps/wcm/connect/CEM/Estado+de+M%C3%A9xico/Investigaci%C3%B3n/C%C3%A1tedras/Tecnolog%C3%ADas+de+la+Informaci%C3%B3n/Investigaci%C3%B3n+e+innovaci%C3%B3n+en+seguridad+computacional/.
- [29] Computer Emergency Response Team (*CERT*[®]). <http://www.cert.org.mx/estadisticas.dsc>.
- [30] Russell S. & Norving P., *Artificial Intelligence (A Modern Approach)*. Prentice-Hall, 1995. pp. 36-45, 57, 110, 151, 157, 161, 163, 165, 166, 171, 196-201, 304, 419.

- [31] Java Agent Development Framework (*JADE*). <http://jade.tilab.com/>.
- [32] Foundation for Intelligent Physical Agents. <http://www.fipa.org/repository/aclspecs.html>.
- [33] Foundation for Intelligent Physical Agents. <http://www.fipa.org>.
- [34] Wooldridge M. & Jennings N.R. *Intelligent agents: Theory and practice. The Knowledge Engineering Review*. Vol. 10 (2). 1995. pp. 116–152.
- [35] Kelley Dean, *Teoría de Autómatas y Lenguajes Formales*. Prentice-Hall, 1995. pp. 30, 105-170.
- [36] V. Aho Alfred, Sethi Ravi & D. Ullman Jeffrey, *Compiladores. Principios, técnicas y herramientas*. Addison Wesley Longman (Pearson), 1998. pp. 25-82.
- [37] De Hoog R., Martil R., Wielinga B., Taylor R., Bright C. & Van de Velde W. *The CommonKADS model set. ESPRIT Project P5248 KADS-II/M1/DM.1b/UvA/018/5.0*. University of Amsterdam, Lloyd's Register, Touche Ross Management Consultants & Free University of Brussels, Diciembre 1993.
- [38] INGENIAS. http://grasia.fdi.ucm.es/ingenias/Spain/ejemplos/ejemplo_actividades/analisis-construccion.php.
- [39] Veloso, M., Carbonell, J., Pérez, A., Borrajo, D., Furk, E., Blythe, J. *Integrating Planning and Learning: The Prodigy architecture*. Journal of Experimental and Theoretical Artificial Intelligence 7 (1). 1995.
- [40] Kinny, D., Georgeff, M., and Rao, A., *A Methodology and Modelling Technique for Systems of BDI Agents*. Informe. 1997.
- [41] Zambonelli, F., Jennings, N. R. and Wooldridge, M. (2003) *Developing multiagent systems: the Gaia Methodology*. ACM Trans on Software Engineering and Methodology, 12 (3). pp. 317-370.
- [42] GAIA. <http://www.auml.org/auml/documents/Gaia.doc>.
- [43] Caire, G., Leal, F., Chainho, P., Evans, R., Garijo, F., Gomez-Sanz, J. J., Pavon, J., Kerney, P., Stark, J., and Massonet, P., *MESSAGE - Methodology for Engineering Systems of Software Agents*. Eurescom P907. <http://www.eurescom.de/~public-webspace/P900-series/P907/index.htm>.
- [44] Ingrand F., Georgeff M. P. *Managing Deliberation and Reasoning in Real-Time AI Systems*. In Proceedings of the Workshop Innovative Approaches to Planning, Scheduling and Control, 284-291. 1990.
- [45] N. Carver and V. Lesser. *The evolution of blackboard control architectures*. Technical report, University of Massachusetts Amherst, Oct. 1992, pp. 19-23. <http://www.cs.siu.edu/~carver/ps-files/tr92-71.ps.gz>.
- [46] UML. <http://www.omg.org/technology/documents/formal/uml.htm>.
- [47] Bauer, B., Müller, J. P., & Odell, J., *Agent UML: A Formalism for Specifying Multiagent Interaction*. International Journal of Software Engineering and Knowledge Engineering (IJSEKE), vol. 11, no. 3, 2001.
- [48] Foundation for Intelligent Physical Agents. *Fipa Request Interaction Protocol Specification*. <http://www.fipa.org/specs/fipa00026/SC00026H.html>.
- [49] Robert Moore. *Logic and Representation*. CSLI Lecture Notes n° 39. CSLI Publications, Stanford, California, 1995. pp. 1-7, 11-14.

