Departamento de Tecnología Electrónica
Escuela superior de Ingeniería Informática
Universidad de Sevilla

TESIS DOCTORAL

INCLUSIÓN DE REGLAS EXPERTAS EN LOS MODELOS NORMALIZADOS PARA LA DESCRIPCIÓN DE LA ESTRUCTURA DE LA MIB.

APLICACIÓN A LA GESTIÓN DE AVERÍAS EN REDES DE COMUNICACIONES.

Autor

Antonio Martín Montes

Ingeniero Informático

Director

Carlos León de Mora

Doctor en Informática

Sevilla, 2005

AGRADECIMIENTOS

Dar las gracias a mi director de Tesis, D. Carlos León de Mora. Su trabajo, apoyo y valiosa orientación han hecho posible la finalización de esta tesis.

Igualmente agradecer a todo el Departamento de Tecnología Electrónica de la Universidad de Sevilla, especialmente a D. Joaquín Luque. Gracias a él amplié mis conocimientos acerca de los modelos de gestión de redes, lo que sin duda ha ayudado a la realización de la presente tesis.

A mi familia, a Carmen, a mis hijos…

Índice General.

Indice de Figuras

Indice de Tablas

Capítulo 1

Introducción y Objetivos.

La gestión de redes y servicios ha sido un campo en el que usualmente se han impuesto soluciones y mecanismos propietarios de distintos fabricantes, que exigían que la gestión de sus equipos y servicios sólo se pudiera realizar con el gestor del propio proveedor. De esta manera, en un entorno con equipos heterogéneos, existían también sistemas de gestión heterogéneos e incompatibles entre sí. A finales de los años 80 surgieron los denominados modelos de *Gestión de Red,* que definen de manera normalizada un protocolo y un modelo de información de gestión que rompe con el acceso de gestión remoto propietario, permitiendo teóricamente la interoperabilidad entre gestores y elementos gestionados, de múltiples fabricantes.

Asimismo, las redes de comunicaciones han crecido fuertemente en los últimos años, tratando de satisfacer las exigencias de calidad y máximo rendimiento, durante el mayor tiempo posible. Para que esto sea posible, es necesario realizar una gestión de la red asistida por un software avanzado. La Inteligencia Artificial se incorpora a la gestión de las redes, con el fin de facilitar las labores de administración y control de toda la información que proviene de los recursos gestionados, dando origen a la denominada Gestión Experta de las Redes de telecomunicaciones. Este nuevo modo, proporciona a los sistemas de gestión de un mayor grado de cohesión con las tecnologías de comunicaciones actuales, a la vez de disponer de todas las posibilidades y ventajas aportadas por los Sistemas Expertos.

Esta tesis tiene como objetivo perfeccionar las técnicas actuales de gestión. Para ello se establecen los mecanismos que permitan una mayor correlación entre las especificaciones de la red y las aplicaciones que efectúan el tratamiento de la información de gestión de una forma óptima y especializada. Presentamos una nueva concepción denominada *"Gestión Experta Integrada"*, que contempla la inclusión de la reglas del Sistema Experto de Gestión en las especificaciones de los objetos gestionados de la red de telecomunicaciones. Este modelo consigue reunir conceptos que actualmente pertenecen a distintos ámbitos de estudio, la Inteligencia Artificial y la Información de Gestión de sistemas de telecomunicaciones. De esta forma se obtiene una solución global, que permite a los administradores de redes utilizar la potencia aportada por la Inteligencia Artificial, en particular de los Sistemas Expertos, de una forma sencilla y transparente.

1.1 Grado de Innovación.

Los estándares de gestión tradicionales no están dotados de todas las características necesarias para efectuar una administración eficiente de los complejos sistemas telemáticos modernos. Es necesario desarrollar modelos de control y supervisión que ofrezcan mayores posibilidades, los denominados *Sistemas de Gestión Experta Integrada.*

Este nuevo paradigma, contiene aspectos claramente diferenciadores a las técnicas de gestión habituales, que hacen uso por separado de los Sistemas Expertos y de las plataformas de gestión. Existe una frontera definida entre el conocimiento aportado por el sistema experto y la plataforma encargada de la aplicación de dicho conocimiento en la administración y control de los Sistemas Telemáticos, pudiéndose considerar incluso cada uno por separado.

En esta tesis proponemos una aproximación original a la gestión de redes, en la cual el sistema experto queda completamente integrado en la plataforma de gestión. Concretamente, las reglas expertas de gestión, que componen la base de conocimientos del Sistema Experto, se fusionan con la definición de los elementos que conforman la red de comunicaciones. Cada una de estas definiciones, contendrá las especificaciones correspondientes a los recursos que componen el sistema de información y el conjunto de reglas expertas que posibilitan el control inteligente de los recursos que representan [León99].

Con la gestión Inteligente Integrada se obtienen las siguientes ventajas:

- ***Abstracción de los usuarios de la plataforma de gestión,*** con respecto a la configuración y las reglas expertas de gestión pertenecientes al sistema experto. No necesitan tener conocimientos de programación del Shell del Sistema Experto.

- ***Mayor rapidez en la resolución de problemas***. Al estar el sistema experto integrado en la plataforma de gestión formando una sola unidad modular, los retrasos y esperas producidas por la comunicación entre ambos elementos serán inferiores.

- ***Compatibilidad entre las distintas plataformas de gestión***. Integración del sistema experto en la norma *ISO 10165-4/ITU-T X.722, Guidelines for the Definition of Managed Objects*[1] [ITUT93]. Los estándares serán más rigurosos, fortaleciendo el grado de coexistencia y entendimiento entre las distintas plataformas y los modelos de gestión.

- ***Estandarización de las nuevas reglas expertas***, por parte de un organismo de normalización como es ISO. Esto asegura la correcta integración de las definiciones de las reglas expertas de gestión, en las especificaciones de los elementos sobre los que actúan.

- ***Reutilización de las reglas expertas***, mediante los mecanismos de herencia, propio de los lenguajes Orientados a Objetos. GDMO facilidad el uso de las definiciones y especificaciones existentes en el estándar. Igualmente proporciona un marco perfecto para la reutilización de las clases y las reglas expertas de gestión definidas. El encapsulamiento y la Modularidad permiten utilizar una y otra vez las mismas clases en dominios de gestión distintos. Las relaciones existentes entre las distintas clases, hacen posible el añadir una nueva clase o un modulo nuevo (extensibilidad), sin afectar al resto de las especificaciones.

- ***Agilidad para la definición y realización de nuevas plataformas de gestión expertas.*** Se afianza la compatibilidad entre las distintas plataformas de gestión de sistemas telemáticos, a la vez que facilitan la comunicación y el trabajo conjunto entre ellas.

[1] *Guía para la definición de objetos gestionables.* Realiza una amplia exposición de los principios básicos para la definición de los objetos gestionados, que van a servir de orientación a quienes realizan la especificación de dichos objetos, así como favorecer la coherencia entre las definiciones que de ellos se establecen.

Para que esto sea posible, es fundamental explotar las capacidades que tienen los modelos de información de gestión. En particular, el estándar GDMO especifica propiedades y características, que facilitan la inclusión de las reglas expertas de gestión como parte de las definiciones de recursos gestionados. De esta forma, los gestores de red inteligentes pueden interpretar las reglas aportadas por el Sistema Experto y realizar un tratamiento inteligente integro de la información de gestión.

1.2 Objetivos de la Tesis Doctoral.

El objetivo y contribución principal de esta tesis, es proveer de un método general y sistemático, que permita a un gestor OSI contener las reglas expertas necesarias para realizar su propia administración y control. Las reglas expertas definidas, representan la base de conocimiento del sistema experto, recogen las acciones y condiciones bajo las cuales se efectúan las operaciones de gestión correspondientes

Se pretende eliminar la ruptura existente entre ambos componentes, sistema experto y plataforma de gestión. Del mismo modo, favorecer que el estándar de gestión OSI integre las bases de conocimientos aportadas por los Sistemas Expertos que pueden corresponder a distintos dominios de gestión [Hegering94].

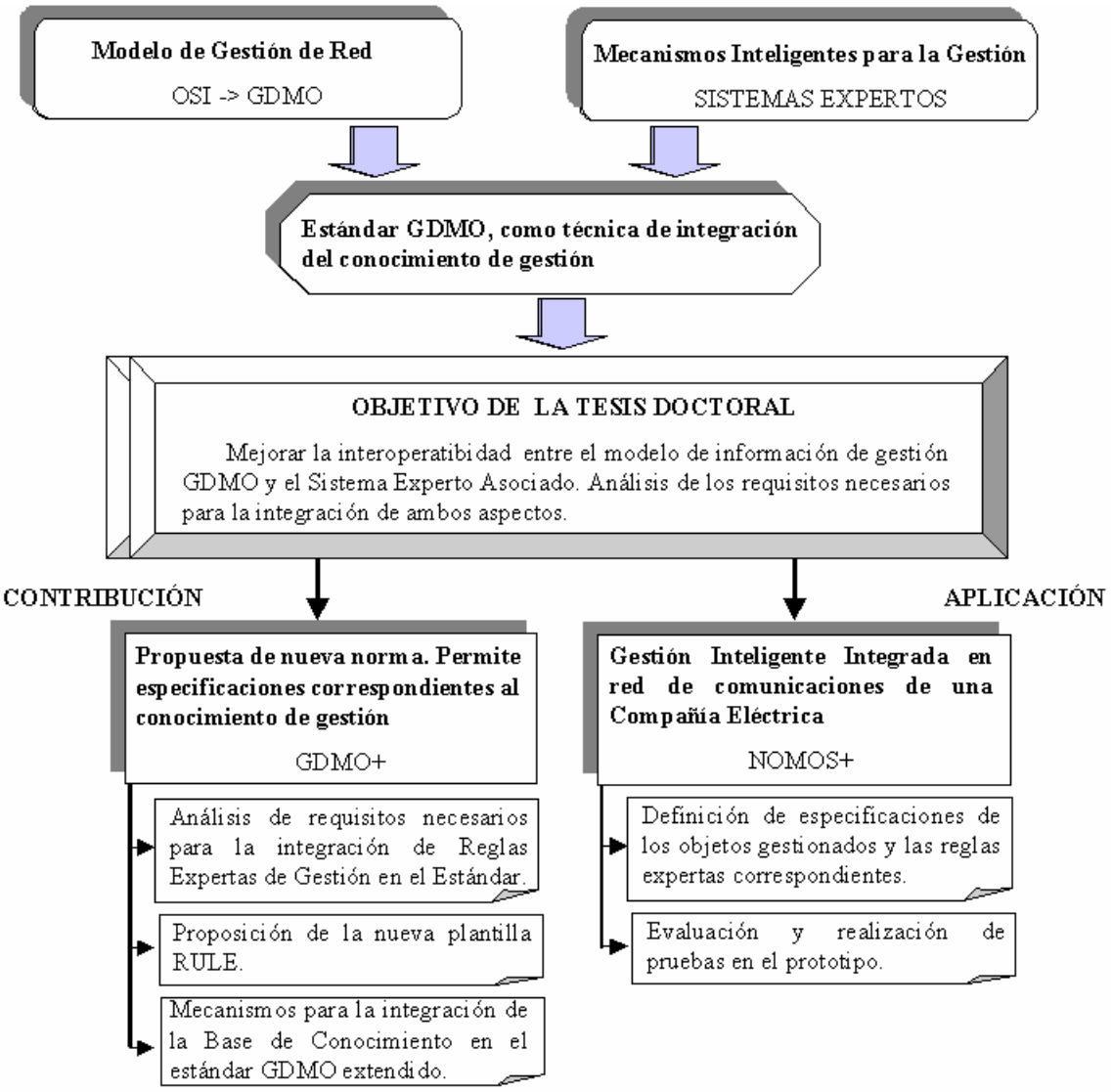

Figura 1.1 Ámbito de desarrollo de la tesis.

La anterior Figura 1.1 ilustra el ámbito de estudio de la tesis doctoral, sus objetivos y las contribuciones con las que pretende alcanzarlos.

Con objeto de alcanzar el objetivo general, este trabajo incluye las siguientes proposiciones:

- **Percepción de las necesidades requeridas por la Gestión Experta Integrada.** Identificación de metas para la integración del Sistema Experto y la definición de los Objetos Gestionados.
- **Plantear medidas que mejoren la calidad global de GDMO y sus especificaciones.**
- **Propuesta de nueva sintaxis de la norma GDMO,** que permita la composición de ambos factores, abriendo nuevas vías de mejoras para el estándar.

1.3 Estructura.

Este apartado presenta la estructura seguida en la tesis. Se organiza en siete capítulos y seis anexos, el contenido correspondiente queda resumido a continuación.

- El Capítulo 2 presenta el estado del arte de la gestión de redes de comunicaciones, su normalización y las principales tendencias futuras. Visión general de los modelos de gestión actuales: OSI, TNM, INTERNET, etc. Se estudian también los distintos elementos que lo constituyen: protocolos, estructuras, definición de información, etc.

- El Capitulo 3 hace un estudio de los Sistemas Expertos y sus áreas de aplicación en la gestión de redes. Los aspectos más importantes para su construcción, adquisición del conocimiento de un dominio, técnicas de representación del conocimiento, ciclo de vida, etc. Para finalizar se estudian los sistemas expertos en el dominio de la gestión de redes y se indican sus características principales. Haremos especial hincapié en la Gestión experta tradicional, en este apartado se desarrollan los aspectos propios de la configuración, que tradicionalmente se establece para la gestión de redes asistida por un sistema experto.

- El Capitulo 4 profundiza en los distintos aspectos del modelo de gestión OSI. Examinamos la sintaxis GDMO, *Guía para la Definición de Objetos Gestionados*. Se estudia la arquitectura y plantillas que conforman el estándar actual. Sobre esta norma se va a basar el estándar propuesto y principal objeto de estudio de la tesis.

- El Capítulo 5 detalla la propuesta original planteada en la tesis, introduce los mecanismos de Integración de las reglas expertas en la especificación de objetos gestionados. Se exponen las ventajas que conlleva unificar la especificación de los objetos gestionados y las reglas para su gestión. Se sugiere una Extensión de

GDMO, se describen los nuevos elementos sintácticos que la extienden y que permiten la inclusión de reglas expertas, en las definiciones de los objetos gestionados.

Paralelamente se expone el proceso para introducir las mejoras, que consiste en la adición de una nueva plantilla RULE y las distintas propiedades que la constituyen. Se realiza un análisis de las características que posee la nueva plantilla RULE y que permiten representar el conocimiento en un formato aplicable al lenguaje de definición de información del modelo OSI. Se proponen los mecanismos válidos para integración de dicho conocimiento.

- En el Capítulo 6 se plantea un caso practico, los resultados de nuestra investigación se aplican en una red de comunicaciones privada perteneciente a una compañía del sector eléctrico. Para ello se modelan los objetos correspondientes al *Sistema de Supervisión de Comunicaciones* (SSC) en el estándar GDMO+. Se describen las reglas de producción en la nueva notación y se procede al análisis y diseño de un prototipo. Para su programación se utiliza Art*Enterprise, herramienta de *Entorno de Desarrollo de SBC* que permite la construcción de Sistemas Expertos (Anexo D).

- Capítulo 7: este último capítulo presenta un resumen de todo el estudio, valoración y análisis del trabajo realizado, así como las conclusiones. Se señalan además los aspectos de la contribución propuesta en esta tesis doctoral, que pueden ser mejorados y que marcan las líneas futuras de investigación.

- Los Anexos recogen información correspondiente a distintos aspectos vinculados con el estudio desarrollado: Relación de los distintos estándares existente en la gestión de redes, Sintaxis Abstracta y Sintaxis de Transferencia ASN.1., especificaciones GDMO+ correspondientes al prototipo desarrollado, y una introducción a la herramienta de construcción de Sistemas Expertos ART*Enterprise.

- Para finalizar se incluyen las Referencias Bibliográficas y el Glosario de Términos.

Capítulo 2.

La Gestión de Redes de Comunicaciones y su Normalización.

Una vez se ha presentado el ámbito y objetivos de la tesis, este capítulo presenta una panorámica del estado del arte de la Gestión de Red Integrada tradicional, desde el punto de vista de la problemática expuesta en el capítulo de introducción y objetivos.

Las redes de equipos surgen como respuesta a la necesidad de compartir datos de forma rápida. Ofrecen capacidades de acceso a servicios de otras redes, posibilidades adicionales de proceso, accesos a bases de datos internacionales, etc. La existencia de dispositivos de comunicaciones dispersos, sobre los que se implementan e interconectan todas estas redes, obliga a disponer de sistemas de gestión para la configuración, supervisión, diagnóstico y mantenimiento de todos los elementos interconectados. Si bien en un principio, la gestión de red significaba gestión más o menos individualizada de los elementos de red, actualmente se tiende a tener una gestión única de ésta. En el camino se ha pasado por la coexistencia de sistemas de gestión y la integración de los mismos.

2.1 Definición de Gestión de Red.

Para determinar qué es la Gestión de Red, se presentan varias descripciones procedentes de distintas fuentes:

Según [Hegering99] *la gestión de sistemas interconectados se compone de todas las medidas necesarias para asegurar la operación efectiva y eficiente de un sistema y sus recursos, según los objetivos de una organización*. El propósito de la gestión es proporcionar los servicios y aplicaciones de un sistema en red con el nivel deseado de calidad, garantizar la disponibilidad, despliegue rápido y flexible de los recursos interconectados. Si la prioridad es la gestión de la red de comunicaciones y sus componentes, se hablará de gestión de red; si el énfasis está en los sistemas finales, se referencia a la gestión de sistemas; finalmente, la gestión de aplicaciones es la responsable de las aplicaciones y servicios que se proporcionan en un sistema distribuido.

La importancia de esta definición está en indicar que el ámbito de la gestión no se restringe a la red, sino a los sistemas que interconecta, las aplicaciones y servicios que proporcionan dichos sistemas. Este mismo autor propone distintas dimensiones de la gestión.

Según [T101] *la gestión de red consiste en la ejecución del conjunto de funciones requeridas para controlar, planear; asignar, desplegar, coordinar y monitorizar los recursos de una red de telecomunicación*, incluyendo funciones de rendimiento tales como planear la red inicial, asignación de frecuencias, encaminamiento de tráfico predeterminado para soportar balance de carga, autorización de la distribución de claves criptográficas, gestión de configuración, gestión de

fallos, gestión de seguridad, gestión de rendimiento y gestión de contabilidad. Aquí se hace hincapié en las áreas funcionales de la gestión de red, siguiendo el modelo FCAPS (*Fault, Configuration, Accounting, Performance and Security)*, Fallos, Configuración, Contabilidad, Rendimiento y Seguridad [ITUT00d], Figura 2.1.

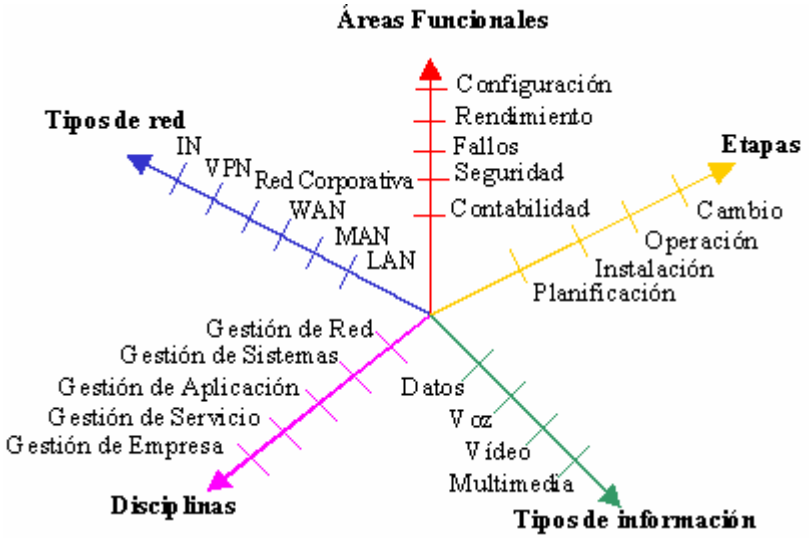

Figura 2.1. Dimensiones de la Gestión

Como se ve, las definiciones son parcialmente similares por lo que se puede concluir que ***la Gestión de Red consiste en la planificación, organización, supervisión y control de elementos de comunicaciones para garantizar un nivel de servicio, y de acuerdo a un coste.***

2.1.1 Objetivos y Necesidades.

La gestión de red, por consiguiente contiene un conjunto de tareas de monitorización, información y control, necesarias para operar efectivamente en una red. Estas operaciones pueden ser distribuidas sobre diferentes nodos de la red, lo cual puede requerir repetidas acciones de recogida de datos y análisis, cada vez que sucede un nuevo evento en el sistema [Staling00]. Las tareas de gestión, se llevan a cabo por el personal responsable o por procesos de administración automáticos.

Algunos de los objetivos de la gestión de red son los siguientes:
- Detección de fallos y corrección con la máxima rapidez posible.
- Gestión de contabilidad y uso que hace del sistema.
- Gestión de seguridad.
- Instalación y distribución de software en el sistema de una manera controlada.
- Gestión de los componentes del sistema y configuración del mismo.
- Planificación y crecimiento del sistema de manera controlada.
- Monitorización del rendimiento, detección de cuellos de botella y optimización de los mismos.

El objetivo final de la gestión de red es garantizar un nivel de servicio en los sistemas de una organización durante el máximo tiempo posible, minimizando la perdida que ocasionaría una parada o funcionamiento incorrecto del sistema. Los elementos que son objeto de control por un Sistema de Gestión en una Red de ordenadores son fundamentalmente los equipos conectados: servidores, terminales, ordenadores personales, estaciones de trabajo, así como los elementos y equipos de interconexión tales como cables, concentradores, repetidores, puentes, routers, etc.

2.1.2 Monitorización y Control de los Sistemas.

Se van ha distinguir dos aspectos diferentes, presentes en la gestión de red: la Monitorización y el Control del sistema administrado.

2.1.2.1 Monitorización.

Las acciones consistentes en obtener información de la red con el fin de detectar anomalías, se definen como monitorización. Dichas acciones son pasivas y su único objetivo es conocer el comportamiento de los recursos gestionados. La monitorización abarca cuatro aspectos:

1.- Definir la información de gestión que se monitoriza. Según su naturaleza temporal podrá ser:

- *Estática,* caracteriza la configuración de los recursos y cambia con muy poca frecuencia.

- *Dinámica,* asociada a eventos que se dan en la red. Dentro de este segundo tipo vamos a tener la información estadística, obtenida al procesar la información dinámica.

2.- Acceso a la información de monitorización. Los módulos gestores necesitan acceder a los módulos agentes que se localizan en los recursos. El acceso se realiza mediante los denominados protocolos de intercambio de información de gestión, punto clave en la gestión de redes.

3.- Diseño de políticas de monitorización. Se fundamenta en un Sondeo periódico por parte del gestor a los agentes, preguntando por los datos de monitorización. Existe además el mecanismo de informe de eventos, en el cual los agentes informan por propia iniciativa a los gestores, ante un cambio de estado significativo.

4.- Procesado de la información de monitorización. Tratamiento de los datos obtenidos y posterior estudio de conclusiones. Dependerá de la función para la que se ha realizado la monitorización.

En definitiva con la monitorización, se va a decidir que información se va ha recoger del sistema, como se accede y que se hace con ella, figura 2.2.

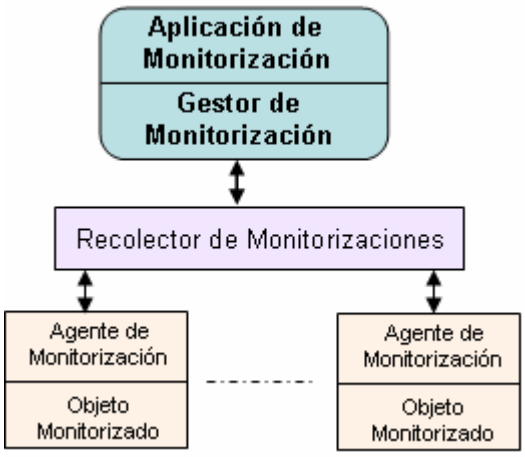

Figura 2.2 Modelo Resumen

2.1.2.2 Control.

Por otro lado el control, determinará el comportamiento de la red, encargándose de modificar parámetros e invocar acciones. Las tareas de control aportan potencia a los sistemas de gestión, permitiendo en todo momento y de forma remota, determinar las características del comportamiento de la red [Sloman95].

Las funciones de control están agrupadas según sus funciones. A la hora de definir cuáles son las funcionalidades básicas que ha de soportar un sistema de gestión, son multitud las clasificaciones que tradicionalmente se han venido haciendo atendiendo a criterios de diversa naturaleza. Sin embargo, la ISO (*International Standarization Organization*) Organización Internacional de Estandarización, realizó una clasificación de las tareas de los sistemas de gestión en cinco áreas funcionales, (norma ISO 7498-4 o su equivalente CCITT X.700). Se agrupan en gestión de fallos, gestión de contabilidad, gestión de configuración, gestión de prestaciones y gestión de seguridad.

Esta clasificación fue originariamente aplicada a los sistemas de gestión de la torre de protocolos OSI (Open Systems Interconnection) *"Interconexión de sistemas abiertos"* y más tarde aplicada directa o indirectamente a la práctica totalidad de los ámbitos de gestión, según veremos en los próximos apartados del capítulo.

2.2 El paradigma Gestor-Agente.

Concepto fundamental compartido por la mayoría de los sistemas de gestión que se pueden encontrar en el mercado. Se basa en la posibilidad de clasificación de los componentes de una red, en dos grandes grupos (ver Figura 2.3):

- Gestores. Son los elementos de un sistema de gestión que interaccionan con los operadores humanos y desencadenan las acciones pertinentes para llevar a cabo las operaciones por ellos invocadas.

- Agentes. Son los componentes de un sistema de gestión que llevan a cabo las operaciones de gestión invocadas por el gestor (o gestores) de la red.

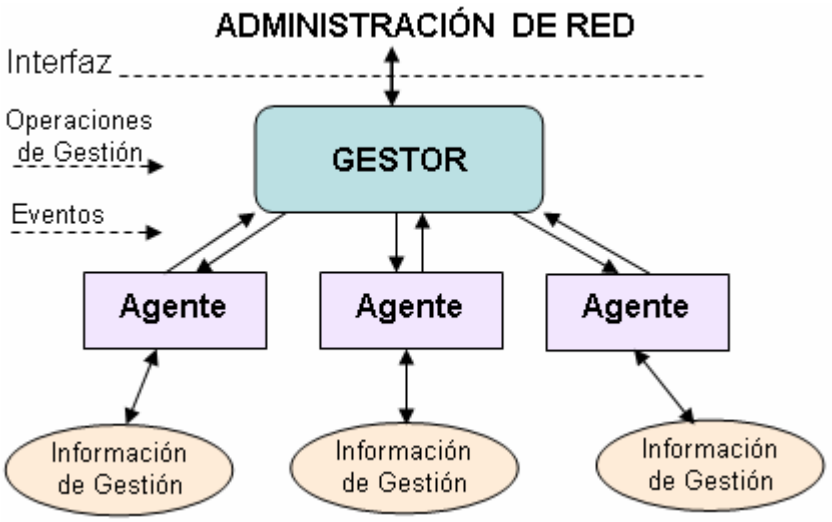

Figura 2.3. El Paradigma Gestor-Agente

De esta forma, los recursos de una red que posean un gestor se denominarán nodos gestores y los que tengan un agente de gestión, recibirán el nombre de nodos gestionados.

La base del funcionamiento de estos sistemas de gestión, reside en el intercambio de información de gestión entre nodos gestores y gestionados, según el paradigma gestor-agente. Habitualmente, los agentes mantienen en cada nodo gestionado información acerca del estado y las características de funcionamiento de un determinado recurso de red. El gestor pide al agente que realice determinadas operaciones con esa información. Gracias a esas operaciones, el gestor podrá conocer el estado del recurso y podrá influir en su comportamiento mediante la alteración de los datos de gestión.

El papel pasivo de los agentes, se rompe cuando se produce alguna situación excepcional en el recurso gestionado. Es entonces cuando estos, por propia iniciativa, emiten los denominados eventos o notificaciones que son enviados a un gestor, para que el sistema de gestión pueda actuar en consecuencia.

2.3 Evolución de los Sistemas de gestión.

La gestión de redes surgió con las redes mismas, puesto que siempre hubo necesidad de controlar, configurar, etc., los recursos de interconexión. Ahora bien, al igual que las redes han ido cambiando, los sistemas de gestión también han ido evolucionando con el paso del tiempo. Básicamente se pueden distinguir tres etapas fundamentales en la evolución de los sistemas de gestión:

- *Gestión Autónoma:* Las primeras redes tenían pocos nodos y cada uno de ellos poseía su propio sistema de gestión local. Las decisiones que afectaban a más de un nodo implicaban la comunicación con cada uno de los administradores correspondientes.

- *Gestión Homogénea:* En una etapa posterior, las redes sufrieron un aumento considerable de tamaño, pero siempre utilizando equipos y protocolos de un mismo fabricante. Ese mismo fabricante aportaba su propio sistema de gestión propietario que, en la mayoría de las ocasiones, estaba centralizado en un único nodo.

- *Gestión Heterogénea:* Más tarde, ya en nuestros días, las redes han ido creciendo y evolucionando mediante la incorporación de una amplia variedad de tecnologías. Ya no se puede hablar de entornos homogéneos sino que en una misma red se pueden encontrar componentes de una gran amalgama de fabricantes que tienen que interoperar entre sí. De esta forma se ha conseguido aumentar los servicios ofrecidos por las redes, a la vez que los usuarios de las mismas han podido maximizar el rendimiento de sus inversiones.

Esta evolución de las redes ha traído consigo la necesidad de que coexistan sistemas de gestión de red, de muy diversa naturaleza.

2.3.1 La Gestión Integrada.

La gestión heterogénea descrita en la sección anterior plantea una serie de importantes problemas:

- Desde el punto de vista del usuario, la necesidad de que la persona o personas encargadas de la administración de la red, conozcan perfectamente todos y cada uno de los sistemas que se deben utilizar.

- Desde el punto de vista de integración de sistemas, la incompatibilidad entre datos de gestión, procedimientos y protocolos de comunicación con funcionalidad similar. También la duplicidad y posible inconsistencia de la información almacenada en las bases de datos.

En respuesta a estas situaciones se han definido los modelos de gestión integrada, que permiten teóricamente, la interconexión de una manera abierta de los recursos de telecomunicación y las aplicaciones de gestión de red. De esa forma se podría evolucionar a modelos capaces de gestionar de una forma integrada redes heterogéneas.

Para llegar a esta integración hay que tener en cuenta los siguientes apartados:

- ***Normalizar las comunicaciones:*** Entre los diferentes componentes del sistema de gestión. Está claro que si un sistema de gestión quiere controlar un router, es necesario que éste sepa entender las preguntas que el sistema de gestión le haga con independencia del tipo de router y del tipo de sistema de gestión.

- ***Normalizar la información.*** Este aspecto es una de las claves de los modelos de gestión de red integradas actual y que diferencian a la gestión de red de otras aplicaciones de comunicación. El objetivo es conseguir una definición sintácticamente uniforme de todos los elementos de la red, con independencia del fabricante.

Esto plantea un gran trabajo, pues es necesario realizar una definición de las propiedades de gestión de todos los recursos de comunicación existentes, por ello la definición de la información de gestión correspondiente, deber ser una tarea más en el diseño de nuevos recursos de comunicaciones.

2.4 Estado del Arte

Las redes de comunicaciones de datos se han convertido en un componente fundamental dentro de la infraestructura corporativa, imponiendo a su vez unas exigencias muy altas a los sistemas de gestión de dichas redes. Las plataformas de gestión actuales se quedan cortas a la hora de responder a estas necesidades, especialmente cuando se aplican a redes a gran escala y en aplicaciones críticas.

A continuación se analizan las principales tendencias que se detectan en el segmento de la gestión de redes.

2.4.1 Sistemas Distribuidos.

Con el fin de evitar que toda la información de gestión confluya en un único puesto central, la tendencia hoy en día se dirige hacia la distribución de la inteligencia y la información por toda la red. Se pretende de este modo simplificar la gestión por medio de la automatización, de forma que las decisiones básicas se tomen cerca del origen del problema. Mediante la gestión distribuida es posible controlar redes de gran extensión de una manera más efectiva, dispersando entre varias estaciones de gestión las tareas de monitorización, recogida de información y toma de decisiones.

La funcionalidad básica que ha de ofrecer un sistema distribuido es la siguiente:

- *Escalabilidad,* para poder satisfacer las necesidades de gestión de redes de complejidad creciente en recursos y en información almacenada.

- *Capacidad para distribuir* entre distintas estaciones remotas de la red las funciones de supervisión, recogida de datos y sondeo de estado.

- *Capacidad para gestionar* entornos enormemente heterogéneos en el tipo de recursos de red y sistemas que los componen.

- *Alta disponibilidad* del sistema de gestión y tolerancia a fallos de componentes.

- *Capacidad para incorporar nuevos servicios* e integrarlos con los existentes.

- *Capacidad para interoperar* con diversos entornos.

En esta línea se están realizando esfuerzos para integrar la arquitectura de objetos distribuidos CORBA (Common Object Request Broker Architecture) en los modelos de gestión tradicionales (CMIP/SNMP). CORBA es más potente que SNMP y menos complejo que CMIP. A esto se añade la ventaja que supone su proximidad a C++ y Java, dos lenguajes de gran difusión.

La mayor dificultad que presenta la integración de CORBA con los sistemas tradicionales de gestión es el modelo de objetos. SNMP es completamente no orientado a objeto mientras que CMIP, a pesar de serlo, utiliza una aproximación que difiere mucho de la empleada en CORBA. A la hora de integrar CORBA y SNMP la opción natural es la adopción de una estrategia de pasarela (gateway) que mapee los paquetes SNMP en tipos de datos del lenguaje de definición de interfaz de CORBA (IDL).

En el caso de CMIP se han planteado dos aproximaciones posibles:

- ***La estrategia de pasarela***, similar a la seguida en el caso anterior y que consiste en mapear cada objeto del modelo CMIP (GDMO) y cada operación dentro del entorno CORBA. La desventaja de esta aproximación radica en que, al existir una correspondencia uno a uno entre uno y otro entorno, no se obtiene ningún valor añadido de la integración.

- ***La aproximación mediante la definición de objetos abstractos***. En este caso, un grupo de objetos del entorno CMIP se mapea mediante un único objeto CORBA, el cual representa entidades de gestión de nivel superior. Mediante este modelo se saca el mejor partido de ambas tecnologías.

CORBA también se perfila como alternativa de implantación de los objetos de nivel de servicio del modelo TMN, todavía por definir [Siegel96].

2.4.2 Gestión orientada a servicios.

La aproximación tradicional a la problemática de la gestión de redes se ha centrado en los dispositivos de red. Esto ha dado lugar en muchos casos, a situaciones en las que a pesar de mantener un alto nivel de rendimiento en los componentes aislados, no se obtenía la calidad del servicio requerido. En gran medida esto se debe a que resulta difícil establecer una conexión entre la gestión de dichos componentes de red y los procesos de negocio a los que están dando soporte dentro de la empresa.

La arquitectura de gestión de redes TMN que contempla en su modelo de niveles de gestión una capa específica de gestión de negocio, parece la mejor posicionada para dar respuesta a estas necesidades.

2.4.3 Gestión basada en Web.

El gran crecimiento de Internet y la introducción en las redes empresariales de las tecnologías que le son propias, está llegando también al ámbito de la gestión de redes. Mediante la adopción de este paradigma se posibilita un acceso universal a los sistemas de gestión desde cualquier plataforma que soporte los estándares de Internet (HTML, Java).

En esta línea, los fabricantes de dispositivos de red (routers, comnutadores, etc.) están integrando en sus equipos el software que les permite actuar como servidores web. Del mismo modo, se están realizando esfuerzos para la definición de nuevos estándares de gestión que, integrando protocolos como SNMP, HTTP y otros en una misma arquitectura, permita la gestión desde cualquier plataforma.

Los esfuerzos para definir un interfaz de gestión basado en Java, también se enmarcan dentro de esta estrategia unificadora. Se trata en este caso de aprovechar la característica de los módulos de software desarrollados en este lenguaje, que puedan ser ejecutados en cualquier plataforma.

2.4.4 Gestión inteligente.

Los nuevos sistemas de gestión de red están basados en desarrollos de inteligencia artificial, de forma que el sistema de gestión permite descargar de trabajo al administrador de la red. Existen dos técnicas básicas de inteligencia artificial que pueden emplearse en la gestión de redes:

- Sistemas Expertos. Los sistemas expertos de gestión de red simulan el proceso humano de toma de decisiones, aplicando una serie de reglas para escoger la mejor respuesta a un conjunto de circunstancias o eventos.

La base de conocimiento y las reglas que utiliza un sistema experto son suministradas por seres humanos, y deben adaptarse a cada red concreta antes de poder usarse con confianza. Estos sistemas no son, por el momento, capaces de aprender por sí mismos cómo gobernar una red, pero han mejorado mucho las capacidades de los gestores de la red.

- Modelado Inductivo. Cada parte del sistema se modela por separado, se caracterizan mediante estructuras de datos y código que representa la función del elemento. Cada elemento interacciona con los demás intercambiando señales y datos.

Para realizar este modelado se utiliza tecnología orientada a objetos. La característica de herencia de la orientación a objetos permite la creación de nuevos objetos basados en los ya existentes. A los datos se les asocian deducciones que se activan cuando se produce un cambio en sus valores. Los eventos activan deducciones que reaccionan con los modelos de los elementos la red, originando otras deducciones sobre el nuevo estado de la red. La estación de gestión no tiene conocimiento de todos los eventos posibles, sólo responde por deducción a cada nuevo conjunto de condiciones.

2.5 Arquitectura de gestión de Red.

El esquema de funcionamiento general de una plataforma de gestión puede verse en la siguiente figura 2.4.

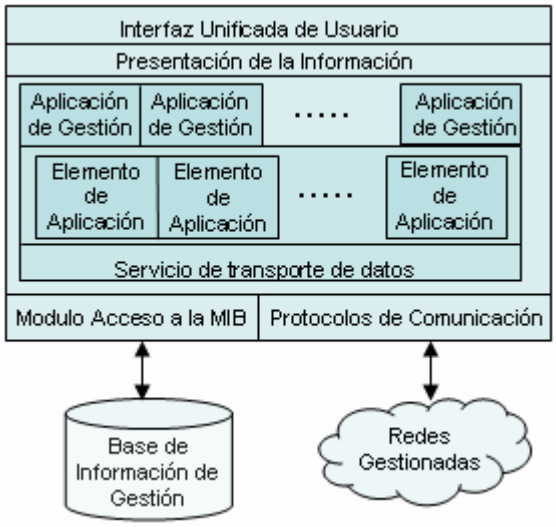

Figura 2.4 Arquitectura de un Sistema de Gestión de red

El usuario a través de una interfaz unificada tiene acceso a la información procedente de las diversas aplicaciones de gestión (gestores). Esto se requiere así, puesto que la diversidad de elementos de red procedentes de diferentes fabricantes, junto con la enorme cantidad de funciones de gestión definidas por los estándares, aconseja el procesado en paralelo.

Los elementos de aplicación corresponden a funciones diferentes que son aprovechables por las diferentes aplicaciones. El servicio de transporte y las pilas de protocolos implementadas suelen ser también de tipo variado, dada la complejidad de interconexión de redes que forman los sistemas de comunicaciones actuales. Los sistemas distribuidos y las redes de área local tienen un carácter abierto, es necesario definir arquitecturas de gestión de red normalizadas, que permitan la gestión de elementos heterogéneos de múltiples proveedores.

Los estándares más extendidos son los siguientes:

- Modelo de Gestión OSI: Arquitectura definida por ISO, utiliza CMIS/CMIP (Common Mangement Informatión Service/Common Management Information Protocols). Constituye un estándar concebido para operar sobre protocolos OSI. Si bien, puede operar a nivel de aplicación sobre otros protocolos. Sea el caso de TCP/IP, para cuyo caso se denomina CMOT (Common management Over TCP/IP).

- Arquitectura TMN: Definida por la ITU-T, se basa en el modelo anterior e incluye el acceso a los recursos de telecomunicación.

- Modelo de Gestión Internet: Utiliza SNMP (Simple Network Management Protocol), estándar de facto que opera sobre el protocolo TCP/IP.

El modelo de gestión OSI e Internet se refieren a redes de computadores, mientras que el TMN es de utilidad para los grandes operadores de redes de telecomunicaciones. Señalar que existen otras arquitecturas propietarias que son interesantes y sólidas, como es el caso de SNA de IBM, UNMA de AT&T, etc. Dedicamos el resto del capítulo al estudio de los principales modelos de gestión actuales: OSI, TMN e Internet.

2.6 El Modelo de Gestión OSI.

El Modelo de gestión OSI (*Open System Interconection*) provee a los fabricantes de un conjunto de estándares que aseguran una mayor compatibilidad e interoperabilidad entre los distintos tipos de tecnología de red utilizados por las empresas a nivel mundial. Proporciona una estructura de red organizada, para conseguir la interconexión de los diversos tipos de Sistemas de Operación y equipos de telecomunicación usando una arquitectura estándar e interfaces normalizadas.

Define una *arquitectura física:* estructura y entidades de la red; un *modelo funcional:* servicios, componentes y funciones de gestión; *modelo de información:* definición de recursos gestionados y un *modelo organizativo:* niveles de gestión, Figura 2.5.

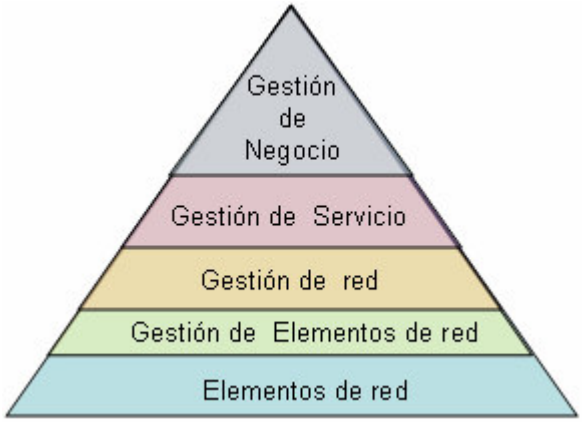

Figura 2.5. Niveles de Gestión según OSI

2.6.1 Conceptos de Gestión de Sistemas OSI.

En este apartado se presentan los conceptos principales, relacionados con el modelo de gestión de redes OSI.

2.6.1.1 Marco de Gestión OSI.

La finalidad es tener una visión general del modelo de gestión OSI. Los conceptos de gestión encerrados en dicho modelo tienen plena vigencia y son la base para la comprensión de otras soluciones de gestión. Propone tres formas distintas de llevar a cabo la gestión de los entornos OSI, Figura 2.6:

1.- Gestión de Sistemas: Pretende llevar a cabo la gestión de toda la torre de comunicaciones OSI, de una manera específica por medio de protocolos del nivel de aplicación.

2.- Gestión de Nivel: Pensado para permitir el intercambio de la información de gestión entre elementos de red, que no implementan toda la torre de protocolos OSI. Ejemplo: puentes, repetidores, etc.

3.- Operación en Nivel: Tiene como objetivo el monitorizar y controlar la comunicación entre los distintos niveles de la torre OSI.

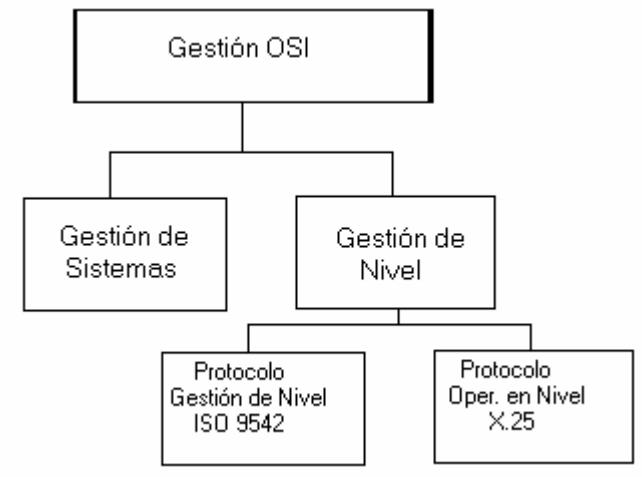

Figura 2.6 Enfoques de Gestión

Sin embargo, restricciones tales como que la gestión OSI solo se ocuparía de recursos con capacidad de comunicación y limitarla únicamente a entornos OSI, se decidió generar una nueva norma denominada visión general de la gestión de sistema, definida en el estándar [ISO92A], que ampliaba el ámbito de actuación y las funcionalidades de la gestión de sistemas, definida en el marco de gestión OSI. La gestión de sistemas se basa en el uso de protocolos del nivel de aplicación para el intercambio de información de gestión, según el paradigma gestor-agente. La entidad agente se encarga de aplicar operaciones actuando sobre los denominados objetos gestionados, que son abstracciones de los recursos controlados a través de los agentes.

2.6.2 Estructura de Gestión OSI.

Los agentes y los gestores, deben mantener una comunicación de la información de gestión, necesaria para que de esta forma el gestor conozca el estado de los distintos objetos que gestiona. Además este debe gestionar información del objeto e incluso información almacenada en él, para lo que debe de establecerse una asociación entre ambos que permita realizar el intercambio de información. Se va a disponer de un conjunto de servicios, reglas y protocolos para la comunicación entre gestores y agentes.

Cada recurso que se monitoriza y controla por el sistema de gestión se representa por un objeto gestionado, por ejemplo: conmutadores, estaciones de trabajo, etc. En el modelo OSI la complejidad de la gestión se traslada al agente. Las recomendaciones permiten hablar de una estructura general del modelo OSI, tal y como se aprecia en la siguiente figura:

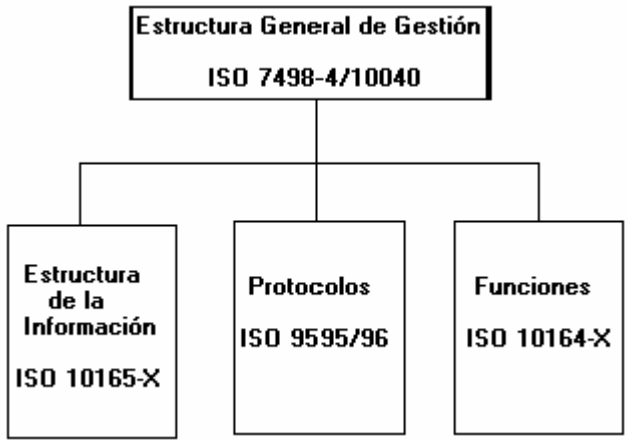

Figura 2.7 Estructura de Gestión OSI

Un sistema de gestión OSI se refiere a una colección de estándares para la gestión de red, que se fundamenta en una base de datos que contiene información relativa a los recursos y elementos de deben ser gestionados (MIB), un conjunto de servicios (CMIS) y un protocolo (CMIP), que permite la comunicación de información y operaciones entre la Base de datos y los gestores/agentes [Stalin96]. En los siguientes apartados procedemos al estudio de éstos.

2.6.3 Estructura de la Información de Gestión.

El fundamento de cualquier sistema de gestión de red, es una base de datos que contiene información acerca de los recursos y los elementos que van a ser gestionados. En la gestión de sistemas OSI, esa base de datos se denomina Base de Información de Gestión (*MIB - Management Information Base*) y la estructura general con la cual se puede definir y construir se llama Estructura de Gestión de Información (*SMI - Structure of Management Information).* La SMI identifica los tipos de datos que pueden ser usados en la MIB y la forma de representar y de nombrar los recursos.

La gestión de sistemas OSI reposa muy fuertemente en los conceptos de diseño orientado a objetos, en donde cada recurso monitorizado y controlado por los sistemas de gestión OSI se representa mediante un objeto gestionado. La MIB entonces, se constituye en una colección estructurada de tales objetos. Un objeto gestionado se define a partir de un recurso, ya sea hardware o software, que una organización desea monitorizar y/o controlar. Los recursos hardware pueden ser switches, estaciones de trabajo, PBXs, LANs, tarjetas de puertos o multiplexores, y los recursos software podrían ser programas de consulta, algoritmos de enrutamiento o rutinas de gestión de memoria.

La recomendación ISO 10165/CCITT X.720, presenta un modelo de información de gestión general para OSI. Su finalidad, según la recomendación, es *"proporcionar una estructura a la información de gestión transportada externamente por los protocolos de gestión de sistemas y modelar los aspectos de gestión de los recursos conexos (p.e. un router)".*

2.6.3.1 Diseño Orientado a Objetos de la Información de gestión.

Las especificaciones OSI de SMI y de MIB, como ya hemos visto están estrechamente relacionadas con los conceptos de diseño orientado a objetos. Esta aproximación permite adherir nuevas funciones y nuevas clases de objetos gestionados. La técnica orientada a objetos también provee extensibilidad a los servicios y protocolos relacionados, ya que la filosofía de este enfoque es que *"todos los aspectos de los sistemas del mundo real pueden describirse mediante un único modelo de objeto*".

Es importante anotar que las especificaciones no dictan que la implementación de las MIBs deba hacerse con tecnología orientada a objetos, su único requerimiento es que la especificación de información entre sistemas en los protocolos de gestión de sistemas (p.e. CMIP) use principios de diseño orientado a objetos. La siguiente tabla define algunos de los términos claves usados en la SMI y relacionados, la mayoría de ellos, con la terminología genérica orientada a objetos.