- [50] Nilsson, U. & Maluszynski, *Logic Programming and Prolog*. 2000. pp. 13-14, 103, 115, 245. <http://www.ida.liu.se/~ulfni/lpp/>.
- [51] Pajares P., Santos P. *Inteligencia Artificial e Ingeniería del Conocimiento*. Alfaomega Ra-Ma. 2006. pp. 48-63, 95-114, 216-226.
- [52] Bib: Stephen A. Thomas. *SSL and TLS Essentials Securing the Web*. 2000. Wiley Computer Publishing John Wiley & Sons, Inc, pp. 17.
- [53] Secure Shell (SSH). <http://www.ssh.com/>.
- [54] OpenSSH. <http://www.openssh.com/es/index.html>.
- [55] Secure Socket Layer (SSL). http://www.netscape.com/eng/security/SSL_2.html.
- [56] Wikipedia, Secure Socket Layer (SSL). http://es.wikipedia.org/wiki/Transport_Layer_Security.
- [57] OpenSSL. <http://www.openssh.com/es/index.html>.
- [58] SSH Communications Security. <http://www.ssh.com/>.
- [59] The Internet Engineering Task Force (IETF). <http://www.ietf.org/>.
- [60] BSD. <http://www.bsd.org/>.
- [61] OpenBSD. <http://www.openbsd.org/es/>.
- [62] OpenVPN. <http://openvpn.net/>.
- [63] Joan Herrera Joancomarti, Joaquín García Alfaro, Xavier Perramón Tornil. *Aspectos avanzados de seguridad en redes*. Fundación para la Universitat Oberta de Catalunya. Barcelona España, 2004. pp. L1 105, 107, 108, 120, 121. <http://www.uoc.edu>.
- [64] Stallng W. *Cryptography and Network Security*. Prentice-Hall, 2006. pp. 28-56, 95-130, 134-165, 259-286, 317-346, 377-393.
- [65] RSA algorithm << Security << Java Tutorial. http://www.java2s.com/Tutorial/Java/0490__Security/0740__RSA-algorithm.htm.
- [66] RSA Laboratorios. <http://www.rsa.com/rsalabs/>.
- [67] Using AES with Java Technology. http://java.sun.com/developer/technicalArticles/Security/AES/AES_v1.html.
- [68] National Institute of Standards and Technology. <http://csrc.nist.gov/archive/aes/index.html>.
- [69] Etherdetect Packet Sniffer. <http://www.etherdetect.com>.
- [70] Mehmed Kantardzic, John Wiley, Sons. *Data Mining: Concepts, Models, Methods, and Algorithms*. pp. 1.
- [71] Joyanes Aguilar, L., *Programación orientada a objetos*. Mc Graw Hill. pp. xvii-xix.
- [72] Friedman, M. y Kendel, A., *Introduction to Patter Recognition*. World Scientific. pp. 1, 3-4, 9-10.
- [73] N.J. Nilsson, *Introduction to Machine Learning*. Septiembre, 1996. pp. 81-96.
- [74] Miguel, A., Piattini, M., Marcos, E., *Diseño de Bases de datos Relacionales*. Alfaomega Ra-ma, 2000. pp. 39, 95.
- [75] Quinlan J., *Induction of Decision Trees*. Kluwer Academic Publishers, Machine Learning, Hingham, MA, USA, Volume 1, Issue 1. ISSN: 0885-6125. (1986). pp. 81-106.

- [76] Quinlan R & Cameron J., *Oversearching and Layered Search in Empirical Learning*. International Joint Conference on Artificial Intelligence. 1995. Montreal, Quebec, Canada.
- [77] Quinlan R., *Comparing Connectionist and Symbolic Learning Methods*. 1994. Cambridge, Massachussetts. pp. 445. <http://www.rulequest.com/see5-info.html>.
- [78] C4.5. <http://www2.cs.uregina.ca/~dbd/cs831/notes/ml/dtrees/c4.5/tutorial.html>.
- [79] Pavel Laskov, Patrick Düssel, Christin Schäfer and Konrad Rieck, *Learning Intrusion Detections Supervised or Unsupervised*. <http://ida.first.fraunhofer.de/~rieck/docs/iciap2005.pdf>.
- [80] Yacine Bouzida, Frederic Cuppens, *Neural Networks vs decision trees for intrusion detection*. <http://www.rennes.enst-bretagne.fr/~fcuppens/articles/monam06.pdf>.
- [81] KDDCUP'99. <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>.
- [82] KDDCUP'99. <http://kdd.ics.uci.edu/databases/kddcup99/kddcup.names>.