Término	*Término O.O.*	*Definición*
Objeto Gestionado	**Objeto**	Es la visión en la gestión OSI de un recurso que debe ser gestionado usando protocolos de gestión OSI. Ejemplos: una entidad de capas, una conexión, un equipo físico de comunicaciones.
Clase de objeto gestionado	**Clase de objeto**	Un conjunto de objetos gestionados que comparten los mismos nombres, conjuntos de atributos, notificaciones y operaciones de gestión (paquetes).
Atributo	**Variable**	Una propiedad de un objeto gestionado. Un atributo tiene un valor.
Operación	**Message**	Una operación sobre un objeto gestionado que efectúa algo sobre un sistema de gestión.
Comportamiento	**Método**	Una descripción de la forma en la cual los objetos gestionados adquieren nombres, atributos, notificaciones y ejecutan acciones con los recursos actuales.
Notificación	**Mensaje**	Información emitida por un objeto gestionado relacionada con un evento que ha ocurrido en dicho objeto.
Plantilla		Un formato estándar para la documentación de las definiciones de adquisiciones de nombres, clases de objetos gestionados y sus componentes, tales como paquetes, parámetros, atributos, grupos de atributos, definiciones de comportamiento, acciones o notificaciones.
Encapsulamiento	**Encapsulamiento**	Una relación cerrada entre un objeto gestionado y sus atributos, notificaciones, operaciones y comportamiento. Dicha relación asegura que el objeto pueda mantener su integridad.
Herencia	**Herencia**	El mecanismo conceptual mediante el cual los atributos, notificaciones, operaciones y comportamiento son adquiridos por una subclase desde su superclase.
Especialización	**Especialización**	La técnica por la cual se derivan nuevas clases de objetos gestionados desde una clase existente mediante la adición de nuevas capacidades (tales como nuevos atributos y notificaciones).
Inclusión	**Contenido**	Una relación estructurada para las instancias de los objetos gestionados en la cual la existencia de una instancia de un objeto depende de la existencia de un contenido de la instancia de un objeto gestionado.
Alomorfismo	**Polimorfismo**	La capacidad de un objeto gestionado de una clase dada para simular una o más clases de objetos diferentes.
Paquete	—	Una colección de atributos opcionales, notificaciones, operaciones y comportamiento que están todos presentes o todos ausentes en un objeto gestionado. La presencia o ausencia de un paquete depende de la capacidad del recurso en sí.

Tabla 2.1 - Terminología de la estructura de gestión OSI

2.6.3.1.1 Objetos gestionados.

Un objeto gestionado es la abstracción de un recurso que representa sus propiedades vistas por la gestión y que están sujetas a ésta, tal como una entidad de capa, una conexión o un elemento de equipo de comunicaciones físicas. Una parte esencial de la definición de un objeto gestionado es la relación que existe entre las propiedades del recurso y el comportamiento del mismo; esta relación no se modela de una manera general.

Un objeto gestionado se define en términos de los atributos que posee, las operaciones que se pueden realizar sobre él, las notificaciones que puede emitir, y las relaciones que tenga con otros objetos gestionados, Figura 2.8.

Figura 2.8 Un objeto gestionado

Los objetos gestionados pueden ser específicos de una capa determinada, en cuyo caso se denominan objetos gestionados de capa (N). Los objetos gestionados que son pertinentes a más de una capa, a una función específica de gestión de sistemas (objeto de soporte de gestión) o al sistema en su totalidad se denominan objetos gestionados de sistemas. En la figura 2.9, se esquematizan los objetos gestionados como abstracciones de recursos físicos.

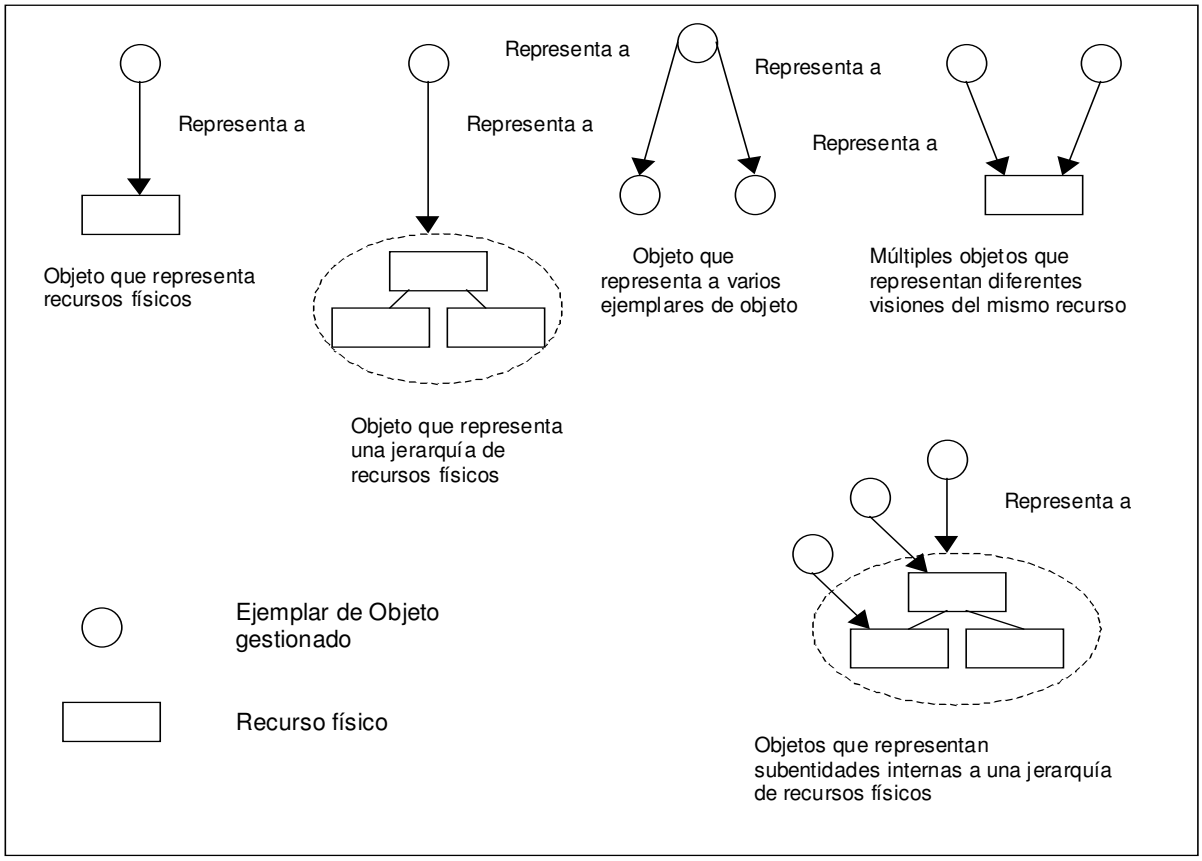

Figura 2.9 Ilustración de Objetos Gestionados y recursos físicos

- Un objeto gestionado representa un recurso. El mecanismo mediante el cual se mantiene la relación entre el objeto gestionado y el recurso físico o lógico, está fuera del alcance de los estándares de gestión OSI.

- Un único objeto gestionado puede representar uno o varios recursos de red.

- El mismo recurso de red puede ser representado por uno o por más objetos gestionados, cada uno de los cuales representa un aspecto del recurso en particular.

- No todos los recursos necesitan ser representados por un objeto gestionado. Eso no significa que dicho recurso no exista, solo que no está disponible para la gestión de sistemas OSI.

- Algunos objetos gestionados están definidos únicamente por el soporte de funciones de gestión y no representan ningún recurso. Ejemplo de ellos son los registros de eventos (*event-log*) y los filtros.

2.6.3.1.2 Clases de Objetos Gestionados.

Según la definición de la estructura de una MIB, cada objeto gestionado es una instancia de una clase de objetos gestionados. Una clase de objetos gestionados es un modelo o plantilla (*template*) de instancias de objetos gestionados que comparten los mismos atributos, notificaciones y operaciones de gestión. Todas las instancias de objetos que comparten los mismos elementos son miembros de la misma clase. Las instancias de objetos individuales pueden diferir en los valores de sus atributos.

El concepto clase también es una facilidad tipo macro que permite a un tipo general de objeto ser definido solo una vez y de esta forma, reutilizar varias veces la definición en cada instancia del tipo de objeto. Según la definición de "*clase de objeto gestionado*", los objetos de una misma clase se caracterizan por:

- Los atributos (datos) que tienen disponibles.
- Las operaciones de gestión que les pueden ser aplicadas.
- El comportamiento exhibido en respuesta a las operaciones de gestión.
- Las notificaciones que puede emitir.
- Las operaciones de gestión (paquetes) que pueden ser encapsuladas en él.
- Su posición en el árbol jerárquico.

Todas estas propiedades están expresadas en las Líneas Guía para la Definición de Objetos Gestionados (recomendación X.722, GDMO), estudiada en el capítulo 4 de la tesis.

2.6.3.1.3 Atributos.

Los elementos de datos actuales contenidos en un objeto gestionado se llaman atributos. Cada atributo constituye una propiedad del recurso representado por el objeto, por ejemplo las características operacionales, el estado actual o las condiciones de operación. La mayoría de las veces los atributos son usados para la monitorización, en donde el valor del atributo refleja el estado del recurso en cuestión. Un atributo también puede ser usado para controlar un recurso, esto se hace mediante la fijación de un valor en un atributo que cause un cambio en el comportamiento o estado de dicho recurso.

Los tipos de datos de un atributo pueden ser enteros, reales, cadena de caracteres o de algún tipo compuesto a partir de los tipos básicos. Además, cada atributo tiene reglas de acceso (lectura, escritura, lectura-escritura) y también reglas por medio de las cuales pueden ser localizados como resultado de una búsqueda (reglas de comparación - *matching rules*).

Un atributo puede ser una variable escalar simple, por ejemplo las operaciones *read* (get) y *write* (set, replace) pueden tener atributos escalares. Por otro lado, un atributo puede contener varios valores, como se define en la construcción SET-OF del ASN.1, *Abstract Syntax Notation One* (Anexo B).

En una clase de objeto, algunos de los atributos definidos pueden ser agrupados para formar un "grupo de atributos". El grupo simplemente es una conveniencia que permite desempeñar la misma operación sobre todos sus miembros con una única orden, sin embargo, dicha orden no está especificada en las recomendaciones.

Los atributos definidos para formar parte de paquetes obligatorios están presentes en todos los ejemplares de clase de objeto gestionado, mientras que los que están definidos para formar parte de los que son condicionales, se encuentran en los ejemplares que satisfacen las condiciones asociadas al lote.

2.6.3.1.3 Grupos de atributos.

Proporcionan el medio para referirse a una colección de características dentro de un objeto gestionado. Una clase de objeto gestionado puede tener más de un grupo de atributos, se definen dos tipos de grupos de atributos:

- *Grupo de atributos fijo,* es aquel cuyo conjunto de características se establece como parte de la definición de grupo de atributos inicial y no puede cambiarse de ninguna manera.

- *Grupo de atributos extensible,* es aquel al que pueden añadirse características como resultado de la especialización.

2.6.3.1.4 Comportamiento.

Un objeto gestionado exhibe ciertas características de comportamiento, incluyen las reacciones a las operaciones de control realizadas sobre él, así como las limitaciones de operación y administración a las que está sujeto. El comportamiento de un objeto gestionado ocurre como respuesta a un estímulo externo o interno. El estímulo externo pueden ser las operaciones de gestión de sistemas, liberadas en forma de mensajes CMIP. El estímulo interno, puede venir de eventos internos realizados con el objeto gestionado y su recurso asociado, tales como temporizadores.

Como se ha mencionado antes, el comportamiento forma parte de la definición de una clase de objeto gestionado y todas las instancias de objeto que pertenezcan a una misma clase, tienen el mismo comportamiento. Conforme la recomendación X.720 del CCITT (anexo A), el comportamiento puede definir:

- La semántica de los atributos, operaciones y notificaciones.
- La respuesta a las operaciones de gestión que se invocan en el objeto gestionado.
- Las circunstancias en las cuales se emitirán notificaciones.
- La dependencia entre valores de atributos particulares.
- Los efectos de relación sobre el objeto gestionado que participa.

2.6.3.1.5 Plantilla.

Formato normalizado para la documentación de enlace de nombres, definiciones de clases de objeto gestionado y sus componentes: parámetros, atributos, grupos de atributos, definiciones de comportamiento, acciones o notificaciones

2.6.3.1.6 Encapsulamiento.

Característica fundamental de un sistema orientado a objetos, que proporciona el acceso a los datos (atributos) por medio de métodos y oculta al usuario la compleja implementación interna del objeto.

En el contexto de gestión de red tiene el siguiente significado: cada tipo de recurso a ser gestionado en el sistema se representa mediante una clase de objeto gestionado. Una instancia específica de ese recurso se representa mediante una instancia de objeto gestionado. La gestión de datos (atributos) relacionada a ese recurso y los procedimientos (operaciones) de gestión aplicable a dicho recurso, se empaquetan juntos (encapsulamiento) en el objeto correspondiente.

Las aplicaciones de gestión tienen acceso al recurso, para control y monitorización, solo por medio del objeto. Además, todas las operaciones sobre el objeto gestionado son realizadas por medio del envío de mensajes hacia él. De esta forma, los datos y procedimientos actuales encapsulados en el objeto se encuentran protegidos del mundo externo. La definición de la clase de objeto gestionado especifica las operaciones que deben realizarse y las limitaciones de coherencia necesarias.

2.6.3.1.7 Especialización.

Una clase de objeto gestionado se especializa a partir de otra clase de estos objetos, definiéndola como una extensión de la otra clase. En el contexto de gestión OSI, la especialización se lleva a cabo mediante la extensión de las características de una clase de objeto en una o más de las siguientes formas:

- La adición de nuevos atributos.
- La extensión o restricción del rango de un atributo existente.
- La adición de nuevas operaciones y notificaciones.
- La adición de argumentos a las operaciones y notificaciones existentes.
- La extensión o restricción de los rangos de los argumentos a las operaciones y notificaciones.

A diferencia de un esquema orientado a objetos de propósito general, la gestión de sistemas OSI no permite la definición de una subclase mediante la eliminación de alguna de las características de su superclase.

Una clase de objeto gestionado que se especializa a partir de otra clase de dicho objeto recibe el nombre de ***subclase*** de esa clase (su ***superclase***). Como última superclase en la jerarquía de clases se designa una clase de objeto gestionado que recibe el nombre de *top*, éste es una clase de objeto gestionado que no es posible ejemplificar. La subclase hereda las operaciones, atributos, notificaciones, lotes y comportamientos de la superclase. En la figura 2.10, la clase *tokenBus* sería una clase donde se han añadido nuevos atributos referentes a un tipo de red local en especial, por lo tanto diremos que es una clase especializada de la clase *lanNet*.

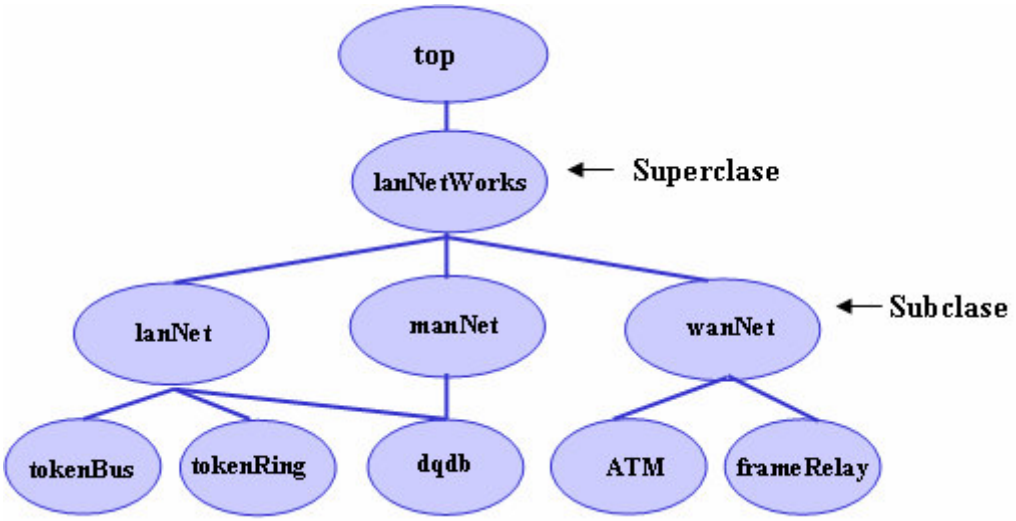

Figura 2.10 - Ejemplo de herencia.

2.6.3.1.8 Herencia.

La construcción de una clase permite la definición de nuevas clases de objetos en términos de las clases existentes, a una nueva clase de objeto se le llama subclase de la clase desde la que fue definida. El uso del concepto subclase tiene dos ventajas significativas:

1. Permite el desarrollo de una jerarquía de clase, con una subclase que puede tener a vez sus propias subclases. En casi todos los casos se forma una estructura espejo de la estructura actual de los recursos.

2. La subclase retiene características de su superclase, este concepto se denomina ***herencia***. Eso minimiza la necesidad de especificar características para objetos individuales.

Como una facilidad opcional, se permite la *herencia múltiple*. Esto significa que una subclase es especializada desde más de una superclase y hereda las operaciones, atributos, notificaciones, paquetes, y el comportamiento de cada superclase. Aunque eso permite una mayor posibilidad de reutilizar la definición de clases, es una técnica de diseño complicada de usar.

Al final, todas las clases de objetos se derivan de una única clase de objeto denominada *top*. Esa es la última superclase, y las otras clases de objetos forman una jerarquía de herencia o árbol jerárquico con el *top* como la raíz. La Figura 2.10 es un ejemplo de una porción de uno de estos árboles. La clase de objeto *lanNetWork* es padre de otra clase de objeto *wanNet*, la primera es conocida como superclase, a la clase hija se le conoce como subclase.

2.6.3.1.9 Alomorfismo.

El alomorfismo, tal como lo afirma la recomendación X.720: "*es la posibilidad que tiene un objeto gestionado, instancia de una clase determinada, de ser gestionado como miembro de una o más clases de objeto gestionado...*", proporcionando así un método para asegurar la interoperabilidad. El alomorfismo es en esencia un caso especial del concepto polimorfismo en la técnica de orientación a objetos. El polimorfismo provee la capacidad de ocultar implementaciones diferentes detrás de una interfase común. Por ejemplo el definir un único método de impresión *print* para cada clase de documento en un sistema, permite que cualquier documento sea impreso mediante el envío del mensaje *print*, sin importarle como este método llevó a cabo la tarea en dicho documento.

Del mismo modo, el alomorfismo provee la capacidad de implantar diferentes objetos que presentan la misma interfase en las estaciones de gestión. Específicamente, el alomorfismo se refiere a la capacidad para que una instancia de una subclase (llamada una subclase alomórfica) asemeje el comportamiento de, o emule, su superclase (llamada una superclase alomórfica) como se observa en los protocolos de gestión de sistemas (p.e. CMIP). Un uso típico del alomorfismo sería soportar la evolución de la MIB. Un nuevo objeto podría ser definido para emular el comportamiento de uno viejo, así que una estación de gestión podría manejar todavía el objeto de la misma forma.

El concepto alomórfico se define usando la convención subclase/superclase, pero con lineamientos de como la subclase se puede derivar de la superclase. Por ejemplo, el rango de valores de un atributo heredado en la subclase debe ser un subconjunto del rango del superconjunto. De este modo, cualquier valor aceptable en la subclase, también lo es en la superclase. La relación alomórfica se indica mediante la inclusión de un atributo de múltiples valores en la subclase, que lista todas las superclases que imita ese objeto. Por tanto, se dice que la subclase es "alomórfica a" la superclase.

2.6.3.1.10 Paquetes.

Un paquete o lote (*package*) es un conjunto de características que constituyen un módulo integral de una definición de clase de objeto gestionado. Los paquetes se especifican como obligatorios o condicionales. Un paquete ***obligatorio*** es aquel que debe estar presente en todos los ejemplares de una clase de objeto gestionado determinada. Un paquete ***condicional*** es una colección de atributos, notificaciones, operaciones y comportamientos opcionales que están todos presentes o todos ausentes en un objeto gestionado. La presencia o ausencia de un paquete la determina la capacidad del recurso que esta siendo modelado por el objeto. Un ejemplo de esto es el conjunto de opciones de una máquina con el protocolo X.25.

El paquete condicional es una conveniencia organizacional y no una construcción independiente del objeto gestionado que lo contiene. En particular, las operaciones y atributos que forman parte de un paquete se crean en el objeto gestionado y no mediante una referencia al paquete en sí. Cuando se crea una subclase, el paquete entero de atributos, notificaciones y demás es heredado por la subclase.

2.6.3.2 Operaciones.

Son las acciones de gestión de sistemas que se aplican a los atributos de un objeto o al objeto gestionado como un todo. La definición del modelo de información de gestión, incluye una especificación de las actividades que pueden ser desempeñadas sobre los objetos. Esas operaciones se realizan mediante una entidad de gestión, por medio de un mensaje que es enviado al objeto, empleando un protocolo de gestión de red.

Las operaciones de gestión se clasifican en dos categorías: aquellas que se aplican a los atributos de un objeto y aquellas que se aplican al objeto como un todo. Una operación sólo puede prosperar si el sistema de gestión invocador tiene los derechos de acceso necesarios para realizar dicha operación y no se violan las constricciones de coherencia (p.e. las relaciones que deben mantenerse entre valores de atributos).

Las operaciones pueden ser *confirmadas*, cuando requieren del envío de una respuesta al invocador de dicha operación indicando éxito o fracaso de la misma, o *no confirmadas*, cuando el invocador no requiere de respuesta alguna.

2.6.3.2.1 Operaciones orientadas a atributos.

Las siguientes operaciones pueden ser enviadas a un objeto para ser aplicadas a uno o varios atributos:

- Tomar (*get*) el valor del atributo. Esta operación se confirma siempre.
- Reemplazar (*replace*) el valor del atributo. Puede ser confirmada.
- Fijar (*set*) el valor del atributo a su valor por defecto. Puede ser confirmada.
- Añadir (*add*) miembros a un atributo de múltiples valores. Puede ser confirmada.
- Eliminar (*remove*) miembros de un atributo multivaluado. Puede ser confirmada.

Cualquier operación puede solicitar que la misma función sea realizada sobre una lista de atributos. Por ejemplo, una operación para reemplazar los valores de los atributos, podría especificar una lista de atributos con el nuevo valor para cada uno de ellos. Una operación puede especificar que las operaciones individuales sean realizadas atómicamente; esto es, que prosperen todas las operaciones, o si esto no es posible, que no se efectúe ninguna. Esta forma de comportamiento es propia de SNMP.

2.6.3.2.2 Operaciones orientadas a objetos.

Las siguientes operaciones pueden ser aplicadas a un objeto gestionado como un todo:

- Creación (*create*): Crea e inicializa un objeto gestionado, es siempre confirmada.
- Supresión (*delete*): Borra un objeto gestionado, es siempre confirmada.
- Acción (*action*): El objeto gestionado realiza la acción especificada e indica el resultado.

Puede ser de tipo confirmado o no. La semántica de esas operaciones es parte de la definición de clases de objetos. En particular, el efecto de esas operaciones sobre otros objetos gestionados relacionados (p.e. objetos superiores o subordinados) se debe especificar.

Todas las operaciones anteriores, tanto para atributos como para objetos (a excepción de *create*), pueden ser aplicadas sobre varios objetos, o atributos del mismo objeto al mismo tiempo. Existen tres nuevos conceptos que se asocian a esta forma de trabajo: el alcance, el filtrado y la sincronización. Procedamos a su explicación.

2.6.3.2.3 Alcance (Scoping).

Identifica a uno o varios objetos sobre los que se efectúa una misma operación. El alcance se refiere a una instancia de objeto gestionado específico como un *"objeto gestionado base",* y sirve como punto de partida para la selección de uno o más objetos a los cuales se aplica la operación. Recordando que los objetos gestionados forman una estructura en árbol, si tomamos el objeto gestionado base como la raíz, obtendríamos un subárbol de todos los objetos subordinados a este.

En la recomendación X.710 del CCITT el objeto gestionado base se define como "*la raíz del subárbol del árbol de información de gestión a partir del cual ha de comenzarse la búsqueda*". Teniendo en cuenta esto, se definen cuatro especificaciones del nivel de alcance:

- Solo el objeto base
- El objeto base y todos sus subordinados hasta el nivel enésimo.
- El objeto base y todos sus subordinados, esto es, el subárbol entero.
- El enésimo nivel subordinado del objeto base, con el objeto base definido como el nivel 0.

2.6.3.2.4 Filtrado.

Un filtro es una expresión boleana consistente en una o más aserciones, acerca de la presencia de los valores de los atributos en un objeto gestionado alcanzado. Si el filtro contiene más de una aserción, las aserciones se agrupan utilizando operadores lógicos.

Las aserciones de los valores de los atributos, pueden requerir la presencia de las siguientes reglas de comparación:

- *Igualdad:* El valor del atributo es igual al afirmado.

- *Mayor o igual:* El valor afirmado es mayor o igual al valor del atributo.

- *Menor o igual:* El valor afirmado es mayor o igual al valor del atributo.

- *Presente*: El atributo está presente.

- *Subcadenas:* El valor del atributo incluye la subcadena especificada en el orden dado.

- *Subconjunto de:* Todos los miembros afirmados están presentes en el atributo.

- *Superconjunto de:* Todos los miembros del atributo estarán presentes en el atributo afirmado.

- *Intersección no nula:* Al menos uno de los miembros afirmados, estará presente en el atributo.

2.6.3.2.5 Sincronización.

El parámetro alcance puede funcionar en la selección de varios objetos que van a ser filtrados. A la vez, si más de un objeto ha sido alcanzado, el parámetro filtrado funciona en la selección de más de un objeto a los cuales se les va a realizar una operación. Surge entonces el interrogante de con qué orden van a ser procesados los objetos. Si el orden con el cuál las instancias de objetos son seleccionadas por el filtro no está especificado, sino que corresponde a una decisión local, ese orden no puede ser usado.

Sin embargo, CMIS incluye el concepto de sincronización. El usuario del servicio CMISE requiere uno o dos tipos de sincronización:

- *Atómica:* Todos los objetos gestionados seleccionados por la operación son revisados para comprobar si son capaces de realizarla en forma exitosa. Si uno o más de ellos no son capaces, entonces no se realiza la operación.

- *Mejor esfuerzo:* Cuando todos los objetos gestionados seleccionados por la operación requieren desempeñarla.

2.6.3.2.6 Notificaciones.

Los objetos gestionados emiten notificaciones cuando se detecta la ocurrencia de un acontecimiento interno o externo que lo afecta, a petición de un agente externo o bien de forma espontánea. Las notificaciones pueden ser emitidas externamente en un protocolo o pueden ser almacenadas en un registro. Los sistemas de gestión pueden solicitar que alguna o todas las notificaciones emitidas por un objeto gestionado le sean enviadas. Las

notificaciones que son enviadas a un gestor o a un log se encuentran almacenadas en un repositorio de eventos.

2.6.3.3 Árbol de Inclusión.

Estos principios están contenidos en el numero 5 de la recomendación X.720 y en la norma equivalente ISO10165-1, (anexo A). Como hemos visto, la facilidad proporcionada por las subclases orientadas a objetos permite la creación de un árbol jerárquico, el cual refleja la relación entre diferentes tipos de objetos. Es importante decir que esa jerarquía simplemente representa una conveniencia para definir una variedad de tipos de objetos con la mínima cantidad de texto.

Un objeto gestionado puede contener otros objetos gestionados de su misma clase o de clases distintas. Esa relación se denomina ***Inclusión*** y consiste en una relación entre los objetos gestionados y no entre las clases. Un objeto gestionado es contenido en un, y solo en un objeto gestionado continente. A su vez, los objetos gestionados continentes pueden estar contenidos en otros objetos gestionados. De esta forma, la condición obliga a la MIB a tener una estructura en árbol. Todo esto se lleva a cabo mediante la inclusión en el objeto superior (continente) de una referencia al objeto inferior (contenido) en forma de atributo. La contenencia puede visualizarse como un gráfico dirigido en el que cada flecha apunta desde un objeto gestionado contenido hacia un objeto gestionado continente, denominado ***Árbol de Contenencia*** (ver Figura 2.11).

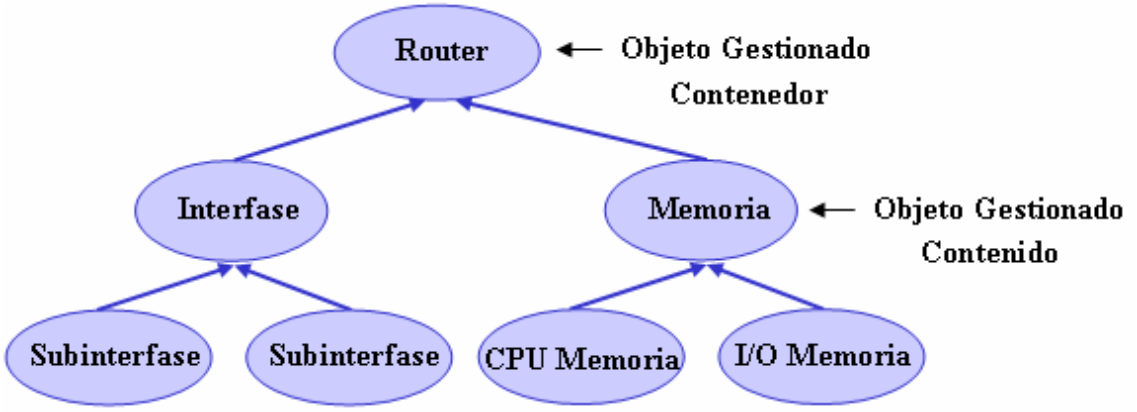

Figura 2.11 Árbol de Contenencia

Se puede apreciar que *Interfase* es contenido con relación al objeto gestionado *Router,* y este será un objeto continente. La relación de contenencia puede utilizarse para modelar jerarquías de piezas del mundo real (p.e. conjunto, subconjunto y componentes) o jerarquías organizacionales del mundo real (p.e. directorio, archivos, registros y campos).

2.6.3.4 Árbol de Nombres.

Para denominar objetos gestionados se utiliza la relación de contenencia. Los nombres se diseñan para que sean únicos en un contexto específico, el cual es determinado por el objeto continente. Los objetos determinados en términos de otro objeto se llaman **objetos subordinados**, y el objeto que establece dicha denominación se llama **objeto superior**.

Un objeto subordinado es denominado por la combinación de:

- El nombre de su objeto superior.

- Información que identifica unívocamente este objeto dentro del ámbito de su objeto superior.

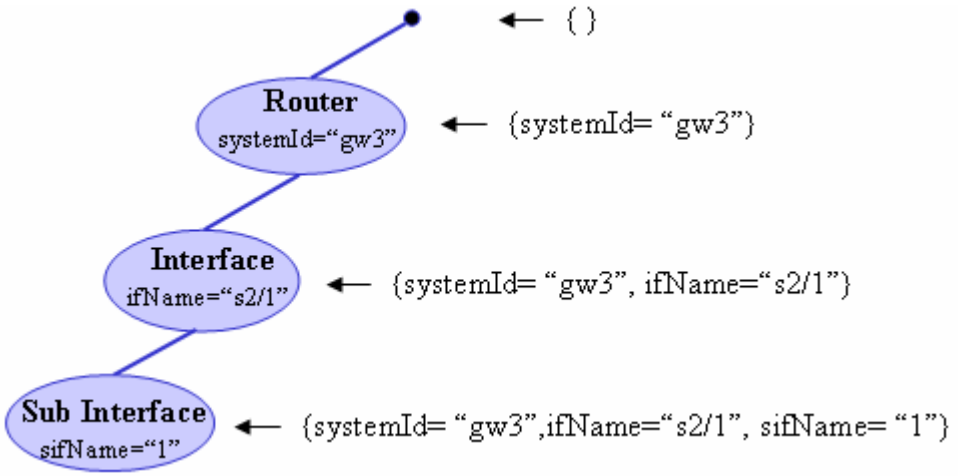

Figura 2.12 Árbol de Nombres

La relación de denominación puede visualizarse como un grafo direccionado, con cada flecha apuntando desde un objeto denominado hacia un contexto de denominación. El contexto de denominación puede estar calificado recursivamente por otro contexto de denominación, de tal forma que la estructura completa puede visualizarse como una jerarquía de una sola raíz. Esta jerarquía recibe el nombre de ***árbol de nombres o denominación***.

Así como se ha visto que hay distinciones entre la jerarquía de ***herencia***, que define la relación entre clases de objetos, y la jerarquía de ***contenencia***, que define la relación entre instancias de objetos, también hay distinción con el esquema dado por la ***denominación*** para las clases y para las instancias de objetos.

Primero consideremos las ***clases de objetos***. Cada clase de objeto registrada en el árbol de registro es identificada por un único identificador de objeto (*object identifier*). Cada identificador de objeto se puede considerar como una secuencia de enteros que navegan a través del árbol identificador de objeto hasta la clase de objeto gestionado. Tiene el mismo esquema usado por la MIB SNMP, a no ser porque en SNMP no hay una diferencia estructural, en cuanto a la denominación, entre clases de objetos e instancias de objetos. La Figura 2.13 muestra una parte del árbol identificador de objeto de ISO/CCITT. A manera de ejemplo, se describe el subárbol *management especification* (ms) con algunos de sus componentes.

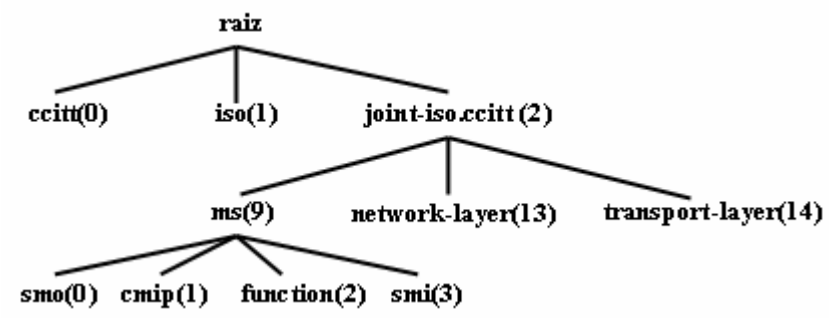

Figura 2.13 Parte del árbol de identificadores de objetos de ISO/CCITT

La denominación para las **instancias de objetos** es completamente distinta a la de las clases de objetos y se rige por la relación de contenencia. Dicho esquema trabaja así:

- Cada clase de objeto gestionado incluye un atributo que es usado en la denominación de instancias de ese objeto.

- El nombre distinguido relativo (RDN - *Relative Distinguished Name*) de una instancia de objeto corresponde a un valor específico del atributo de denominación. Este valor debe tener la propiedad de identificar inequívocamente un solo objeto gestionado en el ámbito de su objeto superior.

- El RDN de una instancia de objeto se construye con la secuencia de otros RDNs, desde la raíz del árbol de denominación hasta ese objeto.

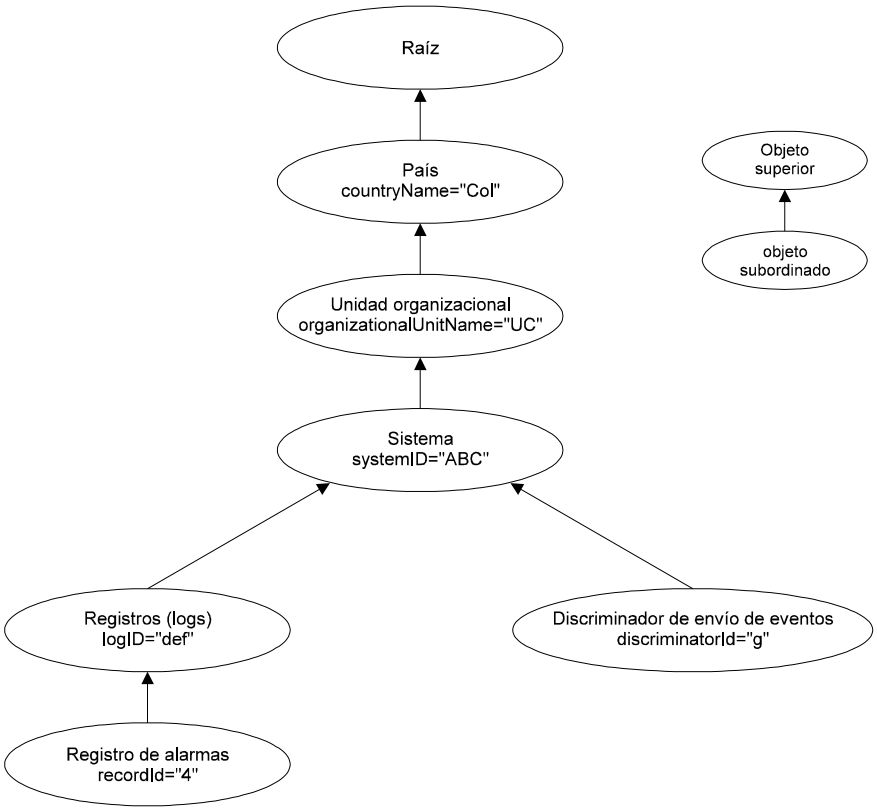

Figura 2.14 Ejemplo de árbol de contenencia

Figura 2.14 y la Tabla 2.2 muestra un ejemplo de un árbol de contenencia en donde se describen los respectivos RDNs, nombres globales y nombres locales de cada instancia.

RDN	Nombre local	Nombre global
countryName="COL"	no aplicable	{countryName="COL"}
organizationalUnitName="UC"	no aplicable	{countryName="COL", organizationalUnitName="UC"}
systemID="ABC"	{}	{countryName="COL", organizationalUnitName="UC", systemID="ABC"}
logID="def"	{logID="def"}	{countryName="COL", organizationalUnitName="UC", systemID="ABC", logID="def"}
recordId="4"	{logID="def", recordID="4"}	{countryName="COL", organizationalUnitName="UC", systemID="ABC", logID="def", recordID="4"}
discriminatorId="g"	{discriminatorId="g"}	{countryName="COL", organizationalUnitName="UC", systemID="ABC", discriminatorId="g"}

Tabla 2.2 - Ejemplo de nombres de las instancias de objetos

2.6.3.5 Resumen de Jerarquías de la Información de gestión en OSI.

En la gestión OSI se distinguen tres estructuras de árbol, figura 2.15:

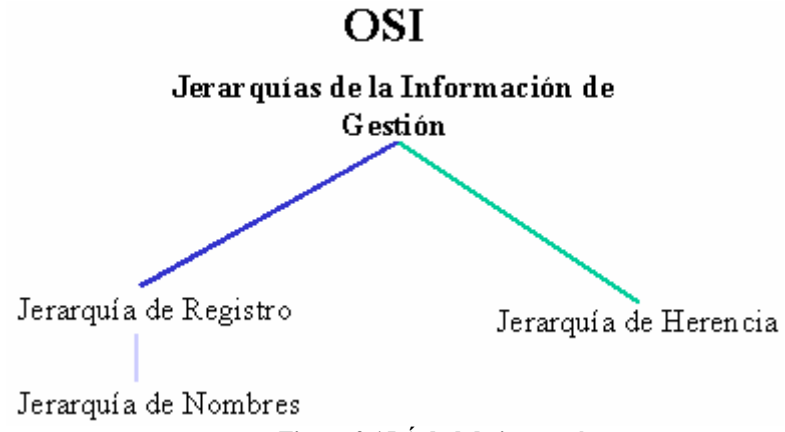

Figura 2.15 Árbol de jerarquías

1. Árbol de registro ISO: Es un árbol de denominación en donde se registran las clases de objetos gestionados, las definiciones de atributos, las acciones, las notificaciones y los paquetes.

Se puede pensar en el árbol de registros como en un diccionario o librería de "*stubs*" (pedazos) que se pueden pegar en la definición de una nueva clase de objeto gestionado. Este es un ejemplo de los beneficios del reutilización de que dispone las técnicas orientadas a objetos.

2. Árbol de herencia: Este árbol muestra cómo la definición de clases de objetos se deriva de otras, usando principios orientados a objetos. La herencia permite la reutilización de una estructura de clase de objeto, con refinamientos para definir una clase de objetos relacionada pero distinta.

3. Árbol de contenencia o nombres: Es la estructura MIB en sí. Muestra los objetos contenidos en un sistema gestionado y la jerarquía de contenencia entre aquellos objetos. Este árbol es usado no solo para definir la estructura MIB, sino para hacer una referencia no ambigua de las instancias de objetos.

Haciendo una comparación con SNMP, éste usará un único árbol: el árbol de registro. Dicho árbol, al igual que en CMIP, sirve para definir los nombres de los objetos; sin embargo, en SNMP todos los objetos incluidos deben ser escalares o tablas y no incluye atributos y notificaciones (como lo hace CMIP). Por lo tanto, los beneficios de la reutilización proporcionados por CMIP, no están disponibles en SNMP

2.6.4 Modelo de Comunicaciones.

La función fundamental en la gestión de sistemas OSI es el intercambio de información de gestión entre dos entidades (gestor y agente) por medio de un protocolo [Raman99]. Esa funcionalidad se denomina elemento de servicio genérico de información de gestión (*CMISE - Common Management Information Service Element*) y se concreta en dos partes:

- Un protocolo que especifica la Unidad de Datos del Protocolo (*PDU - Protocol Data Unit*) y los procedimientos asociados, el Protocolo Genérico de Información de Gestión (*CMIP - Common Management Information Protocol*).

- La interfase con el usuario que detalla los servicios que provee, el Servicio Genérico de Información de Gestión (*CMIS - Common Management Information Service*).

Como se discutirá más adelante, CMIS provee siete servicios para desempeñar operaciones de gestión en forma de primitivas de servicio. Además, los usuarios CMISE necesitan ser capaces de establecer asociaciones para realizar operaciones de gestión. Estos últimos servicios son suministrados por el elemento de servicio de control de asociación (*ACSE - association-control-service element*) a través de CMISE, allí CMIP no interviene. Para los servicios de operación de gestión, CMISE emplea a CMIP en el intercambio de unidades de datos de protocolo (PDUs). CMIP a su vez, se apoya en los servicios del elemento de servicio de operaciones remotas (*ROSE - remote-operation-service element*). Ambos, ACSE y ROSE se apoyan en los servicios de presentación.

2.6.4.1 Protocolo Genérico de Información de Gestión (CMIP).

El *Protocolo Común de Información de Gestión* (CMIP) define los procedimientos para la transmisión de información de gestión y la sintaxis de los servicios de gestión CMIS. CMIP se define en términos de *Unidades de Datos de Protocolo* (PDUs) que son intercambiadas entre un par de elementos de servicio de información de gestión (CMISEs), para llevar a cabo el servicio CMIS.

2.6.4.1.1 Características de CMIP.

Es un protocolo orientado a conexión, que aporta una gran fiabilidad a las comunicaciones entre elementos, pero por el contrario introduce una gran sobrecarga al sistema. Esto no es bueno, dado que los protocolos de gestión deben poder actuar en condiciones extremas de la red, y por tanto se debe de tratar de saturar lo menos posible el ancho de banda con el manejo de las conexiones. Además hay cierto tipo de operaciones de gestión que exigen una acción inmediata, cosa que no es posible si previamente hay que establecer una conexión.

La arquitectura de este protocolo fija en el nivel de aplicación una *Entidad de Aplicación para la Gestión de Sistemas* (SMAE), indica como será la configuración del nivel de aplicación para ser usado por una aplicación de gestión. Asimismo determinará todos los elementos necesarios para poder interactuar con otras entidades, Figura 2.16.

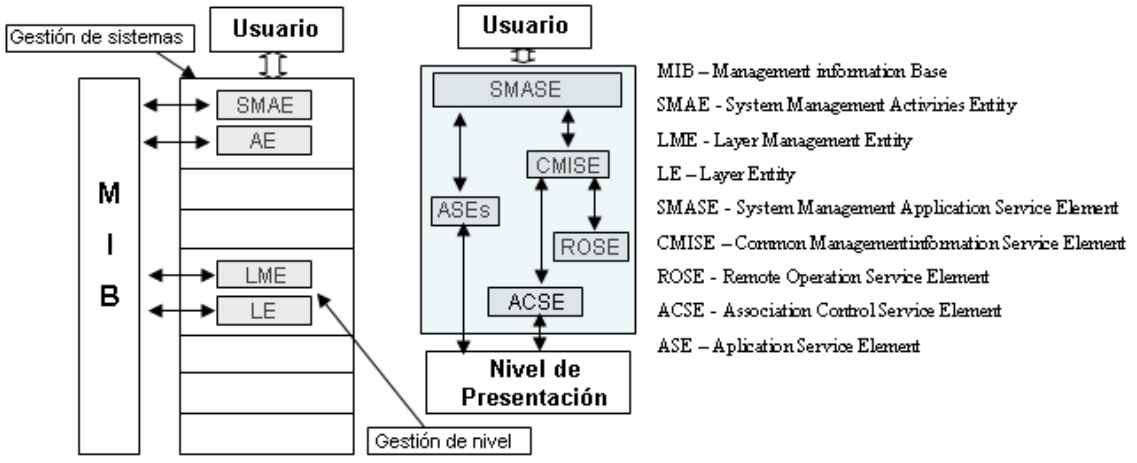

Figura 2.16 Gestión OSI en la capa de aplicación

Entre las características más importantes del protocolo CMIP se pueden destacar las siguientes:

- Requiere gran cantidad de memoria y capacidad de CPU.
- Grandes cabeceras en los mensajes del protocolo.
- Especificaciones difíciles de realizar y tediosas de implementar en aplicaciones.
- Comunicaciones con los agentes orientadas a conexión.
- Estructura de funcionamiento distribuida.
- Permite una jerarquía de sistema de operación.
- El protocolo asegura la llegada de los mensajes a su destino.

Por tratarse de una gestión conducida por eventos, tendremos:

- El agente notifica al gestor de sucesos, la información concerniente a los recursos gestionados.
- El agente es responsable de monitorizar los recursos.

Tiene como ventaja que existe menor gestión de tráfico y la desventaja que los agentes son muy complejos.

Existen tres tipos de servicios ofrecidos por CMIP:

- *Manejo de datos:* Usado por el gestor, para solicitar y alterar información de los recursos del agente.

- *Informe de Suceso:* Usado por el agente, para informar al gestor sobre diversos sucesos de interés.

- *Control Directo:* Usado por el gestor, para solicitar la ejecución de diversas acciones en el agente.

2.6.4.2 Servicio Genérico de Información de Gestión (CMISE).

El CMISE define los servicios proporcionados por la gestión de sistemas OSI (X.710). Esos servicios son invocables por medio de procesos de gestión, con el fin de poder proveer comunicaciones remotas. Los servicios CMIS se describen en términos de primitivas, que pueden ser vistas como comandos o llamadas de procedimientos con parámetros, se listan en la siguiente Tabla 2.3.

Los servicios son de dos tipos:

- *Confirmados,* que requieren que un proceso de gestión remota envíe una respuesta para indicar la recepción y el éxito o fallo de la operación solicitada.

- ***No** confirmados* que, como su nombre lo indica, no proveen respuestas.

(a) Servicio de Notificación de gestión		
Servicio	***Tipo***	***Definición***
M-EVENT-REPORT	confirmado / no confirmado	Reporta un evento acerca de un objeto gestionado a un usuario del servicio CMISE igual.
(b) Servicios de Operación de Gestión		
Servicio	***Tipo***	***Definición***
M-GET	Confirmado	Solicita la retribución de información de gestión desde un usuario del servicio CMISE igual.
M-SET	confirmado / no confirmado	Solicita la modificación de información de gestión por parte de un usuario del servicio CMISE igual.
M-ACTION	confirmado / no confirmado	Solicita a un CMISE igual que desempeñe una acción.
M-CREATE	Confirmado	Solicita a un CMISE igual que cree una instancia de objeto gestionado.
M-DELETE	Confirmado	Solicita a un CMISE igual que borre una instancia de un objeto gestionado.
M-CANCEL-GET	Confirmado	Solicita a un CMISE igual que cancele una solicitud previa y haga una invocación al servicio M-GET

Tabla 2.3 - Servicios CMISE

Existen tres categorías de servicios relevantes en CMIS:

1. Servicio de Asociación: Los usuarios de CMIS necesitan establecer una asociación de aplicación para comunicarse. Ellos se apoyan en el ACSE para el control de las asociaciones de aplicación, el servicio utilizado es A-ASSOCIATE.

2. Servicio de notificación de gestión: Destinado al transporte de la información de gestión aplicable a una notificación. La definición de ésta y el consecuente comportamiento de las entidades de comunicación, dependen de la especificación del objeto gestionado que la generó y está fuera del contexto de CMIS. La primitiva utilizada:

- M-EVENT-REPORT: Es la única primitiva emitida por el servicio de notificación de gestión y usada para reportar una notificación a un gestor que la solicitó. A diferencia de todas las demás operaciones de gestión, el envío de eventos se inicia en el proceso agente y el proceso gestor es el que responde. El servicio M-EVENT-REPORT puede ser confirmado o no.

3. Servicios de operación de gestión: Usado para el transporte de la información de gestión aplicable a las operaciones de administración del sistema. La definición de la operación y el consecuente comportamiento de las entidades de comunicación, dependen de la especificación del objeto gestionado, al cual se dirigían la operación, y está fuera del contexto de CMIS. Tal y como se observa en la tabla 2.3 existen seis primitivas asociadas:

- M-GET: Pide la obtención de información de gestión desde una capa de usuario CMISE.

- M-SET: Permite la modificación de datos en la MIB. Utilizado para cambiar el valor o los valores de uno o más atributos en uno o más objetos gestionados. Puede ser confirmada o no-confirmada. La mayoría de sus parámetros tienen el mismo significado que los del servicio M-GET.

• M-ACTION: Permite la invocación de un procedimiento predefinido especificado como parte de un objeto gestionado. La solicitud indica el tipo de la acción y los parámetros de entrada. Este servicio puede ser confirmado o no-confirmado.

• M-CREATE: Usada para crear una nueva instancia de una clase de objeto. Una instancia ya existente puede ser aprovechada como modelo para crear una nueva.

• M-DELETE: Borra uno o más objetos de la MIB

• M-CANCEL-GET: Detiene una operación GET muy larga. Sólo se puede cancelar la operación GET debido a que es la única operación que no altera la MIB. De este modo se puede asegurar su integridad, una vez cancelada la operación (con las otras operaciones no). Siempre es un servicio confirmado, puede contener los siguientes parámetros:

- *Invoke identifier:* Identifica unívocamente esa solicitud.
- *Get invoke identifier:* Identifica la solicitud M-GET a ser cancelada.
- *Errors:* Error, si la operación no se desarrolla en forma exitosa.

2.6.4.3 Facilidades de CMIS.

Hemos visto en apartados anteriores que pueden darse dos casos a la hora a trabajar con las primitivas (numeral 6 de la recomendación X.710):

1. Una operación confirmada puede generar múltiples respuestas mediante el uso de un parámetro de identificación de enlace (*linked-identification*).

2. Las operaciones se pueden realizar sobre múltiples objetos gestionados seleccionados, para satisfacer un mismo criterio y sujetos a una condición de sincronización.

CMIS provee un potente conjunto de herramientas que permiten a un usuario seleccionar uno o varios objetos para que estén sujetos a una operación de gestión.

2.6.4.3.1 Encadenamiento (linkage).

Las primitivas de servicio M-GET, M-SET, M-ACTION y M-DELETE pueden especificar operaciones sobre múltiples objetos. Cuando se realiza una solicitud de gestión a varios objetos, se retorna una respuesta por cada objeto y se necesitan algunas técnicas de encadenamiento para relacionar las múltiples respuestas con la solicitud inicial que las generó. Ese enlace es necesario una vez que allí puede haber un número considerable de solicitudes y las respuestas que llegan deben ser organizadas con una precedencia correcta.

El encadenamiento es provisto por medio de un parámetro identificador de enlace (link-identifier), el cual aparece en cada una de las respuestas y de las primitivas de confirmación. El valor del parámetro es el mismo que el invoke identifier que aparece en la solicitud y en las primitivas de indicación. El invoke identifier[1] es único para cada operación.

[1] Es necesario que exista un único identificado (presente en el PDU, ROSE) para sucesivas invocaciones a la misma asociación. Es suministrado por el servicio de usuario CMISE, esta identificación única permitirá la detección de pérdidas y duplicidad de respuestas.

2.6.4.3.2 Selección de objetos gestionados.

Las primitivas de servicio M-GET, M-SET, M-ACTION y M-DELETE, permiten a un usuario seleccionar uno o varios objetos mediante el uso de parámetros. La selección de objetos gestionados involucra los tres conceptos previamente estudiados: alcance, filtrado y sincronización.

- *Alcance (Scoping):* Identificación de uno o varios objetos, sobre los que se va ha aplicar el filtro.

- *Filtrado:* Discrimina la información que sale o entra en un objeto.

- *Sincronización:* El parámetro alcance puede funcionar en la selección de varios objetos que van a ser filtrados. Surge entonces el interrogante de con qué orden van a ser procesados los objetos.

2.6.4.4 ASES de Aplicación.

Las funciones de la capa de aplicación y las funciones de las capas inferiores son usadas por los *procesos de aplicación* (AP). Un AP combina todos los aspectos de procesamiento de la información y comunicación, se agrupan y dan un nombre simple para una referencia remota. Por ejemplo una recuperación de información de una base de datos, que se puede encontrar en otro sistema abierto, para lo cual hace falta incluir aspectos de establecimiento una asociación y comando de comunicaciones, etc.

Una AP puede usar una o más *entidades de aplicación* (AE) para representar aspectos de la comunicación. La capa de aplicación aparece como una colección de AEs, cada una de las cuales contiene información de la comunicación con otra AE par. Una AE utiliza los servicios de la capa de presentación para llevar a cabo la comunicación con otra AE. Una conexión lógica cooperativa entre dos AEs se denomina *Asociación de aplicación* (AA).

Las funciones de las AEs requeridas para el procesamiento cooperativo se encuentran separadas en *elementos de servicio de aplicación* (ASEs) y representan una colección de capacidades de comunicación empaquetados en módulos. Usando las capas inferiores una ASE puede comunicarse con otra ASE en la misma capa de aplicación o con otra ASE en otro sistema abierto y tiene su propia definición de servicios y especificaciones de protocolos. Existen ventajas para la clasificación de funciones en ASEs genéricos.

Elementos de Servicio de Administración de Sistemas, ASEs son usados por un SMAE. Un SMASE se usa para formar una MAPDU (unidades de datos del protocolo de la aplicación de administración). Estas MAPDUs y las APDUs del CMIP *pass through,* llevan información de una SMAE a otra SMAE. La comunicación de la información de administración entre SMAEs es llevada a cabo por el CMIP o posiblemente con otros servicios de comunicación provistos por ASEs tales como FTAM o TP. SMASE usa servicios CMIS. El frame se forma en CMISE, entonces transformados en el frame del protocolo CMIP en el CMIPM, la cual es una máquina de estado finito (FSM), figura 2.17.

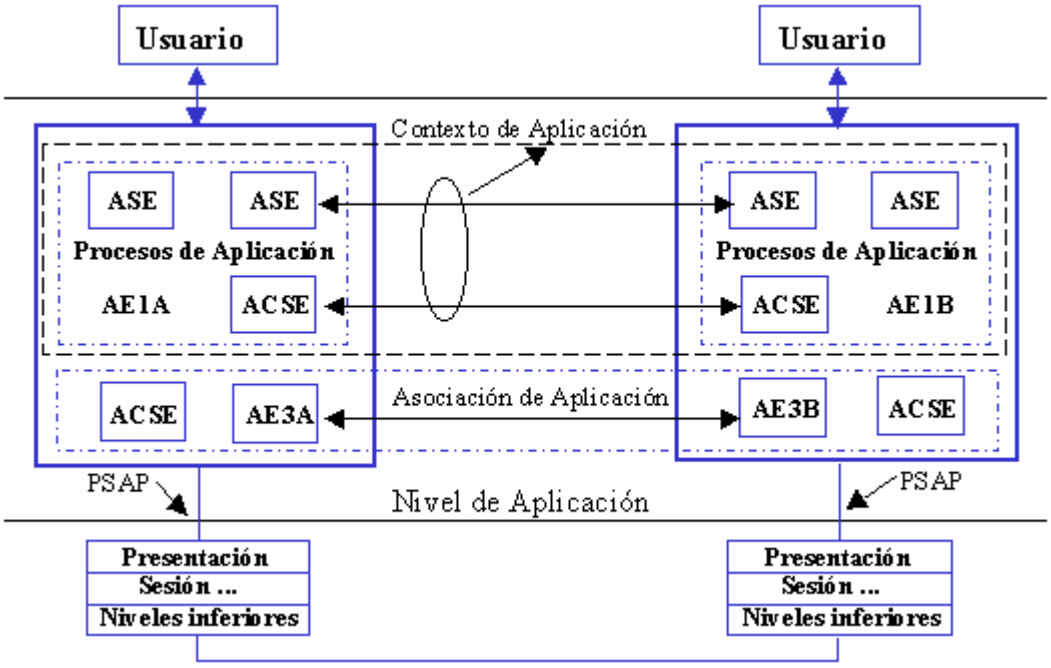

Figura 2.17 Componentes del nivel de aplicación

La APDU (*Aplicattion Protocol Data Unit*) CMIP tiene los encabezados apropiados de ROSE, forman el frame adecuado para transferir datos de administración vía los servicios de las capas subyacentes de la de presentación a otra SMAE. Inicialmente, para que la comunicación entre dos SMAEs sea llevada a cabo, debe formarse una asociación, y es una operación lo suficientemente compleja. Para que dos SMAEs lleven a cabo una asociación deberán soportar unidades funcionales similares. Una unidad funcional es un concepto abstracto usado para la combinación de opciones de servicios. Estas unidades funcionales asisten proveyendo servicios de administración de sistemas. Si las unidades funcionales de dos SMAEs son idénticas, no existe problema. Pero puede haber casos en que sean distintas. Por esta razón, durante la asociación, las características de la adjuntas necesitan ser negociadas, lo que se realiza con la ayuda de los servicios ACSE.

2.6.4.4.1 Association Control Service Element (ACSE).

Para que dos usuarios hagan uso de las operaciones de gestión CMIS, deben primero establecer una asociación de aplicación. Para este propósito, CMIS se apoya sobre el elemento de servicio de control de asociación (ACSE). El establecimiento de la asociación es la primera fase de cualquier manifestación de actividad del servicio de información de gestión. Además deberá disponer de una liberación de la conexión cuando un administrador o un agente no pueden comunicarse.

Actualmente ACSE desempeña dos funciones para los usuarios CMIS:

- Establece una asociación de aplicación para el intercambio de primitivas de servicio CMIS.

- En el tiempo del establecimiento de la asociación, los dos usuarios se ponen de acuerdo en cuales características de CMIS usará esa asociación. Esas características son usadas en términos de unidades funcionales.

Para llevar a cabo esta tarea se dispone de cuatro primitivas:

- A_ASSOCIATE: Establece la asociación entre aplicaciones.

- A-RELEASE: Libera ordenadamente la asociación entre dos aplicaciones.

- A-ABORT: Este servicio genera una liberación inmediata de la asociación, generada a petición del usuario. En este caso como en el anterior, la liberación de la asociación la puede realizar cualquiera de los dos usuarios CMIS.

- A-P-ABORT: Este servicio genera una liberación inmediata de la asociación, generada a petición de un proveedor de servicio ACSE. Puede deberse a un error interno en un ACSE o en la capa de presentación e inferiores, provoca el término de la asociación.

Las dos primeras primitivas A-ASSOCIATE y A_RELEASE, son usadas por otros ASEs para establecer o terminar una asociación con una entidad pareja. Ambos servicios son confirmados, por contra A-ABORT y A-P-ABORT son servicios no confirmados

Los servicios ACSE tienen dos modos de operación:

- *Modo Normal*: Usado en la administración OSI, los servicios ACSE usan los servicios de la capa de presentación y sesión, en la asociación se proporciona los parámetros de ambas capas. Además debe tenerse en cuenta restricciones de capa en los servicios, tales como la longitud.

- *Modo X.410-1984*: No es utilizado en la administración OSI. Este modo usa una capa de presentación nula y los servicios de la asociación no proveen los parámetros de los servicios de sesión y presentación.

2.6.4.4.2 Remote Operation Service Element (ROSE)

Elemento de Servicios de Operaciones Remotas, define un mecanismo común para la realización de operaciones remotas en aplicaciones distribuidas, comprende la invocación de operaciones, resultados y errores.

Una AE requiere una operación a ser realizada por otra AE remota. El éxito de la operación se reporta de vuelta. La operación efectuada es conocida como operación remota. La primera entidad se denomina invocadora y la segunda ejecutante.

Las distintas primitivas de ROSE, figura 2.18:

- RO-INVOKE: Utilizada para invocar una operación en el otro sistema final. No confirmado.

- RO-RESULT: Devuelve el resultado de la operación, no confirmado.

- RO-ERROR: Cuando la operación se interrumpe o falla, se devuelve un error.

- RO-REJECT: Se divide a su vez en dos tipos, RO-REJECT-U: cuando es el usuario el que tiene problemas con la invocación, resultado o error y RO-REJECT-P: cuando es el proveedor de servicio ROSE el que detecta el problema.

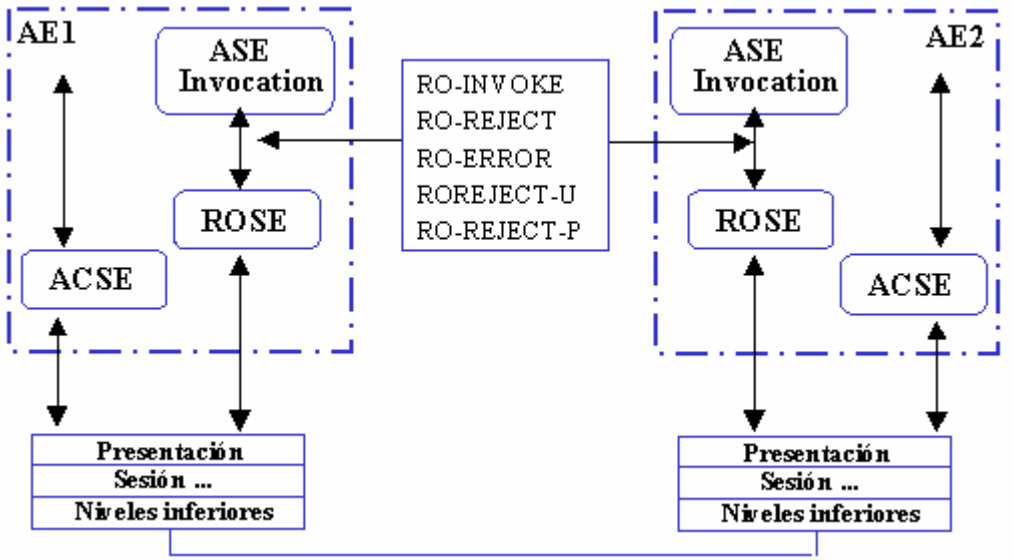

Figura 2.18 Primitivas ROSE

Los servicios son análogos a los RPC *(remote procedure call*, Llamada a Procedimientos Remotos). Primero la asociación se forma entre dos AEs usando A-ASSOCIATE. Después ROSE interactúa con los servicios de la capa de presentación. Las APDUs de ROSE (ROIV, RORS, ROER y RORJ) se forman en una *máquina de protocolo de operación remota* (ROPM). ROIV se usa para invocar una operación en otro sistema ejecutante. Si no hay problemas, los resultados de la operación se devuelven usando RORS, caso contrario se usa la APDU ROER para ejecutar un informe de errores. Si el invocador o el ejecutante no pueden procesar una APDU, entonces se usa RORJ.

2.6.4.5 Relación entre Primitivas.

Para finalizar, se muestra en la Figura 2.19 las relaciones existentes entre los distintos elementos de servicio de aplicación.

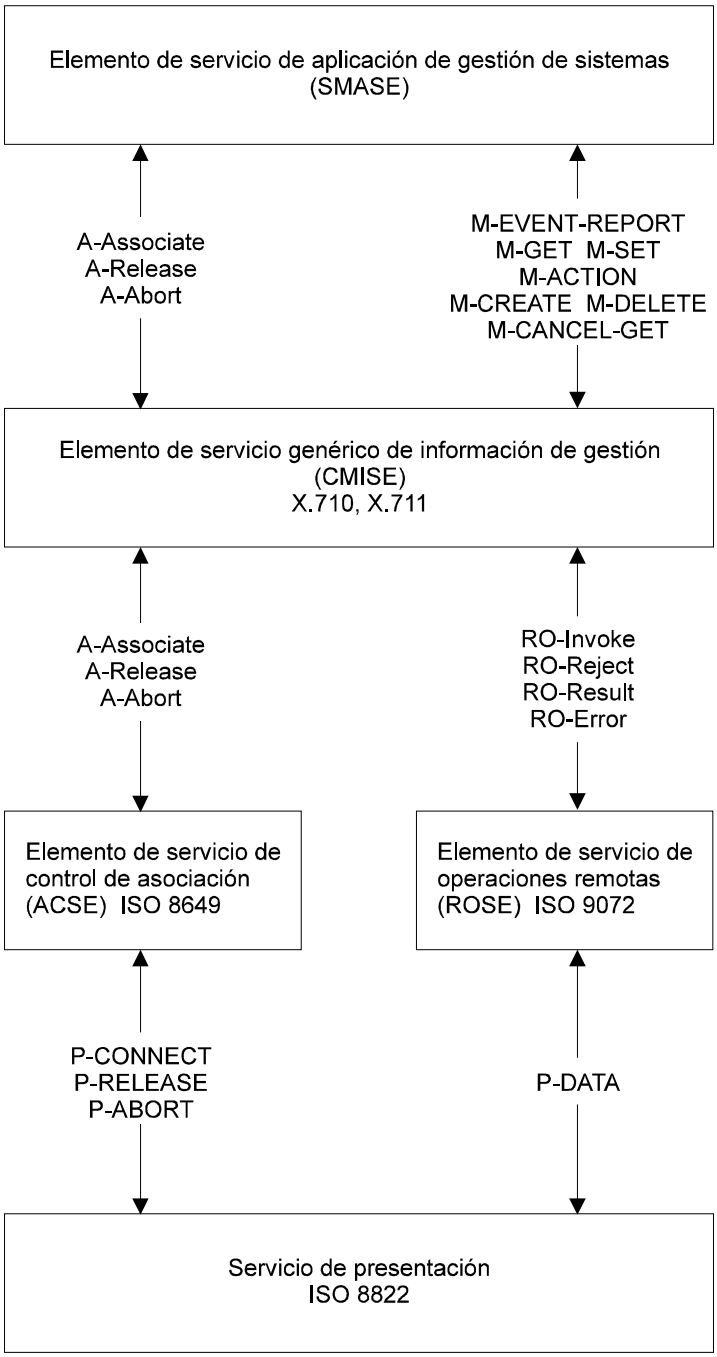

Figura 2.19 Servicios Provistos y usados por CMISE

2.6.5 Funciones de gestión

Define las funciones que proporcionan servicios básicos de gestión a las aplicaciones de gestión, ofrecen una funcionalidad más elaborada.

La recomendación X.700 (numeral 5.5) del CCITT, teniendo en cuenta los diversos aspectos que requieren gestión, divide la gestión de sistemas OSI en las cinco áreas funcionales (SMFAs). Fueron mencionados en la introducción del capítulo, por su interés procedemos a su estudio en mayor detalle.

2.6.5.1 Gestión de fallos.

Los fallos hacen que los sistemas abiertos dejen de satisfacer sus objetivos, por tanto la detección de problemas es una tarea necesaria, a fin de facilitar la labor a los administradores de la red. Esas facilidades incluyen mecanismos para la detección, aislamiento y corrección de operaciones anormales en cualquier componente de la red o en alguna de las capas OSI. La gestión de fallos suministra procedimientos para:

- Detectar y notificar la ocurrencia de fallos. Permiten a un sistema de gestión informar a su administrador la detección de un fallo, usando un protocolo estandarizado de emisión de eventos.

- Mantener un registro o "log" de los eventos emitidos. Este registro se puede examinar y procesar.

- Programar y ejecutar tests de diagnóstico.

- Investigar fallos.

- Iniciar acciones para corregir los fallos.

2.6.5.2 Gestión de Contabilidad.

Permite determinar y localizar cargos y costos por el uso de los recursos de la red. La gestión de contabilidad suministra procedimientos para:

- Informar a los usuarios sobre los costos incurridos mediante el uso de software para la emisión de eventos y manipulación de datos.

- Tolerar los establecimientos de límites de contabilidad y asociar calendarios de tarifas al uso de los recursos gestionados.

- Permitir la combinación de costos, cuando se involucren múltiples recursos de comunicación, para alcanzar un objetivo dado.

2.6.5.3 Gestión de configuración.

Facilita a los administradores ejercer control sobre la configuración de los componentes de red y las entidades de los niveles OSI. Cambiar la configuración podría servir para disminuir la congestión, aislar fallos o hacer cambios que necesiten los usuarios. La gestión de configuración suministra procedimientos para:

- Fijar y modificar los parámetros que controlan la operación rutinaria de los componentes de red y el software de las capas OSI.

- Asociar nombres con objetos y con conjuntos de objetos gestionados.

- Inicializar y cerrar objetos gestionados.

- Recolectar datos concernientes al estado actual de los recursos.

- Obtener anuncios de cambios significativos en la condición del sistema.

- Cambiar la configuración.

2.6.5.4 Gestión de prestaciones.

Facilita a los administradores de red la monitorización y evaluación de las prestaciones del sistema. La gestión de prestaciones pretende suministrar información estadística que permita establecer si un sistema tiene la capacidad de tráfico requerida, si su tiempo de respuesta está acorde con las necesidades, si hay una posible sobrecarga de los recursos o si el sistema está siendo usado en forma efectiva.

La gestión de prestaciones suministra procedimientos para:

- Recolectar datos interesantes del nivel de prestaciones actual de los recursos (información estadística).

- Mantener y examinar registros de prestaciones con el propósito de planear y analizar las condiciones de la red.

- Determinar el rendimiento del sistema en condiciones naturales y artificiales.

- Cambiar los modos de operación del sistema con el fin de realizar las tres actividades anteriores.

2.6.5.5 Gestión de seguridad.

Facilita a los administradores de red gestionar aquellos servicios que proveen protección en el acceso a los recursos de comunicación. La gestión de seguridad provee soporte para la gestión de:

- Facilidades de autorización.
- Control de acceso.
- Encriptación y gestión de claves.
- Autenticación.
- Logs de seguridad.

2.6.5.6 Funciones de Gestión de Sistemas.

Las áreas funcionales definidas en la estructura de gestión OSI describen áreas de responsabilidad muy amplias, por lo tanto éstas no han sido estandarizadas como tales, sino como el conjunto de un número específico de funciones que se denominan *funciones de gestión de sistemas (SMFs)*. Cada SMF estándar soporta los requerimientos de una o más de las cinco áreas funcionales (SMFAs); por ejemplo, la función de gestión "reporte de eventos" (*event-report*) se debe aplicar a todas las SMFAs. Por otro lado, cualquier SMFA requiere de varias SMFs.

Cada uno de los SMFs estándares, definen la funcionalidad del SMF y provee la conexión entre los servicios prestados por él y los prestados por CMISE. Dichas relaciones se encuentran representadas en la figura 2.20. De esta manera, cada una de las cinco áreas funcionales hace uso de una o más de las funciones de gestión de sistemas y cada función de gestión de sistemas puede hacer uso de los servicios de otras SMFs, como también de los servicios de CMISE.

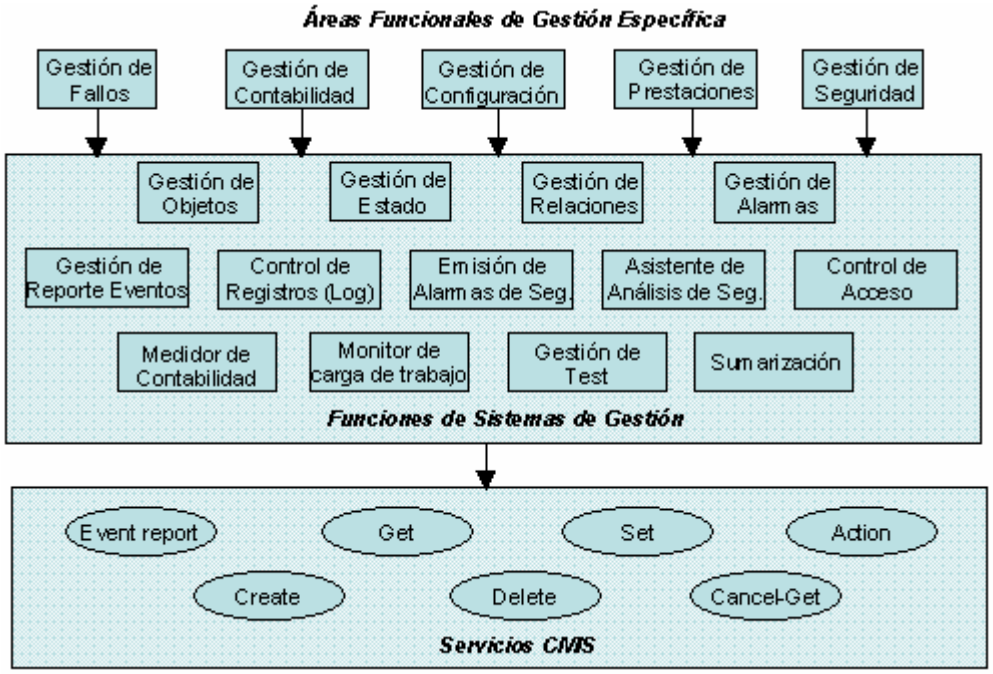

Figura 2.20 - Vistazo a un sistema de gestión

Hasta ahora han sido especificadas 13 funciones:

1. *Gestión de Objetos:* Soporta la creación y eliminación de objetos gestionados, así como la lectura y modificación de los atributos de los objetos. Emite también notificaciones cuando el valor de un atributo presenta algún cambio.

2. *Gestión de Estado:* Especifica un modelo de cómo representar el estado de un objeto. Además, provee servicios para soportar ese modelo.

3. *Gestión de Relaciones:* Especifica un modelo para la representación de las relaciones de gestión entre los objetos manejados. Además, provee servicios para soportar ese modelo.

4. *Emisión de Alarmas:* Soporta la definición de alarmas de fallos y las notificaciones usadas para su envío.

5. *Gestión de Informe de Eventos:* Soporta el control de informes de eventos, incluyendo la especificación del "recipiente" de los reportes, su definición y la especificación de criterios para generarlos y distribuirlos.

6. *Control de logs:* Soporta la creación y almacenamiento de logs, a la vez que especifica los criterios necesarios para hacer registros de este tipo.

7. *Informe de alarmas de seguridad:* Soporta la definición de alarmas de seguridad y las notificaciones usadas para registrarlas.

8. *Asistente de inspección de seguridad:* Especifica las clases de informes de eventos, que podrían ser contenidas en un log usado para evaluar la seguridad.

9. *Control de Acceso:* Soporta el control de acceso a la información y a las operaciones de gestión.

10. *Medidor de Contabilidad:* Provee la contabilidad en el uso de recursos de sistemas y un mecanismo para fijar límites en la cuenta.

11. *Monitor de carga de trabajo:* Soporta la monitorización de atributos de objetos gestionados, relacionados con las prestaciones de un recurso.

12. *Gestión de pruebas:* Soporta la gestión de procedimientos de prueba de diagnósticos.

13. *Sumarización:* Soporta la definición de medidas estadísticas, que son aplicadas a los atributos e indica la información resumida.

Todas estas funciones de gestión de sistemas están bajo el alcance de la estandarización de la ISO y la ITU-T (ver los estándares en el Anexo A), y han sido implementadas mediante la utilización de las facilidades de gestión suministradas por CMISE. La siguiente tabla muestra las relaciones que podrían existir entre las áreas funcionales y las funciones de gestión de sistemas y ha sido basada en las definiciones de R. Aronoff [Aronoff89].

	1	2	3	4	5	6	7	8	9	10	11	12	13
G. Configuración	√	√	√		√				√				
G. Fallos	√	√		√		√						√	
G. Seguridad							√	√	√				
G. Prestaciones	√				√						√	√	√
G. Contabilidad			√			√				√			

Tabla 2.4 Relación entre las áreas funcionales y las funciones de gestión.

Una función de gestión de sistemas puede satisfacer más de un requisito, para lo cual puede ser aplicable más de una función, dando lugar a que exista una relación de muchos a muchos (relación n - m) entre funciones y requisitos. La especificación de una función de gestión de sistemas define las actividades de gestión e información necesarias para cumplir los requisitos. Así las funciones de gestión pueden combinarse para realizar una actividad de gestión específica.

Como en una asociación dada no siempre se requieren todos los servicios, los servicios de una función de gestión de sistemas pueden subagruparse en una o más unidades funcionales, que son unidades básicas de negociación entre usuarios MIS[2], además pueden definirse unidades funcionales que abarcan servicios para más de una función. Así como es posible proporcionar unidades funcionales que atraviesan fronteras de función, para soportar los siguientes conjuntos de capacidades:

[2] Las actividades de gestión se efectúan a través de la manipulación de objetos gestionados. Para la gestión de sistemas, las aplicaciones se clasifican como usuarios del Servicio de Información de Gestión (MIS, Management Information Service) así, cada interacción tiene lugar entre dos usuarios MIS, uno que asume el cometido de gestor y el otro el cometido de agente.

a) Notificaciones solamente
b) Operaciones de gestión solamente
c) Notificaciones y operaciones de gestión.

El agente no puede determinar en general la finalidad de las operaciones de gestión que recibe o las notificaciones que emite. Por ejemplo, un sistema abierto no puede establecer si sus respuestas a peticiones de lectura de contadores de errores, se utilizaran a los efectos de la gestión de averías o de la gestión de calidad de funcionamiento. El agente responde a las peticiones de un gestor individualmente, sin necesitar ningún contexto más amplio dentro del cual realizar la petición

2.7 El Modelo de Gestión TMN.

El término TMN (Telecommunications Management Network) fue introducido por la ITU-T, y está definido en la recomendación M.3010, que buscan apoyar al sector de las telecomunicaciones a obtener un máximo de productividad de los recursos de la red instalada, ofrecer servicios de alta calidad de acuerdo a las exigencias del mercado y sobre todo definir un modelo de arquitectura de gestión común, para todos los fabricantes de equipos [Sloman95].

Aunque en un principio no hubo mucha colaboración entre los grupos de gestión de red de la ISO y el CCITT (germen de la ITU-T), posteriormente fueron incorporados varios conceptos del modelo OSI al estándar TMN. En concreto:

- Se adoptó el modelo gestor-agente del modelo OSI.
- Se siguió el paradigma de la orientación a objetos de la arquitectura OSI.
- Se trabajó conjuntamente en el desarrollo del concepto de dominios de gestión.

Un aspecto diferenciador de ambos modelos consiste en la introducción, en el modelo TMN de una red separada de aquella que se gestiona, con el fin de transportar la información de gestión, Figura 2.21.

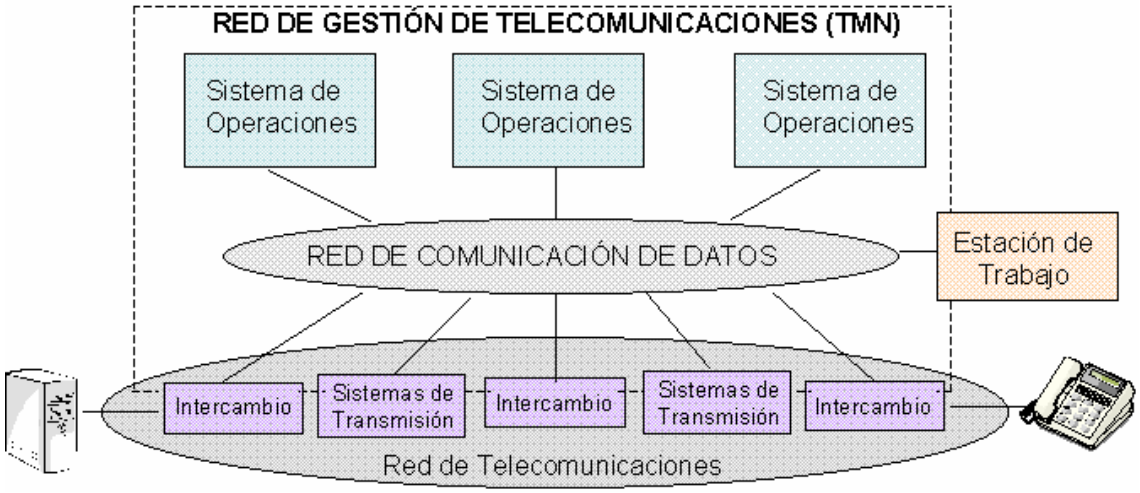

Figura 2.21 Relaciones generales entre una red TMN y una red de telecomunicaciones.

La red de gestión de telecomunicaciones es una red separada lógicamente y opcional físicamente que proporcionando un marco de gestión integrada que recibe, transporta, almacena, procesa y envía información asociada a la gestión de la red de telecomunicaciones (servicios y equipos), para mantener un control sobre ésta y apoyarla en actividades de

OAM&P[3] mediante interfaces normalizadas en diferentes puntos de conexión y funciones de gestión para redes y servicios (clasificadas en cinco áreas funcionales, como ya se vio en el modelo de gestión OSI).

A diferencia del modelo OSI, en el cual se definen cinco áreas funcionales, el estándar TMN no entra en consideraciones sobre las aplicaciones de la información gestionada. Por el contrario se definen las siguientes funcionalidades:

- El intercambio de información entre la red gestionada y la red TMN.
- El intercambio de información entre redes TMN.
- La conversión de formatos, para un intercambio consistente de información.
- La transferencia de información entre puntos de una TMN.
- El análisis de la información de gestión y la capacidad de actuar en función de ella.
- El control del acceso a la información de gestión por los usuarios autorizados.
- La manipulación y presentación de la información de gestión en un formato útil para el usuario de la misma.

2.7.1 Arquitectura TMN.

Los sistemas de gestión deben tener una arquitectura distribuida y modular que facilite la adaptación ante eventuales necesidades [Udupa99]. En TMN existen tres tipos de arquitectura o formas que permiten organizar la gestión de los diferentes sistemas, para realizar actividades de manera eficiente:

- Arquitectura funcional, que describe la distribución de la funcionalidad dentro de la TMN, con el objeto de definir los bloques funcionales a partir de los cuales se construye.

- Arquitectura física, que describe las interfaces y el modo en que los bloques funcionales se implementan en equipos físicos.

- Arquitectura de la información, describe la información a intercambiar entre los elementos de la red, como objetos que representan recursos a gestionar. Esta arquitectura se basa en el paradigma orientado a objetos y sigue los principios de los modelos OSI de gestión (CMIS y CMIP), estudiados en los apartados precedentes del capítulo.

2.7.2 Arquitectura funcional.

Se definen cinco tipos de bloques funcionales. Estos bloques proporcionan la funcionalidad que permite a la TMN realizar sus funciones de gestión. Dos bloques funcionales que intercambian información están separados mediante puntos de referencia.

2.7.2.1 Bloques Funcionales.

A continuación se describen los distintos tipos, Figura 2.22:

- Función de operación de sistemas (OSF): Los OSF procesan la información relativa a la gestión de la red con el objeto de monitorizar y controlar las funciones de gestión. Cabe definir múltiples OSF dentro de una única TMN.

[3] Funciones de gestión propias de TMN la cual comprende el conjunto de procesos asociados a la planificación, diseño, instalación, operación, administración, mantenimiento y aprovisionamiento de redes y servicios de telecomunicaciones.

- Función de estación de trabajo (WSF): Este bloque funcional proporciona los mecanismos para que un usuario pueda interactuar con la información gestionada por la TMN.

- Función de elemento de red (NEF): Es el bloque que actúa como agente, susceptible de ser monitorizado y controlado. Estos bloques proporcionan las funciones de intercambio de datos entre los usuarios de la red de telecomunicaciones gestionada.

- Adaptadores Q (QAF): Este tipo de bloque funcional se utiliza para conectar a la TMN aquellas entidades que no soportan los puntos de referencia estandarizados a ésta, es decir entidades no TMN. Usualmente están representados en equipos que actúan como agentes proxy entre el interfaz CMIP y el equipo que soporta las interfaces propietarias.

- Función de mediación (MF): La función de mediación se encarga de garantizar que la información intercambiada entre los bloques del tipo OSF o NEF cumple los requisitos demandados por cada uno de ellos. Puede realizar funciones de almacenamiento, adaptación, filtrado y condensación de la información.

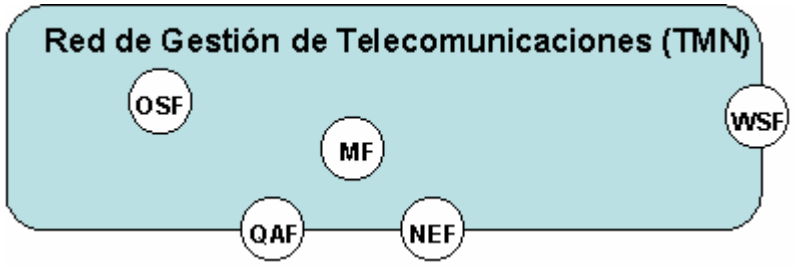

Figura 2.22 Bloques Funcionales

En la siguiente tabla se especifican los puntos de referencia posibles entre los distintos bloques funcionales.

	NEF	OSF	MF	QAF($_{Q3}$)	QAF($_{QX}$)	WSF	no-TMN
NEF		q3	Qx				
OSF	q3	x*, q3	q3	q3		f	
MF	qx	q3	Qx		qx	f	
QAF($_{Q3}$)		q3					m
QAF($_{QX}$)			Qx				m
WSF		f	F				g**
no-TMN				M	m	g**	

Tabla 2.5 Puntos de Referencia entre funciones

* El punto de referencia ***x*** solo se aplica cuando cada OSF está en una TMN diferente
** El punto de referencia ***g*** se sitúa entre el WSF y el usuario, quedando fuera del estándar

2.7.2.2 Componentes funcionales de los bloques.

Cada bloque funcional se compone a su vez de un conjunto de componentes funcionales, considerados como los bloques elementales para su construcción. Estos componentes se identifican en la norma pero no están sujetos a estandarización.

- MAF: Función de Aplicación de Gestión, realiza funciones de soporte y servicios de gestión de la TMN.

- MIB: Base de información de Gestión, representa el conjunto de objetos gestionados internos a un sistema gestionado.

- ICF: Función de Conversión de Información, realiza la traducción sintáctica y semántica entre modelos de información de interfaces diferentes.

- PF: Función de Presentación, convierte el modelo de información propio de un bloque funcional a un formato entendible para una interfaz hombre/máquina.

- HMA: Adaptación Hombre/Máquina, realiza la conversión del modelo de información propio del MAF al modelo de PF, reorganizando o añadiendo datos.

- MCF: Función de Comunicación de Mensajes.

2.7.2.3 Puntos de Referencia.

Los puntos de referencia son puntos conceptuales de intercambio de información entre bloques funcionales adyacentes, Figura 2.23. Los puntos de referencia pueden ser de dos tipos.

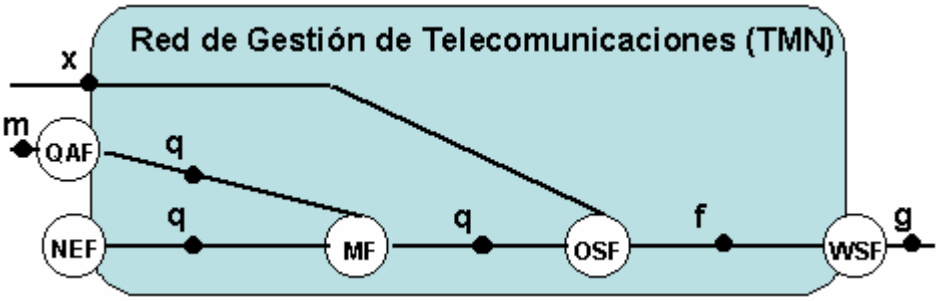

Figura 2.23 Puntos de Referencia TMN

Pertenecientes a la TMN. Existen tres clases.

- *Clase q:* existen dos tipos de punto de referencia:
  q3, para conexión entre bloques OSF con MF, NEF, OSF y QAF.
  qx, para conexión entre bloques MF con MF, NEF y QAF.

- *Clase x:* interconecta bloques OSF de TMN diferentes o entidades funcionales similares.

- *Clase f:* interconecta bloques de función WSF con OSF y MF.

No pertenecientes a la TMN.

- *Clase g:* punto de referencia ubicado entre el WSF y el usuario.

- *Clase m:* ubicado entre QAF y entidades gestionadas no TMN.

2.7.2.4 Interconexión de funciones a través de los puntos de referencia.

En la siguiente figura tenemos un ejemplo de los diversos bloques funcionales que se pueden interconectar mediante los puntos de referencia. Desde la perspectiva de TMN, una interfase se basa en un punto de referencia, a pesar de que las interfases físicas pueden que no sean visibles, por que los puntos de referencia pueden que estén contenidos dentro de los equipos.

Los puntos de referencia **q** se usan para separar el flujo de información entre bloques funcionales. De forma general los bloques funcionales que usan puntos de referencia **q,** pueden no soportar todos los servicios de TMN. Además estos puntos **q**, están definidos entre ciertos conjuntos de bloques funcionales como se muestra en la figura 2.24.

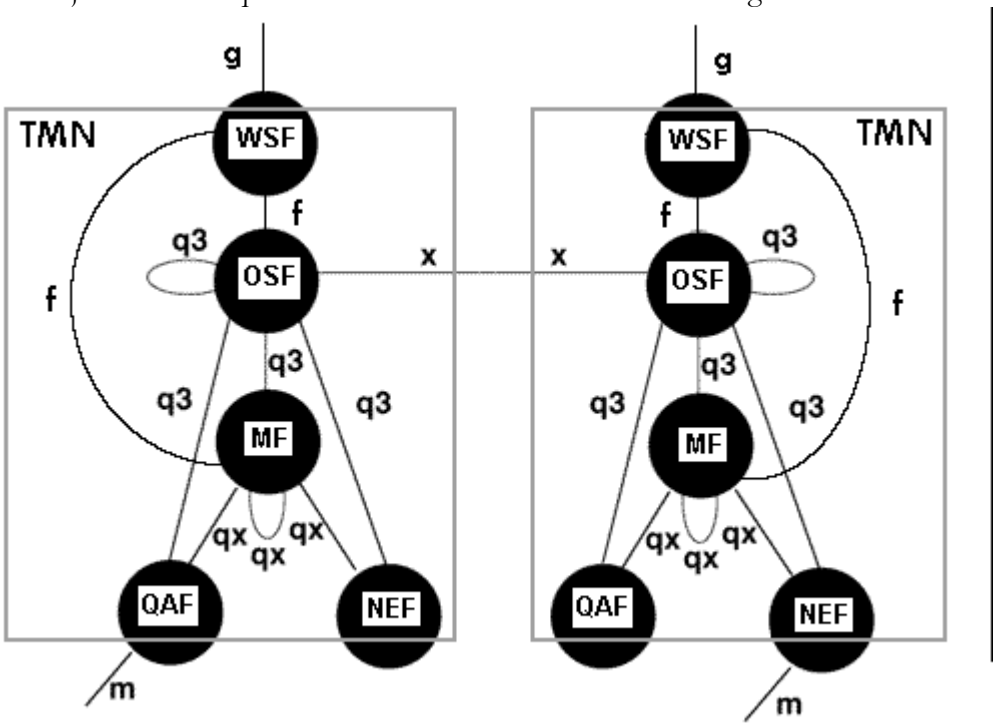

Figura 2.24 Bloques Funcionales y Puntos de Referencia

Los puntos de referencia **f**, se sitúan entre los bloques funcionales WSF y OSF, también entre bloques funcionales WSF y MF.

Los puntos de referencia **x,** se sitúan entre bloques funcionales OSF, residentes en diferentes TMN's.

Los puntos de referencia **g,** se sitúan fuera de TMN. Estos se consideran como una interfase entre un WSF y un usuario humano. Las operaciones en este punto no están definidas por TMN, pero existe información disponible en la serie M.3000. De manera similar los puntos de referencia **m,** se sitúan fuera de TMN, entre un bloque QAF y entidades que no son TMN.

2.7.3 Relación entre objetos y recursos administrados.

Dado que el entorno a gestionar es distribuido, la gestión de red es distribuida, ello hace necesario el intercambio de información entre procesos de gestión. El modelo TMN basado en el modelo OSI para la interconexión de sistemas abiertos, adopta el modelo gestor-agente para el intercambio de mensajes entre entre sistemas o entre sistemas y equipos. TMN identifica dos tipos de roles en los procesos de gestión:

• Gestor: inicia las operaciones de gestión y recibe notificaciones desde los agentes.

• Agente: mantiene los objetos gestionados asociados, responde a las operaciones iniciadas por el gestor y emite notificaciones hacia el gesto

La siguiente figura muestra como se relaciona e interactúan los objetos y recursos a través de TMN.

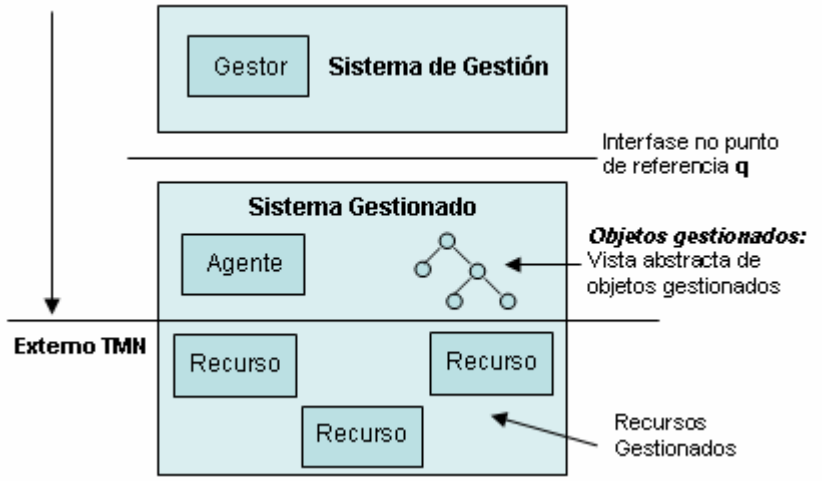

Figura 2.25 Objetos y Recursos Gestionados

Los bloques funcionales TMN pueden actuar en el papel de gestor y/o agente, siguiendo un comportamiento análogo a lo visto para CMIP en OSI. Por otro lado, un objeto puede ser visto por un segundo objeto como gestor, en tanto un tercer objeto lo puede estar viendo como un agente. Una Función de Elemento de Red (NEF), puede efectuar el papel de agente, sin embargo el bloque Funcional de Estación de Trabajo (WSF), que traduce entre TMN y una interfase de usuario no es una entidad de gestión y no actuará ni como agente ni como gestor.

2.7.4 Arquitectura física.

La arquitectura física se encarga de definir como se implementan los bloques funcionales mediante equipamiento físico y los puntos de referencia en interfaces. En la arquitectura física se definen los siguientes bloques constructivos:

- Elemento de red (NE).
- Dispositivo de mediación (MD).
- Adaptador Q (QA).
- Sistema de operaciones (OS).
- Red de comunicación de datos (DCN).

Cada uno de estos bloques puede implementar uno o más bloques funcionales (excepto el DCN que se encarga de realizar el intercambio de información entre bloques), pero siempre hay uno que ha de contener obligatoriamente y que determina su denominación.

2.7.4.1 Interfaces.

Las interfaces son implementaciones de los puntos de referencia y son comparables a las pilas de protocolos. Existe una correspondencia uno a uno entre los puntos de referencia y las interfaces, excepto para aquellos que están fuera de la TMN, es decir, los puntos de referencia g y m.

- Interfaz Q: Esta interfaz determina las pilas de protocolo a utilizar para intercambiar información dentro de una TMN. Existen dos tipos de interfaz provenientes de las dos clases de puntos de referencia mencionados en la arquitectura funcional TMN (Qx y Q3).

- Interfaz F: aplicable a los puntos de referencia f.

- Interfaz X: aplicable al punto de referencia x, define la pila de protocolo a utilizar para intercambiar información entre entidades TMN.

2.7.5 Arquitectura lógica de niveles.

En el estándar TMN se definen una serie de capas o niveles de gestión mediante las cuales se pretende abordar la gran complejidad de la gestión de redes de telecomunicación. Cada uno de estos niveles agrupa un conjunto de funciones de gestión. A continuación se define cuáles son esos niveles y las relaciones entre ellos, Figura 2.26.

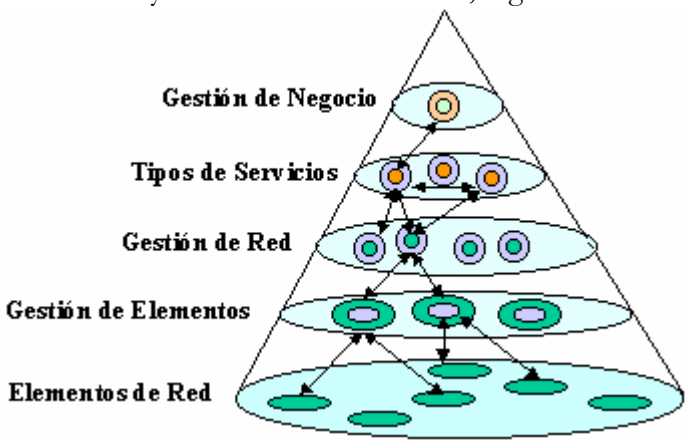

Figura 2.26 Arquitectura Lógica de Niveles

- Nivel de Elementos de Red. Incluye las funciones que proporcionan información en formato TMN del equipamiento de red, así como las funciones de adaptación para proporcionar interfaces TMN a elementos de red no-TMN.

- Nivel de Gestión de Elementos. Referente a la gestión remota e individual de cualquier elemento de red que se precise para el establecimiento de conexiones entre dos puntos finales, proporcionando un servicio dado. Este nivel suministra las funciones de gestión para monitorizar y controlar elementos de gestión individuales, en la capa de elemento de red.

- Nivel de Gestión de Red. Proporciona el control, supervisión, coordinación y configuración de grupos de elementos de red, constituyendo redes y subredes para la realización de una conexión.

- Nivel de Gestión de Servicios. Contiene las funciones que proporcionan un manejo eficiente de las conexiones entre los puntos finales de la red, asegurando un óptimo aprovisionamiento y configuración de los servicios prestados a los usuarios.

- Nivel de Gestión de Negocio. Soporta la gestión completa de la explotación de la red, incluyendo contabilidad, gestión y administración, basándose para ello en las entradas procedentes de los niveles de Gestión de Servicios y de Gestión de Red.

2.7.6 Ejemplo simplificado de arquitectura TMN.

El Sistema de Operaciones, lleva a cabo las funciones OSF, y además se puede encargar de las MF, QAF y WSF.

El Dispositivo de mediación efectúa funciones de MF, y puede realizar además las QAF y WSF.

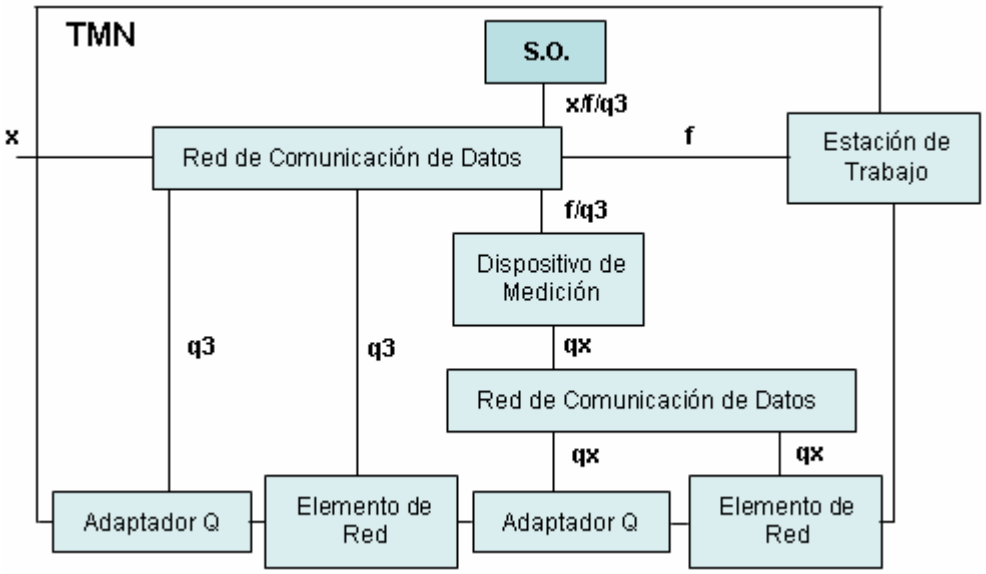

Figura 2.27 Arquitectura TMN Simplificada

Un adaptador Q, es el encargado de conectar elementos de red de sistemas que no tienen interfaces TMN, es decir, puntos de referencia **m** e interfaces qx o q3.

La red de comunicaciones de datos opera en los niveles 1,2 y 3 de OSI.

La Estación de trabajo es la responsable de la función WSF, traduce la información de un punto de referencia **f** a un punto de referencia **g** y viceversa.

2.7.7 Como se comunica la Información de Gestión no TMN.

En la siguiente figura 2.28, los agentes y gestores que hay entre los sistemas A, B y C forman parte de la red de gestión de telecomunicaciones TMN. Las operaciones entre gestor y agente del sistema B no forman parte de TMN. Es decir los recursos reales no forman parte de TMN, pero su representación a través del modelo de información si lo es. El sistema A administra al sistema B mediante referencias al modelo de información, análogamente B gestiona a C.