ANEXO 1

Diagramas *AUML* correspondientes al ejemplo implementado en *SISAG-W*.

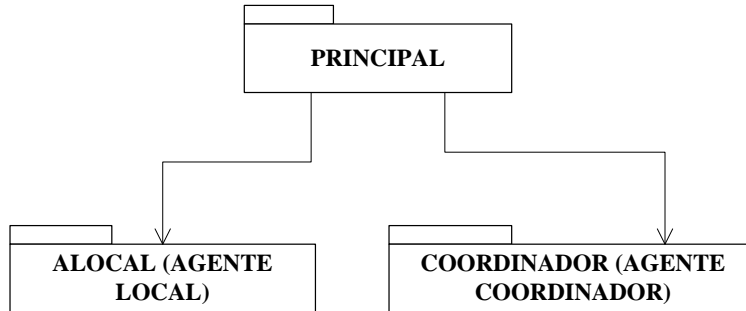


Fig. 66.- Diagrama de 'packages' de *SISAG-W*.

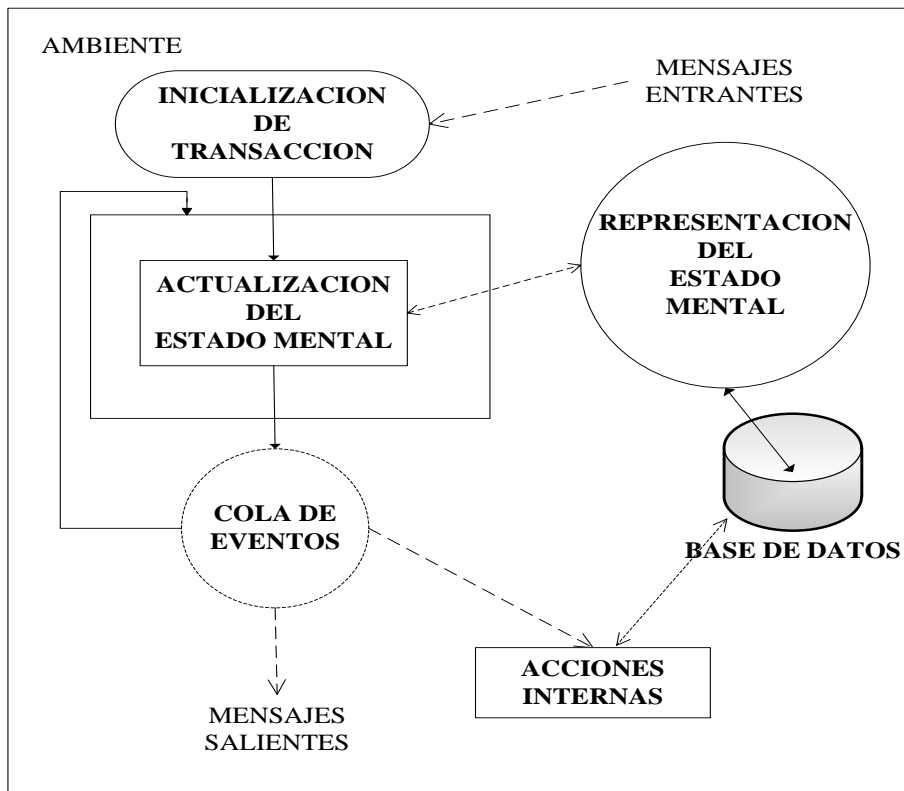


Fig. 67.- Flujo de control para registro y ejecución de transacciones.