La situación es que una entidad gestiona a otra entidad, y esta segunda a su vez gestiona a otra tercera entidad, a este modo se le llama cascada de sistemas. Por lo tanto, como en ejemplo de la figura queda patente, un sistema puede ser a la vez gestor y agente.

TMN usa principios de orientación a objetos e interfaces estándar para definir la comunicación entre bloques funcionales que hay en una red. Dado que TMN se basa en el marco de gestión OSI, las funciones de gestión son ejecutadas por operaciones que se basan en el Elemento de servicio genérico de información (CMISE). La información que se gestiona recibe el nombre Base de Información de Gestión (MIB).

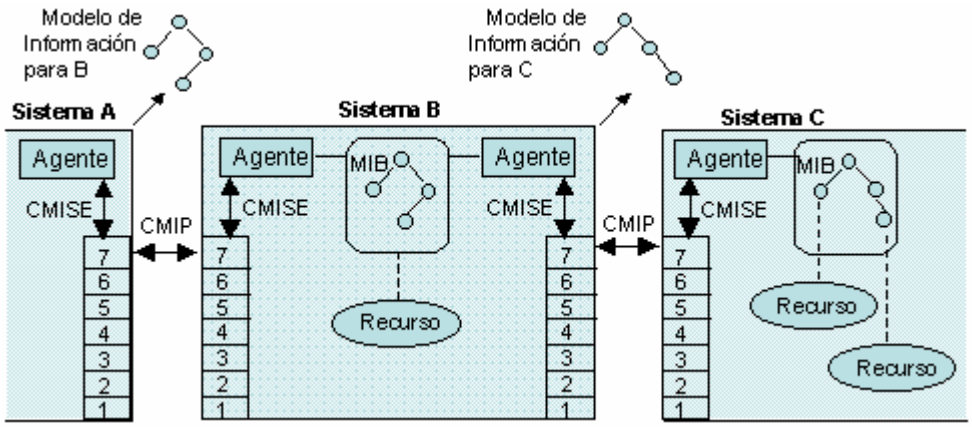

Figura 2.28 Iteración entre Funciones TMN (Cascada de Sistemas)

Los procesos que administran información se denominan entidades. Una entidad de gestión puede tomar un papel de gestor o agente, que a su vez envían y reciben peticiones y notificaciones a través del Protocolo de Información de Gestión Común (CMIP).

2.8 El Modelo de Gestión de Internet.

Basado en el conjunto de protocolos TCP/IP y diseñado para ser usado en entornos de proceso de datos distribuidos con ordenadores personales, estaciones de trabajo y ordenadores centrales, se encuentra ampliamente extendido entre distintos fabricantes para la gestión de redes.

Este modelo no es el resultado de ningún fabricante en particular, fue concebido en abril de 1987 por cuatro personas: Jefferi Case (Universidad de Tennesse), James Davin (del Massachussets Institute of Technology) y Martín Achoffstall y Mark Fedor (Performance Systems International). Una de sus características principales es que utiliza el protocolo SNMP, *Protocolo Simple de Gestión de Red,* para el intercambio de la información de gestión.

2.8.1 El Modelo.

Al igual que el modelo OSI utiliza el modelo gestor-agente, pero con la salvedad de que las comunicaciones serán lo más simples posibles. Las funciones del agente consisten únicamente en la alteración o inspección de ciertos valores, limitando de esta forma a sólo dos las funciones esenciales de gestión y eludiendo la necesidad de un protocolo complejo. En la otra dirección, desde el agente a gestor, se utiliza un número limitado de mensajes no solicitados para informar sobre sucesos asíncronos.

SNMP se fija como principal objetivo, que ***"el impacto de añadir gestión de red a los nodos debe ser mínimo"***. Tal y como TCP, impone unos requisitos mínimos de implementación impuestos a las capas inferiores del modelo.

Tiene como consecuencia, la imposibilidad de plantearse un modelo de gestión como OSI, que exige gran capacidad a los agentes para realizar las operaciones de gestión. El modelo Internet interpreta la gestión desde el punto de vista de la simpleza, lo que permite su implantación en todo tipo de nodos.

2.8.2 Arquitectura de Gestión.

Aunque tiene una estructura mucho más sencilla que la correspondiente a los sistemas de gestión OSI, aparecen los cuatro elementos principales de esta: gestor, agente, base de información de gestión (MIB) y el protocolo, en este caso SNMP, Figura 2.29.

Para el modelo de gestión Internet, únicamente se definen los modelos de comunicación e Información, estando ausentes los modelos de organización y funcional, que aparecían en el modelo de gestión OSI.

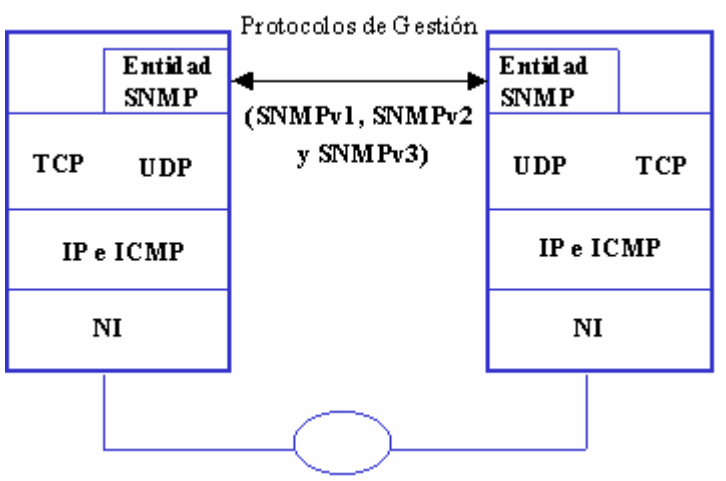

Figura 2.29 Arquitectura SNMP

2.8.3 Modelo de Información.

Pretende elaborar unas definiciones de objetos, para poder representar de la misma manera todos los recursos de comunicaciones susceptibles de ser gestionados.

El modelo OSI se construye desde la perspectiva de diseño de recursos complejos y dota a dichos recursos de una gran parte de la funcionalidad de la gestión. Por contra el modelo Internet, se diseña con la premisa de conservar la simplicidad de los agentes y una escasa funcionalidad de gestión de estos.

Con el objetivo de mantener la sencillez, el modelo de información no se ciñe a las técnicas de diseño orientado a objetos (usada por la gestión OSI), sino que utiliza la idea de *Tipo de Objeto* (Object type). Estos tipos de objetos son simples variables escalares sobre los que se definen como únicas operaciones la lectura y la escritura. Cada tipo de objeto debe estar caracterizado por los siguientes apartados:

- *Nombre:* Cada objeto se representa unívocamente por un identificador de Objeto (OID, Object Identifier), es la secuencia de enteros que se obtiene al recorrer los nodos del árbol global de información.

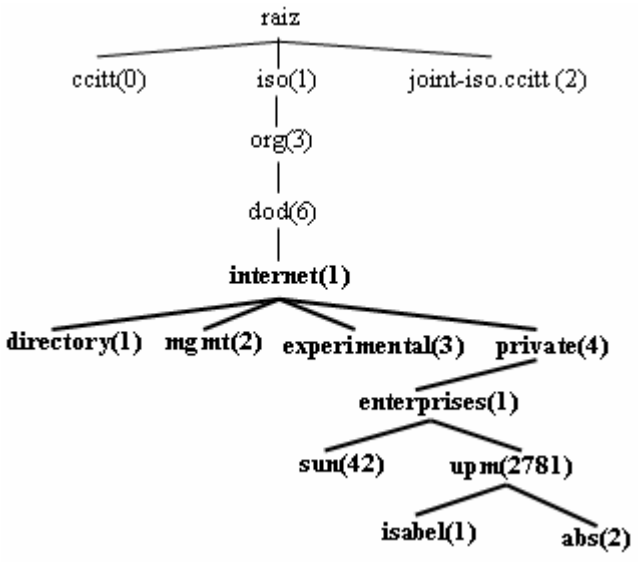

Figura 2.30 Árbol de identificadores de objetos

- *Sintaxis:* Indica la estructura de los valores mantenidos por los objetos gestionados.

- *Reglas de Codificación:* Basic Encoding Rules (BER) de codificación ASN.1, *Abstract Syntax Notation #1*, Notación de Sintaxis Astracta Nº 1 (AnexoB), se utilizan para la trasmisión de los valores de los objetos definidos mediante SMI.

La información de gestión que se intercambia en las relaciones que se establecen entre un gestor y un agente, se describen desde dos puntos de vista complementarios:

- Estructura de la información de Gestión. *(SMI, Strcuture of Mangement Information):* Define la estructura lógica de la información, descripción e identificación, es decir la estructura sintáctica de esta.

- Base de Información de Gestión, *(MIB).* Describe los recursos que son objeto de las relaciones de gestión, para ello utiliza la Estructura de la Información de Gestión (SMI). Los objetos se organizan en grupos, según su temática.

2.8.3.1 MIB (Management Information base).

A una colección de objetos, referentes a una área de gestión común, se le conoce con el nombre de "*módulo MIB*" o simplemente MIB, también se le da este nombre a la base de objetos de los agentes que pueden ser observados por y controlados desde los gestores SNMP. La MIB define y contiene los objetos susceptibles de ser gestionados. La información de gestión usada con el SNMP se define como un conjunto de objetos gestionados, utilizando la sintaxis abstracta ASN.1.

La RFC 1155, *Strucutre and Identification of Management Information for TCP/IP-Based Internets* (ver anexo A), presenta las reglas para la definición de los objetos (*Structure and Identification of Management Information,* SMI), por otro lado la RFC 1212, *Concise MIB definition* proporciona un formato para la definición de los módulos MIB (ver anexo A).

Conceptualmente, la MIB representa una base de datos de "parámetros de gestión", almacenados en una base de datos arborescente, donde las entradas a raíz definen las organizaciones de las cuales dependen los diferentes parámetros. Alejándose de la raíz del árbol, vamos encontrando el emplazamiento de los verdaderos parámetros de gestión (los

objetos), donde cada uno de los dispositivos que se gestionan proporciona diferente información. Por ejemplo un encaminador proporcionará su tabla de encaminamiento, un puente una tabla de filtros, etc.

Actualmente tenemos los siguientes tipos de MIB, las estándares como la MIB-I y MIB-II, las experimentales y en un tercer grupo las privadas, que incorporan la información de los diversos fabricantes de equipos (ejemplo Sun).

2.8.3.2 PROXY.

En coherencia con la directiva del IAB *(Internet Activities Board)*, de producir a corto plazo sistemas manejables y simples, la lista de objetos definidos en la MIB estándar está constituida por aquellos objetos considerados como esenciales.

SMI permite la adición de nuevos objetos estándar mediante la definición de nuevas versiones de la MIB (MIB-I y MIB-II), también añadiendo estos nuevos objetos a la rama experimental, aunque no serán estándares, o un tercer caso como objetos privados en MIB privadas, para lo cual es necesario un nuevo elemento conocido como Proxy.

Un Proxy permite mantener una MIB privada e interactúa con un proceso SNMP, para comunicarse con uno o más gestores. El protocolo de comunicación usado entre el proxy y el proceso SNMP no será SNMP, sino una interfaz programada que implementará un protocolo común.

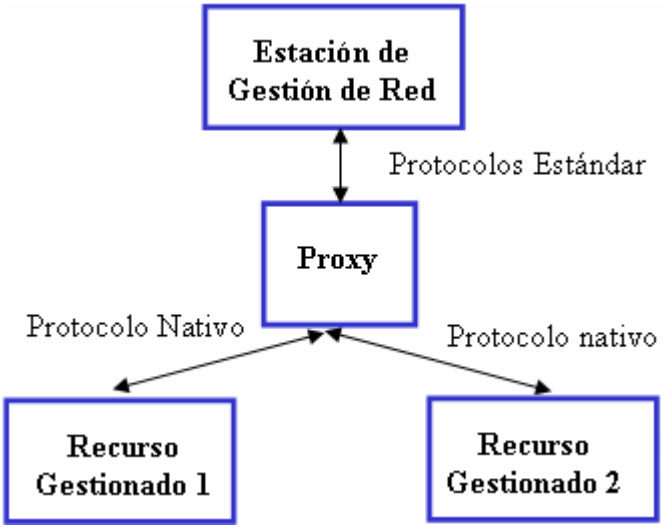

Figura 2.31 El Concepto Proxy

2.8.4 Modelo de comunicaciones

En Internet además de una visión común de todos los recursos por parte de los gestores y agentes, es necesario definir el mecanismo que permita el intercambio de información útil para la gestión, en nuestro caso el protocolo SNMP.

El requisito imprescindible para SNMP, es que cuando el resto de la red falle, la gestión deberá continuar funcionando siempre que sea posible [Stallings00]. Esto provoca que algunas de las funciones habituales del nivel de transporte, sea tratada directamente por las aplicaciones gestoras y que el protocolo no sea orientado a conexión. Lo normal es utilizar SNMP sobre UDP, como protocolo de transporte e IP como protocolo de red.

Para el intercambio de información se utilizan las PDUs (Protocol Data Units, Unidades de datos de Protocolos), definidas en ASN.1.

En SNMP el agente se comunica con el gestor mediante las siguientes primitivas:

- get: Enviado por la estación gestora para obtener las variables MIB específicas de un nodo gestionado. El agente responde con un ***get-response*** conteniendo las variables requeridas o un mensaje de error.

- get-next: Es enviado por la estación gestora a la gestionada para obtener la variable MIB siguiente a la especificada. El agente responde con un ***get-response*** conteniendo las variables requeridas o un mensaje de error. Mediante el uso de una serie de mandatos ***get-next*** (cada uno con el nombre de la variable obtenida en el get-next previo), la estación gestora puede obtener todas la variables de la MIB de un nodo gestionado.

- get-response: Es enviado por el nodo gestionado a la estación gestora como respuesta a los mandatos ***get, get-next o set.***

- set: Enviado por la estación gestora para dar un valor determinado a una variable MIB de un nodo gestionado. El nodo gestionado responde con un mandato ***get-response*** idéntico al solicitante o con una indicación de error.

- trap: Además el agente bajo ciertas condiciones críticas puede enviar un mensaje no solicitado al gestor, será no confirmado.

La figura 2.32 muestra el flujo de mandatos entre la estación y el nodo gestionado.

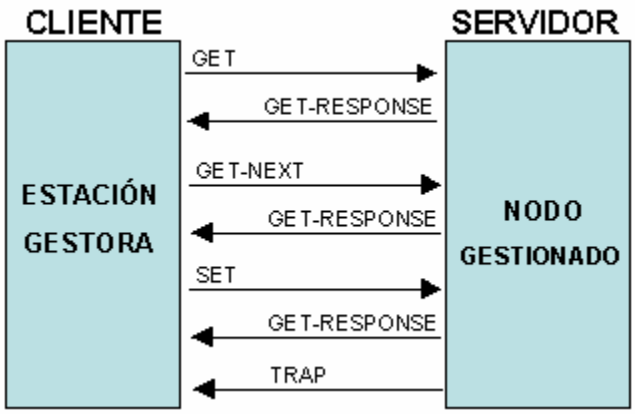

Figura 2.32 Flujo de mandatos SNMP.

En general, la estación gestora obtiene y guarda información actualizada sobre el estado de los objetos que gestiona, mediante muestreo *(polling)* a intervalos regulares de tiempo. En aquellos casos en los que los nodos gestionados desean informar a su/s gestor/es de un acontecimiento extraordinario, sin esperar a ser preguntados, hacen uso del mecanismo de los *traps*. Por ejemplo, cuando un nuevo nodo es activado puede que se tarde mucho tiempo hasta que la estación gestora lo descubra; este tiempo se reduce si el nuevo no envía un trap al gestor que le avise del evento.

Los Trap de SNMP, no tienen la misma funcionalidad que los eventos tendrán en la gestión OSI, sólo se definen unos pocos con el propósito de informar de situaciones específicas.

En SNMP, a diferencia de OSI, se utiliza una política de seguridad basada en el concepto de Comunidad. Esta identifica una relación entre varios gestores y un agente. Los agentes pertenecientes a una comunidad poseerán unos determinados derechos de acceso a los objetos mantenidos. Cada comunidad se identifica por un nombre consistente en una cadena de octetos, que se transmitirá en todos los mensajes SNMP, controlando en todo momento dos aspectos fundamentales de la seguridad: la Autentificación y el Control de Acceso.

2.8.5 Common Management Information Protocol over TCP/IP (CMOT).

CMOT es la arquitectura de gestión de red desarrollada con vistas a mantener una relación más estrecha con el CMIP (*Common Management Information Protocol*) de OSI. Con esta premisa, CMOT se divide, como en OSI, en un modelo organizacional, funcional e informacional.

En los dos primeros el mismo concepto de OSI se usa en CMOT y SNMP. Todos los objetos de gestión se definen en el MIB (*Management Information Base*), y se representan con el SMI (*Structure and Identification of Management Information*), un subconjunto de ASN.1 ("Abstract Syntax Notation 1" de OSI).

En el modelo funcional, CMOT adopta el modelo OSI que divide los componentes de gestión en gestor y agentes. El agente recoge información, realiza comandos y ejecuta tests, y el gestor recibe datos, genera comandos y envía instrucciones a los agentes. El gestor y el agente están constituidos por un conjunto específico de entidades de información de gestión por cada capa de comunicación, denominadas LME (*Layer Management Entities*).

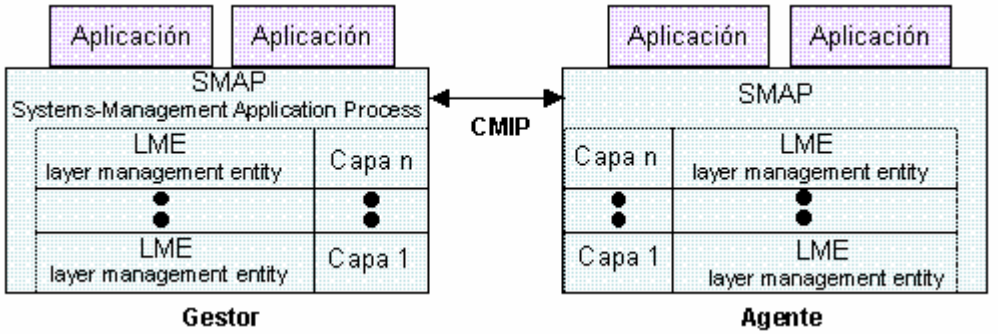

Figura 2.33 Componentes de CMIP sobre TCP/IP.

Todos los LME los coordina el SMAP (*System Management Application Process*), Aplicación de gestión de sistemas, software local de un equipo (sistema) gestionado que implementa las funciones de gestión para ese sistema (host, router, etc.). Tiene acceso a los parámetros del sistema y puede, por tanto, gestionar todos los aspectos del sistema y coordinarse con SMAPs de otros sistemas a través de CMIP(*Common Management Information Protocol*).

En el mundo OSI, la gestión sólo se puede producir sobre conexiones establecidas por completo entre gestores y agentes. CMOT permite el intercambio de información de gestión usando servicios no orientados a conexión (datagramas). Pero para mantener la misma interfaz del servicio que requiere CMIP, denominada CMIS (*Common Management Information Services*), la arquitectura de CMOT define una nueva capa, el LPP (*Lightweight Presentation Protocol*). Esta capa se ha definido para proporcionar los servicios de presentación que necesita CMIP, de tal forma que la totalidad de los estándares OSI para la gestión de red se adapten a la arquitectura TCP/IP de CMOT.

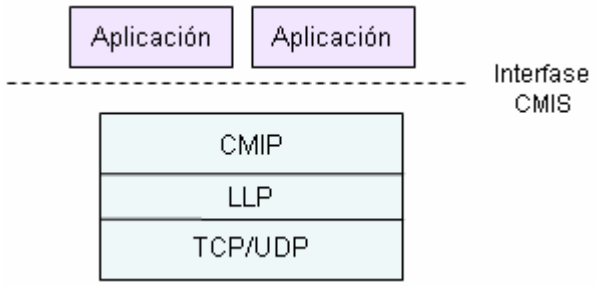

Figura 2.34 LPP ("Lightweight Presentation Protocol")

Capítulo 3

Sistemas Expertos y su Aplicaciones a la Gestión de Redes de Comunicaciones.

Una vez descritos los principales modelos de gestión de redes existentes, en este capítulo se exponen conceptos importantes sobre la inteligencia artificial. Se analizará la importancia de los sistemas expertos, su construcción, los elementos que lo componen y su aplicación en el campo de la gestión de redes de comunicaciones.

No hace mucho se creía que algunos problemas como la demostración de teoremas, el reconocimiento de la voz y el de patrones, ciertos juegos (como el ajedrez o las damas), sistemas altamente complejos de tipo determinista o estocástico, como la gestión de redes de telecomunicaciones, debían ser resueltos por personas, dado que su formulación y resolución requieren ciertas habilidades que sólo se encuentran en los seres humanos, (por ejemplo, la habilidad de pensar, observar, memorizar, aprender, ver, oler, etc.). Sin embargo, el trabajo realizado en las tres últimas décadas por investigadores procedentes de varios campos, muestra que muchos de estos problemas pueden ser formulados y resueltos por máquinas.

El amplio campo que se conoce como *Inteligencia Artificial* (IA), trata de estos problemas, que en un principio parecían imposibles, intratables y difíciles de formular utilizando ordenadores. A. Barr y E. A. Feigenbaum, dos de los pioneros de la investigación en IA, definen ésta como sigue [Negnevitsky02]:

La Inteligencia Artificial es la parte de la Ciencia que se ocupa del diseño de sistemas de computación inteligentes, es decir, sistemas que exhiben las características que asociamos a la inteligencia en el comportamiento humano que se refiere a la comprensión del lenguaje, el aprendizaje, el razonamiento, la resolución de problemas, etc.

Hoy en día, el campo de la IA engloba varias subáreas tales como los sistemas expertos, la demostración automática de teoremas, el procesamiento del lenguaje natural, la visión artificial, la robótica, las redes neuronales, el reconocimiento de la voz y de patrones, etc.

3.1 ¿Qué es un Sistema Experto?

En la literatura existente se pueden encontrar muchas definiciones de sistema experto. Por ejemplo, [Meystel02]:

Los sistemas expertos son máquinas que piensan y razonan como un experto lo haría en una cierta especialidad o campo. Por ejemplo, un sistema experto en diagnóstico médico requeriría como datos los síntomas del paciente, los resultados de análisis clínicos y otros hechos relevantes, y, utilizando éstos, buscaría en una base

de datos la información necesaria para poder identificar la correspondiente enfermedad. Un Sistema Experto de verdad, no sólo realiza las funciones tradicionales de manejar grandes cantidades de datos, sino que también manipula esos datos de forma tal que el resultado sea inteligible y tenga significado para responder a preguntas incluso no completamente especificadas.

Aunque la anterior es todavía una definición razonable de un sistema experto, han surgido desde entonces otras definiciones, debido al rápido desarrollo de la tecnología. El sentido de estas definiciones puede resumirse como sigue:

Un sistema experto puede definirse como un sistema informático (hardware y software) que simula a los expertos humanos, en un área de especialización dada.

Como tal, un sistema experto debería ser capaz de procesar y memorizar información, aprender y razonar en situaciones deterministas e inciertas, comunicar con los hombres y/u otros sistemas expertos, tomar decisiones apropiadas y explicar por qué se han tomado tales decisiones. Se puede pensar también en un sistema experto, como un consultor que puede suministrar ayuda a (o en algunos casos sustituir completamente) los expertos humanos con un grado razonable de fiabilidad.

Durante la última década se han desarrollado muy rápidamente numerosas aplicaciones de sistemas expertos a muchos campos. Durkin examina unos 2.500 sistemas expertos y los clasifica por criterios, tales como áreas de aplicación, tareas realizadas, etc. Tal como puede verse en la Figura 3.1, la economía, la industria y la medicina continúan siendo los campos dominantes, entre aquellos en los que se utilizan los sistemas expertos [Callan03].

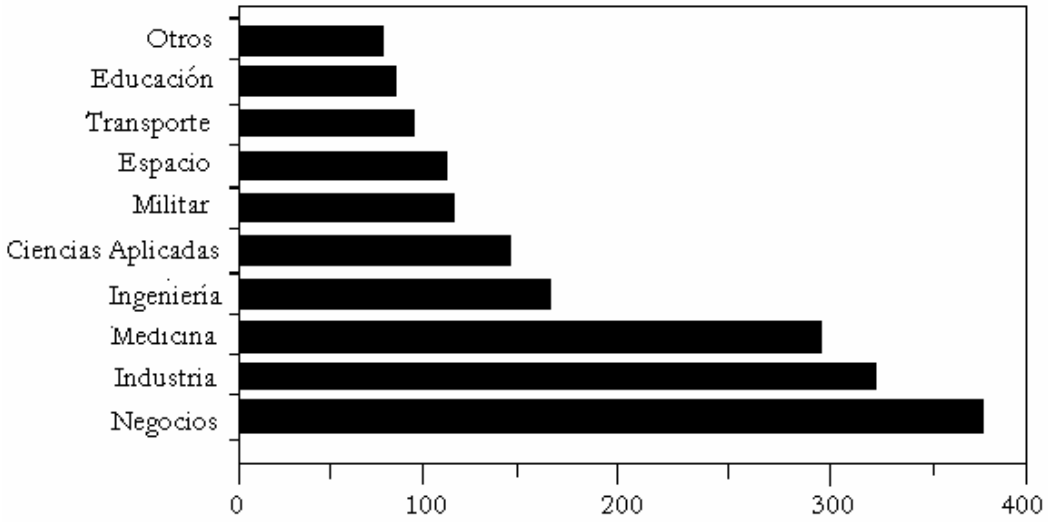

Figura 3.1 Campos de aplicación de los sistemas expertos.

3.1.1 ¿Por Qué los Sistemas Expertos?

El desarrollo o la adquisición de un sistema experto es generalmente caro, aunque la ganancia en términos monetarios, tiempo y precisión resultantes del uso de los sistemas expertos pueden llegar a ser muy altas y la amortización de este relativamente rápida. Sin embargo, antes de desarrollar o adquirir un sistema experto debe realizarse un análisis de factibilidad y de coste-beneficio. Hay varias razones para utilizar sistemas expertos, las más importantes son:

1. Con la ayuda de un sistema experto, personal con poca experiencia puede resolver problemas que requieren un conocimiento de experto. Esto es también importante en casos en los que hay pocos expertos humanos. Además, el número de personas con acceso al conocimiento aumenta con el uso de sistemas expertos.

2. El conocimiento de varios expertos humanos puede combinarse, lo que da lugar a sistemas expertos más fiables, ya que se obtiene un sistema experto que combina la sabiduría colectiva de varios expertos humanos en lugar de la de uno sólo.

3. Los sistemas expertos pueden responder a preguntas y resolver problemas mucho más rápidamente que un experto humano. Por ello, los sistemas son muy valiosos en casos en los que el tiempo de respuesta es crítico.

4. En algunos casos, la complejidad del problema impide al experto humano resolverlo. En otros casos la solución de los expertos humanos no es fiable. Debido a la capacidad de los ordenadores de procesar un elevadísimo número de operaciones complejas de forma rápida y aproximada, los sistemas expertos suministran respuestas rápidas y fiables en situaciones en las que los expertos humanos no pueden.

5. Se pueden obtener enormes ahorros mediante el uso de sistemas expertos.

6. Para terminar señalar que los sistemas expertos pueden ser utilizados para realizar operaciones monótonas, aburridas y tediosas para los humanos. En verdad, los sistemas expertos pueden ser la única solución viable en una situación en la que la tarea a realizar desborda al ser humano (por ejemplo, un avión o una cápsula espacial dirigida por un sistema experto).

Pese a su innegable potencia y utilidad, los sistemas expertos también presentan inconvenientes:

1. Los costes y duración del desarrollo de un sistema experto son bastante considerables, aunque se suelen amortizar rápidamente.

2. Dificultad y elevado coste en tiempo y dinero para extraer el conocimiento de los especialistas humanos. La representación del conocimiento también suele ser problemática.

3. Otro hecho importante a tener en cuenta es que el campo de aplicación actual es restringido y específico. La existencia de problemas que no son resolubles algorítmicamente, o cuya solución mediante un sistema experto u otro método convencional no es suficientemente buena. Si un problema sobrepasa la competencia de un sistema experto, sus prestaciones se degradan de forma notable.

4. Las estrategias de razonamiento de los motores de inferencia suelen estar programadas procedimentalmente y se adaptan mal a las circunstancias. Están limitados para tratar problemas con información incompleta. Un experto humano no estudia progresivamente una hipótesis, sino que decide de inmediato cuando se enfrenta a una situación análoga a otra ocurrida en el pasado. Por el contrario los sistemas expertos no utilizan este tipo de razonamiento por analogía.

5. Poca flexibilidad a cambios y dificultades presentes por la manipulación de información incompleta, inconsistente o errónea.

6. Finalmente, hay que tener en cuenta los problemas sociales que acarrean al ser susceptibles de influir en la estructura y número de empleos, en los distintos contextos de aplicación.

3.1.1.1 Situaciones de Utilización.

Una vez expuestas las ventajas e inconvenientes derivados de la utilización de los sistemas expertos, veamos las situaciones en las que se aconseja el uso de éstos. Especialmente en los casos siguientes:

• Cuando el conocimiento es difícil de adquirir o se basa en reglas que sólo pueden ser aprendidas de la experiencia.

• Cuando la mejora continua del conocimiento es esencial y/o cuando el problema está sujeto a reglas o códigos cambiantes.

• Cuando el costo de emplear expertos humanos es elevado o difíciles de encontrar.

• Cuando el conocimiento de los usuarios sobre el tema es limitado.

3.1.2 Tipos de Sistemas Expertos.

Los problemas con los que pueden tratar los sistemas expertos, pueden clasificarse en dos tipos: problemas esencialmente deterministas y problemas esencialmente estocásticos. Consecuentemente, se puede hacer una primera clasificación de los sistemas expertos en función de la naturaleza de problemas para los que están diseñados:

- Deterministas: Los problemas de este tipo pueden ser formulados usando un conjunto de reglas que relacionen varios objetos bien definidos. Los sistemas expertos que tratan problemas deterministas son conocidos como sistemas basados en reglas, porque sacan sus conclusiones basándose en un conjunto de reglas, utilizando para ello un mecanismo de razonamiento lógico.

- Estocásticos: En situaciones inciertas, es necesario introducir algunos medios para tratar la incertidumbre. Por ejemplo, algunos sistemas expertos usan la misma estructura de los sistemas basados en reglas, pero introducen una medida asociada a la incertidumbre de las reglas y a la de sus premisas. En este caso se pueden utilizar algunas fórmulas de propagación para calcular la incertidumbre asociada a las conclusiones. Durante las últimas décadas han sido propuestas algunas medidas de incertidumbre, como por ejemplo *factores de certeza*, *la probabilidad, la lógica difusa y la teoríua de la evidencia de Dempster y Shafer* entre otros, estudiados en el apartado 3.6 del presente capítulo.

3.2 La Ingeniería del Conocimiento.

Uno de los aspectos más importantes para la construcción de un Sistema Experto es la adquisición del conocimiento, centenares de reglas y millares de hechos que se obtienen generalmente con la intervención de expertos en el dominio de aplicación y para lo cual es necesario disponer de técnicas que automaticen este proceso. La Ingeniería del conocimiento trata de definir y de formalizar un conjunto de métodos que nos permitan adquirir conocimiento de alto nivel y representarlo según un esquema computacional eficaz, para resolver problemas difíciles, en dominios de aplicación concretos.

Los sistemas expertos no sólo representan al dominio que tratan de modelar, también deben conservar representaciones de su propia estructura interna y de su funcionamiento, lo denominamos autoconocimiento. Esto permite a los Sistemas Expertos justificar sus conclusiones, explicar sus procesos de razonamiento e incrementar el conocimiento que

poseen. La ingeniería del conocimiento (IC) sugiere arquitecturas que separan claramente los conocimientos del dominio, de los mecanismos de inferencia y control.

Podríamos establecer que *"La Ingeniería del Conocimiento es la disciplina tecnológica que se centra en la aplicación de una aproximación sistemática, disciplinada y cuantificable al desarrollo, funcionamiento y mantenimiento de Sistemas Basados en Conocimiento. En otras palabras, el objetivo ultimo de la IC es el establecimiento de metodologías que permitan abordar el desarrollo de Sistema Basado en Conocimientos (SBC)[1] de una forma más sistemática".*

El propósito de la Ingeniería del Conocimiento es construir sistemas del conocimiento y el propósito de la Adquisición del Conocimiento es obtener el conocimiento que se requiere para construir el sistema. Todo lo que en esta disciplina se plantea puede ser aplicado para desarrollar otro tipo de sistemas, ya que proporciona técnicas que son adecuadas para el manejo del conocimiento organizacional y su mantenimiento.

Se debe tener en cuenta que el proceso de adquisición del conocimiento se realiza durante todo el desarrollo del sistema, desde el mismo momento en que se comienza a estudiar el problema y su solución, hasta cuando se lleva a cabo su evolución. Por lo tanto, se realiza durante todas las etapas del desarrollo, en unas con mayor intensidad que en otras. Se puede decir que es un proceso que no termina.

3.2.1 Equipo Humano. Roles y Perfiles.

La Ingeniería del Conocimiento involucra una variedad de personas:

● ***El Ingeniero del conocimiento***: Persona encargada de la construcción y puesta en marcha de un sistema inteligente. Éste debe tener conocimientos profundos sobre cómo desarrollar el sistema, conocer las herramientas de desarrollo de este tipo de sistemas, conocer algunas estrategias efectivas de comunicación y saber un poco sobre psicología. En el libro de Anna Hart, ella dice:

"*El ingeniero del conocimiento debe tener ciertas habilidades, entre las cuales están las siguientes: buenas herramientas de comunicación, inteligencia con mente abierta, tacto y diplomacia, empatía y paciencia, persistencia, versatilidad e inventiva, conocimiento del dominio y conocimientos de sistemas inteligentes" [Anderson02].*

Por lo tanto, este papel puede ser desempeñado por una persona o por un grupo de Ingenieros que comparten los conocimientos definidos anteriormente. Entre sus tareas más importantes podemos enumerar las siguientes:

- Ayudar al experto en la labor de explicar los conocimientos.
- Codificar y representar el conocimiento, adecuado a la herramienta de desarrollo.
- Conocer la fortaleza y debilidades de las herramientas y metodología a utilizar.
- Detectar inconsistencias.
- Etc.

[1] De forma general denominación dada a los sistema computacionales que utiliza un "*mecanismo de inferencia"* y "*una Base de Conocimientos*" para la solución de problemas. Se verán con más detalle en el apartado 3.5, donde se explican los elementos principales de los sistemas expertos.

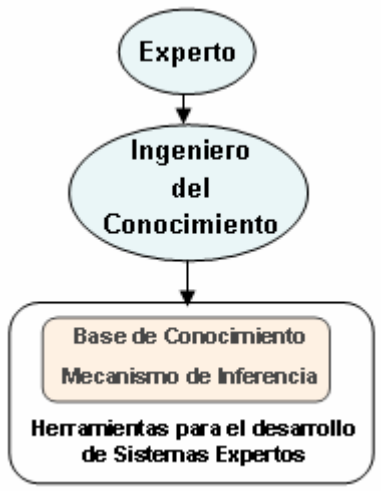

Figura 3.2 El ingeniero del conocimiento.

● ***El Experto***: Provee del conocimiento del dominio y de la tarea específica, proporciona el "*know how*". Debe poseer cualidades como las siguientes:

- Estar familiarizado con la especialidad.
- Tener experiencia real adquirida con la práctica
- Buena capacidad de introspección y de comunicar a otros sus conocimientos.
- Etc.

● ***Los usuarios***: Normalmente no son expertos, pueden ser aprendices del dominio o para algunos sistemas puede ser cualquier persona sin conocimientos.

● ***La administración:*** Patrocinadores del proyecto:

- Identifican el problema.
- Aprueban la realización del proyecto.
- Asignan los recursos.

Adicionalmente, se puede incorporar al grupo de desarrollo un "*aprendiz ingeniero del conocimiento*", con el fin de formar personal en diseño de sistemas expertos, y para apoyo al futuro mantenimiento del sistema desarrollado.

3.2.2 El Cuello de Botella de la Adquisición de Conocimiento.

El termino cuello de botella se utiliza para hacer referencia al hecho de que la adquisición de conocimiento es el punto que plantea una mayor dificultad a la hora de crear una base de conocimiento. Las causas principales de esta dificultad radican en los siguientes problemas:

1. El problema del conocimiento tácito. Aunque no se sabe exactamente como el conocimiento implícito es codificado psicológicamente en la memoria humana, se da por hecho que la resolución de problemas por parte de los expertos involucra el uso de este tipo de conocimiento. Por lo tanto, no es de extrañar que el conocimiento tácito sea el objeto de estudio de la adquisición de conocimiento.

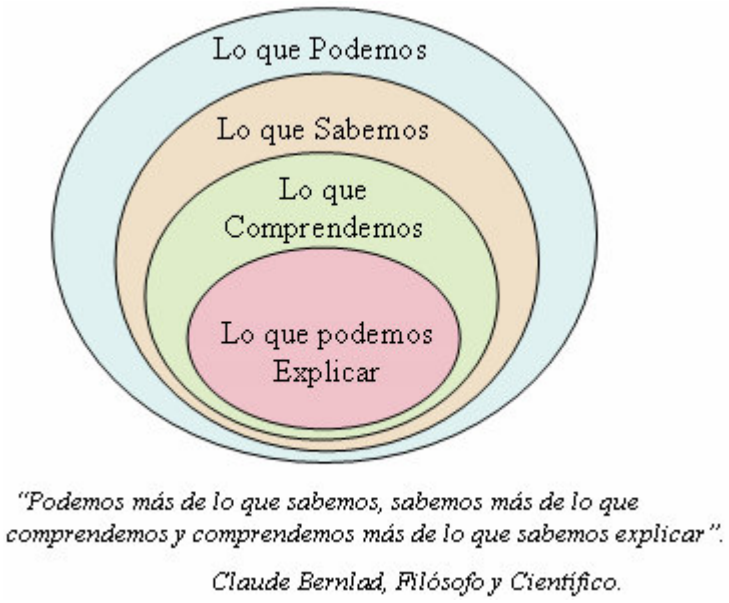

Figura 3.3 El problema del conocimiento implícito.

Además, a medida que las personas se vuelven mas experimentadas en su área de conocimiento y aplican repetidamente su conocimiento declarativo a determinadas tareas, éste se vuelve oculto, perdiendo de esta forma conciencia de lo que realmente saben. Consecuentemente, el conocimiento que es más idóneo para su incorporación a un SBC, y por lo tanto el objeto de la adquisición de conocimiento, es el tipo de conocimiento sobre el que los expertos tienen menos conciencia, con lo cual su adquisición se hace una tarea bastante compleja.

2. El problema de la comunicación. Este problema radica en que los expertos en una determinada área de aplicación y los ingenieros del conocimiento no utilizan el mismo lenguaje para comunicarse. Para poder representar el conocimiento relevante en un determinado tema, el ingeniero del conocimiento se debe familiarizar con el domino de la aplicación, debe de aprender el vocabulario utilizado en dicha área, y quizás, una nueva forma de ver los problemas.

Por lo tanto, el ingeniero del conocimiento debe esforzarse en comprender el área sobre la que el experto va a hablar, o puede no entender lo que el experto trata de comunicar. Por otro lado, el problema de la comunicación no solo recae en la falta de conocimiento sobre el dominio que posee el ingeniero del conocimiento, sino que también se debe al hecho de que el experto, en la mayor parte de los casos, carece de conocimientos de programación, y por lo tanto, no sabe que conocimiento es el idóneo para ser codificado en el ordenador.

3. El problema de utilizar representaciones del conocimiento. Además de las barreras lingüísticas y cognitivas, otro componente adicional al cuello de botella de la adquisición del conocimiento radica en el lenguaje elegido para la codificación del mismo, ya que los lenguajes usados para codificar el conocimiento carecen generalmente de la suficiente potencia expresiva. Este aspecto es bastante importante, existen experimentos que demuestran que la forma en la que es modelado el conocimiento del experto, depende fuertemente del lenguaje de representación utilizado. Por lo tanto, las primitivas ofrecidas por las herramientas de construcción de SBC tienen una influencia bastante fuerte sobre la forma de adquirir el conocimiento del dominio en estudio.

Estos problemas se hicieron cada vez más evidentes a medida que se intentaba abordar la construcción de SBC más complejos, dando lugar a que las funcionalidades esperadas no se

cumplieran y que tampoco se cumplieran con los plazos establecidos para el desarrollo. Esto evidenciaba el hecho de que se había llegado a un punto, donde el tamaño de los problemas que se intentaban abordar, requería de unas metodologías de desarrollo más elaboradas de lo que existían en esos momentos.

Como solución a las cuestiones enumeradas surge la Ingeniería del Conocimiento (IC), que tiene como objetivo principal precisamente la investigación sobre los métodos para adquirir, elaborar y representar en el ordenador el conocimiento del experto humano, a través de la estrecha colaboración entre expertos y profesionales de la informática.

3.2.3 Fase de la adquisición.

Para cada ciclo de desarrollo existen distintas fases de adquisición. En la mayoría de las ocasiones, estas fases estarán relacionadas con el nivel de especificación de la información que sea necesario. En la figura 3.4 se muestran dichas fases de adquisición del conocimiento, los trabajos más importantes en cada fase, las tareas específicas a realizar durante cada fase y una estimación del trabajo a realizar por el ingeniero del conocimiento y el experto [Schreiber01].

- ***Fase I:*** Los trabajos de adquisición del conocimiento son por naturaleza más generales y cuando se completan, proporcionan una base sobre la que estructurar el conocimiento que se irá adquiriendo.

En esta primera fase, el trabajo realizado depende en un 80% del ingeniero del conocimiento, y tan sólo un 20% del experto. En la primera etapa el ingeniero debe familiarizarse con el dominio, para el cual se realiza básicamente lo siguiente:

- Se conviene en mantener una información centralizada.
- Se elabora un glosario con los términos del dominio.
- Se intercambian conocimientos entre los miembros del grupo.

- ***Fase II:*** Entre las tareas a realizar en la segunda fase se incluyen el desarrollo de un diccionario de los términos y conceptos específicos del dominio. En esta etapa las técnicas apropiadas son las entrevistas no estructuradas y los análisis de tareas. El ingeniero del conocimiento será el que más tenga que trabajar en esta fase.

- ***Fase III:*** A medida que la información extraída sea más específica y el desarrollo entre en la tercera fase, las técnicas más apropiadas serán el análisis de protocolos, seguido de entrevistas estructuradas.

- ***Fase IV:*** Las tareas propias de la cuarta fase incluyen el que los expertos trabajen con una simulación o prototipo para refinar el conocimiento.

El primer producto de la Ingeniería del Conocimiento es un mapa de conocimientos que provee una vista del conocimiento del experto. Luego, se usa esa información para desarrollar la base de conocimientos. Una vez se ha introducido el conocimiento inicial en el sistema, se debe probar. Dependiendo de cómo el sistema se ejecuta, entonces se modifica, refina y expande el conocimiento hasta que el sistema alcanza un nivel predeterminado de pericia.

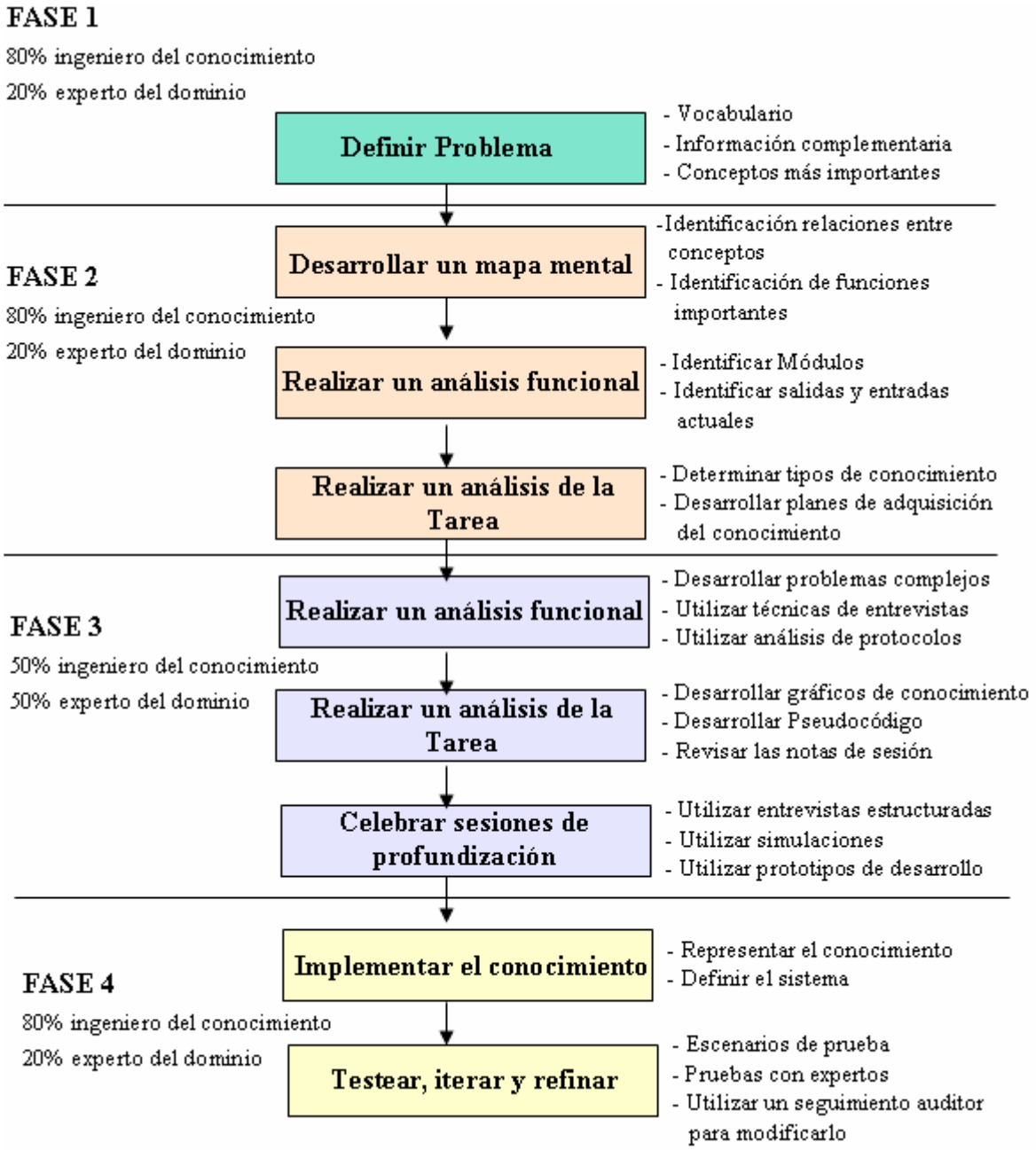

Figura 3.4 Fases de la Adquisición del conocimiento.

3.2.4 Las Fuentes del Conocimiento.

Para hablar de este proceso, es fundamental especificar que el conocimiento se encuentra *"guardado"* en diversas partes, denominadas fuentes de conocimiento. Para la Ingeniería del Conocimiento estas fuentes son de dos tipos:

- ***Fuente de conocimiento dinámica*** (*fuente primaria*): Refleja las características del conocimiento tales como, la variabilidad y el hecho de ser cambiante e inexacto, entre otras. El hombre forma parte de este tipo de fuente, en particular el experto.

- ***Fuente de conocimiento estática*** (*fuente secundaria*): Es rígida en cuanto a que su contenido, no se puede variar, por ejemplo un libro, una revista, un artículo, una película, etc.

La meta de la adquisición del conocimiento es entender cómo una persona lleva a cabo alguna actividad de modo que esa misma actividad pueda ser automatizada. Con esto, la adquisición del conocimiento se refiere a la labor de extracción del conocimiento de dichas fuentes. Dependiendo de su tipo, se siguen procedimientos distintos:

- *Adquisición del Conocimiento de una Fuente Estática.* El propósito es que el ingeniero del conocimiento y el experto puedan tener un vocabulario común para que logren una comunicación efectiva y eficiente. Se consigue cuando el ingeniero del conocimiento ha adquirido conocimientos del dominio del experto a través de los libros, revistas, etc., y cuando el experto del dominio a su vez, ha obtenido el conocimiento relacionado con las bases de los sistemas basados en el conocimiento. Esto es para que se pueda entender completamente el objetivo del proyecto y puedan realizar una labor apropiada.

Lo primero que se debe realizar es seleccionar las fuentes más apropiadas para adquirir el conocimiento del dominio y relacionadas con el problema. Se evalúan todos los recursos que se tengan disponibles bien sean del interior de la empresa o fuera de ella. Comúnmente, el experto en el dominio es quien aconseja cuáles son las fuentes a estudiar. Después de ello, se hace un estudio minucioso de dichas fuentes para que así el (los) ingeniero(s) del conocimiento pueda(n) adquirir ese conocimiento básico y fundamental del dominio del experto y poder realizar un proceso de adquisición eficiente y eficaz. Por último, se debe hacer una validación del conocimiento para saber si fue correcto o no lo que se extrajo.

- *Adquisición del Conocimiento de una Fuente Dinámica.* Esta labor se realiza una vez se ha adquirido el conocimiento básico del dominio por parte del (los) ingeniero(s) del conocimiento. La idea es que tanto el ingeniero del conocimiento como el experto, deben ser capaces de expresar el conocimiento, tanto profundo como superficial que se tiene acerca del dominio y de la solución de los problemas que se presenten en él.

El sistema basado en el conocimiento tratará de actuar como el experto humano cuando se está enfrentado a una situación en el dominio, en la cual se requiere tomar una decisión. Es importante expresar que otro de los objetivos del proceso de adquisición del conocimiento es precisar las actividades o procesos mentales que el experto realiza con su conocimiento, con el fin de llegar a una conclusión. Esto es una tarea ardua para el ingeniero del conocimiento, por ello debe llevarse a cabo con exactitud y precisión para que se pueda concretar el conocimiento heurístico del sistema, es decir las reglas de buen juicio usadas por el experto en el dominio.

Si con anterioridad se determinaron muy bien las bases conceptuales del dominio, se entendió el problema y se hicieron lecturas profundas de temas relacionados con el dominio, el ingeniero del conocimiento estará en capacidad de comprender la forma como el experto maneja su conocimiento.

3.2.5 Estrategias de Adquisición del Conocimiento.

Para hacer la adquisición del conocimiento de las fuentes dinámicas, hay diferentes estrategias. A continuación se presentan las más usuales:

- ***Entrevistas Directas o Formales:*** Consistente en la realización de conversaciones personales entre el ingeniero del conocimiento y la fuente del conocimiento, bien sea el experto o un usuario. El ingeniero del conocimiento establece un plan de la reunión en el que se determina el objetivo principal de la reunión, el tema a tratar, los recursos que se necesitan para registrar (guardar) la entrevista, la fecha, la hora y el lugar en donde se llevará a cabo dicha

entrevista. Este plan debe ser luego enviado a la persona que se va a entrevistar para que lo revise, lo corrija y lo apruebe, así tiene la oportunidad para prepararse con anterioridad.

- ***Entrevistas Informales:*** Se realiza en forma personal pero no planeada. Se aprovecha la oportunidad del encuentro entre el ingeniero del conocimiento y la persona que tiene el conocimiento, en donde el primero le hace una pequeña entrevista al segundo.

- ***Observaciones del trabajo real del Experto:*** Se denomina método de la observación. Es examinar la labor del experto en su ambiente de trabajo, solucionando un problema como el que se está tratando de simular. La ventaja del conocimiento que se adquiere en esta forma es que es muy espontáneo, ya que el experto está tomando las decisiones sin tener mucho tiempo para analizar el por qué de ellas. Además, no se le permite cuestionar si está haciendo lo correcto o no, solamente él hace lo que cree que es mejor en esa situación.

- ***Cuestionarios:*** Son encuestas muy bien diseñadas, que se utilizan especialmente para cuando se requiere obtener las ideas que tienen varias personas sobre el tema. Puede llegar a ser muy difícil de diseñar e inclusive, de manejar.

Cada una de estas estrategias posibilita la extracción del conocimiento. La elección de cuál utilizar y en qué momento, dependerá tanto del ingeniero del conocimiento como del experto.

3.2.6 Representación del Conocimiento.

Este proceso consiste en coger el conocimiento extraído de las fuentes estáticas y dinámicas y llevarlo a una forma entendible, primero por el ingeniero del conocimiento y luego por la herramienta de software que se vaya a utilizar [Truemper04].

La representación del conocimiento en general, *"es una serie de convenciones sintácticas y semánticas que hacen posible describir las cosas. En donde la sintaxis es aquello que especifica una serie de reglas para combinar símbolos de tal forma que se formen expresiones válidas. La semántica es la especificación de cómo tales expresiones son interpretadas".*

El proceso se realiza en paralelo con el de adquisición del conocimiento. Inicialmente la representación que hace el ingeniero del conocimiento la debe hacer en papel, por medio de la documentación, para que luego pueda ser llevada al ordenador. Cuando el ingeniero del conocimiento hace la adquisición del conocimiento lo va registrando de alguna forma, es así como comienza a realizar su representación. Después, de acuerdo con la forma elegida, lo lleva al lenguaje del ordenador, para que así quede reflejado en el software. Por lo tanto, el ingeniero del conocimiento debe conocer muy bien la herramienta de desarrollo.

Quizá lo más complejo de este proceso no es el conocer la herramienta o no, sino la elección de la forma más apropiada de representación interna del conocimiento en el sistema informático, según el problema y el experto. Para los sistemas de conocimiento se han determinado algunas representaciones que se han vuelto estándar para ello, como la lógica proposicional y la lógica de predicados, las reglas de producción, las redes semánticas, los marcos (frames), los guiones (scripts), los lenguajes orientados por objetos, las redes neuronales, etc.

El proceso de representación de conocimiento es de gran importancia, ya que podemos decir que se está construyendo la base de conocimientos del sistema. El siguiente apartado 3.3 está dedicado al estudio de las distintas técnicas mencionadas en el párrafo anterior.

3.2.7 Pruebas y Comprobación.

Mediante el proceso de ejecución se deben hacer todas los ensayos posibles para evitar el mal manejo del conocimiento, bien sea por problemas de interpretación de los hechos, las heurísticas, las relaciones o por problemas de obtención de malas conclusiones y explicaciones.

Después de representar el conocimiento, éste debe ser validado tanto por el ingeniero del conocimiento como por el experto del dominio. Siempre se debe asegurar que el conocimiento que se adquiere y que se represente sea igual al proporcionado por el experto. Básicamente, lo que se hace es evaluar el conocimiento del sistema por medio de pruebas de casos reales, con el fin de confrontarlos entre sí. Este proceso también se debe hacer durante la vida útil del sistema de conocimientos.

3.3 Técnicas de Representación del Conocimiento.

Para que un Sistema Experto pueda utilizar el conocimiento que se ha adquirido sobre un dominio concreto, será necesario disponer de una representación de este. Además de que en buena parte, la potencia del programa vendrá determinada por la eficacia y consistencia del esquema que se haya elegido. Una característica que está presente en todas las representaciones, es que de forma general se van ha manejar dos tipos de entidades:

- ***Hechos:*** Verdades en un cierto mundo, es aquello que se quiere representar.

- ***Representaciones de los hechos***: Son estructuras internas, entidades que los programas de Inteligencia Artificial van a manipular.

Para la manipulación computacional de "*hechos*", es necesario definir un conjunto de procedimientos que conviertan tales hechos en representaciones. Una vez ejecutado el programa de Inteligencia Artificial, de nuevo son necesarios procedimientos que conviertan las representaciones internas en hechos, comprensibles para nosotros. Véase la siguiente figura:

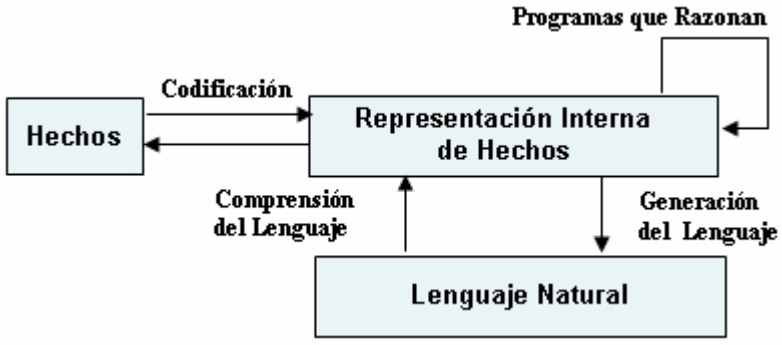

Figura 3.5 El ciclo de vida de codificación-decodificación

3.3.1 Criterios de Evaluación de la Representación.

Para que un sistema de representación sea efectivo debe poseer una serie de cualidades:

- *Suficiencia de la Representación:* capacidad para representar todos los tipos de conocimientos necesarios en el dominio.

- *Suficiencia Deductiva:* La capacidad para manipular las estructuras de la representación, con el fin de obtener nuevas estructuras que se correspondan con un nuevo conocimiento deducido a partir del antiguo.

- *Eficiencia Deductiva:* Capacidad de incorporar información adicional en las estructuras de conocimiento, con el fin de que los mecanismos de inferencia puedan seguir las direcciones más prometedoras.

- *Eficiencia en la Adquisición:* La capacidad de adquirir nueva información con facilidad. El caso más simple es aquel en que una persona inserta directamente el conocimiento en la base de datos. Idealmente, el programa seria capaz de controlar la adquisición del conocimiento por sí mismo.

Desgraciadamente no existe una forma de representación que optimice todos estos aspectos y que sea aplicable a cualquier tipo de conocimiento, en consecuencia existen múltiples técnicas para la representación del conocimiento. Muchos de los sistemas actuales no se decantan por una en concreto y pueden llegar a utilizar simultáneamente varias técnicas de representación.

3.3.2 Esquemas de representación del conocimiento.

Las técnicas de representación del conocimiento se pueden clasificar en dos categorías.

● ***Métodos Declarativos:*** Hacen énfasis en la representación del conocimiento como una acumulación de hechos estáticos, a los que se añade cierta información limitada que describe como se va ha emplear el conocimiento. Es decir, se especifica el conocimiento, pero no la manera de utilizarlo. Por tanto existirá la necesidad de un programa que indique que hacer y de qué modo.

En esta categoría se engloban las denominadas *"Formas estructuradas"*, donde el conocimiento se representa como una colección estática de hechos para cuya manipulación se define un conjunto genérico y restringido de procedimientos. Los esquemas de este tipo presentan ciertas ventajas. Así, las verdades del dominio se almacenan una sola vez y además, es fácil incrementar e incorporar nuevo conocimiento sin modificar ni alterar el ya existente. Las propiedades que deben cumplir:

- *Adecuación Representacional:* El esquema elegido debe ser capaz de representar las distintas clases del dominio.

- *Adecuación Inferencial:* El esquema debe permitir la manipulación del conocimiento para obtener conocimiento nuevo.

- *Eficiencia Inferencial:* El esquema debe ser versátil utilizando aquella información que permita optimizar el proceso inferencial.

- *Eficacia Adquisicional:* El esquema debe suministrar vías que permitan la incorporación de información y nuevos conocimientos.

Algunos de los métodos declarativos existentes:

- *Redes Semánticas:* Permiten describir simultáneamente acontecimientos y objetos.

- *Los modelos de Dependencia Conceptual:* Estructuras especializadas que proporcionan mecanismos para representar relaciones entre los componentes de una acción.

- *Los Marcos*: Estructuras genéricas que permiten representar objetos completos, desde diferentes puntos de vista.

- *Los Guiones:* estructuras especializadas que derivan de los Marcos y que son útiles para representar secuencias comunes de acontecimientos.

- *Representación Orientada a Objetos.*

● ***Métodos Procedimentales:*** Enfatizan en la representación del conocimiento en forma de estructuras dinámicas, que describen procedimientos de utilización de los conocimientos. La información de control necesaria para utilizar el conocimiento, se encuentra embebida en el propio conocimiento. Para utilizarla, hace falta un intérprete que siga las instrucciones dadas por el conocimiento.

Esta categoría se suele englobar en lo que se podría llamar la "*Lógica Formal*", de la cual forma parte la Lógica Proposicional, Lógica de predicados y las Reglas de Producción, entre otras. Se le dará una especial relevancia a las reglas de producción, por ser la técnica de representación utilizada para el desarrollo de nuestras investigaciones.

3.3.3 Representación mediante Reglas de Producción.

Los sistemas basados en reglas son los más comúnmente utilizados. Su simplicidad y similitud con el razonamiento humano, han contribuido para su popularidad en diferentes dominios. Las reglas son un importante paradigma de representación del conocimiento.

Las reglas representan el conocimiento utilizando un formato *SI-ENTONCES (IF-THEN)*, consta de dos partes:

- La parte *SI (IF)*, es el antecedente, premisa, condición o situación.
- La parte *ENTONCES (THEN)*, es el consecuente, conclusión, acción o respuesta.

Las reglas pueden ser utilizadas para expresar un amplio rango de asociaciones, por ejemplo:

***SI** está manejando un vehículo **Y** se aproxima una ambulancia**, ENTONCES** baje la velocidad **Y** hágase a un lado para permitir el paso de la ambulancia.*

***SI** hay una elevada tasa de errores en la línea, **ENTONCES** puede haber un fallo en la conexión.*

***SI** el drenaje del lavabo está tapado **Y** la llave de agua está abierta, **ENTONCES** se puede inundar el piso.*

3.3.3.1 Inferencia Basada en Reglas.

Una declaración de que algo es verdadero o es un hecho conocido, es una *afirmación.* El conjunto de afirmaciones se conoce a menudo con el nombre de *memoria de trabajo o base de afirmaciones*. De igual forma, al conjunto de reglas se lo denomina *Base de Reglas* [Padgman01].

Un sistema basado en reglas utiliza el *"Modus Ponens"* para manipular las afirmaciones y las reglas durante el proceso de inferencia. Mediante técnicas de búsqueda y procesos de

unificación, los sistemas basados en reglas automatizan sus métodos de razonamiento y proporcionan una progresión lógica desde los datos iniciales, hasta las conclusiones deseadas. Esta progresión hace que se vayan conociendo nuevos hechos o descubriendo nuevas afirmaciones, a medida que va guiando hacia la solución del problema, el siguiente apartado 3.4 se estudiará esta y otras técnicas de búsqueda y resolución de problemas.

En consecuencia, el proceso de solución de un problema en los sistemas basados en reglas, va realizando una serie de inferencias que crean un sendero entre la definición del problema y su solución. Las inferencias están concatenadas y se realizan en forma progresiva, por lo que se dice que el proceso de solución origina una *cadena de inferencias*. Los sistemas basados en reglas difieren de las otras técnicas procedimentales, en las siguientes características principales:

- Son en general no-monotónicos, es decir hechos o afirmaciones derivadas, pueden ser retractados en el momento en que dejen de ser verdaderos.

- Pueden aceptar incertidumbre en el proceso de razonamiento.

3.3.3.1.1 Metareglas.

El conocimiento acerca de las reglas de producción se denomina *metaregla,* se utilizan para facilitar y acelerar la búsqueda de soluciones. Son reglas que permiten controlar el razonamiento del problema (meta-reglas), diferenciándose de las reglas que resuelven el problema (reglas a nivel objeto). Un ejemplo de metaregla:

Si (red conexión_ fallida)
entonces (reglas de_establecimiento_de_conexión)

Estas meta-reglas pueden ser de dos tipos:

- *Específicas para el dominio:* Sólo se aplican a un problema en particular.
- *Independientes del dominio:* Se aplican a cualquier problema o a un tipo de problemas.

Es fácil de intuir que las segundas son más difíciles de encontrar, existiendo actualmente vías de investigación para encontrar metareglas o formas de razonamiento que se apliquen a clases de problemas de diagnóstico, diseño, planificación, etc. A continuación dos nuevos ejemplos que ayudan a su comprensión:

Si (red fallos múltiples)
Entonces (reglas fallos mayor_gravedad)

En general:

Si (objeto problemas múltiples)
Entonces (reglas problemas mayor_prioridad)

3.3.3.2 El Proceso de Razonamiento.

El proceso de razonamiento en un sistema basado en reglas, es una progresión desde un conjunto inicial de afirmaciones y reglas hacia una solución, respuesta o conclusión. Como se llega a obtener el resultado, sin embargo, puede variar significativamente:

- Se puede partir considerando todos los datos conocidos y luego ir progresivamente avanzando hacia la solución. A este proceso se lo conoce con el nombre de "guiado por los datos" y se denomina de *encadenamiento hacia adelante o progresivo* (forward chainning).

- Se puede seleccionar una posible solución y tratar de probar su validez buscando evidencia que la apoye. Este proceso es conocido como "guiado por el objetivo" y denominado de *encadenamiento hacia atrás o regresivo* (backward chainning).

3.3.3.2.1 Razonamiento Hacia Delante.

En el caso del razonamiento hacia delante, se empieza a partir de un conjunto de datos recolectados a través de observación y se evoluciona hacia una conclusión. Se chequea cada una de las reglas para ver si los datos observados satisfacen las premisas de alguna de las reglas. Si una regla es satisfecha, es ejecutada derivando nuevos hechos que pueden ser utilizados por otras reglas para derivar hechos adicionales. Este proceso de chequear reglas para ver si pueden ser satisfechas, se denomina interpretación de reglas.

La interpretación de reglas es realizada por una máquina de inferencia en un sistema basado en conocimiento. La interpretación de reglas o inferencia, en el razonamiento progresivo involucra la repetición de los pasos que se indican en la siguiente figura 3.6.

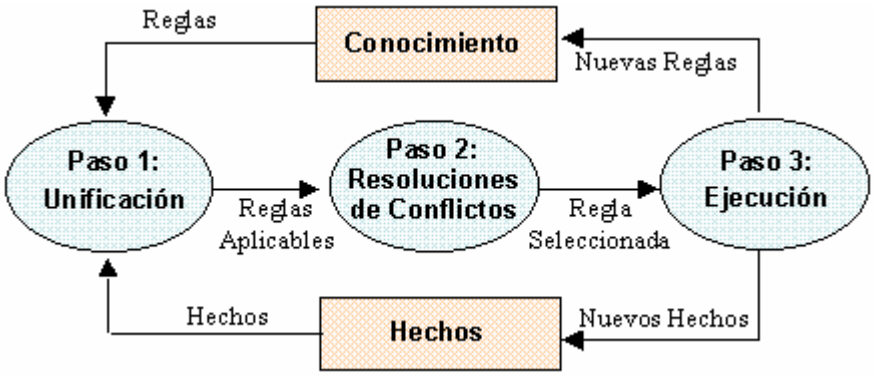

Figura 3.6 Proceso de Razonamiento hacia delante.

- *Unificación (Matching):* Inicialmente en las reglas de la base de conocimientos, se prueban los hechos conocidos al momento para ver cuáles son las que resultan satisfechas. Para decir que una regla ha sido satisfecha, se requiere que todas las premisas o antecedentes de la regla resuelvan a verdadero.

- *Resolución de Conflictos:* Es posible que en la fase de unificación resulten satisfechas varias reglas. La resolución de conflictos involucra la selección de la regla que tenga la más alta prioridad de entre el conjunto de reglas que han sido satisfechas, (Estrategias de Resolución de Conflictos estudiadas en el apartado D.4 del anexo D.

- *Ejecución:* El último paso en la interpretación de reglas es la ejecución de la regla. La ejecución puede dar lugar a uno o dos resultados posibles: nuevo hecho (o hechos) pueden ser derivados y añadidos a la base de hechos, o una nueva regla (o reglas) pueden ser añadidas al conjunto de reglas (base de conocimiento) que el sistema considera para ejecución. En esta forma, la ejecución de las reglas procede de una manera progresiva (hacia adelante) hacia los objetivos finales.

Un conjunto de aplicaciones adecuadas al razonamiento progresivo incluye supervisión y diagnóstico en sistemas de control de procesos en tiempo real, donde los datos están

continuamente siendo adquiridos, modificados y actualizados. Estas aplicaciones tienen dos características importantes:

1.- Necesidad de respuesta rápida a los cambios en los datos de entrada.

2.- Existencia de pocas relaciones predeterminadas entre los datos de entrada y las conclusiones derivadas.

Otro conjunto de aplicaciones adecuadas para el razonamiento progresivo está formado por: diseño, planeamiento y temporización, donde ocurre la síntesis de nuevos hechos basados en las conclusiones de las reglas. En estas aplicaciones hay potencialmente muchas soluciones que pueden ser derivadas de los datos de entrada. Debido a que estas soluciones no pueden ser enumeradas, las reglas expresan conocimiento como patrones generales y las conexiones precisas entre estas reglas, no pueden ser predeterminadas.

3.3.3.2.2 Razonamiento Hacia Atrás.

Este tipo de mecanismo de inferencia o intérprete de reglas, difiere significativamente del mecanismo de razonamiento hacia delante. Si bien es cierto ambos procesos involucran el examen y aplicación de reglas, el razonamiento hacia atrás empieza con la conclusión deseada y decide si los hechos que existen pueden dar lugar a la obtención de un valor para esta conclusión. Sigue un proceso muy similar a la búsqueda primero en profundidad (apartado 3.4.5.2).

El sistema empieza con un conjunto de hechos conocidos que típicamente está vacío. Se proporciona una lista ordenada de objetivos (o conclusiones), para las cuales el sistema trata de derivar valores. El proceso de razonamiento regresivo utiliza esta lista de objetivos para coordinar su búsqueda a través de las reglas de la base de conocimientos. Esta búsqueda consiste en los siguientes pasos:

1. Conformar una pila inicialmente compuesta por todos los objetivos prioritarios definidos en el sistema.

2. Considerar el primer objetivo de la pila. Determinar todas las reglas capaces de satisfacer este objetivo, es decir aquellas que mencionen al objetivo en su conclusión.

3. Para cada una de estas reglas examinar en turno sus antecedentes:

   a.- Si todos los antecedentes de la regla son satisfechos (esto es, cada parámetro de la premisa tiene su valor especificado dentro de la base de datos), entonces ejecutar esta regla para derivar sus conclusiones. Debido a que se ha asignado un valor al objetivo actual, removerlo de la pila y retornar al paso (2).

   b.- Si alguna premisa de la regla no puede ser satisfecha, buscar reglas que permitan derivar el valor especificado para el parámetro utilizado en esta premisa.

   c.- Si en el paso (b) no se puede encontrar una regla para derivar el valor especificado para el parámetro actual, entonces preguntar al usuario por dicho valor y añadirlo a la base de datos. Si este valor satisface la premisa actual entonces continuar con la siguiente premisa de la regla. Si la premisa no es satisfecha, considerar la siguiente regla.

Si todas las reglas que pueden satisfacer el objetivo actual se han probado y todas no han podido derivar un valor, entonces este objetivo quedará indeterminado. Removerlo de la pila y retornar al paso (2). Si la pila está vacía parar y anunciar que se ha terminado el proceso.

El razonamiento hacia atrás es mucho más adecuado para aplicaciones que tienen mucho mayor número de entradas, que de soluciones posibles. La habilidad de la lógica regresiva para trazar desde las pocas conclusiones hacia las múltiples entradas la hace más eficiente que el encadenamiento hacia delante. Una excelente aplicación es el diagnóstico, donde el usuario dialoga directamente con el sistema basado en conocimiento y proporciona los datos a través del teclado. Los problemas de clasificación también son adecuados para ser resuelto mediante el razonamiento hacia atrás.

3.3.3.3 Arquitecturas basadas en Reglas.

El tipo de conocimiento descrito con sistemas basados en reglas varía significativamente en complejidad. Algunas veces las conclusiones derivadas de las reglas pueden ser "hechos" que se identifican en forma exacta con las premisas de otras reglas. En estos casos, se puede visualizar una base de conocimientos como una red de reglas y hechos interconectados. En otros casos, las conclusiones derivadas pueden ser más generales. Como resultado, la visualización de la base de conocimiento como una red, no es posible aplicarla. En lugar de esto, nos vemos forzados a pensar que las conclusiones derivadas de las reglas, son una colección de hechos que podrían o no unificarse o identificarse con los patrones descritos por las premisas de otras reglas [Giarratano01].

Esto da como resultado dos tipos de estructuras y organizaciones al conocimiento contenido dentro de un sistema basado en reglas: redes de inferencia y sistemas de unificación de patrones. Cabe señalar que ambas arquitecturas pueden trabajar con encadenamiento progresivo o regresivo. Sin embargo, tradicionalmente se ha utilizado el proceso de razonamiento regresivo en redes de inferencia y el proceso de razonamiento progresivo en sistemas de unificación de patrones.

3.3.3.3.1 Redes de Inferencia.

Una red de inferencia puede ser representada como un gráfico en el que los nodos representan parámetros que son los hechos obtenidos como datos o derivados de otros datos. Cada parámetro es una declaración acerca de algún aspecto del problema bajo análisis y puede servir como un antecedente o consecuente de una regla. Estas declaraciones pueden copar un rango que va desde la conclusión final de un sistema, hasta hechos simples, observados o derivados. Cada uno de estos parámetros puede tener uno o más valores asociados, donde cada valor tiene una medida correspondiente de incertidumbre que representa, cuan creíble es el valor particular de un parámetro.

Las reglas en el sistema están representadas dentro del gráfico por las interconexiones entre los distintos nodos. Este conocimiento es utilizado por el proceso de inferencia, para propagar resultados a través de la red. Nótese que todas las interconexiones entre los nodos de la red de inferencia, son conocidas previamente a la ejecución del sistema. Esto tiene como consecuencia, la minimización del proceso de búsqueda de hechos que se identifiquen con las premisas. Adicionalmente, simplifican la implementación del mecanismo de inferencia y el manejo de las facilidades de explicación.

Las redes de inferencia son muy útiles para dominios donde el número de diferentes soluciones alternativas es limitado. Por ejemplo, la clasificación de elementos y problemas de

diagnóstico. Una red de inferencia es fácil de implementar, pero es menos poderosa ya que se debe conocer de antemano todas las relaciones entre reglas y hechos. Sistemas comerciales de desarrollo basados en esta arquitectura son los siguientes: Personal Consultant, EXSYS y VP-Expert, por ejemplo.

3.3.3.3.2 Sistemas de Unificación de Patrones.

Estos sistemas utilizan procesos de búsqueda extensivos para unificar y ejecutar las reglas. Típicamente usan complejas implementaciones de asociación de patrones para asignar valores a variables, condicionar los valores permisibles para ser asociados a una premisa y para determinar las reglas a ejecutar. Las relaciones entre las reglas y los hechos se forman durante la ejecución basadas en los patrones que se identifican con los hechos.

La unificación de patrones es una idea importante y poderosa en razonamiento automatizado que fue utilizada por primera vez en el lenguaje PROLOG. Los sistemas basados en reglas que utilizan la unificación de patrones son extremadamente flexibles y poderosos. Son más aplicables a dominios en los que las posibles soluciones son ilimitadas o muy grandes en número, tales como diseño, planeamiento y síntesis. Sin embargo, el uso de procesos de búsqueda para encontrar reglas aplicables en los sistemas de unificación de patrones, los puede volver ineficientes en implementaciones grandes.

Comercialmente existen varios sistemas de desarrollo basados en esta arquitectura: XCON, OPS-5, ART, CLIPS, KEE, etc.

3.3.3.4 Ventajas y Desventajas de las Reglas de Producción.

Los problemas existentes en los sistemas basados en reglas, estarán dentro de una de las siguientes categorías:

- Encadenamiento infinito.
- Incorporación de conocimiento nuevo contradictorio.
- Modificación de reglas existentes.

Desventajas adicionales pueden ser

- *Ineficiencia:* necesidad de modularizar o de introducir metarreglas.
- *Opacidad:* dificultad de establecer relaciones.
- *Adaptación al dominio:* rápido crecimiento del número de reglas.

A pesar de las desventajas anotadas, los sistemas basados en reglas han permanecido como los esquemas más comúnmente utilizados para la representación del conocimiento. Como ventajas significativas se pueden mencionar la modularidad, uniformidad y naturalidad para expresar el conocimiento

3.4 Problemas y Técnicas de Búsqueda.

A la hora de la creación de sistemas programables utilizando las técnicas tradicionales de desarrollo, se comprobó la existencia de problemas que caen dentro de los límites de la Inteligencia Artificial. Son demasiado complejos para resolverlos mediante técnicas directas; es mejor intentar resolverlos mediante procedimientos adecuados de búsqueda, apoyadas por cualquier técnica directa que esté disponible para guiar la búsqueda.

En este apartado se describen métodos de búsqueda y se explican algunas técnicas de búsqueda de propósito general. Se explicará cómo actúan los agentes mediante la definición de

metas y la consideración de secuencias de acciones que les permiten alcanzarlas. Una meta y su correspondiente conjunto de medios que permiten lograrlo se denomina *"Problema"*. Al procedimiento de explotación para determinar qué es lo que se puede obtener mediante los anteriores medios se denomina *"Búsqueda"*.

Para construir un sistema básico de inteligencia artificial capaz de resolver un problema específico, es necesario realizar una seria de acciones previas:

- *Definir de una forma precisa el problema:* incluyendo especificaciones de las condiciones iniciales y de las situaciones finales que pueden considerarse como soluciones aceptables al problema.

- *Analizar el problema:* puede darse el caso de que unos muy pocos rasgos importantes, puedan tener un gran impacto en la identificación de la técnica más apropiada para resolver el problema.

- *Identificar y representar el conocimiento:* para que es necesario para resolver el problema.

- *Escoger la mejor técnica:* para aplicarla a la resolución del problema.

En los siguientes apartados se tratan los fundamentos sobre los que se basan cada una de las acciones indicadas.

3.4.1 Definición del Problema.

El primer paso hacia el diseño de un programa que resuelva un problema en IA debe ser la creación de una forma descriptiva formal y manipulable del problema, a partir de la descripción informal del mismo [Friant00].

La definición del problema como una búsqueda en el espacio de estados forma la base de la mayoría de los métodos que se utilizan para la solución de problemas en IA. Los estados del sistema o descripción de estados, son representaciones que contienen el conjunto de toda la información que describe la situación actual del sistema. En cambio, el espacio de estado del sistema o espacio del problema, es el dominio que contiene todos los posibles estados del sistema. El espacio de estados puede ser finito o infinito.

Los mecanismos que se emplean para modificar o transformar un estado del sistema, toman el nombre de operadores, producciones o acciones y se utilizan para enlazar un estado actual con otro estado objetivo. Cuando existen varias posibles soluciones, representadas por diversas secuencias de operadores que enlazan dos estados, es necesario desarrollar algoritmos de IA que sean capaces de identificar las mejores secuencias.

En general, el posible número de secuencias de operadores a ser exploradas en el desarrollo de una solución puede ser muy grande, por lo que los algoritmos que se empleen no deben requerir la enumeración de todas las secuencias posibles. Esto sugiere el empleo de un proceso de búsqueda para tratar de encontrar una solución aceptable. La búsqueda es un proceso de gran importancia en la resolución de problemas difíciles para los que no se dispone de técnicas más directas. Los procesos de búsqueda están cercanamente relacionados con los procesos de optimización.

3.4.1.1 Características.

En resumen, la representación como espacio de estados ofrece una estructura que permite:

- Definir formalmente el problema, al poder convertir alguna situación dada en una situación deseada, utilizando un conjunto de operaciones permitidas.

- Definir la resolución de un problema como la combinación de dos componentes:

  - Un conjunto de operadores que, al modificar o transformar un estado, representan movimiento en el espacio del problema.

  - Un proceso de búsqueda, que explorando el espacio, intenta encontrar alguna ruta desde el estado actual hasta un estado objetivo.

3.4.2 Descripción del problema.

Al proceso que se encarga de convertir una descripción informal, en una descripción formal del problema, se lo denomina *Operacionalización*. Los pasos a seguir son los siguientes:

- Definir el espacio de estado que contiene todas las configuraciones posibles de los objetos relevantes. Esto es posible, sin necesidad de enumerar en forma explícita todos los estados que contiene.

- Especificar uno o más estados dentro de ese espacio que correspondan a posibles situaciones desde donde el proceso de resolución pueda arrancar, *Estados Iniciales*.

- Especificar uno o más estados que podrían ser aceptables como soluciones al problema, *Estados Objetivos*.

- Especificar un conjunto de operadores que describan las acciones posibles. Para esto se debe considerar los siguientes aspectos:

  - ¿Qué suposiciones implícitas están presentes en la descripción informal del problema?

  - ¿Qué generalidad deben tener los operadores?

  - ¿Qué cantidad del trabajo requerido para resolver el problema debería estar incluido y representado en los operadores?

El problema puede ser resuelto utilizando el conjunto de operadores, en combinación con una estrategia de control apropiada, para moverse dentro del espacio de estados del problema hasta encontrar un sendero entre el estado inicial y el estado objetivo. Como se indicó anteriormente, el mecanismo de búsqueda es fundamental para el proceso de solución del problema. Adicionalmente, proporciona un marco donde pueden intercalarse métodos más directos de resolución de partes del problema, en caso de ser esto posible.

3.4.3 Análisis del Problema.

Para poder escoger el o los métodos más apropiados para resolver un problema, es necesario analizarlo en algunos aspectos claves y buscar las respuestas necesarias a las siguientes preguntas:

- ¿Puede el problema ser descompuesto en un conjunto de subproblemas pequeños y posiblemente, independientes?

- ¿Podrían ignorarse pasos de solución o ser corregidos si resultaran inútiles?

- ¿Es posible predecir el o los resultados del problema?

- ¿Una buena solución al problema es suficientemente obvia sin necesidad de compararla con otras posibles soluciones?

- ¿La solución deseada es un estado o una ruta desde un estado inicial hasta un estado objetivo?

- ¿Es absolutamente necesaria toda una cantidad de conocimiento para resolver el problema, o es importante sólo para restringir la búsqueda?

- ¿Puede un ordenador al que se le ha dado el problema retornar por sí sólo la solución, o será necesario que haya una interacción entre el sistema y una persona?

Las respuestas que se den a las preguntas planteadas no sólo afectan a la definición del problema en sí mismo, sino también a las características de la solución deseada y a las circunstancias bajo las cuales debe darse la solución.

3.4.4 Representación.

En general, una representación es un conjunto de convenciones sobre la forma de describir algún tipo de cosa. El hallar una representación apropiada es una parte fundamental de la resolución de un problema. El principio de la representación establece que "*Una vez que un problema es descrito mediante una buena representación, el problema está casi resuelto".*

3.4.4.1 Descripciones Explícitas.

La descripción explícita de una buena representación está caracterizada por los siguientes aspectos importantes:

- Hace explícitos los objetos y las relaciones de importancia: de una sola mirada se puede apreciar lo que sucede.

- Pone de manifiesto las restricciones inherentes al problema.

- Agrupa los objetos y las relaciones.

- Suprime los detalles insignificantes.

- Es transparente, se puede entender lo que representa.

- Es completa, contiene todo lo que es necesario que debe expresar.

- Es concisa, expresa todo lo necesario con eficacia.

3.4.4.2 Aspectos Fundamentales.

Las representaciones tienen cuatro ingredientes fundamentales:

- El léxico, que determina los símbolos que están permitidos en el vocabulario de la representación.

- Una parte estructural que describe las restricciones sobre la forma en que los símbolos pueden ordenarse.

- Una parte operativa que especifica los procedimientos de acceso que permiten crear descripciones, así como la forma de modificarlas y utilizarlas para responder preguntas.

- Una parte semántica que establece una forma de asociar el significado con las descripciones.

5.4.4.3 Descripciones Implícitas.

En general, una red semántica de cualquier tipo no necesariamente puede presentar todos sus estados y transiciones. Puede haber ocasiones en las que es preferible no trabajar con la representación completa o en las que es prácticamente imposible obtenerla. En estos casos se desarrolla sólo la sección necesaria, partiendo de un estado inicial y generando los siguientes estados, según se los requiera. Una descripción de red semántica que está compuesta por un estado inicial y un mecanismo para generar los estados sucesores, se dice que es una descripción implícita del espacio de estado.

3.4.5 Técnicas de Solución.

Las técnicas de solución de problemas en IA, en general, incorporan un proceso de búsqueda. Todo proceso de búsqueda puede ser visualizado como el recorrido por un árbol en el que cada nodo representa un estado y cada rama representa las relaciones entre los estados cuyos nodos conecta.

En general, las reglas contienen en forma implícita el árbol y se genera en forma explícita sólo aquellas partes que se decide explorar. Las principales diferencias que pueden aparecer en las diferentes técnicas de búsqueda, son:

- La dirección en la cual se conduce la búsqueda (hacia adelante o hacia atrás).

- La estrategia de control, o forma de seleccionar las reglas que pueden ser aplicables. Los principales requerimientos de una buena estrategia de control son: que cause desplazamiento en el espacio de estado, y que sea sistemático.

- La forma de representar cada nodo del proceso de búsqueda (representación del conocimiento). Muchas veces, tratar el proceso como búsqueda en un grafo en lugar de una búsqueda en un árbol, puede reducir el esfuerzo que se gasta en explorar senderos, esencialmente iguales, varias veces. Algunos requisitos asociados, son:

- Cada vez que se genere un nodo se debe chequear para ver si ha sido generado antes.

- Se deben introducir procedimientos especiales para que la búsqueda no quede atrapada en algún lazo.

A continuación se describen los algoritmos de tres procesos básicos de búsqueda de soluciones en el espacio de estado.

3.4.5.1 Algoritmo Primero a lo Ancho (BREATH-FIRST).

Consiste en revisar todas las trayectorias de una determinada longitud antes de crear una trayectoria más larga. Descripción del procedimiento:

1.- Crear una variable NODE_LIST y asignarle el estado inicial.

2- Hasta que se encuentre el objetivo o NODE_LIST esté vacía, hacer:
- Eliminar el primer elemento de NODE_LIST, y llamarlo E.
  Si NODE_LIST está vacía, salir.

- Para cada regla emparejada con el estado E, hacer:
  - Aplicar la regla para generar un nuevo estado.
  - Si nuevo estado es objetivo, salir y devolver este estado.
  - Sino, añada el nuevo estado al final de NODE_LIST.

De forma gráfica se puede ver reflejado en la siguiente figura:

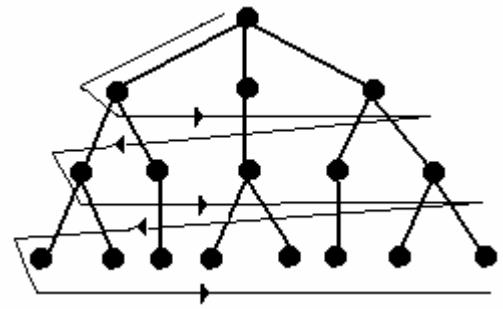

Figura 3.7 Procedimiento de búsqueda primero a lo ancho.

3.4.5.2 Algoritmo Primero en Profundidad (DEPTH-FIRST).

En cada nivel, a lo largo del proceso de búsqueda, se selecciona un único nodo para ser expandido, por tanto como diferencia con el anterior sólo se considera un único camino al ir avanzando en la búsqueda. El proceso a seguir es el siguiente:

1. Si el estado inicial es el objetivo, salir y devolver éxito.

2.- Sino, haga lo siguiente hasta que se obtenga éxito o fracaso:
- Generar sucesor E del estado inicial. Si no hay sucesores devolver fracaso.
- Llame recursivamente al algoritmo, esta vez con E como el estado inicial.
- Si la señal es éxito, devolver, de otra manera, continuar con el ciclo.

Visto de forma gráfica:

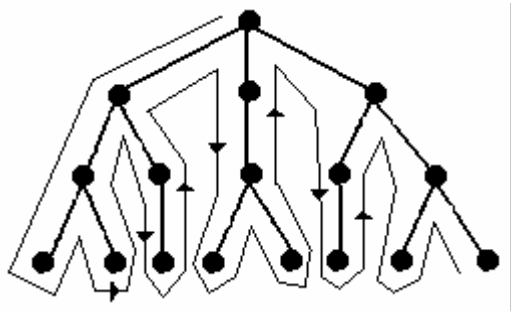

Figura 3.8 Procedimiento de búsqueda primero en profundidad.

3.4.5.3 Algoritmo Generación y Prueba (GENERATE-AND-TEST).

La estrategia de generación y prueba es la más sencilla de todas, consiste en realizar los siguientes pasos:

1. Generar una posible solución (estado o camino).
2. Comprobar para ver si es una solución, mediante comparación con los elementos del conjunto de objetivos aceptables.
3. Si la solución ha sido encontrada salir, de otra manera, retornar al paso 1.

Este algoritmo utiliza el procedimiento de búsqueda primero en profundidad, ya que las soluciones completas deben generarse antes de que sean comprobadas.

3.4.6 Búsqueda Heurística.

Para resolver muchos problemas difíciles (explosión combinatoria), es necesario muchas veces llegar a un compromiso de los requerimientos de movilidad, operatividad y construir una estructura de control que no necesariamente garantiza el encontrar la mejor respuesta, sino que casi siempre encuentra una buena respuesta. La técnica heurística mejora la eficiencia del proceso de búsqueda. Las consideraciones que sirven de soporte a un proceso de búsqueda heurística, son:

- Rara vez se requiere en realidad una solución óptima. Una buena aproximación, normalmente, sirve muy bien.

- A pesar que una aproximación heurística no puede resultar muy buena en el peor de los casos, raras veces aparecen los peores casos en la práctica.

- El tratar de comprender por qué un heurístico funciona o por qué no funciona, a menudo conduce a una mejor comprensión del problema.

Las técnicas heurísticas de búsqueda son como guías de turismo. Buenas, en el sentido que señalan aspectos de gran interés general, pero malas ya que pueden no satisfacer aspectos de interés particular. Existen varias técnicas heurísticas buenas de propósito general que son útiles para una diversidad de problemas. Adicionalmente, es posible construir heurísticos especiales que exploten conocimiento específico en cada dominio, para resolver problemas particulares. Algunos de los algoritmos tipos:

- ***Ascenso a Colina (Hill Climbing)*****:** Es una variante del algoritmo de búsqueda de generación y prueba. Del procedimiento de prueba existe una realimentación que ayuda al generador a decidirse por cual dirección debe moverse en el espacio de búsqueda. En estos procesos se abandona la búsqueda si no existe un estado alternativo razonable al que se pueda

mover. Típicamente locales, ya que deciden qué hacer mirando únicamente a las consecuencias inmediatas de sus opciones. Puede que nunca lleguen a encontrar una solución, si son atrapados en estados que no son el objetivo, desde donde no se puede hallar mejores estados.

- Primero el Mejor (Best-First): Consiste en recorrer el grafo de búsqueda eligiendo en cada momento el nodo que tenga un mejor valor para una determinada función heurística *f*. Se pueden retomar caminos de exploración abandonados anteriormente. La finalidad del proceso es encontrar el camino de menor coste, por lo que la función *f* debe medir la distancia a la meta. Este algoritmo, combina las ventajas de los algoritmos primero en profundidad y primero en amplitud. Sigue un sendero a la vez, pero puede cambiarse a otro sendero que parece más prometedor que el que está siguiendo.

En este sentido, puede considerarse que es un algoritmo que realiza su proceso de búsqueda en un grafo, en el cual todos sus ramales representan una alternativa de solución. Para su operación, el algoritmo necesita dos listas de nodos y una función heurística que estime los méritos de cada nodo que se genere.

- El algoritmo A*: Es el mejor de tipo *Primero el Mejor,* que sirve para resolver el problema que conlleva el no considerar el camino recorrido hasta un momento dado. La clave de éste método está en definir una función para que tenga en cuenta el coste del camino recorrido [Winston94].

3.5 Elementos y Ciclo de Vida de los EE. SS.

Llegado este punto es conveniente examinar los distintos componentes que están presentes y que conforman la mayoría de los sistemas expertos actuales, así como su ciclo de vida. Esto permitirá un mejor entendimiento de todos los conceptos e ideas vistas hasta el momento.

3.5.1 Estructura de un Sistema Experto.

Los elementos fundamentales de un sistema experto son mostrados esquemáticamente en la Figura 3.9.

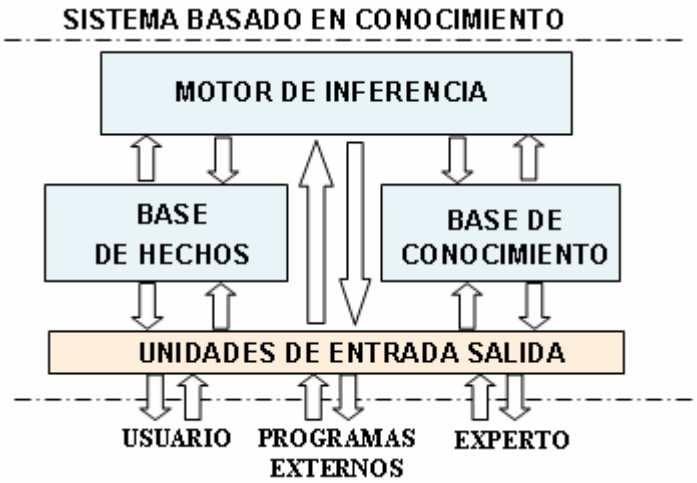

Figura 3.9 Estructura de un sistema basado en conocimiento

Se puede resumir en los siguientes módulos:

- La Base de Conocimientos.
- La Memoria Activa.
- El Motor de Inferencia.

- La Interfaz de Entrada/Salida.

Observaremos que una característica decisiva de los Sistemas Expertos es la separación entre conocimiento (reglas, hechos) por un lado y su procesamiento por el otro. A ello se añade una interfase de usuario y un componente explicativo. Procedamos a la descripción de cada uno de ellos.

3.5.2 La Base de Conocimientos.

Describe el universo de discurso o dominio en el cual el sistema de producción tiene que plantear soluciones, contiene el conocimiento del dominio en el cual el programa es competente. El conocimiento tiene que estar representado en la forma que resulte más adecuada para el dominio de su competencia. La base de conocimientos está constituida por bases de hechos y por bases de reglas. Los sistemas de producción de entidad suelen tener varias bases de hechos y varias bases de reglas relativas a diferentes aspectos del dominio. Hay que tratar que la representación del conocimiento sea:

- Sencilla.
- Independiente.
- Fácil de modificar.
- Transparente: justificación de soluciones y explicación de los procesos.
- Relacional.
- Potente: poder expresivo y eficiencia de cálculo.

Un aspecto importante de una base de conocimientos es su capacidad expresada sobre la base del número de reglas que posee:

- Demostración interesante: 50 reglas.
- Prototipo funcional: 250 reglas.
- SE operacional medio: 500 - 1000 reglas.
- SE operacional especial: 4000 reglas.

Una base de conocimientos debe ser coherente, rápida, modular, fácil de desarrollar y cómoda de mantener.

3.5.2.1 La Base de Hechos.

Las bases de hechos forman el esqueleto declarativo del sistema de producción, y su misión es la de articular a todos los hechos potencialmente relevantes del dominio, es el conjunto de información invariable de una a otra resolución. Los hechos se diferencian de los datos en el sentido que los hechos forman parte del Sistema Basado en Conocimiento, mientras que los datos, al poder variar de una solución a otra, conviene agruparlos en archivos externos al SBC.

Algunos autores no consideran a la base de hechos en forma independiente. Los conocimientos y los hechos pueden aparecer conjuntamente en una sola base, la de conocimientos.

3.5.2.2 La Base de Reglas.

La bases de reglas constituye el esqueleto procedimental del sistema de producción, y a través de ellas se posibilita la construcción de los circuitos inferenciales[2] que nos van a permitir

[2] O recorridos posibles entre unidades de conocimiento relacionadas

obtener conclusiones válidas. Obviamente, la estructura de las bases de hechos y de las bases de reglas debe ser tal que ambas entidades puedan "*comprenderse*" entre sí.

Cómo se lleva a cabo la clasificación en grupos de las características y de los procedimientos alrededor de un objeto con las técnicas de programación, y cómo deben ser las relaciones entre los objetos, pueden variar mucho de aplicación a aplicación. Las reglas disponibles en la Base de Conocimiento según se ha visto en apartados anteriores, se representan de la siguiente manera:

```
Si premisas Entonces Conclusión y/o Acción.
```

En la zona de las premisas se solicitan vinculaciones lógicas referentes a las cualidades de los objetos. En la zona de la conclusión se añaden nuevos hechos y cualidades a la base de conocimientos y/o se ejecutan acciones. Esta terminología se define a menudo como programación orientada a reglas.

3.5.3 Memoria Activa.

También se la puede denominar "*memoria de trabajo*"[3], es la estructura que contiene toda la información de naturaleza estática necesaria para resolver un problema concreto. Esta información incluye:

- Datos iniciales del problema.
- Datos incorporados con posterioridad.
- Hechos establecidos durante los procesos inferenciales.
- Hipótesis de trabajo, metas o submetas que todavía no han sido establecidas.

En la memoria activa es donde se producen todos los cambios de estado de nuestro sistema, de forma que es la memoria activa la que representa siempre nuestro estado actual. Por esta razón, la memoria activa es la responsable de interaccionar con el mundo exterior, aceptando la entrada de información de naturaleza no inferencial, es también el foco permanente de atención de las reglas del sistema.

3.5.4 El Motor de inferencia (MI).

Selecciona, decide, interpreta y aplica el conocimiento de la base de conocimientos sobre la base de hechos, con el fin de obtener la solución buscada. Un mecanismo de inferencia debe ser independiente del conocimiento y de los hechos [Nilsson01]. El motor de inferencia consta, fundamentalmente, de dos entidades: un interprete y una estrategia de control, que están físicamente separados de la unidad de conocimientos del dominio de aplicación y contiene los mecanismos necesarios para:

- Examinar la memoria activa y determinar que reglas debe ejecutarse. Este proceso de selección se realiza en función de la estrategia de búsqueda elegida y de los modelos implementados para la resolución de conflictos (Anexo D) que pudiesen aparecer.

- Controlar y organizar el proceso de ejecución de las reglas seleccionadas en el paso anterior.

- Actualizar la memoria activa cuando sea preciso.

[3] Algunos autores consideran que la base de hechos y la memoria activa son la misma estructura. Se le denomina *bases de hechos permanentes*, cuando representan el esqueleto declarativo del sistema, y *bases de hechos temporales,* cuando sólo constituyen conocimiento declarativo al problema en curso.

- Asegurar que el sistema tiene autoconocimiento. Ello implica saber en todo momento qué reglas han sido activadas, cuáles han sido ejecutadas[4], cual ha sido el último hecho incorporado a la memoria activa, etc.

El proceso global de trabajo del motor de inferencias se produce por ciclos denominados "*ciclos básicos del sistema de producción*". La naturaleza de tales ciclos básicos es netamente diferente si el proceso de búsqueda está dirigido por los datos o sí, por el contrario, está dirigido por los objetivos.

En cualquier caso, todo motor de inferencias debe ser considerado un intérprete, y como tal, no es más que un programa de naturaleza secuencial cuya misión es decidir que es lo que hay que hacer en cada momento, reconociendo y activando las reglas apropiadas en función de:

- Los criterios de activación elegidos.
- Las estrategias de búsqueda implementadas.
- La dirección de tránsito por el espacio de estados.

Además, para tratar de optimizar el proceso de explotación del espacio de estados correspondiente, la estrategia de control de motor de inferencias debe observar los criterios siguientes:

- Producir movimientos válidos en el espacio de estados.
- Ser sistemática.
- Ser eficiente.

El motor de inferencias es, pues, quien gobierna los procesos inferenciales en los sistemas de producción. Dado que los sistemas de producción están basados esencialmente en reglas, tendremos que ser capaces de definir como va ha poder materializarse la propagación del conocimiento en el sistema. En este sentido y desde una perspectiva completamente general se consideran dos procedimientos de propagación:

- *El encadenamiento progresivo de reglas*: Se traducirá en un proceso de búsqueda dirigido por los datos.

- *El encadenamiento regresivo de reglas:* Se traducirá en un proceso de búsqueda dirigido por los objetivos.

Aparte del tipo de encadenamiento de reglas, como norma general todo motor de inferencias debería incluir:

- *El emparejador o intérprete:* Activa las reglas relevantes en cada momento, de acuerdo con el estado actual de la memoria activa.

- *La estrategia de búsqueda:* Incluye las heurísticas de explotación del espacio de estados.

[4] Obviamente, no todas las reglas activadas son siempre ejecutadas. Un ejemplo lo constituyen los procesos de búsqueda en profundidad, en los que sólo se ejecuta una de las reglas activadas, manteniéndose las demás en este estado para posibilitar una eventual vuelta atrás (backtraking).