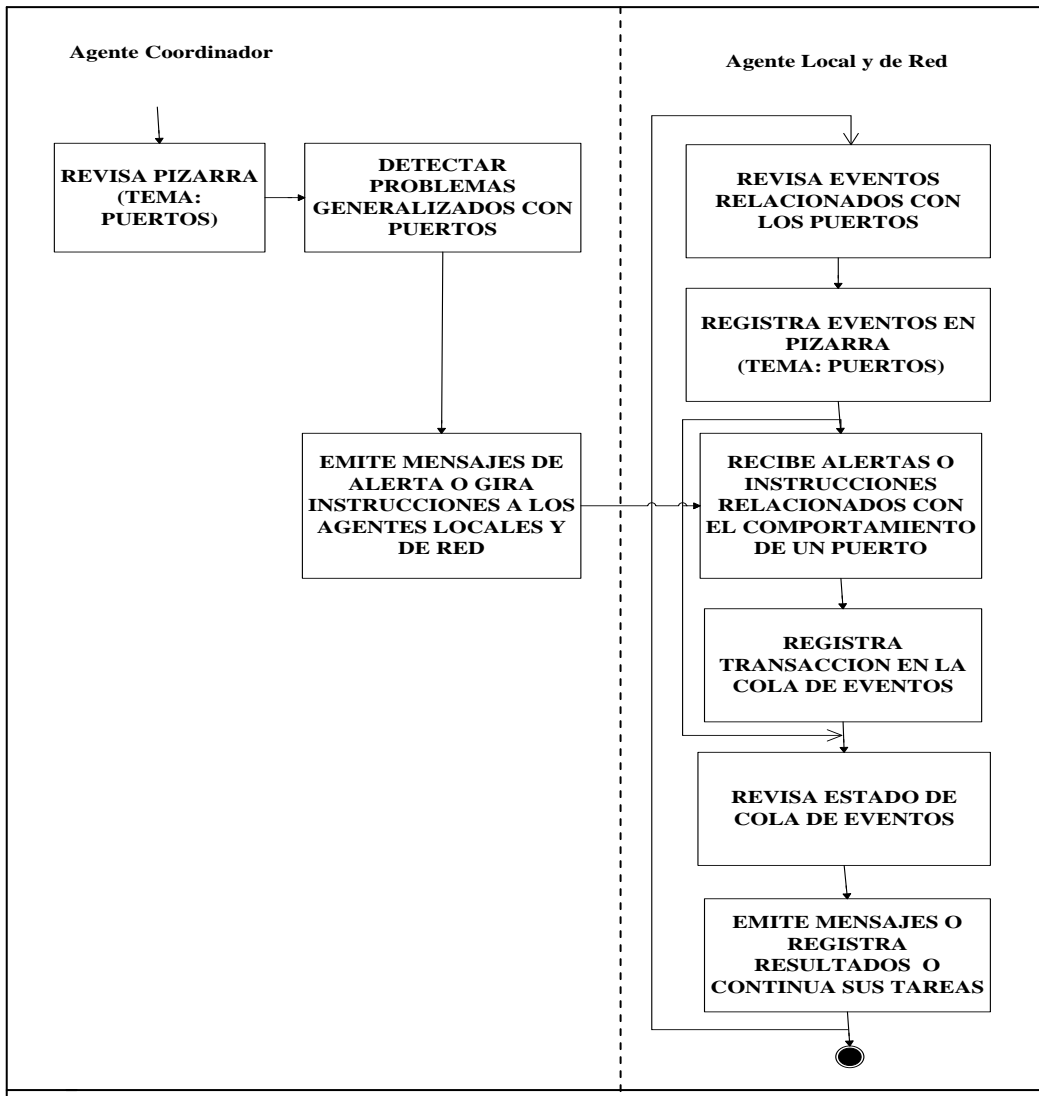


Fig. 68.- Diagrama de actividad de revisión de pizarra con el tema 'puertos'.

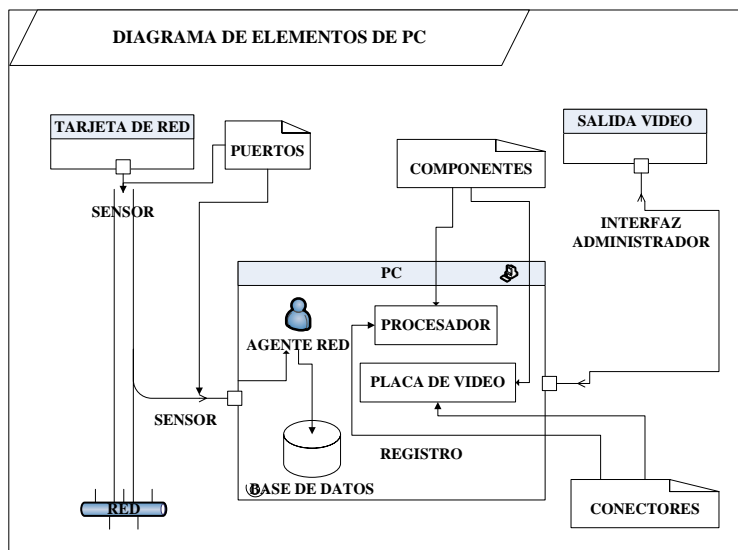


Fig. 69.- Diagrama general de elementos de Agente de Red.

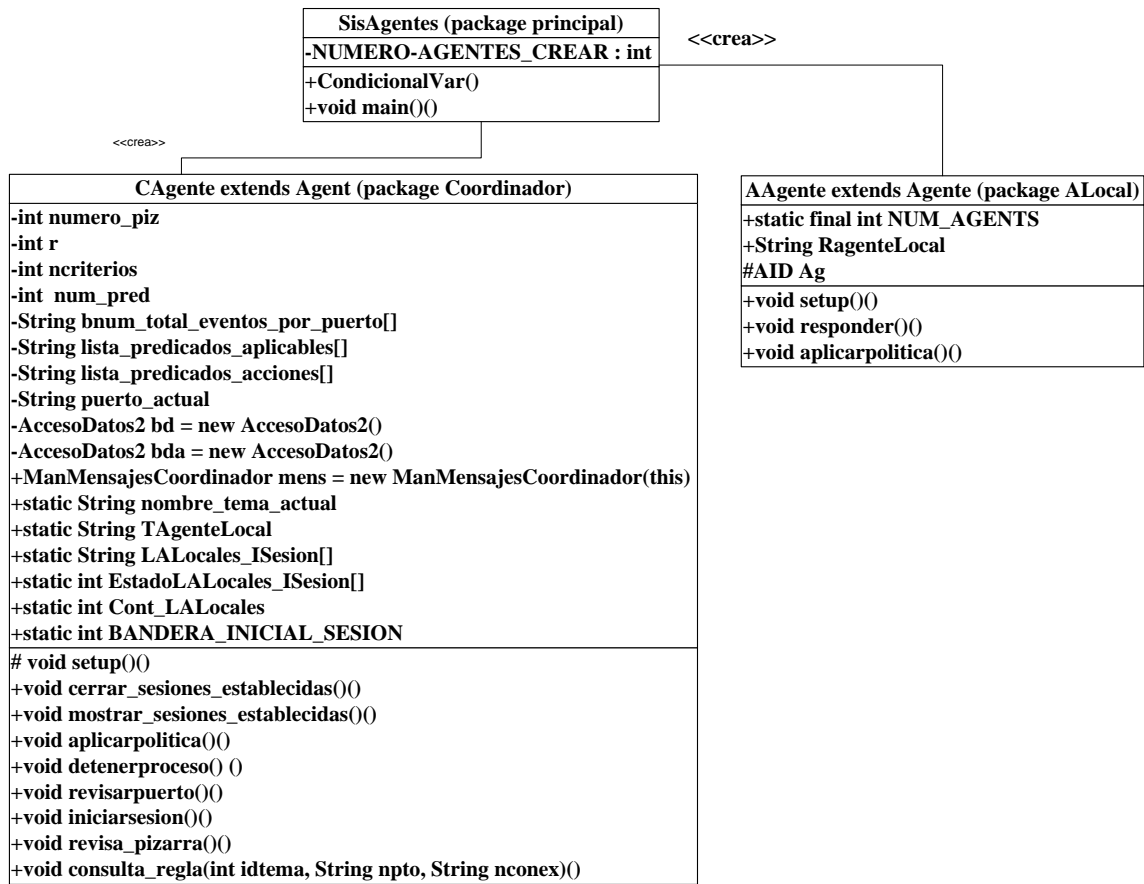


Fig.70.- Diagrama General de clases.