- *Los mecanismos de autoconocimiento:* Permiten identificar estructuras utilizadas, estados del problema, cambios en la memoria activa, órdenes de prioridades de acciones y hechos inferidos, etc.

- *Los mecanismos de terminación:* de los procesos inferenciales.

3.5.4.1 Características.

A continuación se señalan una serie de aspectos relacionados con los motores de inferencia, y que dada su importancia hay que tener en cuenta:

• El lenguaje en que ha sido escrito.

• La velocidad de trabajo: Inferencias/segundo.

• Las estrategias de búsqueda de soluciones: Que pueden ser:

- ***No ordenada***: aleatoria, heurística.

- ***Ordenada***: *Encadenamiento hacia delante,* guiado por los datos, deductivo.
Encadenamiento hacia atrás, guiado por los objetivos, inductivo.

• La forma en que elige el conocimiento.

• La posibilidad de incorporar metaconocimiento.

• El tipo de lógica que emplea en el razonamiento:

- Booleana, trivalente, multivalente, difusa.
- Monotónica o no monotónica.
- Atemporal o temporal.
- Lógica de orden 0, orden 0+, orden 1.

• El método que utiliza para la evaluación del conocimiento incompleto o incierto: Determinístico, Probabilística, Aproximado, Difuso, etc.

3.5.5 Los Módulos de Comunicación.

Un SBC necesita medios y canales adecuados, sencillos y potentes para comunicarse:

- *Con el usuario*: Para permitir el diálogo en forma sencilla.

- *Con el experto*: Para la configuración del sistema; para la adquisición, mantenimiento, depuración y validación del conocimiento.

- *Con archivos externos:* Bases de datos, hojas electrónicas de cálculo, archivos de texto y programas.

3.5.6 Otros Componentes.

La mayoría de los sistemas expertos actuales disponen de otros elementos además de los ya señalados y que tienen distintas funciones. La siguiente figura 3.10, representa una arquitectura en más detalle.

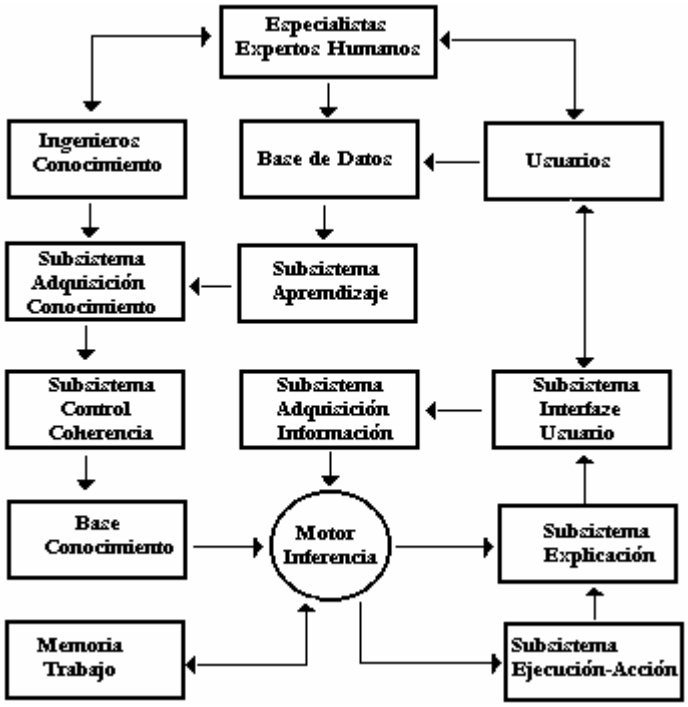

Figura 3.10 Componentes de un Sistema Experto y Flujo de Información.

3.5.6.1 La Componente Humana.

Un sistema experto es generalmente el resultado de la colaboración de uno o varios expertos humanos especialistas en el tema de estudio y los ingenieros del conocimiento, con los usuarios en mente. Los expertos humanos suministran el conocimiento básico en el tema de interés, y los ingenieros del conocimiento trasladan este conocimiento a un lenguaje, que el sistema experto pueda entender. La colaboración de los expertos humanos, los ingenieros del conocimiento y los usuarios es, quizás, el elemento más importante en el desarrollo de un sistema experto. Esta etapa requiere una enorme dedicación y un gran esfuerzo debido a los diferentes lenguajes que hablan las distintas partes y a las diferentes experiencias que tienen.

3.5.6.2 Subsistema de Adquisición de Conocimiento.

El subsistema de adquisición de conocimiento controla el flujo del nuevo conocimiento que fluye del experto humano a la base de datos. El sistema determina qué nuevo conocimiento se necesita, o si el conocimiento recibido es en realidad nuevo, es decir, si debe incluirse en la base de datos y, en caso necesario, incorpora estos conocimientos a la misma.

Si el conocimiento inicial es muy limitado y no se pueden sacar conclusiones, el motor de inferencia necesita obtener el conocimiento necesario y continuar con el proceso de inferencia hasta que se hayan sacado conclusiones. En algunos casos, el usuario puede suministrar la información requerida para éste y otros objetivos. Se realizará una comprobación de la consistencia de la información suministrada por el usuario, antes de introducirla en la memoria de trabajo.

3.5.6.3 Control de la Coherencia.

El subsistema de control de la coherencia ha aparecido en los sistemas expertos muy recientemente. Sin embargo, es una componente esencial de un sistema experto. Este susbsistema controla la consistencia de la base de datos y evita que unidades de conocimiento inconsistentes entren en la misma. En situaciones complejas incluso un experto humano puede formular afirmaciones inconsistentes. Por ello, sin un subsistema de control de la coherencia,

unidades de conocimiento contradictorio pueden formar parte de la base de conocimiento, dando lugar a un comportamiento insatisfactorio del sistema.

Es también bastante común, especialmente en sistemas con mecanismos de propagación de incertidumbre, que se llegue a conclusiones absurdas o en conflicto como, por ejemplo, situaciones en las que el sistema genera probabilidades mayores que la unidad o negativas. Por ello, el subsistema de control de la coherencia comprueba e informa a los expertos de las inconsistencias. Por otra parte, cuando se solicita información de los expertos humanos, éste subsistema informa sobre las restricciones que ésta debe cumplir para ser coherente con la existente en la base de conocimiento. De esta forma, ayuda a los expertos humanos a dar información fiable.

3.5.6.4 El Subsistema de Ejecución de Ordenes.

Permite al sistema experto iniciar acciones basadas en las conclusiones sacadas por el motor de inferencia. Como ejemplos, un sistema experto diseñado para analizar el tráfico en una red de telecomunicaciones puede modificar las tablas de enrutamiento en un router determinado para optimizar el tráfico global en el sistema, o un sistema para controlar una central nuclear puede abrir o cerrar ciertas válvulas, mover barras, etc., para evitar un accidente. La explicación de las razones por las que se inician estas acciones, puede darse al usuario mediante el subsistema de explicación.

3.5.6.5 El Subsistema de Explicación.

El usuario puede pedir una explicación de las conclusiones sacadas o de las acciones iniciadas por el sistema experto. Por ello, es necesario un subsistema que explique el proceso seguido por el motor de inferencia o por el subsistema de ejecución. Por ejemplo, si un conmutador de acceso decide rechazar la palabra clave asociada a un usuario de la red (una acción), el sistema puede mostrar un mensaje (una explicación) como la siguiente:

> *¡Lo siento!, su palabra clave es todavía incorrecta tras tres intentos.*
> *Se procede a la desconexión, con el equipo.*
> *Por favor, póngase en contacto con el administrador de la red.*

En muchos dominios de aplicaciones, es necesaria la explicación de las conclusiones debido a los riesgos asociados con las acciones a ejecutar. En el campo de la gestión de redes, los expertos son responsable últimos de los diagnósticos, independientemente de las herramientas técnicas utilizadas para sacar conclusiones. En estas situaciones, sin un subsistema de explicación, los expertos pueden no ser capaces de explicar las razones de su diagnóstico.

3.5.6.6 El Subsistema de Aprendizaje.

Una de las principales características de un sistema experto es su capacidad para aprender. Diferenciaremos entre aprendizaje estructural y aprendizaje paramétrico.

- Aprendizaje Estructural: Nos referimos a algunos aspectos relacionados con la estructura del conocimiento (reglas, distribuciones de probabilidad, etc.). Por ello, el descubrimiento de nuevos efectos relevantes para un fallo de red o la inclusión de una nueva regla en la base de conocimiento son ejemplos de aprendizaje estructural.

- ***Aprendizaje Paramétrico***: Nos referimos a estimar los parámetros necesarios para construir la base de conocimiento. Por ello, la estimación de frecuencias o probabilidades asociadas a efectos o fallos de red es un ejemplo de aprendizaje paramétrico.

Otra característica de los sistemas expertos es su habilidad para obtener *experiencia* a partir de los *datos* disponibles. Estos datos pueden ser obtenidos por expertos y no expertos, además pueden utilizarse por el subsistema de adquisición de conocimiento y por el subsistema de aprendizaje. De las componentes antes mencionadas puede verse que los sistemas expertos pueden realizar varias tareas. Estas tareas incluyen, pero no se limitan a, las siguientes:

• Adquisición de conocimiento y la verificación de su coherencia; por lo que el sistema experto puede ayudar a los expertos humanos a dar conocimiento coherente.

• Almacenar (memorizar) conocimiento.

• Preguntar cuándo se requiere nuevo conocimiento.

• Aprender de la base de conocimiento y de los datos disponibles.

• Realizar inferencia y razonamiento en situaciones deterministas y de incertidumbre.

• Explicar conclusiones o acciones tomadas.

• Comunicar con los expertos/no expertos humanos y con otros sistemas expertos.

3.5.7 El Ciclo de Vida.

Al igual que otras formas de software, tienen como objetivo crear soluciones computacionales a problemas. A pesar de que los Sistemas Expertos se basan principalmente en procesos heurísticos antes que algorítmicos, el desarrollo tiene un ciclo de vida similar al de un sistema de software convencional. Sin embargo, existen varias diferencias significativas entre la ingeniería cognoscitiva y la ingeniería de software:

- Una de las mayores diferencias (estudiadas en apartado 3.2), es el tipo de conocimiento que se representa. La ingeniería de software involucra la representación de procedimientos algorítmicos bien definidos y típicamente bien conocidos por muchas personas; mientras que la ingeniería cognoscitiva involucra la representación del conocimiento heurístico amplio, impreciso, mal definido, que está almacenado en la mente de pocos expertos.

Debido a que el conocimiento heurístico no es ampliamente conocido, ni entendido, tienen que ser utilizadas ciertas técnicas para lograr transferirlo desde las mentes de los individuos que lo poseen, hasta una representación computarizada. Esta transferencia, como ya se vio, denominada Adquisición de Conocimiento, es elaborada y consume mucho tiempo.

- Otra de las diferencias significativas está relacionada con la naturaleza y la cantidad de conocimiento. Mientras la naturaleza y la cantidad del conocimiento requerido para resolver un problema algorítmico tradicional pueden ser razonablemente bien estimados, este no es el caso para Sistemas Expertos. Típicamente, la naturaleza y la cantidad de conocimiento requeridos para resolver un problema no es bien conocido, aún por los propios expertos. Esto dificulta la predicción del esfuerzo total requerido para desarrollar un Sistema Experto. Además, puede ser difícil llegar a un diseño adecuado desde las etapas iniciales del proyecto, dando lugar al problema denominado *"Dilema del Cambio"*.

El dilema del cambio, puede aparecer durante el proceso de desarrollo, cuando un ingeniero del conocimiento descubre que la estructura de representación del conocimiento, la herramienta u otros aspectos de diseño del sistema resultan inadecuados. Este descubrimiento

es debido, generalmente, a la falta de comprensión inicial de las complejidades del dominio del problema o al hecho de haber subestimado su magnitud. Si esto ocurre dentro de una etapa avanzada del desarrollo, puede ser un serio problema. El ingeniero del conocimiento puede vérselas ante el dilema de continuar con la infraestructura inadecuada que puede resultar en graves dificultades al final del proyecto o volver a iniciar el desarrollo con lo que parece ser una mejor estructura de representación del conocimiento y otros aspectos más adecuados de diseño.

Cualquiera que sea la decisión, sin lugar a dudas, se retardará la ejecución del proyecto. Sin embargo si el dilema del cambio aparece en las etapas iniciales del proceso, puede ser beneficioso ya que permitiría corregir los errores identificados, antes de que se invierta demasiado tiempo y recursos en el paradigma inicial. Para evitar estos obstáculos y proporcionar confianza al equipo de desarrollo, muchos ingenieros del conocimiento utilizan técnicas combinadas de Prototipado Rápido y Desarrollo Incremental.

3.5.7.1 Prototipado Rápido.

Puede utilizarse la flexibilidad y el poderío de LISP, PROLOG o de otras herramientas de desarrollo para crear rápidamente un prototipo funcional del sistema final deseado. Permite realimentar en forma temprana aspectos de profundidad y tipo de conocimiento, de las necesidades de los usuarios, así como ayudar a verificar la validez de las decisiones tomadas durante la etapa de diseño. Con esto, si se presenta un dilema de cambio, su impacto será mínimo debido a lo temprano de su ocurrencia.

En la figura 3.11, se puede apreciar que el desarrollo del prototipo inicial utiliza un ciclo de vida modificado del que normalmente se emplea en ingeniería de software. Las fases de análisis y especificación deben realizarse teniendo en cuenta el sistema completo, pero el diseño y la implementación del prototipo son realizados de una manera rápida y preliminar. Esto proporciona un sistema funcional que puede ser evaluado para obtener la realimentación necesaria [Louis00].

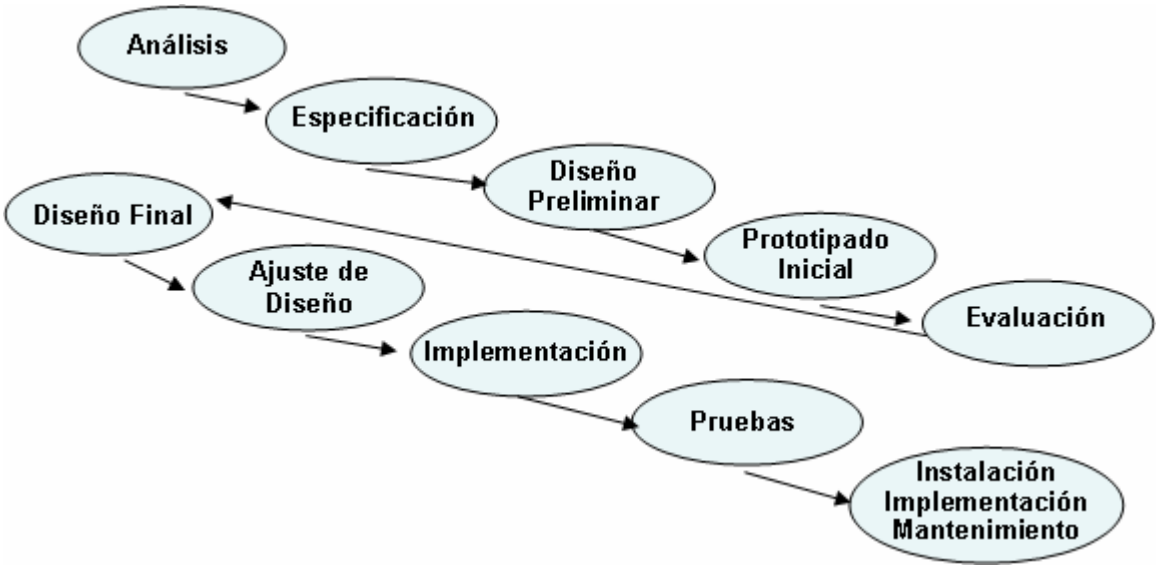

Figura 3.11 de vida de un Sistema Basado en Conocimiento Ciclo

El prototipo inicial puede ser desechado una vez que se ha obtenido la realimentación esperada o puede ser mejorado en forma incremental para constituirse en un subsistema del Sistema Experto final. En la mayoría de casos se opta por desecharlo debido a las dificultades que existen para modificarlo y acomodar todas las ideas generadas durante su desarrollo y

evaluación. En general, es más fácil empezar un nuevo proceso de desarrollo siguiendo las diferentes etapas del ciclo de vida con mayor detalle.

3.5.7.2 Desarrollo Incremental.

Una vez que el prototipo del sistema ha sido evaluado, se realiza su diseño final y se lo somete a un ciclo continuo de modificaciones para que mejore sus características y su calidad. A este proceso se conoce como desarrollo incremental. La estrategia que se sigue puede ser definida como:

Un proceso iterativo de extracción, representación y confirmación del conocimiento en un limitado subconjunto del dominio del problema, con el objetivo de construir en forma incremental el SBC en segmentos autocontenidos.

El desarrollo incremental se centra en dos conceptos:

- *Dividir para conquistar:* Donde un segmento de conocimiento manejable pero completo, es seleccionado y desarrollado.

- *Desarrollo iterativo:* Donde el concepto de dividir para conquistar es aplicado iterativamente sobre los diferentes segmentos que conforman el problema completo.

En consecuencia, el desarrollo incremental involucra varios ciclos de recopilación del conocimiento de los expertos, incorporación del conocimiento en el sistema, revisión de la implementación resultante con los expertos y corrección de los problemas encontrados. A través de la progresiva incorporación de módulos independientes al sistema en desarrollo, un SBC puede ser rápidamente puesto en funcionamiento, aunque sea en forma parcial. En cambio un programa convencional debido a su naturaleza procedimental, en general debe ser completamente implementado antes de ser utilizado.

3.5.8 Etapas del Ciclo de Vida de un SBC.

A continuación se describen los detalles de las etapas del ciclo de vida de un SBC, tal como se muestra en la figura anterior 3.11 [Rousel04].

- **Análisis del Problema:** Evaluar el problema y los recursos disponibles para determinar la aplicabilidad de una solución basada en conocimiento. Se debe realizar un análisis costo/beneficio del SBC propuesto para saber si su desarrollo puede ser garantizado. Puede también requerirse de una investigación de mercado o un examen profundo del propósito del sistema pedido para determinar la efectividad del costo del sistema.

- **Especificación de Requerimientos:** Formalizar y poner por escrito lo que fue adquirido durante la fase de análisis. Esto permite determinar los objetivos del proyecto, de una manera inequívoca y establece los medios para obtener dichos objetivos. La experiencia acumulada en el desarrollo de SBC's demuestra que sin tener especificaciones no es posible diseñar un SBC de real utilidad. El documento de especificaciones debe plantear claramente y discutir los objetivos y las características del sistema, el entorno del usuario y las limitaciones.

- **Diseño Preliminar:** Esta etapa considera únicamente las decisiones de alto nivel necesarias para preparar y desarrollar rápidamente el prototipo inicial. Específicamente, esta etapa determina el paradigma de representación del conocimiento, la herramienta escogida para construir el prototipo y la selección de los expertos. En esta etapa puede ser necesaria una considerable recopilación de conocimiento tanto de los expertos, como de fuentes impresas,

para poder tomar decisiones sólidas acerca del paradigma de representación del conocimiento y por lo tanto de la herramienta necesaria.

● **Prototipo Inicial (Rápido) y Evaluación:** Esta es una etapa clave del desarrollo del sistema experto. En esta fase se confirman, rectifican o desechan las decisiones que han sido tomadas por los expertos durante el diseño preliminar. El prototipo inicial debe verse como el sistema completo, excepto que estará limitado en su cobertura. Debe incluirse una relativamente bien definida interfaz con el usuario y un robusto subconjunto de conocimiento de tal forma que los usuarios puedan juzgar su aceptabilidad. Esto no significa que el prototipo debe ser altamente robusto, simplemente debe reflejar la forma que tendría el sistema final que será construido.

En general se recomienda que el prototipo inicial sea desechado una vez que se haya completado su evaluación. El desarrollo del sistema final debe partir, en lo posible desde el inicio. La clave en la etapa del prototipo es que se debe extraer tanto conocimiento y opiniones de expertos y usuarios como sea posible para poder validar satisfactoriamente las decisiones de diseño. Cualquier error cometido en las etapas anteriores debería ser detectado y corregido en esta etapa.

● **Diseño Final:** El diseño final comprende la selección de las herramientas y de los recursos necesarios para desarrollar el sistema a ser entregado. En esta etapa también se incluye la selección del paradigma para representar el conocimiento. Esta decisión tiene un impacto directo en la herramienta que será seleccionada. En muchos casos puede ser aplicable y muy útil realizar una descripción gráfica de los diferentes módulos del sistema, empleando las herramientas CASE propias de los sistemas de software convencionales. Para cada uno de estos módulos el diseño debe incluir las especificaciones de las entradas típicas y las salidas o conclusiones esperadas. Como es muy factible que una misma entrada se use en más de un módulo, es primordial preparar una descripción del subsistema de interfaces.

● **Implementación:** Esta es la etapa que puede consumir la mayor parte del tiempo del ciclo de vida de un SBC, aun cuando exista un excelente diseño y abarca el proceso completo de adquisición del conocimiento, para todos los módulos o subsistemas.

● **Pruebas**: El asegurar la calidad de un SBC es una tarea muy importante que debe ser cuidadosamente planificada, especialmente a medida que el SBC es más grande y complejo, o es de aplicación crítica. El plan de pruebas de un SBC, es bastante similar al que se prepara para un sistema de software convencional. Debe incluir procesos de verificación y validación**,** elementos diferentes en naturaleza, pero ambos muy necesarios.

- La *verificación* puede ser definida como una ayuda para que el sistema sea construido correctamente (*Building the system right*).

- La *validación* se define como el proceso que nos indica si hemos construido el sistema correcto para las necesidades planteadas (*Building the right system*).

Sobre la base de los resultados de las pruebas realizadas al sistema, el grupo responsable del desarrollo y el usuario deberán determinar finalmente si el sistema está listo para ser aceptado. De ser este el caso, el usuario deberá notificar que el SBC es apto para su uso como sistema funcional. Además, deberá suscribir un documento de aceptación preliminar y consignar cualquier ajuste final que debe ser realizado, en caso de existir alguno.

● **Ajustes al Diseño:** A medida que el trabajo avanza y los ingenieros del conocimiento tienen a la vista los problemas detectados, deben realizar los ajustes necesarios al inicio de cada iteración. Si estos ajustes cada vez son relativamente más pequeños y no son retroactivos, se tiene una buena medida de que se está progresando. Pero si ocurre lo contrario, puede representar un serio retardo al proyecto y posiblemente requerir un cambio de paradigma.

3.5.9 Instalación, Implantación y Mantenimiento

En la etapa final del ciclo de vida de un SBC se traslada el sistema desarrollado como un producto operativo, hacia el entorno de los usuarios. Para ello, se deben realizar varias actividades de instalación, implantación y mantenimiento similares a las de un sistema de software convencional.

● **Aceptación del producto final:** Durante la etapa de aceptación, se debe completar todo el trabajo que falte. Por ejemplo:

- Preparar los programas para instalación del SBC.

- Completar la documentación del sistema, especialmente el Manual del Usuario.

- Identificar los cambios procedimentales que pueden ser requisitos necesarios, para que el SBC se incorpore fácilmente al flujo de trabajo de las áreas operacionales de los usuarios.

- Si es del caso, preparar la interfaces necesarias para que el SBC se comunique con otros sistemas de la organización.

● **Formación a los usuarios del producto final.** El plan de adiestramiento de usuarios debe incluir los siguientes aspectos:

- Presentación del producto final, exponiendo los aspectos más importantes del sistema desarrollado.

- Entrega del Manual de Usuario.

- Explicación de los procedimientos organizacionales requeridos.

- Puesta en práctica del producto. En lo posible debe realizarse con el sistema integrado dentro del flujo de trabajo normal de la organización.

● **Pruebas de aceptación de los usuarios finales.** Luego que los usuarios están debidamente capacitados, se pueden realizar pruebas para determinar si el sistema está operando satisfactoriamente dentro de su ambiente normal de trabajo. Cualquier potencial problema que sea identificado por el usuario, debe ser resuelto previo a la aceptación final.

● **Constancia de aceptación de los usuarios finales.** Una vez que los usuarios han completado su formación y las pruebas realizadas los ha convencido que el sistema está listo para operación normal, se debe suscribir un documento de aceptación final.

● **Implantación y Mantenimiento.** Finalmente, el dueño del SBC debe ejecutar las acciones requeridas para que el sistema aceptado entre en producción. Asesoramiento y consultoría especializada pueden ser necesarios para planificar y ejecutar la implantación y el mantenimiento del sistema. A medida que se gane en experiencia y comprensión acerca del

problema que ayuda a resolver el SBC, el conocimiento adquirido que debe irse incorporando en el sistema, a través de las actividades previstas en el plan de mantenimiento.

3.5.10 Herramientas para Desarrollo de SBC.

El paso desde el diseño abstracto, hasta un sistema implementado y listo para ser ejecutado en un sistema informático, debe ser expresado de forma que el sistema computacional lo entienda. Para ello se puede utilizar un lenguaje programación o un sistema especialmente diseñado para desarrollar SBC. La pregunta inmediata es ¿Cuál de ellos utilizar? En las siguientes secciones se puede encontrar una posible respuesta.

3.5.10.1 Lenguajes de Programación.

En principio, cualquier lenguaje de programación puede ser utilizado. Siendo así de amplio el espectro del cual se puede escoger un lenguaje para programar un SBC, se debe considerar como factor importante de decisión, la extensión en la cual el lenguaje cubre o se adecua a los requerimientos de diseño. Atendiendo a la forma de estructurar sus instrucciones se pueden dividir en:

- Imperativos: PASCAL, C/C++.
- Funcionales: LISP.
- Declarativos: PROLOG, CHIP, OPS5.
- Orientados a Objetos: SmallTalk, Hypercard, CLOS.

Tradicionalmente LISP y PROLOG han sido los lenguajes que se han utilizado, ya que estos lenguajes ofrecen características especialmente diseñadas para manejar problemas generalmente encontrados en IA y por este motivo se los conoce como *lenguajes de IA*.

Una de las principales características que comparten los lenguajes LISP y PROLOG, como consecuencia de su respectiva estructura, es que pueden ser utilizados para escribir programas capaces de examinar a otros programas, incluyendo a ellos mismos. Esta capacidad se requiere por ejemplo para hacer que el programa explique sus conclusiones. Esto sólo puede hacerse si el programa tiene la capacidad de examinar su propio modo de operación.

3.5.11 Sistemas de Desarrollo.

Históricamente, los primeros SBC fueron desarrollados utilizando lenguajes de programación como el LISP y el PROLOG. A medida que el desarrollo de SBC iba aumentado en cantidad y complejidad, la comunidad científica comenzó a buscar formas de desarrollar los sistemas en menor tiempo y con menor esfuerzo. Esto dio lugar la aparición en primer lugar de sistemas vacíos como el EMYCIN, que se denominaron *shells*, que ofrecían toda la arquitectura de un SBC a la que hay que incorporar la base de conocimientos.

Posteriormente aparecieron en el mercado otras herramientas que incorporaron, además de opciones de representación del conocimiento, esquemas de inferencia y control. Estas herramientas tomaron el nombre de *Entornos de Desarrollo de SBC*. A continuación se dan algunos ejemplos de sistemas comerciales:

- Sistemas Vacíos (*shells*): EMYCIN, Crystal, Leonardo, XiPlus, EXSYS, VP-Expert, Intelligence Compiler, etc.

- Entornos híbridos de desarrollo: CLIPS, KEE, ART, EGERIA, Kappa, Nexpert Object, Goldworks, LOOPS, Flavors, etc.

3.6 Manejo de la Incertidumbre.

En determinados tipos de situaciones reales, no siempre es posible contar con toda la información, inclusive la información disponible puede ser incorrecta, incompleta o cambiar muy rápidamente. Todo esto da lugar a diferentes formas de inconsistencia e incertidumbre. El tratamiento de la incertidumbre constituye uno de los campos fundamentales que afecta en mayor o menor medida a los sistemas expertos. En particular, es una de las características esenciales que deben ser tratadas con detenimiento y que representan una mayor complejidad [Martin01].

3.6.1 Causas que lo motivan.

Existen distintas razones por las cuales se pueden presentar incertidumbre en el conocimiento:

- *Conocimiento incierto:* El experto puede tener solamente un conocimiento heurístico o empírico respecto a algunos aspectos del dominio. Por ejemplo, que cierto conjunto de evidencias implican probablemente determinada conclusión.

- *Datos inciertos*: Aun cuando tengamos certidumbre del conocimiento, por ejemplo de una relación causa-efecto, la certeza de los datos puede ser cuestionable.

- *Información incompleta:* De forma frecuente hay que tomar decisiones a partir de información incompleta, teniendo a veces que asumir algún tipo de información por defecto.

- *Azar:* El dominio puede estar compuesto por información aleatoria.

En resumen el tratamiento de la incertidumbre es esencial en el diseño de sistemas expertos. Diversos métodos han sido desarrollados para evaluar los grados de certeza o de verdad de las conclusiones. Uno de los más generalizados consiste en asignar coeficientes de certeza o de confianza a los hechos que intervienen en las condiciones y en la conclusión de una regla. Los principales modelos desarrollados son:

- Modelo estadístico/probabilístico.
- Modelo de lógica difusa.
- Factores de certeza.
- Etc.

3.6.2 Razonamiento Estadístico Probabilístico.

Las técnicas de Teoría de Probabilidades se han empleado ampliamente en varias disciplinas diferentes en un intento de cuantificar la incertidumbre. El aspecto atractivo de la probabilidad se basa parcialmente en el hecho de que se ha establecido sobre una base matemática sólida, la técnica para emplear probabilidades han sido ampliamente difundidas [Pino01].

La medida utilizada es la *probabilidad*, en la que la distribución conjunta de un conjunto de variables se usa para describir las relaciones de dependencia entre ellas, y se sacan conclusiones usando fórmulas muy conocidas de la teoría de la probabilidad. Los sistemas expertos que utilizan la probabilidad como medida de incertidumbre se conocen como sistemas *expertos probabilísticos* y la estrategia de razonamiento que usan se conoce como *razonamiento probabilístico,* o *inferencia probabilística.*

3.6.3 Factores de Certeza.

Surge como consecuencia de las dificultades que presentaba el método probabilístico clásico. Es un modelo que en vez de buscar un fundamento teórico, trata de reproducir la forma en que el ser humano combina intuitivamente distintas fuentes de información.

Resulta una escala de valores entre –1 y +1 para la certeza de un hecho, correspondiendo el –1 a los hechos ciertamente falsos y el 1 a los hechos verdaderos. Un hecho que tiene alguna evidencia de ser falso recibe un factor negativo, y un hecho que tiene alguna evidencia de ser verdadero recibe un factor positivo. Se puede considerar que los valores cercanos a cero, significan que el hecho no tiene suficiente evidencia para considerarse falso o verdadero, Figura 3,12.

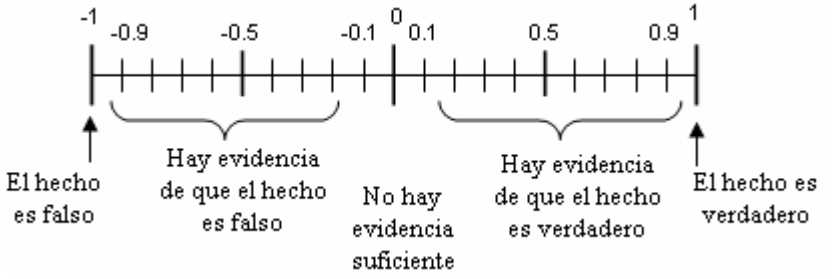

Figura 3.12 Representación de los factores de certeza.

3.6.4 Lógica Difusa.

En las técnicas que se han estudiado, no se han modificado los fundamentos matemáticos que proporcionan la lógica y la teoría de conjuntos, un objeto es miembro del conjunto o no lo es, no existen posibilidades intermedias. Se han extendido esas ideas mediante construcciones adicionales que proporciona la teoría de probabilidades.

Ahora se proporciona un enfoque diferente y se introducen cambios fundamentales en la idea miembro de un conjunto, haciendo los correspondientes cambios en las definiciones de operaciones lógicas. La motivación de la lógica difusa surge por la necesidad de representar proposiciones como las siguientes:

- El servidor es rápido.
- La red falla un poco.
- Los conmutadores trabajan a velocidades análogas.

La lógica borrosa tiene una historia corta pero un rápido crecimiento debido a su capacidad de resolver problemas relacionados con la incertidumbre de la información o del conocimiento de los expertos. Además proporciona un método formal para la expresión del conocimiento en forma entendible por los humanos. Estas cualidades le aseguran un amplio campo de aplicaciones y un alto interés para las aplicaciones industriales presentes y futuras, según el principio enunciado por algunos investigadores actuales: *"Cuanto más humano ha de ser un sistema, más lógica borrosa contendrá".*

La teoría de conjuntos tradicional define la pertenencia a un conjunto como un predicado booleano, la teoría de conjuntos difusos permite representar el ser miembro de un conjunto como una distribución de posibilidades, tal y como se muestra en la siguiente figura 3.13(a) con el conjunto de la servidores rápidos y el conjunto de servidores muy rápidos. Nótese como esto contrasta con la definición booleana normal para los servidores rápidos que

se muestra en la figura 3.13(b). En esta última, un servidor puede ser o no rápido, por los que hay que definir una rapidez que represente la frontera, ocurre lo mismo para muy rápido.

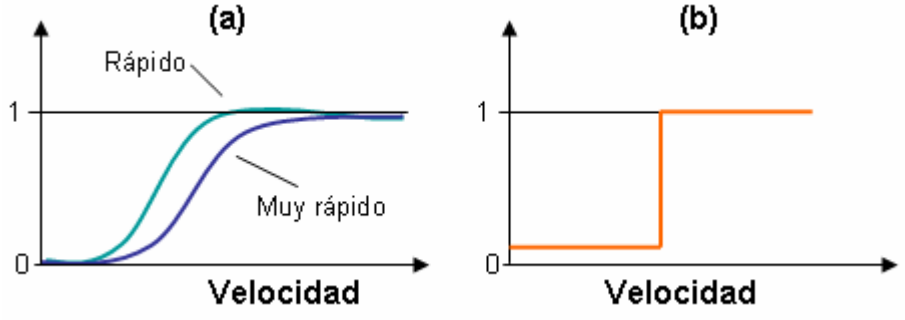

Figura 3.13 Miembro de un conjunto convencional frente a uno difuso

3.6.4.1 Conjuntos Difusos.

Dentro de los sistemas borrosos se incluyen diversas perspectivas: la Teoría de Conjuntos Borrosos, como extensión de la teoría de conjuntos clásica y la Lógica Borrosa, que puede ser considerada una ampliación de las lógicas n-valuadas propuestas por Lukasiewiez en 1930, que son a su vez extensión de la lógica trivaluada (verdadero, falso e indeterminado).

También se habla de sistemas de control borrosos, que utilizan las expresiones de la lógica borrosa para formular reglas orientadas al control de sistemas. Estos sistemas de control borroso pueden considerarse una extensión de los sistemas expertos, pero superando los problemas prácticos que estos sistemas tienen para el razonamiento en tiempo real, causados por la explosión exponencial de las necesidades de cálculo para el análisis lógico completo de amplias bases de reglas.

La teoría de conjuntos borrosos parte de la teoría clásica e introduce una función de pertenencia al conjunto, definida como un número real entre 0 y 1. Se introduce el concepto de valor lingüístico asociado a cada subconjunto borroso, y definido por una palabra o etiqueta lingüística A. Para cada valor lingüístico A se define una función $\mu A\ t\ (\ )$ que indica el grado de probabilidad de que la variable *t* esté incluida en el concepto representado por el valor lingüístico A.

En la lógica difusa, se generaliza el concepto de pertenencia a un determinado conjunto permitiendo que las funciones características de confianza asuman valores reales, dentro del intervalo [0 .. 1], (totalmente FALSO = 0 y totalmente VERDADERO = 1). Estos valores indican el grado o nivel de pertenencia del objeto dentro del conjunto difuso. La teoría de los conjuntos difusos permite que un elemento sea parcialmente miembro de un determinado conjunto. Por ejemplo:

```
REDES = {ATM, FDDI, RDSI, ETHERNET}
```

Un subconjunto difuso REDES_RAPIDAS de REDES, puede ser descrito como:

```
REDES_RAPIDAS = {1,0/ATM, 0,9/FDDI, 1,0/RDSI, 0,0/ETHERNET}
```

Donde los valores indicados representan el grado de pertenencia de cada tecnología al subconjunto difuso REDES_RAPIDAS. Como la Ethernet tiene un grado de pertenencia igual a cero, bien podría ser eliminado del subconjunto:

```
REDES_RAPIDAS = {1,0/ATM, 0,9/FDDI, 1,0/RDSI}
```

3.6.5 Variables Lingüísticas.

Los conjuntos difusos nos proporcionan bloques constructivos para resolver problemas complejos e imprecisos del mundo real. Por ejemplo, la variable lingüística es una herramienta poderosa para procesar lenguaje natural impreciso y difuso. Se constituye en un puente entre el mundo numérico preciso y la forma difusa en que los humanos nos expresamos.

Las variables lingüísticas son similares a las variables numéricas ya que tienen ciertos valores asociados a ellas. Pero, a diferencia de las variables numéricas, los valores de las variables lingüísticas no son números sino expresiones del lenguaje natural que describen alguna cantidad abstracta de interés. Estas expresiones del lenguaje natural son los nombres de conjuntos difusos que consisten de valores numéricos. A estos conjuntos difusos, se los llama también restricciones difusas.

Para aclarar estos conceptos, asumamos que el conjunto de las velocidades de transmisión a través de una determinada red contiene los números enteros del 10M a 100M. En esta base podríamos describir las restricciones difusas bajo, medio y alto, como sigue:

bajo: {*1,0/10M, 0,8/20M, 0,4/30M, 0,1/40M, 0,0/50M* }
medio: {*0,1/20M, 0,4/30M, 0,8/4, 1,0/50M, 0,8/60M, 0,4/70M, 0,1/80M*}
alto: { *0,0/50M, 0,1/60M, 0,4/70M, 0,8/80M, 1,0/100M* }

Nótese que las definiciones de las restricciones difusas son subjetivas, dependen de los gustos de cada persona y de la comprensión que ésta tenga del término que desea describir.

3.6.6 Otros métodos.

Entre las técnicas numéricas para el tratamiento de la incertidumbre hay que citar la teoría de Dempster Shafer, una extensión de la teoría de la probabilidad que mide para cada proposición tanto la evidencia a favor como la evidencia en contra. Sin embargo su formalismo matemático es complejo, por lo que no la vamos a desarrollar aquí. Además esta teoría debe considerar cada combinación posible de hipótesis que desea evaluar, la explosión combinatoria resultante hace que este método sea inaplicable en problemas del mundo real, a menos que se utilicen aproximaciones. Por esta razón aunque el método de Dempster Shafer sigue estudiándose, en la teoría prácticamente no se emplea hoy en día para la construcción de sistemas expertos.

Existen además otros métodos *no numéricos* para el tratamiento de la incertidumbre, basados fundamentalmente lógicas no monotónicas tales como los modelos de razonamiento por defecto, los sistemas de suposiciones razonadas y la teoría de justificaciones de Cohen y Grinberg. Estos métodos consisten en que cuando no hay información suficiente se hacen suposiciones que posteriormente podrán ser corregidas al recibir nueva información. El problema principal que presentan se debe a su naturaleza cualitativa, por lo que no pueden considerar los distintos grados de certeza o incertidumbre de las hipótesis. Suelen presentar además problemas de explosión combinatoria En consecuencia se estudian más por su importancia teórica (fundamento de la inteligencia artificial) que por las aplicaciones prácticas a que puedan dar lugar.

3.7 Sistemas Expertos y la Gestión de Redes.

Gracias a las investigaciones realizadas en la inteligencia artificial, con el desarrollo de los sistemas basados en el conocimiento y los sistemas expertos, observamos que se han producido grandes avances en el tratamiento del conocimiento, factor fundamental para la toma de decisiones. Esto sólo es posible, hoy en día, utilizando los ordenadores electrónicos y los medios que proporciona la tecnología de la información.

En las últimas décadas, se han producido grandes cambios en el entorno de las telecomunicaciones, como consecuencia de los avances producidos por las nuevas tecnologías de la producción, de la información y de las comunicaciones. En este nuevo entorno, tan complejo y cambiante, para poder tomar decisiones de una manera eficaz y satisfacer las necesidades que plantean las redes modernas, es necesario disponer, en todo momento y de una forma rápida de información suficiente, actualizada y oportuna.

En este y en los próximos apartados se van ha estudiar los sistemas expertos desde la óptica de la gestión y control de las redes de telecomunicaciones. A través de este estudio se va a mostrar una visión de conjunto de la aplicación de los sistemas expertos en el dominio de la gestión de redes. Para ello, se estudian los sistemas expertos en los distintos dominios de la gestión de redes, indicando sus características principales.

3.7.1 Adecuación de Sistemas Expertos a la Gestión de Red.

Los sistemas expertos, como ya se ha explicado en apartados anteriores, son programas de ordenador que capturan el conocimiento de un experto e imitan sus procesos de razonamiento, cuando resuelven los problemas en un determinado dominio. Los sistemas expertos se han venido aplicando con éxito en múltiples campos: medicina, geología, química, ingeniería, etc. [Moret00], para realizar tareas muy diversas (ejemplo, interpretación, predicción, diagnóstico, diseño, planificación, instrucción, control, etc.).

Las actividades de gestión en las telecomunicaciones también son campos en los que se pueden aplicar los sistemas expertos, pues se realizan muchas de las tareas antes descritas y, además, éstas cumplen la mayoría de los requisitos que son necesarios para poder desarrollar un sistema experto, las tareas requieren conocimiento especializado, existen auténticos expertos en la materia, los expertos son escasos, la pericia necesita ser localizada en distintos lugares, la mayoría de las tareas requieren soluciones heurísticas, etc.

En este sentido, y teniendo en cuenta el área de gestión de redes sobre los que se van a aplicar los sistemas expertos, obtendremos las siguientes características:

1.- Pueden resolver problemas complejos de gestión y administración en las redes actuales, tan bien o mejor que los expertos humanos.

2.- Razonan heurísticamente, usando lo que los expertos consideran que son reglas empíricas efectivas, e interactúan con los administradores y operadores del sistema de telecomunicaciones de forma adecuada incluyendo el lenguaje natural.

3.- Manipulan y razonan sobre descripciones simbólicas, que representan los elementos que componen el sistema gestionado.

4.- Pueden funcionar con datos que contienen errores, usando reglas de enjuiciamiento inciertas.

5.- Pueden contemplar múltiples hipótesis en competición simultáneamente, dando mayor eficiencia a la gestión.

6.- Pueden explicar por qué están formulando una pregunta.

7.- Pueden explicar su proceso de razonamiento y justificar sus conclusiones, para una mayor comprensión del personal con el que interactúa.

3.7.2 Funciones de Gestión.

Hay que tener en cuenta, que no en todas las tareas que se realizan en el campo de la gestión de redes es necesario utilizar los sistemas expertos. Así, existen tareas que están perfectamente estructuradas, son muy mecánicas y pueden expresarse en forma algorítmica (copias de seguridad, obtención de estadísticas, etc.), se puede, y es conveniente, utilizar la informática convencional (programas informáticos normales, tratamientos de textos, bases de datos, ...)

En las tareas que estén semiestructuradas se pueden utilizar los sistemas de ayuda a la decisión (ejemplo, hojas de cálculo, sistemas de consulta de archivos, sistemas de representación y análisis de datos, etc.), reservándose los sistemas expertos para las tareas que estén muy poco o nada estructuradas, pues en este tipo de tareas se requiere mucho del juicio de un experto y se utilizan reglas heurísticas para llegar rápidamente a una solución, dado que el campo de soluciones puede ser muy amplio.

Se utiliza el marco de gestión OSI, CCITT, X.701 que define las cinco áreas funcionales en las cuales se divide la gestión de red [ITUT92], estudiadas en el capítulo segundo de la tesis y que resumimos a continuación:

- *Gestión de fallos:* Tiene como objetivo fundamental la localización y recuperación de los problemas de la red. Abarca la detección e identificación de los fallos y la corrección de los mismos.

- *Gestión de configuración:* Es el proceso de obtención de datos de la red y utilización de los mismos para incorporar, mantener y retirar los diferentes componentes y recursos que la integran.

- *Gestión de contabilidad*: Tiene como misión la recolección de estadísticas que permitan generar informes de tarificación que reflejen la utilización de los recursos por parte de los usuarios. Requiere la realización de tareas de recolección de datos sobre la utilización de los recursos, establecimiento de cuotas y cobro a los usuarios por la utilización de los recursos.

- *Gestión de prestaciones*: Consiste en realizar cuatro tareas básicas: recogida de datos, análisis de datos, establecimiento de umbrales cuando se supera un determinado grado de utilización de un recurso se dispara una alarma y modelado de la red. Su principal objetivo es el mantenimiento del nivel de servicio de la red. Basa sus tareas en la definición de unos indicadores de funcionamiento. Es necesario fijar una serie de criterios que permitan conocer cuál es el grado de utilización de un recurso.

- *Gestión de seguridad:* El objetivo es ofrecer mecanismos que faciliten el mantenimiento de políticas de seguridad. La Gestión de Seguridad se ocupa de la identificación de la información a proteger y dónde se encuentra, identificación de los puntos de acceso a la información, protección de los puntos de acceso y del mantenimiento de los puntos de acceso protegidos.

3.8 Implementación de Sistemas Expertos para la Gestión de Redes.

Como ya se ha visto, la Inteligencia Artificial (IA) y en particular los Sistemas Expertos, están siendo progresivamente incorporadas para la mejora de los Sistemas de Explotación de las redes de Telecomunicación.

La eficiencia en las tareas de administración y mantenimiento de redes de telecomunicación actuales depende, en gran parte, del grado de cooperación y sinergia existente entre las funciones del sistema de gestión de la red y los operadores humanos. El sistema de gestión desempeña un papel de primer orden, encargándose de supervisar continuamente los elementos de la red, informando sobre posibles anomalías y ejecutando los comandos de control. Sin embargo, toda la responsabilidad recae sobre el operador, que se ocupa de interpretar la información recibida, verificarla mediante pruebas, y realizar las acciones de control necesarias para localizar el fallo y repararlo. Excepto en las tareas de supervisión, el sistema de gestión actúa como una sofisticada herramienta que obedece en todo momento las indicaciones del operador.

3.8.1 Problemas que se presentan.

Esta situación aparentemente razonable, dado que el operador conserva todo el control sobre el sistema, puede presentar distintos inconvenientes [Rich99]:

- En primer lugar el sistema de monitorización puede inundar al operador con información, a veces redundante e intrascendente, que puede incluso impedirle ver los hechos más relevantes en las situaciones más criticas.

- En segundo lugar, la comunicación con el sistema no es todo lo ágil y sencilla que desearía el operador. Los lenguajes de comandos están pensados para que el sistema los interprete con facilidad; son extremadamente rígidos y obligan al operador a adaptarse a la lógica estricta de la máquina. Se necesitan en consecuencia largos periodos de formación y de práctica con objeto de asimilar la sintaxis y el significado de los comandos, así como la casuística concreta para su correcta aplicación.

- Por ultimo, el sistema no ayuda a los operadores en la toma de decisiones sobre que acciones realizar.

Los frecuentes cambios debidos a la evolución de los sistemas de gestión actuales (nuevas funciones, nuevos comandos y mensajes) junto con la gran movilidad de los operadores dentro del esquema organizativo responsable de la explotación de la red, constituyen elementos adicionales que subrayan la necesidad de mejoras tecnológicas capaces de aportar soluciones a la situación actual.

En este sentido, las técnicas de Inteligencia Artificial (IA) e Ingeniería del Conocimiento vienen experimentándose desde el comienzo de los 80. Se trata de incrementar el grado de autonomía e iniciativa de los subsistemas encargados de la operación y conservación de la red,

de forma que actúen como un ayudante "informador" del operador. Esto implica que sean capaces de filtrar y seleccionar la información, presentándola al operador de forma sencilla y ordenada, de manera que facilite la toma de decisiones, más aun, el sistema debe ser capaz de ayudar al análisis de los posibles fallos, proponer soluciones, colaborar en la reparación y justificar en todo momento las propuestas realizadas.

3.8.2 El Problemas de la Correlación.

Una laguna de la mayoría de las plataformas de gestión de redes de telecomunicaciones es la ausencia de comunicación entre las aplicaciones de gestión de elementos que se ejecutan en la misma plataforma. En la consola de gestión del administrador, pueden aparecer de repente 101 alarmas que indican anomalías en los sistemas. Sin embargo, sólo una es la causa del problema, las restantes 100 son sólo síntomas [Klementtinen99].

En muchos casos, se deja al administrador la tarea de determinar cuál es cual. Este problema ha creado un mercado marginal de software de gestión que correlaciona dificultades de la red y presenta al administrador la información sobre la causa. Algunas veces la falta de integración de datos entre las aplicaciones de gestión dificulta poder encontrar el origen de un problema de la red. Así, se suelen dar situaciones de acusación mutua, que provoca una queja común en los usuarios de las plataformas de gestión.

A pesar de que la correlación es todavía una idea relativamente nueva, diversos productos llevan ya algunos años en el mercado, ofreciendo la denominada correlación entre distintos eventos.

Muchos fabricantes de estas plataformas confían en que este problema se solucione cuando vuelvan a crear sus aplicaciones de gestión de redes utilizando tecnología orientada a objetos. Esta posibilidad, que ya está en desarrollo en compañías como Groupe Bull e IBM/Tivoli, permitirá a los usuarios compartir datos entre varias aplicaciones.

3.8.3 Tareas y Objetivos.

Las actividades realizadas por un operador en el mantenimiento de una red, son las siguientes:

- *Detección de averías.* Consiste en la recogida de cualquier tipo de información (alarmas, reclamaciones de abonado, pruebas rutinarias,...) que indique la posible degradación del servicio.

- *Protección del sistema de red y aislamiento de órganos.* Se trata de aislar el conjunto de órganos o programas en fallo, para evitar que otros elementos de la red se vean afectados.

- *Localización de averías.* El objetivo de esta tarea es identificar la parte o partes constituyentes del sistema que causan el fallo.

- *Reparación de averías.* Conjunto de acciones orientadas al restablecimiento del servicio por sustitución de la unidad en fallo y la verificación del sistema una vez reparado. Consiste en comprobar que el problema ha desaparecido.

- *Restablecimiento del servicio.* Reanudación de los servicios que se suspendieron por motivo del problema detectado.