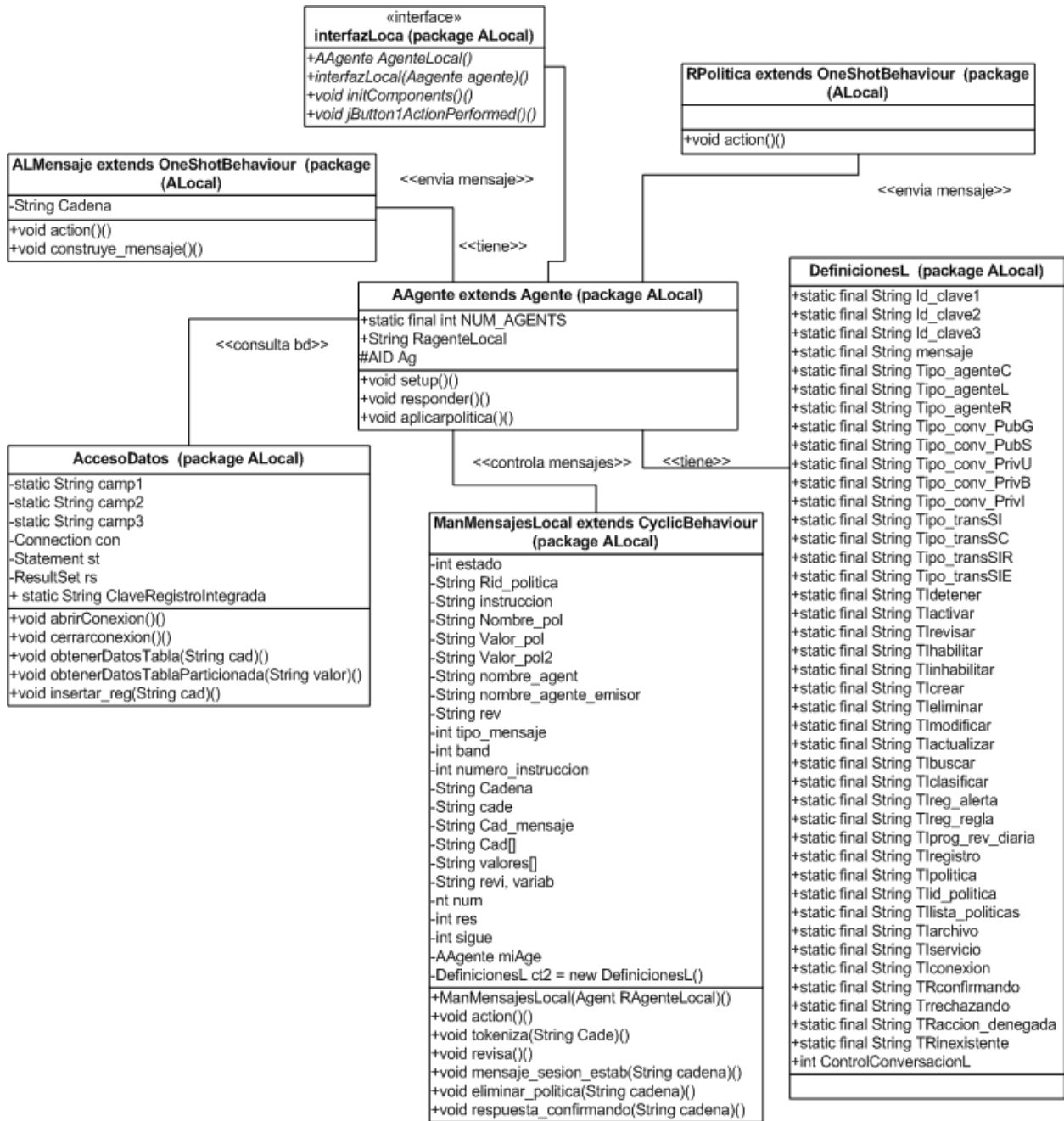


Fig. 71.- Diagrama General de clases del Agente Local.



Fig. 72.- Diagrama General de clases del Agente Coordinador.