La realización correcta de las tareas precedentes requiere por parte del operador el manejo de numerosos manuales de operación, distintos tipos de listados, bases de datos e

incluso la cooperación con otros operadores pertenecientes a distintas organizaciones de la red. El principal objetivo del sistema experto es suministrar al operador de red ayuda en las tareas de detección, localización, reparación de averías y verificación. La ayuda facilitada se concreta en:

- Seleccionar y estructurar la información relevante.

- Sugerir hipótesis de fallos, pruebas a realizar, interpretación y verificación de resultados.

- Facilitar la selección de recursos a utilizar en cada momento: instrumentos de medida, componentes a sustituir, etc.

- Informar sobre el uso de herramientas complejas, tanto hardware (por ejemplo analizador de protocolos), como software (por ejemplo bases de datos, o funciones del sistema de control de red).

- Realizar una síntesis del proceso mediante la confección de los boletines de avería.

3.8.4 Colaboración en la Gestión.

Describe las distintas tareas a realizar entre usuario y sistema, indicando además el modo de cooperación entre ambos, es decir, la forma en que usuario y sistema, colaboran para realizar conjuntamente una tarea. Para ello se requieren una distribución de subtareas entre los agentes cooperantes y la definición de los canales de comunicación. La siguiente relación representa el modelo de cooperación para realizar las tareas del sistema:

- *Detección de averías.* Esta tarea será responsabilidad del operador, pero asistido por el sistema.

- *Localización de la avería.* Se puede dividir a su vez en subtareas: obtener datos administrativos y ejecutar pruebas (responsabilidad del operador) y diagnosticar (realizada por el sistema).

- *Reparación de averías.* Requerirá primero determinar la reparación (realizada por el sistema) y ejecución de la reparación (realizada por el operador).

- *Verificación del sistema una vez reparado.* La verificación de la avería supondrá, entre otras cosas, informar al operador del proceso seguido. A esta subtarea se la denomina franqueo de averías y podrá ser responsabilidad del sistema.

Los canales de comunicación pueden ser los siguientes:

- Observaciones sobre equipos, aparatos de medida, etc.

- Información inicial sobre la anomalía en el servicio: diagnóstico de la avería, información sobre los trabajos recientes o situaciones anómalas detectadas.

- Información sobre el dispositivo donde se detecta el fallo, necesario para el seguimiento del problema: dirección IP, tráfico existente, ubicación, etc.

- Interpretación de la información para introducirla en el sistema.

- Datos administrativos en formato entendible por el sistema.

- Plan de acciones para realizar una prueba.

- Observación obtenida.

- Diagnostico.

- Plan de acciones a llevar a cabo para la reparación de la avería.

- Explicaciones y conclusión del proceso de diagnostico.

- Emisión final de documentos, de término de la reparación y solución de problemas.

3.8.5 Métodos y Algoritmos de Razonamiento.

Los sistemas expertos utilizan las técnicas disponibles en la Inteligencia Artificial para emular el comportamiento de los expertos y automatizar las operaciones que estos realizan sobre los sistemas. Determinar cual es la mejor técnica para una aplicación específica es crucial para cualquier producto basado en Inteligencia Artificial. Hoy en día existen muchas técnicas que están siendo y pueden ser aplicadas con resultados satisfactorios en el dominio de las redes de telecomunicaciones.

A continuación se hace un resumen de las distintas técnicas utilizadas por los sistemas expertos realizados para la gestión de redes de telecomunicaciones. Algunas de ellas ya han sido estudiadas a lo largo del trabajo.

3.8.5.1 Sistemas Basados en Reglas.

Hoy en día, el método más utilizado para la construcción de sistemas expertos, consiste en el encadenamiento de reglas. El conocimiento general de cierta área se representa mediante un conjunto reglas, que tendrán relevancia para situaciones particulares, constituida por hechos que se expresan mediante aserciones que se almacenan en una base de datos.

Su principal ventaja es su modularidad, que se traduce en la posibilidad de seguir añadiendo nuevas reglas a medida que se obtiene nueva información. El motor de inferencia se encargará después de encadenar y combinar todas las reglas que se encuentran en la base de datos de conocimiento. Otra gran ventaja es su flexibilidad, que deriva de utilizar una representación modular.

Las reglas se utilizan para examinar un conjunto de datos y solicitar nueva información hasta llegar a un diagnóstico. Hoy en día las reglas, ampliadas con el uso de marcos y con algunos conceptos propios de las redes semánticas, constituyen un método potente para la construcción de sistemas expertos.

Los sistemas basados en reglas se utilizan en una amplia gama de dominio de comunicaciones: la correlación de alarmas, aislamiento de fallos, diagnosis, reparación, mantenimiento en la red, control de tráfico y routing, asignación de recursos, etc.

A continuación podemos ver un ejemplo de reglas:

```
- Sistema de control de servicios para red banda ancha ATM,
Norwegian Telecom Research.

- Si alguien llama a un abonado ocupado, se debe notificar a ambos
abonados del hecho.

IF SETUP. analizado is FALSE and
       <|abonado|>.numero is equal to SETUP.num_llamado and
       <|abonado|>.estado is 'ocupado' and
       <|abonado|>.numero is equal to SETUP.num_llamante

THEN SETUP.analizado is FALSE and
       Create Object abonado_ocupado <|abonado|> and
       Create Object abonado_esperando <<|abonado|>> and
       Notify (abonado_ocupado, "Esperando llamada") and
       Notify (abonado_esperando, "Abonado llamado ocupado")
```

3.8.5.2 Modelo Basado en Razonamiento (MBR).

Debido al carácter dinámico del comportamiento de las redes de telecomunicaciones, este modelo es idóneo para su modelado. Muy usado en dominios de gestión de redes y se encuentra presente en no pocos de los sistemas expertos aplicados a la gestión de redes.

Los principios de MBR, consisten en la representación de un sistema a través de su modelo estructural y su modelo funcional, a diferencia del modelo clásico de sistemas basados en reglas, donde las reglas se basan en asociaciones empíricas (relacionadas a la experiencia sin teoría ni razonamiento), [Hicks90].

Para el caso de los sistemas de gestión de telecomunicaciones, la representación estructural incluye una descripción de los elementos de red y la topología (es decir la conectividad y las relaciones entre los elementos). La representación de la conducta funcional describe los procesos de automatización de tareas. En la actualidad se utilizan en la definición de sistemas físicos, así como la correlación de alarmas, supervisión de red, gestión de configuración, traducción y reconocimiento de voz.

3.8.5.3 Lógica Borrosa.

Debido a la complejidad existente en la gestión de redes, no siempre es posible precisar un modelo completo de estas. A menudo se dan situaciones en las cuales las ocurrencias de alarmas en un equipo, puede ofrecer información incompleta o incorrecta. También suele ocurrir que el conocimiento de las relaciones entre las causas y los efectos sea impreciso. Por otro lado las reglas aportadas por los especialistas pueden ser confusas y a veces inconcretas. Sean los siguientes ejemplos:

```
      - Si el tráfico en el router A es muy alto y el tráfico en el
router B es normal, entonces desviar ¼ del tráfico del router A al router
B.

      - La ocurrencia de una alarma tipo C, a veces indica que el equipo
D tiene fallos.
```

Tal y como se puede intuir las expresiones "muy alto", "normal" y "a veces", son indeterminadas y no se pueden incorporar directamente en un sistema convencional de razonamiento basado en reglas.

La lógica borrosa es la solución que permite el tratamiento de la imprecisión, característica de algunas de las aplicaciones de gestión de redes de telecomunicaciones. Esta se funda en el concepto "Todo es cuestión de grado", lo cual permite manejar información vaga o

de difícil especificación si quisiéramos hacer cambiar con esta información el funcionamiento o el estado de un sistema especifico. Es entonces posible con la lógica borrosa gobernar un sistema por medio de reglas de 'sentido común' las cuales se refieren a cantidades indefinidas.

Las reglas involucradas en un sistema borroso, pueden ser aprendidas con sistemas adaptativos que aprenden al "observar" como operan las personas los dispositivos reales, o estas reglas pueden también ser formuladas por un experto humano. En general la lógica borrosa se aplica tanto a sistemas de control como para modelar cualquier sistema continuo de ingeniería, física, biología, telecomunicaciones, etc.

La lógica borrosa es entonces definida como un sistema matemático que modela funciones no lineales, que convierte unas entradas en salidas acordes con los planteamientos lógicos que usan el razonamiento aproximado. Se fundamenta en los denominados conjuntos borrosos y un sistema de inferencia borroso basado en reglas de la forma " SI..... ENTONCES..... ", donde los valores lingüísticos de la premisa y el consecuente están definidos por conjuntos borrosos, es así como las reglas siempre convierten un conjunto borroso en otro.

Para el problema anterior, se presenta la siguiente cuestión: suponiendo que 100 representa la máxima capacidad de tráfico y definiendo como "tráfico muy alto" a partir de 85, ¿se puede decir 84'9 no es tráfico muy alto, mientras 85,1, sí? La solución la tenemos en la siguiente tabla 3.1, donde se define una función aproximada para el conjunto "*tráfico muy alto*".

Tráfico	**Grado de pertenencia al conjunto**
Sobre 30	0
30 a 50	0.3
50 a 60	0.5
70 a 80	0.7
80 a 90	0'9
90 a 100	1

Tabla 3.1 Definición de un conjunto borroso.

Dada la imprecisión natural de las telecomunicaciones, contiene dominios idóneos para la utilización de la lógica difusa. Algunos de los más característicos pueden ser el aislamiento de fallos, análisis de comportamientos, planificación, etc.

3.8.5.4 Redes Bayesianas.

Técnica para tratar el razonamiento con incertidumbre, su base es el teorema de Bayes, que es el método matemático exacto para tratar las probabilidades. Consiste en una red donde los nodos son hechos ciertos o no y los enlaces entre los nodos son las probabilidades condicionadas de unos hechos con respecto a otros. Propagando las probabilidades a través de la red, se pueden obtener los resultados más probables a partir de los hechos que se conocen (razonamiento).

Constituyen un método interesante para el tratamiento de la incertidumbre, a través de estas, se realiza el proceso de inferencia en aquellos casos en los que la información aportada por el sistema en estudio, es incompleta e inexacta.

3.8.5.5 Pizarra.

Algunos sistemas expertos para gestión de redes, están siendo implementados utilizando la arquitectura denominada de "pizarra", donde el conocimiento está estructurado en módulos que juegan el papel de expertos/especialistas en la resolución de un problema. Dichos módulos se denominan fuentes de conocimiento (FC).

Los problemas se resuelven de forma cooperativa a través de la intervención incremental de cada especialista. La comunicación entre ellos se realiza a través de una memoria común, la pizarra, donde cada especialista obtiene los datos necesarios para realizar su trabajo, y anota los resultados obtenidos, Figura 3.14.

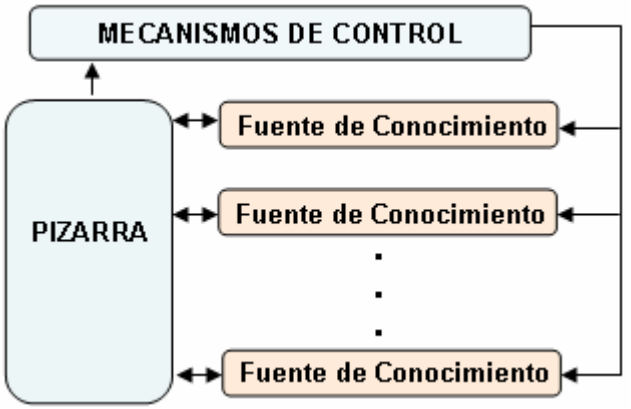

Figura 3.14 Las comunicaciones entre la pizarra y las fuentes del conocimiento

Los resultados pueden ser informaciones que resuelvan definitivamente el problema planteado, o pueden ser subproblemas que requieran la intervención de nuevos expertos. Cada vez que ocurre un cambio en la información de la pizarra, un monitor se ocupa de verificar a que especialista puede interesar la nueva información y convocarle para que coopere.

El modelo admite la existencia de distintos especialistas para la resolución de un mismo problema, con lo cual, pueden aparecer varios candidatos para resolver un subproblema planteado. Se necesita en consecuencia un mecanismo de decisión que resuelva las situaciones de competitividad, seleccionando el más adecuado, de acuerdo a unos criterios bien definidos. El proceso de resolución de un problema se resume a continuación:

- El proceso se inicia mediante la aparición en la pizarra de un problema. El monitor del sistema invoca a varios expertos para su resolución.

- De los expertos se selecciona el más idóneo, que toma de la pizarra los datos necesarios para su trabajo y deja en ella sus conclusiones.

- La anotación de los resultados produce cambios en la pizarra, el monitor buscará a nuevos expertos que puedan aportar soluciones.

- De todos ellos, se seleccionará uno que dejará sus resultados de nuevo en la pizarra. Se continúa con el ciclo, hasta que ya no haya trabajo para nadie o cuando el problema haya sido resuelto definitivamente.

En la siguiente figura 3.15, se muestra el diagrama de bloque de un sistema basado en una arquitectura de pizarra.

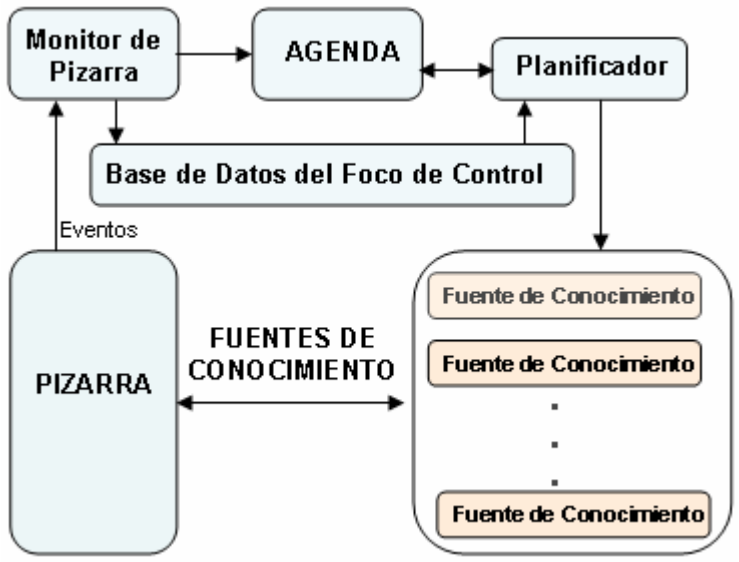

Figura 3.15 Arquitectura de un sistema de pizarra

Las ventajas del modelo desde un punto de vista de ingeniería son importantes: el conocimiento correspondiente a los especialistas se expresa de forma modular y declarativa, admitiendo también redundancia, facilitándose el proceso de experimentación sin afectar a la robustez y facilidad de modificación del sistema.

La solución eficiente de un problema con el modelo de pizarra depende de tres factores:

- *Suficiencia de conocimiento*. Deben existir los especialistas necesarios para resolver el problema.

- *Oportunidad de actuación*. Los especialistas que intervienen deben actuar siempre en el momento adecuado.

- *Eficiencia de actuación*. Los especialistas que intervienen deben hacerlo de forma eficiente.

Como puede observarse, tanto la oportunidad como la eficiencia dependen de la forma en que se seleccionen los expertos. Por ello, supuesta la suficiencia de conocimiento, el control de la resolución de un problema depende de los criterios utilizados para la selección de los expertos.

3.8.5.6 Filtrado.

Cuando se gestiona un gran número de objetos, se cuenta con el problema añadido del gran número de mensajes y alarmas que estos pueden generar. Para evitar la masiva llegada de mensajes al administrador de la red, se crean los filtros inteligentes, encargados de procesar de las alarmas recibidas. Un filtro inteligente necesita de los siguientes requisitos:

- *Funcionales:* Encargado de la comprensión y supresión de notificaciones irrelevantes para el sistema, así como del mantenimiento del orden de importancia de los eventos.

- *No funcionales:* Capacidad de modificación, dar el mayor número de prestaciones posibles y ser escalables (adaptarse a futuras ampliaciones o cambios).

Los sistemas de gestión de red utilizan los filtros para seleccionar cuales son los mensajes y eventos que se mostrarán al operador o administrador de red, de acuerdo con

criterios tales como áreas geográficas, áreas técnicas (cobertura de equipos de comunicaciones) o por el grado de importancia de las alarmas recibidas.

El concepto de filtro es equivalente a la definición proporcionada por ITU-T, la cual lo define como un conjunto de aserciones para los valores de los atributos del objeto gestionado [ITUT91].

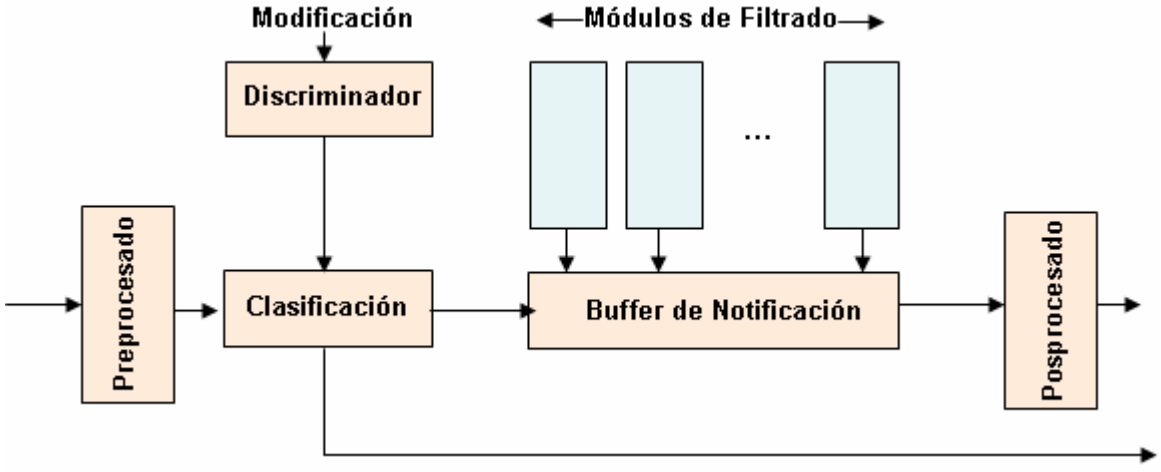

Figura 3.16 Arquitectura de un filtro inteligente.

Esta técnica es apropiada para el tratamiento de las denominadas "*tormentas de eventos*", que se pueden producir en ciertos momentos, en un sistema de gestión como consecuencia de un problema. Consiste en la llegada de cientos o miles de mensajes y eventos que pueden ser generados en un corto periodo de tiempo como consecuencia de un fallo. Este fenómeno es muy frecuente en redes de alta velocidad de transmisión.

```
- Información de la topología:
      atu(mo.Ho_Ac_040:ATU.0), /* adaptation termination unit */
      atu(mo.City_Ac_04:ATU.3), /* managed object */
      line(mo.Ho_Ac_040:ATU.0, mo.City_Ac_04:ATU.3)

- Causalidad entre eventos:
      causes(LossOfSignal, AlarmIndicationSignal)

- Agregación de eventos:
      aggregate(mo.Ho_Ac_040:ATU.0, LossOfSignal,
      mo.City_Ac_04:ATU.3, LossOfSignal, Line_Ac_12, LineFailed)

- Reglas de filtrado:
      causes(E1,E2)&ocurred(MO,E1)&ocurred(MO,E2)->forward(MO,E1)

 - Reglas con tiempo:
      (ocurred(MO1,E1)&ocurred(MMO2,E2)/0.05/0.01->forward(L,E3)
```

3.8.5.7 Event Forwading Discriminator (EFD).

Los Servicios de notificación, se usan cuando un agente quiere informar a un gestor acerca de una notificación generada por un determinado objeto gestionado. Hay que informar del modo de conexión, del objeto que la ha generado, del tipo de notificación y otros parámetros con información adicional. Para este servicio de notificaciones va a existir una clase de objetos gestionados denominada discriminador de eventos (EFD, *Event Forwarding Discriminator*), que implementan un mecanismo de filtrado de las notificaciones que recibe y poseen un atributo que contiene la dirección del agente destino (recomendación X.734), Figura 3.17.

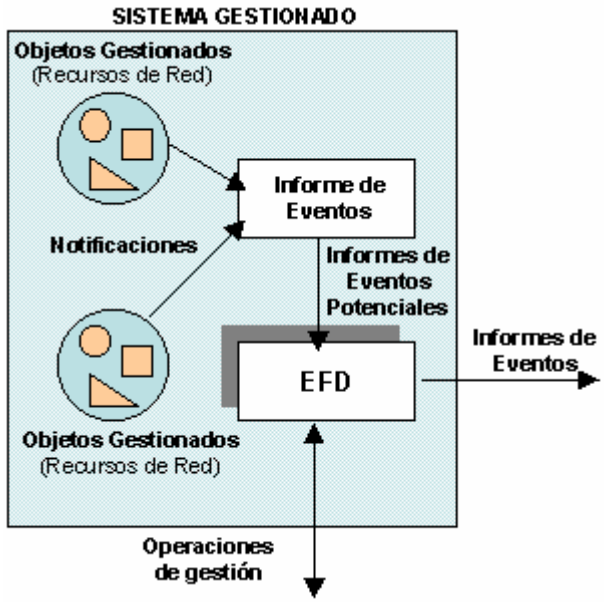

Figura 3.17 El discriminador de eventos.

La recomendación X.734 define los servicios, los protocolos y las unidades funcionales de un sistema de gestión, hasta donde concierne a los servicios de notificación [ITUT93]. Suele existir otro tipo de objetos que registran las notificaciones y poseen un mecanismo similar de filtrado, pero en este caso, para almacenar o no las notificaciones. El modelo de gestión OSI realiza gestión por notificaciones y es por tanto este servicio, uno de los que más caracteriza a la gestión OSI.

3.8.5.8 Razonamiento Basado en Casos, (CBR).

Algunos autores la proponen como una técnica alternativa al razonamiento basado en reglas, en este caso la unidad básica de conocimiento será un caso y no una regla. Los casos contienen los aspectos relevantes de episodios pasados que se almacenan en una base de datos, que serán recuperados y adaptados para la resolución de nuevos problemas. La experiencia obtenida con estos nuevos problemas, constituyen nuevos casos que son añadidos a la base de datos, para su uso posterior.

Así el sistema, por sus propios medios, es capaz de adquirir nuevos conocimientos sin necesidad de la intervención de los expertos humanos. Otra característica importante es su capacidad de modificar el comportamiento futuro aprendiendo de los errores producidos, pudiendo construir la solución a problemas mediante el procesado adaptativo de problemas antiguos a la situación actual de fallo.

El problema de la adaptación de casos al problema en curso, se resuelve aplicando una técnica denominada "*adaptación parametrizada*", basada en la expedición de un "*trouble ticket*" o partes de incidencia, que contiene una serie de relaciones entre las variables que describen el problema y las variables que especifican la solución correspondiente.

A continuación se muestra la estructura de un parte de incidencias:

```
Síntomas
      Servicio (valores);
      Classificación(sin_servicio, problema_QoS);
      Usuario (máquina, etc.);
      Tiempo (instante en que se reconoce el problema);
      Descripción (texto libre);
```

```
Actividades de diagnosis (pruebas)
       Actividad(es): (conj. de actividades);
       Parámetros: (conj. de objetos);

Fallo
       Fallo: (fallos detectados).
       Parámetros (conj. de objetos).

Actividades de reparación
       Actividad(es): (conj. de actividades);
       Parámetros: (conj. de objetos);
```

Los pasos necesarios para el proceso de diagnosis con partes de incidencias, se detallan a continuación:

1.- Generación de partes "*maestros*" de incidencias (partes completos).
- Basándose en descripciones de productos.
- Basándose en experiencias previas.

2.- Diagnosis para un Parte de problemas PP1, con síntoma S1.
Se Recuperan todos los partes relacionados PM1. ... PMn con el mismo síntoma.
Para cada parte maestro PMi:
2.1. Sustituir todos los parámetros y ejecutar la actividad de diagnosis.
2.2. Si no falla (no indica fallo), pasar al siguiente PMi.
2.3. Si falla (indica fallo).
- Si PMi sin fallo asociado, generar un ticket interno MPIi describiendo el test negativo.
- Si tiene fallo y actividad reparadora, realizamos la actividad reparadora.

2.4. Si ningún PMi puede ser obtenido, llamar a un experto humano.

En la siguiente figura 3.18, se observa un ejemplo gráfico:

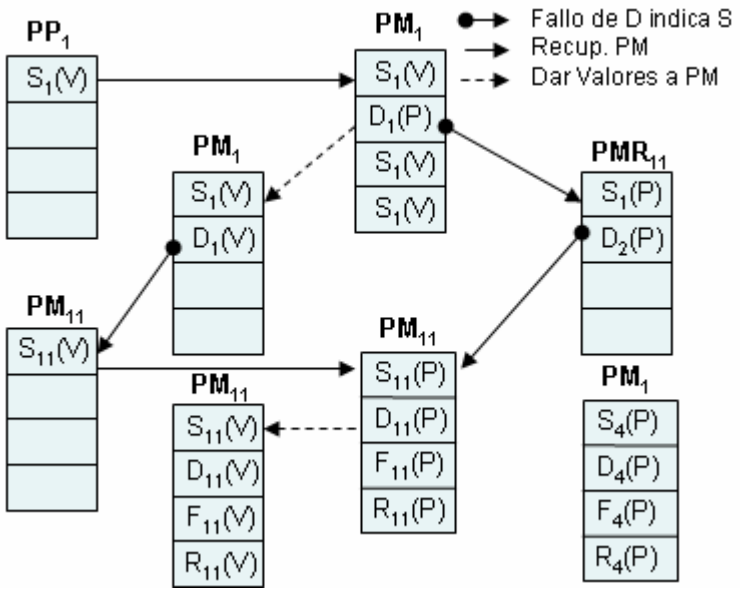

Figura 3.18 Ejemplo de parte de incidencias.

3.8.5.9 Gestión Distribuida.

Con el crecimiento de las redes de comunicaciones, tanto en tamaño como en complejidad, puede ser necesario dividir la red en un cierto número de dominios de gestión (geográficos, funcionales u organizacionales), para así obtener unos requerimientos óptimos de operación y mantenimiento.

La adopción de la arquitectura distribuida para la gestión, facilita la implementación de gestión distribuida y justifica el desarrollo de algoritmos distribuidos. La red se divide en dominios cada uno de los cuales a su vez, es gestionado por su propio centro de gestión. Cada centro de gestión tiene una visión limitada del estado de los demás dominios, mediante el intercambio información entre gestores de dominios diferentes. La gestión distribuida aporta a la gestión de telecomunicaciones, entre otras las siguientes características:

- Resolución cooperativa de problemas.

- *Solución adecuada en entornos distribuidos*: permite compartir modelos de red, aunar criterios de expertos e interoperabilidad entre aplicaciones.

- *Diferentes paradigmas de interacción:* sistemas de pizarras y sistemas de contratos.

- *Diferentes iniciativas de estandarización para interoperabilidad software:* KQML (Knowledge Query and Manipulation Language), Lenguaje de Solicitud y Manipulación de Conocimiento para la comunicación entre agentes y KIF (Knowledge Interchange Format), Lenguaje para el intercambio de conocimientos. Ambos de DARPA y resultantes de investigaciones sobre técnicas, métodos y herramientas para compartir y reutilizar conocimiento entre sistemas [Tiwana99].

La Inteligencia Artificial Distribuida se centra en la resolución de problemas mediante la aplicación tanto de técnicas de la Inteligencia Artificial como de múltiples resolvedores de problemas. Una definición mínima de un sistema IAD establece que debe incluir al menos dos agentes, que tales agentes tengan cierto grado de información y/o autonomía, y que parte de los agentes considerados sean sofisticados en un sentido IA (capacidad de razonamiento, planificación, etc.).

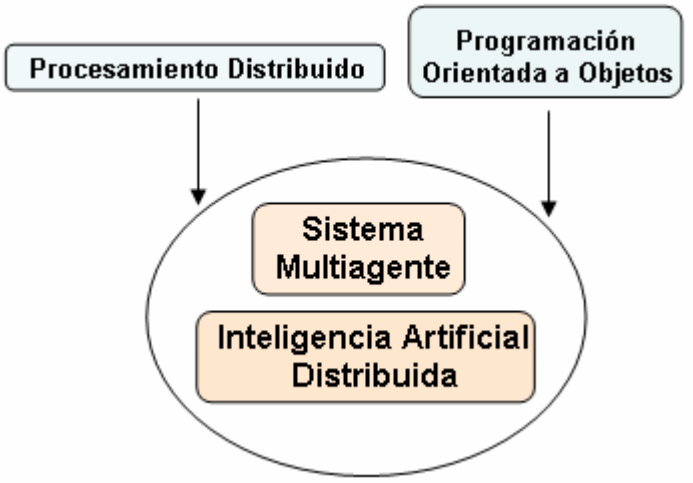

Figura 3.19 Visión de la gestión distribuida.

Se han caracterizado los siguientes problemas básicos en IAD:

- Describir, descomponer y asignar los problemas a los distintos agentes inteligentes.

- Comunicación entre los distintos agentes, lenguaje y protocolos necesarios para el correcto traspaso de la información.

- Asegurar que los agentes actúen racionalmente cuando se toman decisiones o se ejecutan operaciones de control, equilibrando los efectos globales de las decisiones locales y evitando interacciones perjudiciales.

- Diseño de agentes individuales de manera que puedan representarse y razonar acerca de las acciones, los planes y el conocimiento de otros agentes, de forma coordinada.

- Reconocimiento de puntos de vista diferentes e intenciones conflictivas.

Para ser más precisos, la investigación en IAD se centra en cómo diseñar agentes automáticos para que interactúen eficientemente. Por tanto, su principal objetivo es el de la coordinación de planes y no tanto el de la planificación en sí, más propio de la IA. ¿Cómo son resueltos tales problemas en IAD?, básicamente siguiendo distintas técnicas de metaplanificación. Por metaplanificación entendemos cualquier procedimiento de coordinación, desde distintas variantes de planificación hasta organizaciones y negociación. Estas técnicas pueden ser:

- ***Centralizadas:*** es sólo un agente el que genera un meta-plan a ser ejecutado por un colectivo de agentes. Su lema sería ``uno planifica, todos ejecutan".

- ***Descentralizadas:*** tanto la ejecución como la formación de los meta-planes está repartida entre los agentes. Su lema sería ``*todos planifican, todos ejecutan*".

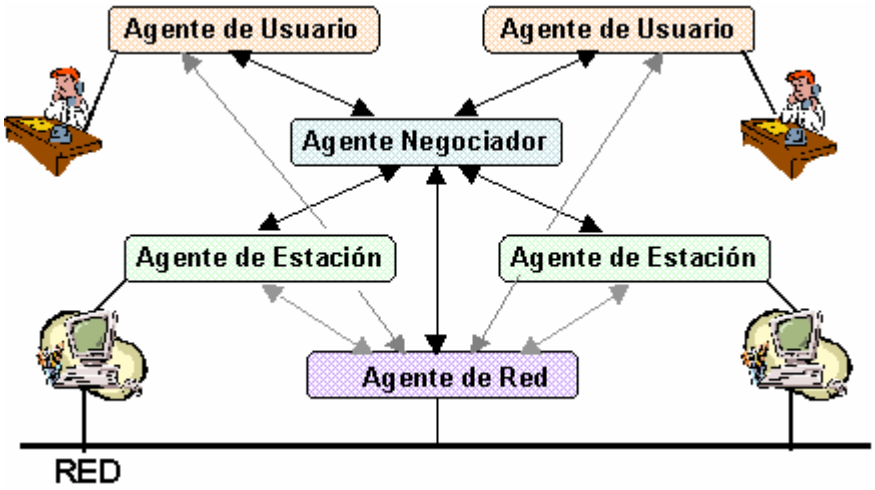

Figura 3.20 Sistema genérico de negociación.

Algunos ejemplos de las aplicaciones de la IAD más próximas a las telecomunicaciones:

- *Diseño de red:* Varios sistemas expertos trabajando de forma débilmente acoplada, utilizando negociación o pizarra.

- *Gestión de red:* Con un agente en cada unidad (segmento/estación/...), se puede conseguir realizar las siguientes operaciones:

  • Reconfiguración dinámica (p.ej. reestablecer rutas).
  • Gestión de fallos.
  • Gestión de prestaciones.
  • Integración de sistemas (interoperabilidad).
  • Búsqueda de información.

• Secretaría inteligente.

3.8.5.10 Redes Neuronales.

Desde el punto de vista de las aplicaciones prácticas de las RNA, su principal ventaja frente a otras técnicas reside en el procesado paralelo, adaptativo y no lineal. Se han desarrollado aplicaciones de RNA para fines tan variados como visión artificial, procesado de señales e imágenes, reconocimiento del habla y de caracteres, sistemas expertos, análisis de imágenes médicas, control remoto, control de robots, inspección industrial y exploración científica y últimamente también en la gestión de redes de comunicaciones.

Lo más interesante es que la propia estructura y modo de operar de las redes neuronales artificiales las hace especialmente interesantes y fáciles de implementar sobre multiprocesadores, algo que puede resultar bastante más complicado cuando se trata de adaptar algoritmos tradicionales sobre dichas máquinas. Conociendo los fundamentos y los algoritmos de aprendizaje de las redes neuronales artificiales y un poco de programación, es posible desarrollar programas específicos para la gestión de redes.

En la gestión de redes se están aplicando en tareas de clasificación e interpretación, seguridad, puesta a punto, ingeniería del conocimiento, control de tráfico y routing, asignación dinámica de canales, detección de fraude, configuración de redes y reconocimiento de voz.

3.8.6 Comparativa de las Distintas Técnicas.

Cada una de las herramientas estudiadas tienen sus ventajas y desventajas, y en la práctica un sistema experto puede utilizar más de una técnica para así obtener las ventajas que cada una de ellas ofrece. Por ejemplo es normal encontrar sistemas basado en reglas y redes neuronales, reglas y en razonamiento, reglas distribuidas, razonamiento y lógica difusa, etc. Además cuando varios sistemas expertos trabajan de forma conjunta, se puede dar el caso de acoplamientos de distinta índole. Así tendremos los casos:

- *Débiles:* Cuando las salidas de uno es la entrada de la otro. Se puede observar sistemas expertos basados en reglas cuyas salidas sirven de entrada a un sistema basado en casos.

- *Fuertes:* Cuando se entremezclan las técnicas. Por ejemplo puede haber un sistema basado en reglas difusas.

-*Hermética:* Cuando se comunican en todo momento las dos técnicas. En el paradigma de pizarra se permite la comunicación de varias técnicas entre sí, durante la resolución del problema actual.

Para finalizar con este tema, hay que destacar que ninguna de las técnicas estudiadas es efectiva en todos los dominios, sino que cada una tiene dominios donde su uso es mejor que en otros, por lo tanto será de vital importancia localizar cual es la técnica mejor para el dominio sobre el que se quiere desarrollar el sistema experto.

3.9 Telecomunicaciones, Aplicaciones Inteligentes.

Las redes de telecomunicaciones son sistemas sumamente complejos que requieren un alto grado de fiabilidad y disponibilidad. La gestión efectiva de estos sistemas es un elemento complejo y crítico, para lo cual se requiere del uso de tecnologías llamadas inteligentes, entre ellas los sistemas expertos. Este apartado describe algunos sistemas expertos, cuyo ámbito de aplicación se enmarca dentro de la gestión de las Redes de Telecomunicaciones.

En la imagen siguiente se muestra esquemáticamente los distintos dominios de aplicación de los sistemas expertos, en las que hemos dividimos la gestión de redes de telecomunicaciones.

Figura 3.21 Visión de los sistemas expertos en la gestión de redes

En el primer apartado se aprecian las cinco áreas funcionales, basadas en el modelo de gestión OSI y en el cual se centran la mayoría de las aplicaciones estudiadas en este tema. Sobre los demás apartados, daremos unas breves pinceladas, sin caer en el error de creer por ello que son de menor importancia.

3.9.1 Aplicaciones de Gestión de Averías.

Se refiere al conjunto de funciones relacionadas con la gestión de redes y que son necesarias para la detección, diagnóstico, recuperación y corrección de fallos en los elementos que conforman una red de telecomunicaciones. El objetivo principal de estas aplicaciones será por tanto, dar fiabilidad y calidad al funcionamiento de la red. Algunos de las funciones que se pueden incluir:

- Detección y análisis de los mensajes de error y ofrecer alternativas de funcionamiento, en caso de errores que afecten a determinados servicios.

- Creación, mantenimiento de una base de datos con información histórica, útil para la gestión futura.

- Diagnosticar y aportar distintas posibilidades de solución al problema en curso, así como la restauración de los servicios afectados y resolución de problemas en los equipos correspondientes.

A medida que la red cuenta con mayor número de dispositivos, se hace cada vez más necesario el uso de estas aplicaciones, que evitan en lo posible, fallos que disminuyen la productividad de la red. Por ello, la investigación está cada vez más presente en campos de estudios como la detección de fallos, la diagnosis y restauración de servicios. A continuación se muestran algunos de los sistemas expertos actuales, aplicados en este campo de la gestión.

3.9.1.1 Max & Opti-Max, Localización de problemas de conexión.

El sistema *Max* es considerado un experto administrador de mantenimiento de líneas telefónicas, desarrollado por NYNEX a mediados de los 90. Es el encargado de detectar problemas de conexión telefónica entre el aparato terminal de un cliente y la centralita de zona, para ello debe supervisar la línea de conexión entre ambos [Merz96].

Max es un sistema experto basado en reglas, su motor de inferencia utiliza reglas de inferencia, que emite diagnósticos de la red en función de los resultados obtenidos por un test realizado en la línea del cliente. Este test electrónico, contiene información referida a la propia línea de conexión y a los equipos que se encuentran conectados a ella. El sistema experto determina en donde se encuentra el problema y selecciona el tipo de solución aplicable para su inmediata resolución.

Max ha sido capaz de reducir el número de diagnósticos incorrectos por encima de sus predecesores, utilizando una mejor metodología. Por contra, cuenta con el problema de que la actuación del sistema se ve gravemente afectada por las características locales del lugar donde actúa, por lo que previamente a su utilización hay que proceder a la reconfiguración de una gran cantidad de parámetros, que serán de utilidad al motor de inferencia para obtener el diagnóstico y estimar la solución oportuna.

Este proceso no es fácil y consume bastante tiempo, por esta razón se ideó el sistema ***Opti-Max***, que cuenta con la novedad de la reconfiguración automática de parámetros, previa a cualquier medida. Se hace de forma automatizada por el propio sistema, que obtendrá por si sólo los valores apropiados para lo parámetros necesarios.

Para ello utiliza un método de *"entrenamiento por ejemplos",* es decir cada uno de los problemas que se pueden presentar incluye un ejemplo asociado, que dispone de una descripción del problema y la solución correspondiente ofrecida por un experto en el dominio. El algoritmo de búsqueda utilizado para la localización de los valores apropiados, para el conjunto de parámetros en cada caso, es el *"Ascenso a la Colina"*, estudiado en el apartado 3.4.6. El sistema *Opti-Max* realiza un tipo de descubrimiento automatizado del conocimiento, lo que le servirá para la resolución de problemas futuros.

3.9.1.2 Trouble Locator, localizador de problemas en una red.

La Compañía Pasific Bell dispone de un sistema inteligente para localizar problemas en el nivel físico de una red local de telefonía, este sistema utiliza la información generada por un test nocturno, que sirve para determinar potenciales defectuosos y equipos en los que se registran fallos [Weiss98].

El sistema utiliza redes Bayesianas e inferencia Bayesiana para el manejo de la incertidumbre presente en el problema actual. El sistema comienza generando un diagrama de la topología de la red local y a partir de esta genera una red Bayesiana, donde cada nodo de la red corresponde a un estado que contiene información (posibilidades de fallos) de cada uno de los componentes de la red. Para la emisión del diagnóstico, el sistema tiene en cuenta la

información histórica acerca de los componentes, además de los datos obtenidos en el test nocturno.

A través de la red, se propaga un mensaje de diagnóstico para el problema dado, hasta que se alcance el equilibrio en esta. Llegado este punto se emite un informe con los posibles elementos causantes del fallo. Este sistema se utiliza como herramienta para mantenimiento preventivo y como sistema de soporte a la toma de decisiones en casos de averías.

3.9.1.3 ESS-ES, gestión de red para conmutadores 4ESS.

La mayoría del tráfico que fluye a través de redes AT&T, está soportado por conmutadores del tipo 4ESS. Esto hizo necesario una herramienta que facilitara la labor administrativa de dichos equipos. El sistema experto 4ESS-ES es el responsable de su supervisión y mantenimiento, contando para ello con un conjunto de programas de testeo y filtrado de alarmas, que se ejecutan en los propios equipos [Hopgood01].

El sistema experto 4EES-ES, es considerado de primera generación, fue desarrollado a principios de los 90 usando las versiones C5 y C del lenguaje basado en reglas OPS-5[5]. Está basado en un conjunto de reglas adquiridas por los expertos del dominio, a través de los años de experiencia. Cuenta con los problemas típicos de los sistemas expertos de primera generación, dificultades presentes en el mantenimiento y la modificación de la base de conocimientos. Este sistema fue rediseñado en 1996 utilizando técnicas híbridas orientadas a objetos y reglas basadas en ejemplos.

3.9.1.4 Sistema CRITTER.

Utilizado para la diagnosis y detección de fallos en red, el sistema CRITTER utiliza el paradigma de razonamiento basado en casos (CBR) sobre un sistema de registro de problemas simples (trouble ticket). Cada uno de los registros (ticket) de problemas constituye un caso [Lewis93A] [Lewis93B]. El componente CBR del sistema provee los mecanismos necesarios para la recuperación de un ticket, adaptarlo si es necesario y generar una solución recomendada al problema. El CBR también es el encargado de añadir los nuevos tickets a la librería de casos.

El primer paso consiste en obtener la información sobre el problema en estudio y crear el ticket correspondiente. Esta información se puede obtener manualmente por el usuario o de forma automatizada través de una aplicación que pregunte al sistema. Algunos de los campos se utilizan para propósitos de gestión, tales como la prioridad y el estado del registro, mientras otros se utilizan para describir la naturaleza del problema, como el nombre, tipo de equipo, dirección IP, etc. A continuación se muestra un ejemplo de un ticket de problema:

[5] Este sistema experto posee un motor de inferencia basado en reglas.

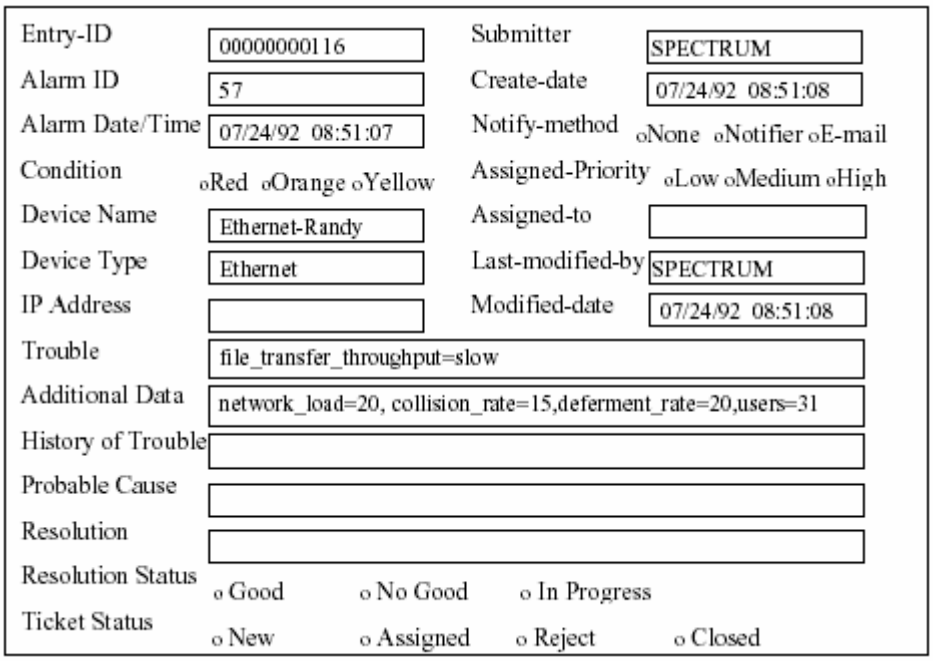

Figura 3.22 Ejemplo de un registro de incidencia.

La recuperación del conjunto de registros almacenados que contiene características similares al problema en curso, se realiza de manera transparente al usuario, haciendo uso de *determinadores,* estos contienen información de relevancia que relaciona las clases de problemas de la red y los conjuntos de atributos de los tickets. Un determinador permite que el sistema localice los registros de incidencias con más probabilidad de contener estrategias que nos lleven a la solución del problema en curso. Véase el siguiente ejemplo:

> *La solución para el problema "la salida de transferencia de archivos es lenta" se determina observando el ancho de banda, la carga de colisiones y la tasa de retardo de paquetes.*

Un conjunto inicial de reglas puede ser ofrecido por los especialistas del dominio o se pueden extraer de un documento de diagnósticos. Estas reglas puede que no sean perfectas, pudiendo servir como solución inicial a un problema y mejoradas con el uso continuado del sistema.

Una vez que los registros afines a nuestro ticket han sido recuperados, el registro con mayor similitud se selecciona. Si este posee un emparejamiento total con los campos relevantes del ticket de nuestro problema, entonces la solución es idéntica en ambos casos. Por el contrario si el emparejamiento total no es posible, entonces se le aplica una técnica de adaptación parametrizada o por abstracción. Las soluciones potenciales encontradas por el sistemas se le presentan al usuario, el cual podrá inspeccionar las soluciones y se le brinda la posibilidad de adaptarlas manualmente si lo así lo estima. A continuación se muestra un ejemplo de adaptación.

Caso Recuperado

```
Problema: Salida_Transferencia_Archivo=F
Datos Adicionales: Ningunos
Solución: A=f(F), Ajustar_Carga_Red=A
Estado de solución: Bueno
(a)
```

Caso Actual

```
Problema: Salida_Transferencia_Archivo=F'
Datos Adicionales: Ningunos
Solución: A=f(F), Ajustar_Carga_Red=A'
Estado de solución: Bueno
(b)
```

Donde *F'* y *A'* tendrían la misma relación que *F* y *A* para el problema recuperado. Habría varios modos de representar la función *f*, la manera más simple sería una tabla donde aquellos valores de *F* que no están en la tabla se calcularían por interpolación. Al final del proceso, el nuevo ticket se almacena en la base de datos de casos, con el fin de dar solución a futuros problemas donde se presenten características análogas.

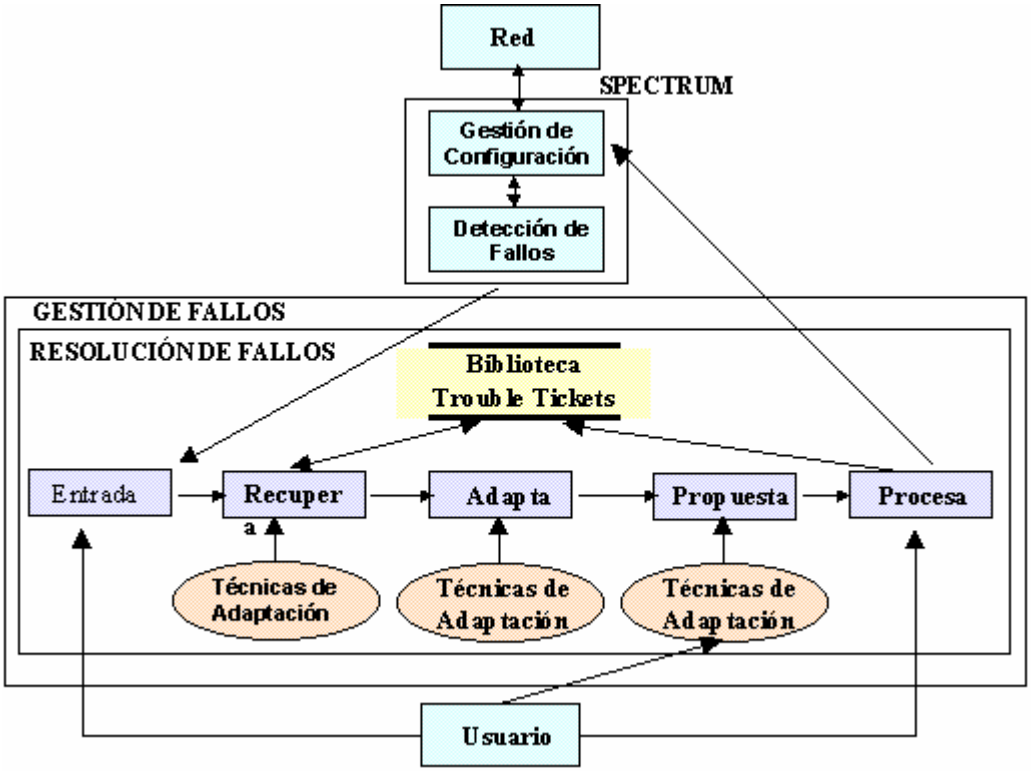

Figura 3.23 Arquitectura del Sistema Critter.

El sistema CRITTER tiene una arquitectura como la represente en la figura anterior, compuesta de cinco módulos:

- *Módulo de entrada*: para la adquisición de la información de los problemas que ocurren en la red. La entrada de información puede ser automática o manual a través del administrador u operador de la red.

- *Módulo de recuperación de ticket*: mediante los determinadores recupera un grupo de tickets desde la librería donde se almacenan, que tienen similitudes al ticket del problema actual.

- *Módulo de adaptación*: Actúa sobre el ticket recuperado que más se asimila al del problema en curso. Si el ticket recuperado tiene sus campos análogos al ticket del problema que se quiere resolver, entonces la solución ofrecida por el ticket recuperado se aplica directamente al problema. Si por el contrario existen diferencias, este módulo se encargara de proceder a las adaptaciones necesarias.

- *Módulo Proposición:* Presentará al usuario la solución potencial al problema encontrada por el módulo anterior, permitiéndole la inspección y las adaptaciones manuales que crea necesarias.

- *Módulo de proceso:* Pone al día la librería de ticket.

3.9.1.5 COMPASS.

Central Office Maintenance Printout Analysis and Suggestion System. Analiza los mensajes impresos por la central No. 2 EAX de GTE [Prerau90].

3.9.1.6 SMART.

Switching Maintenance Analysis and Repair Tool. Es un sistema construido por BellCore para el diagnostico de la central IAESS, que ayuda a localizar el fallo y sugiere acciones para su reparación [Loberg88].

3.9.1.7 ECXpert.

Desarrollado por Lucent Technologies [Nygate95], facilita la gestión de red mediante la detección rápida de problemas y su inmediata reparación. Para ello se analizan las alarmas anticipándose a la aparición de los problemas y toma decisiones sobre las operaciones a realizar para su posible solución. ECXpert es parte de la familia de productos Total Network Management (TNM), que constituye una plataforma de gestión de red actualmente utilizada por grandes compañías, como PacBell, Bell South, etc.

Los grandes centros de telecomunicación, en los que puede haber varios administradores y operadores de red, puede recibir simultáneamente un gran número de mensajes (aprox. 15.000 por hora). Los operadores son incapaces en la mayoría de las ocasiones, de captar toda la información que reciben, así como discernir cual es la verdaderamente importante. El paquete ECXpert fue concebido para automatizar e interpretar toda las alarmas recibidas, filtrando aquellas que no contienen información trascendental y mostrando tan sólo aquellas que son importantes para el tratamiento de un problema. De esta forma los operadores reciben únicamente las alarmas con información importante sobre los problemas, permitiéndoles así centrarse en los problemas cruciales para el sistema.

ECXpert está constituido por cuatro módulos principales:

- *Compilador de grupo de correlación:* Escrito en prolog, convierte las reglas de correlación definidas por un experto en instrucciones Prolog.

- *Interfase de usuario*: Facilita información sobre las alarmas.

- *Proceso Test de correlación:* Comprueba las relaciones entre las distintas alarmas, para evitar mostrar alarmas que dependen de otras más importantes, que si serán mostradas.

- *El proceso de correlación*: realizado en C++, ejecuta el algoritmo de correlación y manipula la base de datos que contiene el árbol de correlación.

ECXpert modela la situación del mundo real por medio de la correlación de eventos. Cuando llegan mensajes (un fallo de la red), se construye un árbol que representa la dependencia entre los distintos eventos. Puede que ciertos eventos puedan ser causados como resultado directo de otros eventos, los cuales pueden representar la verdadera causa del fallo en la red.

Poniendo en correlación los eventos, los problemas importantes pueden ser aislados y pueden ser tratados por los operadores con una prioridad mayor. Una vez vistos cuales son las alarmas con mayor prioridad se envían los algoritmos de reparación para arreglar lo antes posible el problema y evitar un posible aumento de este, con lo que eso supondría para la red (caída total de servicios, zonas incomunicadas, perdidas de información, etc.). Este sistema

experto utiliza un motor de inferencia basado en reglas y un método de ejecución de encaminamiento hacia delante (forward-chaining).

3.9.1.7 ANSWER.

Automated Network Surveillance with Expert Rules. Sistema experto desarrollado por AT&T [Singhal98], es responsable de supervisar los conmutadores 4ESS, presentes en la mayoría de redes AT&T. Utiliza técnicas híbridas orientadas a objetos y reglas basadas en ejemplos.

Cuando un componente conectado a un conmutador experimenta un problema, el conmutador genera una alarma que se envía a uno de los dos centros de control técnico de AT&T. En el centro de control técnico, la alarma se almacena en una base de datos y es remitida a ANSWER. Este analiza la alarma utilizando reglas aportadas por un experto en el dominio. Si ANSWER determina que la alarma requiere alguna acción, remite una alerta describiendo el problema a un técnico para que realice el proceso.

Los conmutadores 4EES conectados en la red AT&T, pueden generar mensajes en número superior a 100.000 por semana, es importante que los mensajes que no revisten importancia sean filtrados para que así no produzcan alarmas innecesarias.

Uno de las funciones más importantes de ANSWER es distinguir entre los fallos que son recurrentes (que se repiten) y los que no. El principal objetivo es minimizar el número de servicios afectados por un incidente, como un bloqueo o caída de todas las conexiones y evitar de esta forma la caída de la productividad en la red, así como el aumento de los costes de personal y de mantenimiento.

3.9.2 Gestión de Contabilidad.

Permite determinar y localizar cargos y costos por el uso de los recursos de la red. La gestión de contabilidad suministra procedimientos para:

- Informar a los usuarios sobre los costos incurridos mediante el uso de software para reporte de eventos y manipulación de datos.

- Permitir establecimientos de límites de contabilidad y asociar calendarios de tarifas al uso de los recursos gestionados.

- Permitir la combinación de costos cuando se involucren múltiples recursos de comunicación para alcanzar un objetivo dado.

3.9.2.1 APRI, predicción de deudas incobrables.

La industria de las telecomunicaciones cada año, incurre en millones de dólares de deuda incobrable. El sistema APRI (*The Advanced Pattern Recognition and Identification)*, desarrollado por AT&T predice la probabilidad de caer en nuevas deudas sin posibilidad de cobro, en función del estudio de una base de datos histórica que contiene información referente a impagos [shen01].

La salida de APRI proviene de a un sistema de soporte de decisión, que puede realizar acciones variadas tales como restricciones e incluso bloqueos de llamadas antes de que esta pueda ser completada. APRI construye automáticamente modelos de redes Bayesianas para clasificar los problemas, para lo cual hace uso de una gran base de datos. Las redes Bayesianas son escogidas para este problema, debido a la inherente naturaleza probabilística de predicción del problema y la gran distribución de clases.

Llegado este punto, hay que tener muy en cuenta el coste desigual de una clasificación errónea de la información, es decir, etiquetar una cuenta equivocadamente como incobrable versus etiquetándola equivocadamente como cobrable.

3.9.3 Gestión de Configuración.

Facilita a los administradores ejercer control sobre la configuración de los componentes de red. Cambiar la configuración podría servir para disminuir la congestión, aislar fallos o hacer cambios que necesiten los usuarios. La gestión de configuración suministra procedimientos para:

- Fijar y modificar los parámetros que controlan la operación rutinaria de los componentes de red y el software.

- Asociar nombres con objetos y con conjuntos de objetos gestionados.

- Inicializar y cerrar objetos gestionados.

- Recolectar datos concernientes al estado actual de los recursos.

- Obtener anuncios de cambios significativos en la condición del sistema.

3.9.3.1 ACE (gestión de configuración).

Es un sistema experto desarrollado por AT&T, que ayuda al experto en la detección y diagnostico de fallos en los cables instalados en la planta exterior y red de acceso telefónico [Liebowitz98].

ACE es un analizador que se ejecuta en segundo plano (background), realizando preguntas a una base de datos donde se guardan los resultados de los testeos diarios de la red. Se buscarán los patrones que indiquen que puede existir un problema en la red. ACE genera una salida con el diagnóstico al problema, junto con los detalles referidos al problema que se ha observado en la base de datos.

Es un sistema basado en reglas, que utiliza la estrategia de encadenamiento hacia delante, que rompe el problema global en subproblemas independientes. Cada subproblema por consiguiente se puede resolver de forma independiente y los distintos resultados ensamblados para dar una solución completa al problema inicial.

3.9.3.2 NEMESYS.

Network Management Expert System (Routing). Está diseñado para evitar congestiones en la red de AT&T. Recoge la información necesaria de la red recomendando las acciones a seguir para mejorar la calidad de servicio en casos de sobrecarga [Celestino93].

3.9.3.3 NMCS.

Network Monitoring and Control System. Desarrollado por GTE, explora bases de datos de gestión avanzadas y hace uso de avanzadas interfases gráficas, que interactúa con los usuarios para la recolección de datos y la evaluación de estados [Ross88].

Es un Sistema Experto basado en reglas, utiliza la interfaz de usuario para la adquisición de los conocimientos necesarios para la configuración y optimización de los recursos de la red. La investigación futura tiende a integrar este tipo de sistemas dentro de grandes plataformas

software que desarrollan múltiples funciones de gestión y administración de red. A continuación se muestra un ejemplo de regla NMCS:

```
SI <enlace X/Y está caído>
   <enlace X/Y es siempre una ruta final desde el nodo X>
ENTONCES
   <ordena al nodo X que aplique un 100% de cancelaciones al enlace X/Y>
```

3.9.3.5 XCON.

Sistema experto para configuración de ordenadores. Su usuario fue la *Digital Equipament Corporation (DEC)*. El cometido de XCON sería configurar todos los ordenadores que saliesen de la DEC. El proyecto presentó resultados positivos y se empezó a trabajar en el proyecto más en serio en diciembre de 1978. En abril de 1979 el equipo de investigación que lo había diseñado pensó que ya estaba preparado para salir, y fue entonces, cuando se hizo una prueba real, esperando resolver positivamente un 95% de las configuraciones, este porcentaje tal alto se quedó en un 20% al ser contrastado con la realidad. XCON volvió al laboratorio, donde fue revisado y a finales de ese mismo año funcionó con resultados positivos en la DEC.

En 1980 se instauró totalmente en DEC y en 1984 el XCON había crecido hasta multiplicarse por diez. El XCOM supuso un ahorro de cuarenta millones de dólares al año para la DEC. En 1987 XCON empieza a no ser rentable y los técnicos de DEC tuvieron que actualizar XCOM rápidamente llegándose a gastar más de dos millones de dólares al año para mantenimiento, [Ghosh00].

Sistema experto basado en reglas, contiene una da las bases de reglas más grandes del mercado, por encima de 4000 [Lutsky95].

3.9.3.5 ExSim, Routing.

El sistema ExSim fue desarrollado para ayudar en el proceso de enrutamiento de una red de telecomunicaciones. Utiliza un programa de simulación que representa la red que se está gestionando, y que se compone de los gateways que la conforman y el tráfico existente entre ellos [Tsing-Hwa91].

Una sobrecarga de enrutamiento en una red, puede provocar la caída del rendimiento del sistema. Cuando esto se produce, la tarea principal del módulo de razonamiento consistirá en detectar la información basura y funcionamientos anómalos. Para ello se efectúa la clasificación de los estados de la red y se comparan con los problemas ocurridos anteriormente y que están almacenados en la base de datos de casos.

Este sistema utiliza el paradigma de razonamiento basado en casos, cuenta con la ventaja de que en él, no hace falta la intervención del usuario, el proceso de configuración del sistema es completamente automático.

La descripción de un problema consiste en un conjunto de tablas de enrutamientos de gateways, con información que representa la carga de los enlaces de red, tabla de topológica de la red y estados de los distintos gateways.

Una recuperación comienza cuando el sistema detecta un estado crítico en la red, la información sobre el estado actual se almacena en tablas de enrutamiento y tablas de estados de los gateways. Esta información sirve como patrón, para la localización en la base de casos del conjunto de tablas con características similares. Se elige el caso que más se aproxime, el cual contiene el conjunto de tablas de enrutamiento correcta y la solución correspondiente para

eliminar la sobrecarga de routing en el sistema. La siguiente figura contiene el diagrama de bloques del sistema.

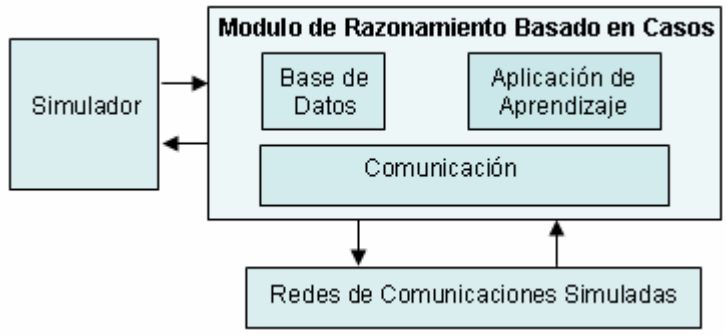

Figura 3.24 Arquitectura del sistema ExSim

3.9.4 Gestión de prestaciones.

Facilita a los administradores de red monitorizar y evaluar las prestaciones del sistema. La gestión de prestaciones pretende suministrar información estadística que permita establecer si un sistema tiene la capacidad de tráfico requerida, si su tiempo de respuesta está acorde con las necesidades, si hay una posible sobrecarga de los recursos o si el sistema está siendo usado en forma efectiva. La gestión de prestaciones suministra procedimientos para:

- Recolectar datos interesantes del nivel de prestaciones actual de los recursos (información estadística).

- Mantener y examinar registros de prestaciones con el propósito de planear y analizar las condiciones de la red.

- Determinar el rendimiento del sistema en condiciones naturales y artificiales.

- Cambiar los modos de operación del sistema, con el fin de realizar las tres actividades anteriores.

3.9.4.1 TASA.

Telecommunication Network Alarm Sequence Analyzer. Es un sistema que extrae información sobre el comportamiento de una determinada red, desde una base de datos que almacena información sobre las alarmas, que se pueden producir en una red de telecomunicaciones [Albus01]. El objetivo principal del sistema es localizar las regularidades en las secuencias de alarmas mediante filtros, localizar problemas y predecir fallos futuros. TASA opera en dos fases distintas:

- *Primera fase*: Se utilizan algoritmos especiales que determinan reglas que describirán la frecuencia de ocurrencias de episodios de alarmas a través de los datos extraídos. Un episodio describe una secuencia de alarmas que se produce en periodo de tiempo determinado. Un ejemplo de regla descrita en un episodio:

```
Si se producen alarmas de tipo A y B en un periodo de 5 segundos,
entonces la probabilidad de que se produzca una alarma tipo C en los 60
siguientes segundos es de 0.7.
```

- *Segunda fase:* La colección de episodios son interactivamente manipulados y ordenados para localizar los episodios de más interés, eliminando los que no merecen la pena. Cada episodio se estudia en función de la prioridad o confidencialidad de las alarmas producidas.

TASA es un sistema basados en reglas, cuando detecta una nueva regla que estima interesante la añade al sistema de gestión de alarmas, para su uso posterior.

3.9.4.2 Scout.

Identificación Fallos de Red mediante Ingeniería del Conocimiento. Desarrollado por AT&T a mediados de los 90, se utiliza para identificar problemas persistentes en las redes de telecomunicaciones, para ello utiliza técnicas de aprendizaje y de correlación [Sasisekharan96]. En una primera aproximación, Scout identifica los problemas, a través de del análisis de los datos extraídos de la observación del comportamiento de la red durante días o semanas. Para ello, se extraen características variantes en el tiempo que resumen el comportamiento histórico de la red, para facilitar la utilización de algoritmos estándar de aprendizaje, (es decir, algoritmos que no son capaces de utilizar razonamiento temporal explícito) [Malheiros98].

Utilizan dos ventanas de tiempo fijas, consecutivas, llamémoslas W1 y W2. El objetivo, es utilizar las distintas medidas realizadas en w1 para predecir los posibles problemas que se produzcan en w2, es decir observar el comportamiento en una de ellas para predecirlo en la segunda. Para realizar esta predicción hay que realizar un gran número de medidas en ambas ventanas de tiempo. Una forma de reducir dicho número, es contar el número de veces que cada característica ocurre dentro de una ventana y utilizar reglas que prevean los comportamientos anómalos en la segunda ventana de tiempo.

Otra versión de Scut, utiliza un mapa topológico de la red en que trabaja, mejorando así su actuación. Al añadir el conocimiento topológico de la red, permite realizar el aprendizaje en intervalos de tiempo más pequeños (de minutos y horas) en lugar de días y semanas, algo que es sumamente importante a la hora de poder predecir problemas y fallos puntuales (agudos). Este conocimiento también le permite al Scut identificar eficazmente el problema, así como la raíz del mismo.

Desde el momento en que fue desarrollado este sistema ha tenido un exitoso incremento de la productividad técnica, en la identificación de la raíz de los problemas y en la prevención de problemas del tipo transeúnte. La arquitectura de Scout está basada en el paradigma de pizarra.

3.9.4.3 Net/Advisor y Net Command

Monitoriza el estado de la red en tiempo real y sugiere las acciones a realizar para mejorar el comportamiento de la misma.

3.9.4.4 NETTRAC.

Desarrollado por GTE Laboratories, el sistema Nettrac se aplica a la gestión del flujo de tráfico a través de una red telefónica. Se realiza asignando un mayor o menor número de recursos de red en función de la demanda de llamadas [Melchiors00]. El control y modificaciones de la configuración para dar una mejor atención a las necesidades actuales, es responsabilidad de un grupo de gestores expertos que se ubican en un punto central de la red.

En una red existe una gran cantidad de nodos que soportan el tráfico de red (Swicht), capaces de dar servicio a millares de conexiones de forma simultanea. Cuando uno de estos dispositivos falla o sufre una caída, todas las conexiones a este nodo se reparten entre los demás de la red. También tenemos el problema de los intervalos horarios de mayor demanda, en los cuales hay que buscar rutas alternativas para satisfacer las necesidades.

El sistema NETTRAC utilizada el paradigma de razonamiento basado en casos (Case-Based Reasoning, CBR). La información de la red es interpretada y formalmente representada, para así obtener una visión general del comportamiento de la red y de los problemas existentes. Cada uno de los distintos casos se almacenan en una base de datos de registros, que será utilizada para la obtener la solución a los problemas que se puedan presentar. Los casos que más se aproximen al problema en cuestión serán recuperados y entre ellos se selecciona el caso potencialmente más efectivo.

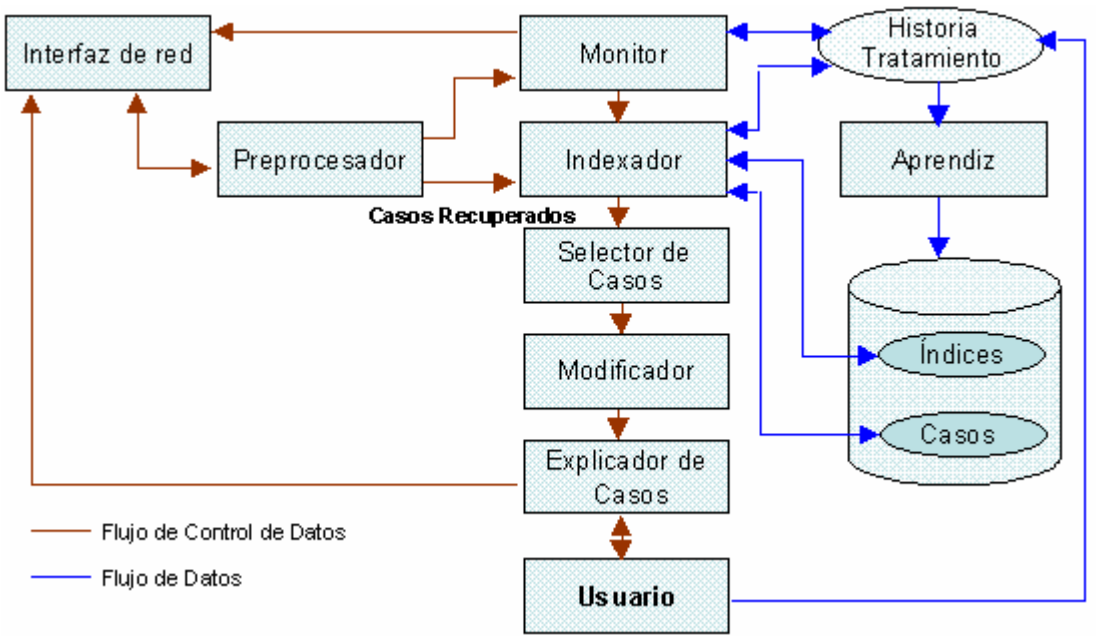

Figura 3.25 Arquitectura del sistema NETTRAC

El preprocesador interpreta la información de la red y la realiza una representación del problema *(PS, Problem Statement),* escoge entre los PS existentes para ver si el problema ya se produjo antes y existe información al respecto. En este caso coge el Problem Statemen y aplica la solución propuesta a la actual, o se lo pasa al monitor para que realice los ajustes o modificaciones en el tratamiento.

La experiencia obtenida del caso recuperado se aplicará a nuestra situación, siendo alterada cuando sea necesario. Si el diagnóstico y tratamiento propuesto es aceptado por el usuario, entonces se emitirán una serie de llamadas de control a los distintos *switches* de la red que servirán para controlar el tráfico. Por el contrario, si el preprocesador detecta un nuevo problema, lo transfiere al Indexador, para que seleccione de la base de datos los casos que más se parecen al nuestro. El selector escoge uno de ellos, el que potencialmente será útil para la resolución del problema.

Este caso se modifica y adaptada a las circunstancias actuales y se le envía al módulo explicador que se lo propone al usuario. Si el usuario no está del todo convencido por la solución aportada, puede solicitar otra. De nuevo el selector escoge otro de los casos seleccionados, e incluso puede realizar una nueva búsqueda para recuperar un nuevo grupo de casos, si el indexador así lo estima oportuno.

Cuando el usuario está de acuerdo con el tratamiento del caso, a través de una interfaz de red realiza los controles sobre el conmutador en cuestión. Todo el proceso de resolución del problema se registra y comunica al preprocesador, con el fin de proceder al almacenamiento

como una nueva entrada en la base de datos que servirá para localizar la solución a problemas de las mismas características.

3.9.5 Gestión de Seguridad.

Facilita a los administradores de red gestionar aquellos servicios que proveen protección en el acceso a los recursos de comunicación. La gestión de seguridad provee soporte para la gestión de:

- Facilidades de autorización.
- Control de acceso.
- Encriptación y gestión de claves.
- Autenticación.
- Logs de seguridad.

Los sistemas expertos desarrollan una detección de ataques mediante la descripción de un conjunto de reglas. Los eventos detectados mediante auditoria del sistema son convertidos para que el sistema experto los pueda entender y solucionar. El motor de inferencia dará los resultados y conclusiones después de usar las reglas y hechos correspondientes. A continuación se describen algunos de los Sistemas Expertos dedicados a la seguridad en redes.

3.9.5.1 IDES.

Intrusion Detection Expert System. Desarrolló por SRI, es un sistema experto que codifica escenarios conocidos de intrusión, así como las vulnerabilidades del sistema y las políticas de seguridad específicas de nuestro entorno de trabajo [Garcia01] [Lai98].

Está dirigido a la detección de ataques externos por usuarios desautorizados, usuarios autorizados que se hacen pasar por otros usuarios y usuarios autorizados que abusan de sus privilegios, traspasando sus cotas de acceso.

3.9.5.2 NIDX.

Network Intrusion Detection Expert-System. Desarrollado por Bell Communication Research [Bauer88], es un prototipo de Sistema Experto basado en reglas, detecta intrusiones para el Sistema de Unix V. Combina conocimiento del sistema designado, historial de la actividad de usuarios y detección heurística de detección de acciones no permitidas.

El resultado es un sistema basado en el conocimiento capaz de descubrir violaciones específicas que ocurren en un sistema determinado. Una característica única de NIDX es que incluye hechos y reglas para un sistema en concreto, por lo cual podemos decir que NIDX es dependiente del sistema operativo sobre el que se trabaje.

3.9.5.3 P-BEST.

Es un sistema experto basado en reglas con razonamiento, que está siendo utilizado desde hace tiempo con resultados satisfactorios. La idea principal consiste en la especificación de las características típicas procedentes de comportamientos malévolos y la supervisión del flujo de eventos generados por la actividad del sistema, para el posterior reconocimiento de la identidad del intruso [Lindqvist99].

Consiste en un Shell programable de sistema de experto de propósito general, muy empleado por usuarios inexpertos, que utiliza un lenguaje básico de definición de reglas.

3.9.6 Otras Aplicaciones Inteligentes

Además de las aplicaciones de gestión, la industria de las telecomunicaciones ha desarrollado muchos otros tipos de aplicaciones inteligentes entre las que se incluyen: márketing y comercialización, aplicaciones de configuración, routing, aplicaciones de escritorio, etc.

3.9.6.1 NetHELP.

Interfaz avanzada de usuario. Desarrollado por Pacific Bell, es un sistema experto que proporciona ayuda a los usuarios de la red. Cuando un usuario está teniendo problemas con la red, ellos pueden llamar al sistema experto, este solicita al usuario información detallada sobre el estado actual de las comunicaciones y le sugiere la/s estrategias ha seguir para la resolución [Derbort91].

El sistema puede estimar la conveniencia de que el problema o los procedimientos de reparación a seguir, por su complejidad pueden no ser apropiados para que el usuario los realice. En este caso el usuario recibe un "*código de identificación*", que le servirá de referencia. A continuación se emite una notificación a la agencia correspondiente, con los datos necesarios para que proceda a la resolución de la incidencia relacionada con el usuario en cuestión. Proporciona funcionalidad las 24 horas del día, detectando y coleccionado informes de las incidencias producidas en la red.

3.9.7 Resumen de Plataformas y Conclusiones.

En la siguiente tabla 3.2 puede observarse un resumen de algunas de las aplicaciones estudiadas, indicando su área de aplicación y la técnica utilizada. Como se puede apreciar, el grueso de los sistemas expertos se han desarrollado para la gestión de averías y utilizan modelos basados en reglas.

	Gestión de Averías	**Gestión de Contabilidad**	**Gestión de Configuración**	**Gestión de Prestaciones**	**Gestión de Seguridad**	**Interfaz Expertata**
Basado en Reglas	***Max & Opti*** *ANSWER* *ESS-ES* *ECXpert*		*ACE* *XCON* *NMCS*	***TASA***	*NIDES* *P-BEST* *NIDX*	*NetHELP*
Redes Bayesianas	*Trouble Locator*	***APRI***				
CBR	*CRITTER*		***EXSim***	*NETTRA*		
Pizarra				*Scout*		

Tabla 3.2 Cuadrante de áreas de gestión y técnicas de IA

En este capítulo se ha efectuado una revisión de las principales aplicaciones para el análisis y gestión de redes de telecomunicaciones, y se ha visto que los sistemas expertos tienen una gran implantación, especialmente en el área de detección de fallos y diagnostico de avería.

La mayor parte de los sistemas expertos están basados en reglas de clasificación, las cuales se obtienen a partir de la experiencia acumulada por uno o varios expertos humanos. No obstante, en los últimos años ha comenzado a aplicarse una nueva metodología, los sistemas

basados en casos (case based reasoning) entre otros, en los cuales el análisis realiza por aproximación al caso más parecido de los existentes en la base de conocimientos.

En un intento de reducir los costes de desarrollo de un sistema de gestión de las redes basado en técnicas de Inteligencia Artificial, para que éstos sean más asequibles a organizaciones pequeñas y medianas, que de otro modo no podrían permitírselos, se han aplicado en los últimos años al análisis de la solvencia técnicas de aprendizaje automático, en las cuales el conocimiento no proviene de expertos humanos, sino que se induce a partir de la información contenida en un banco de datos.

Dentro de éstas es posible destacar las redes neuronales, las cuales constituyen un enfoque que pese a su relativamente reciente aplicación al campo de las telecomunicaciones, también ha originado algunos modelos que están funcionando actualmente en algunas compañías.

Si bien los sistemas basados en computación neuronal están dotados de gran poder predictivo y capacidad de adaptación a un entorno cambiante, presentan el problema de su gran complejidad. Este problema puede ser solventado si se acude a otro de los enfoques del aprendizaje automático, los algoritmos de inducción de reglas y árboles de decisión, que posibilitan realizar una clasificación de los distintos problemas.

Para finalizar, a continuación se ofrecen algunas consideraciones finales a tener en cuenta:

- Hoy en día es normal encontrarse con plataformas software, que realizan un conjunto de funciones, entre las que se puede contar la gestión de redes de comunicaciones, esto ocasiona que a menudo los sistemas expertos vengan integrados dentro de estos paquetes, siendo un elemento importante pero no único.

- Otro aspecto importante, es que cada vez es más usual que los sistemas expertos requieran del acceso a una base de datos con la cual poder compartir información importante.

- La existencia de sistemas expertos que están estrechamente relacionados a otros sistemas propietarios, sin los cuales no puede funcionar, creando la necesidad a los usuarios de realizar un aprendizaje de dichas herramientas.

- También es habitual que los sistemas expertos se comuniquen con otros sistemas expertos y paquetes de software, debiendo disponer de una plataforma de comunicación y envió de mensajes apropiada a las necesidades.

- El otro punto a tener en cuenta, es que la interfaz con el usuario debe ofrecer alta capacidad de interacción, para así facilitar la labor lo máximo posible al equipo humano encargado de su gestión.

- Por último recordar que los sistemas expertos no es la panacea y no siempre resuelven todos los problemas, ofrecen la mejor aproximación a la solución de una serie de alternativas, pero no tienen por que resolver de modo exacto todas las cuestiones que se le presenten.

Capítulo 4

GDMO, Guía para la Definición de Objetos Gestionados

Desde el punto de vista de la gestión de Sistemas de Telecomunicaciones, es usual encontrarse con redes heterogéneas que integran diversas plataformas de gestión y donde equipos de diferentes características y tecnologías pueden estar interactuando de forma conjunta. Es por tanto necesario, uniformizar de alguna manera los distintos recursos de gestión, así como la información de gestión que cada recurso aporta y la forma en que se va a acceder a esta.

En los años 90 ISO e ITUT desarrollan de forma conjunta un modelo para la gestión de redes, que comprende toda una familia de normas que cubren aspectos tan amplios como los protocolos, servicios básicos ofrecidos, modelado de la información de gestión, etc.

El modelo de información de gestión, ofrece una estructura que permite compartir la información de gestión entre los distintos sistemas abiertos, usando para ello el protocolo CMIP. Los recursos reales del sistema de comunicaciones se representan mediante una colección de los objetos gestionados, que pueden ser por ejemplo conexiones, equipos de comunicaciones, ordenadores, etc. Para la definición de estas Clases de Objetos Gestionados, aparecen las Guías para la Definición de Objetos Gestionados, *Guidelines for Definition of Manageed Object*, GDMO. Este lenguaje de especificaciones define los objetos gestionados y la información de gestión, que se transfiere entre los sistemas informáticos, a través del protocolo CMIP de OSI. Utiliza un metalenguaje de plantillas simple basado en ASN.1 (ISO 8824), descritos en el estándar GDMO y que se recoge en la recomendación ISO 10165-4 / CCITT X.722.

El presente capítulo ofrece una descripción de la norma GDMO, sus plantillas y los elementos básicos que las conforman.

4.1 El Lenguaje de Especificación GDMO.

El leguaje GDMO especifica los recursos pertenecientes a una red de comunicaciones y la información relacionada con su gestión. Para cada arquetipo de información existe una plantilla asociada que los define, la denominada Clase de Objetos Gestionados. Cada Clase define un conjunto denominado de objetos gestionados que comparten los mismos atributos, notificaciones, operaciones de gestión y que participan de las mismas condiciones en presencia de estas operaciones.

Existe una serie de principios básicos para la definición de objetos gestionados [1], que buscan servir de orientación a quienes realizan la especificación de dichos objetos, así como favorecer la coherencia entre las definiciones que de ellos se establecen. Los objetivos que se persiguen con la normalización son los siguientes:

- Potenciar la consistencia entre las distintas definiciones de Objetos Gestionados.

- Asegurar el desarrollo de tales definiciones de manera compatible con las Recomendaciones OSI y con los Estándar Internacionales.

- De igual forma, la reducción de dobles esfuerzos entre los diferentes grupos de trabajo. Para ello se definen procedimientos y esquemas comunes de responsabilidades.

4.1.1 Qué Gestionar.

La definición de clases de objetos gestionados y sus componentes adoptan una serie de requisitos relacionados con determinados objetivos de gestión. En cada definición de Clase de Objeto Gestionado, se incluirán todos los aspectos de interés para su gestión. Si en la definición de un objeto gestionado se ha establecido que dicho objeto representa un recurso determinado (por ejemplo, una conexión), la información concerniente a ese recurso debe reflejarse a través del objeto u objetos gestionado(s) correspondientes y no en otra parte.

4.1.2 Estructuración.

Existen varias técnicas que permiten representar la constitución de los objetos gestionados, reflejando agrupaciones de datos o funcionalidades. Cada técnica tiene sus ventajas e inconvenientes, la elección de la(s) técnica(s) más apropiada(s) para cumplir un determinado requisito de una especificación, dependerá de criterios diversos.

Las técnicas empleadas para la representación de los objetos gestionados son las siguientes:

- Grupos de atributos.

- Subclases (especialización).

- Herencia múltiple.

- Objetos gestionados contenidos.

- Lotes o paquetes.

Un criterio importante para la elección de una de las técnicas, se puede basar en la presencia estática o dinámica de una agrupación. Si la presencia de una agrupación es fija en el momento de la especificación del objeto, la utilización de grupos de atributos, subclases, herencia múltiple u objetos gestionados contenidos puede ser apropiada. Sin embargo, si la presencia es fija en el momento de la realización, la instalación o la ejemplificación, la mejor técnica puede ser la utilización de objetos gestionados contenidos o lotes condicionales.

[1] Representación abstracta de un recurso sobre el cual se realizan operaciones de gestión.

Para el caso en que la agrupación pueda cambiar durante el curso de la existencia de objetos gestionados contenedores/encapsulantes, puede ser recomendable la utilización de objetos gestionados contenidos, que pueden ser creados y suprimidos de forma dinámica.

Un segundo criterio es ver si hay múltiples ejemplares de la agrupación dentro del objeto gestionado. Si esto ocurre, es recomendable la utilización de objetos gestionados contenidos.

4.1.3 Los Objetos Gestionados.

A continuación se describen aspectos peculiares de los objetos gestionados. Se pretenden exponer de un modo sencillo y claro, las reglas a seguir para la definición de las clases de objeto gestionado y la herencia entre éstas.

4.1.3.1 Ejemplificación de Superclases.

Hay veces en que es conveniente definir clases de objetos gestionados aunque no se obtengan instancias de ellas. Estas Clases pueden ser útiles para ofrecer una base común, a partir de la cual se especialicen distintas subclases. Sea por ejemplo la definición de una clase genérica de objeto gestionado de *"circuito virtual"* de la que nunca se crean instancias, de esta clase se derivan las subclases: *"circuitos permanentes"* y *"circuitos conmutados"*, de las que sí aparecerán las instancias correspondientes.

4.1.3.2 Superclases sin Restricciones.

Cuando se define una subclase a partir de una superclase, podemos establecer cuales son las modificaciones permitidas sobre de las características heredadas. Se denomina restricciones al conjunto de reglas que limitan el conjunto de valores permitidos y el conjunto de valores requeridos de estas propiedades. De forma análoga, limitarán la posibilidad de añadir parámetros a las acciones y notificaciones existentes.

Las restricciones aseguran que la subclase sea compatible con la superclase de la cual desciende. Por esta razón, cuando se define una clase de objeto gestionado que puede ser una superclase, de la que a su vez descienden clases que pueden volver a ser superclases de otras, es conviene prever la posibilidad de asegurar estos tipos de extensión.

Aunque no pueden asegurarse ni preverse todas las extensiones posibles, existen unas técnicas generales que permitirán asegurar la compatibilidad de la mayoría.

- Definir la sintaxis (el tipo) de cada atributo, para incluir todos los valores que se ajusten de forma razonable a la semántica del atributo, aun cuando algunos de estos valores no vayan a ser utilizados de forma inmediata.

- Ofrecer capacidades de extensión en todas las definiciones de acciones y notificaciones, que incluyan la posibilidad de incorporar nuevas funcionalidades.

- Determinar una "*superclase sin restricciones*" con estás características, sin ningún tipo de limitaciones. Simplemente se definirán las limitaciones básicas que permitan a partir de esta superclase la definición de nuevas subclases más restringidas. Para el caso de los atributos se especificarán con un conjunto de valores vacíos, dejando para más tarde la definición de las posibles limitaciones sobre estos valores.

- Las subclases se definen a partir de esta superclase sin restricciones. Las limitaciones necesarias se imponen de forma individual sobre los atributos, las acciones y las notificaciones de cada superclase, que así lo requiera.

Las subclases también pueden ser definidas para posibilitar capacidades de extensión, tan sólo en algunos de los atributos, acciones y notificaciones, pertenecientes a la superclase sin restricciones.

4.1.4 Los Atributos.

En este apartado se describen aspectos referentes a los atributos determinantes para la notación GDMO.

4.1.4.1 Conjunto de Valores de Atributos

Los objetos gestionados deben tener propiedades que sean significativas de los recursos representados. Estas propiedades se conocen como atributos. Un atributo debe tener un valor. Los valores de un atributo puede ser univaluado (cuando tiene un solo valor) o multivaluado (set, es decir, conjunto es un concepto matemático; multivaluado significa que puede tener más de un valor del mismo tipo). En los conjuntos, el orden no es importante y no existen repeticiones de valores.

Los valores de los atributos son visibles hacia la frontera de un tipo de objeto. Cuando ejecutamos ciertas operaciones sobre una clase de objeto, estos valores pueden ser modificados o recuperados.

Para la identificación de un objeto, la clase de objeto debe tener un último atributo usado para el nombrado, que será obligatorio. Este atributo identificador identifica unívocamente a un objeto gestionado y el valor que puede tomar debe ser único. Este atributo es de sólo lectura. Si este atributo puede ser borrado, entonces es necesario definir un atributo adicional de identificación único.

La definición del conjunto de valores permitidos en un atributo, puede efectuarse de varias maneras:

- Definir estáticamente el conjunto de valores del atributo como parte de la definición de la clase de objeto gestionado.

- Definir un segundo atributo, cuyo valor indica el conjunto de valores que el atributo puede contener.

La primera de estas dos técnicas minimiza el número de definiciones de atributos asociadas con una clase de objeto gestionado. Sin embargo si se requiere cierto número de variantes del atributo, la segunda técnica evita la necesidad de definir múltiples subclases, para tratar cada posible variante de un conjunto de valores.

4.1.4.2 Tipos de Atributo.

Un atributo puede tener un valor simple, pero puede también tener varios. Cuando esto suceda se necesita usar el tipo de dato SET OF. En los atributos estructurados, el tipo *sequence, sequence-of o set,* se utilizan como el tipo de base en una definición de sintaxis de atributo. Deben emplearse cuando no se requiera modificar individualmente elementos del atributo, ya que estos tipos ASN.1 corresponden a tipos de atributo con un solo valor.

Los valores de atributo pueden ser un conjunto de valores del mismo tipo. Estos se restringen a un conjunto permitido. El conjunto de valores permitidos menciona los valores que un atributo puede tomar, también llamados valores permitidos (allowed values). El valor de un atributo no puede cruzar los límites de un conjunto de valores permitidos cuando es modificado. Un conjunto de valores permitidos puede ser más restringido por lo que tenemos conjunto de valores requeridos. Este conjunto puede ser vacío si no se requiere un valor.

Cuando una clase de objeto gestionado tiene muchos atributos, puede ser mejor subdividirlos en clases subordinadas, grupos de atributos. Esto provee eficiencia en las operaciones que son ejecutadas sobre esta clase.

4.1.5 Relaciones entre Valores de Atributo.

El valor de un atributo puede verse limitado a causa de alguna función de dependencia de otros valores de atributo. A continuación se identifican las relaciones de esta naturaleza.

- Cuando un valor de un atributo esté limitado por otros atributos del mismo objeto gestionado, puede ser necesario realizar una sincronización previa a la ejecución de una operación de gestión, evitando inconsistencias y contradicciones entre los atributos dependientes. De esta forma cuando se realice un cambio de valor en los atributos, se evita la ocurrencia de errores y la posible aparición de valores no permitidos. Cuando existan estas dependencias, será necesario documentarla como parte de la definición de comportamiento de la clase de objeto gestionado.

- Cuando un valor de atributo tenga dependencias con atributos pertenecientes a otras clases objetos gestionados, puede existir la necesidad de realizar una sincronización entre los distintos atributos. Esta relación también hay que documentarla en el comportamiento de la clase de objeto gestionado asociada. Cuando los objetos gestionados pertenecen a un mismo sistema gestionado, para modificar los atributos utilizando una sola operación de gestión se puede utilizar la sincronización atómica cross-object aportada por CMIS.

4.1.6 Registro de una Clase de Objeto Gestionado.

El proceso de definición de clases de objetos gestionados requiere de la asignación de identificadores globalmente unívocos conocidos como identificadores de objetos. Estos identificadores operan sobre los diversos aspectos de la clase de objeto gestionado, como es el caso del nombre de clase de objeto gestionado, tipos de atributo, etc.

Los valores de estos identificadores son utilizados por los protocolos de gestión, para identificar inequívocamente cada uno de los aspectos de los objetos gestionados. Por tanto, como paso previo a la elaboración de una definición de clase de objeto gestionado, es preciso que el órgano de normalización en cuestión, identifique o establezca un mecanismo de registro adecuado capaz de emitir valores únicos de identificador de objeto para su posterior utilización.

Una vez asignado un valor identificador de objeto a un elemento de información de gestión, es necesario que cualquier revisión que se haga de la definición de dicho elemento, no modifique la semántica de la información. En la práctica, esto significa que se permite modificar el contenido textual de las definiciones de información de gestión registradas,

pero estas definiciones no deberán afectar para nada al protocolo de gestión, deben ser transparentes a éste.

4.2 La Notación GDMO.

Para que las redes heterogéneas puedan interactuar de forma conjunta con propósitos de gestión y administración, debe existir unos prerrequisitos básicos de interoperabilidad, para lo cual, será necesario la utilización de métodos que ayuden a modelar y describir los Objetos Gestionados, así como una definición estándar de estos.

ISO aplica un enfoque totalmente orientado a objetos para construir su modelo de información, este modelo es llamado *Structure of Management Information* SMI, (ISO 10165x). Los Objetos Gestionados MO (*Managed Objets),* son instancias de las Clases de Objeto Gestionados MOC (*Managed Object Classes*), en donde se describen las propiedades visibles externamente.

Cada clase incluye los atributos, las operaciones definidas, notificaciones y descripciones de comportamiento de la misma, e implementa una abstracción de los recursos desde el punto de vista de la administración. Los valores "reales" de los atributos y las operaciones se "mapean" del recurso real.

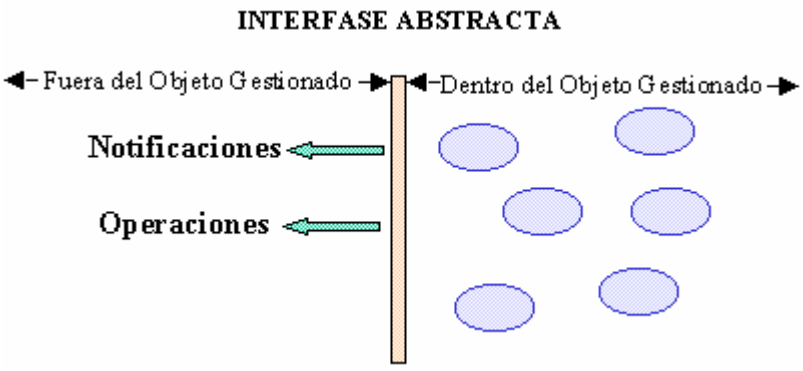

Figura 4.1 Representación de un Recurso

4.2.1 Estructura de la Información de Gestión.

Se encuentra enmarcada en el área de los sistemas orientados a objeto, donde cada objeto poseerá un conjunto de comportamientos, atributos y acciones.

- El comportamiento de un objeto está directamente relacionado con el recurso que representa.

- Los atributos contenidos en un objeto describen el estado, la condición del comportamiento de ellos y pueden incluir referencias a otros objetos.

- Las acciones por su parte, son los servicios que el objeto puede proveer a petición del sistema de gestión.

Cada uno de estos parámetros que determinan el comportamiento de los objetos, se define haciendo uso de las plantillas de la Guía par la Definición de Objetos Gestionados GDMO y la Notación Sintáctica Abstracta Uno, ASN.1, *Abstract Syntax Notation. One* (Anexo B*).* Estas definiciones de plantilla pueden incluir referencias a otras plantillas, a su vez cada una de estas referencias, pueden ser reemplazadas en línea por la definición correspondiente.

4.2.2 Tipos de Plantillas.

GDMO provee nueve estructuras genéricas de plantillas, las enumeramos y explicamos brevemente a continuación:

- *Plantilla de Clase de Objetos de Gestionados:* En esta plantilla interaccionan relaciones, que son importantes para reutilizar características de otras clases de objetos de gestión que ya existen. Las clases de objetos de gestión contienen paquetes de comportamiento, atributos, grupos de atributos, acciones y notificaciones. Las platillas, pueden incluir propiedades de otras clases de objetos de gestión.

- *Plantilla de Paquete:* Contiene las definiciones de atributos, grupos de atributos, operaciones, notificaciones, comportamientos y los parámetros que se pueden de alguna manera coleccionar para formar plantillas. Una plantilla de paquete puede incluirse dentro de las plantillas de clases de objetos gestionados.

- *Plantilla de Atributo:* Esta plantilla se utiliza para proveer la información referente a los atributos, es decir las distintas características que son necesaria definir de un recurso real. Contiene su sintaxis, las reglas para comprobar el valor de los atributos, los comportamientos, los parámetros y por último el identificador de atributo.

- *Plantilla de Grupo de Atributos:* Hay ocasiones en las que es conveniente agrupar los atributos para formar una colección de atributos, que pueden ser tratados de forma conjunta. La plantilla de grupos de atributos indica el conjunto de atributos que conforman el grupo y en este caso un valor para identificar al grupo.

- *Plantilla de Acción:* Esta plantilla define el comportamiento y la sintaxis de los tipos de operaciones que porta la primitiva M_ACTION de CMIS, referentes a un tipo de objeto gestionado.

- *Plantilla de Comportamiento:* Para extender la semántica de las plantillas previamente definidas. De utilidad para explicar con más detalle la conducta de las clases de objetos gestionados, enlaces de nombre, atributos, parámetros, acciones y notificaciones que se han definido en otro lugar.

- *Plantilla de Notificaciones:* Los mensajes enviados por la primitiva M-EVENT-REPORT de CMIS, se definen con en esta plantilla. Son los eventos emitidos por los distintos tipos de objetos.

- *Plantilla de Parámetros:* Para definir los distintos parámetros que se utilizan en las definiciones de los atributos, las operaciones y las notificaciones. Las especificaciones y la sintaxis de los parámetros se listan junto con los comportamientos.

- *Plantilla de Enlaces de Nombres:* Utilizada para el nombramiento y manejo de objetos, se deberá especificar el atributo nombre, que identifica al objeto superior.

4.2.3 Estructura de las Plantillas.

Existe una estructura general para la definición de las distintas plantillas GDMO. En la siguiente figura 4.2 se puede apreciar gráficamente los elementos principales para la definición de una clase de objeto de gestión y comunes a las demás plantillas de la norma.

A continuación se explican brevemente:

● *Etiqueta de plantilla:* Identifica de forma única la instancia que se define con la plantilla. Una cláusula obligatoria.

● *Nombre de plantilla:* Identificará el tipo de plantilla que se está utilizando: atributo, comportamiento, acción, parámetro, etc. También es un elemento obligatorio.

● Uno o más *Constructores:* Cada uno de los cuales estarán formados a su vez por:

- Un Nombre de Constructor.
- Un argumento, que a su vez puede estar constituido por varios campos.

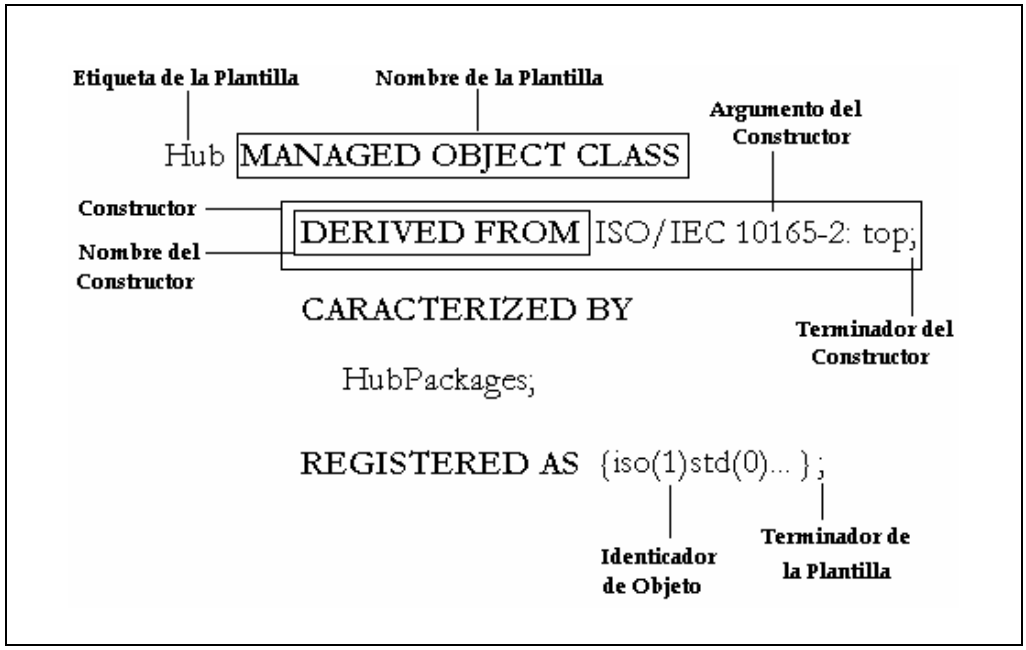

Figura 4.2 Componentes de una Plantilla GDMO

4.2.4 Convenios para la Definición de Plantillas.

Cuando se usan las plantillas para realizar las diferentes definiciones de objetos gestionados, es necesario seguir una serie de convenciones que están recogidas en la norma X.722. A continuación presentamos las más importantes:

- Un punto y coma (;) marca el final de cada construcción y el final de una plantilla.

- Las variables y palabras claves son sensibles a las mayúsculas y las minúsculas, es decir, una palabra en minúscula y la misma en mayúscula tendrán distinto significado. Ejemplo "art enterprise" y "Art Entrerprise" se consideran distintas.

- Los comentarios comienzan y finalizan con "--" al comienzo y final de una línea, respectivamente.

- Los espacios, el final de una línea, una línea en blanco, o comentarios son delimitadores válidos.

- Siempre que se incluya texto en una plantilla, es necesario utilizar uno de los delimitadores: ¡"#& ^,?@, teniendo en cuenta que hay que utilizar el mismo delimitador tanto al comienzo como al final del texto.

- La etiqueta de una plantilla debe ser única en un documento. Cuando se define una plantilla, la sintaxis para la etiqueta será la siguiente:

<etiqueta-plantilla> Etiqueta de Plantilla.

Las cadenas incluidas entre corchetes [ ], son opcionales, pueden estar presentes o no en cada una de las instancias de la plantilla, delimitan las distintas partes de una definición. Si los corchetes van seguido de un asterisco "[ ]*" significa que el contenido de los corchetes puede aparecer cero o más veces.

4.3 Plantilla de Clase de Objeto Gestionado.

Para el estudio de las distintas plantillas comenzamos por el nivel jerárquico superior, la Plantilla de *Clase de Objeto Gestionado.* Esta plantilla representa el nivel principal, se emplea para definir un tipo de objeto gestionado, las demás plantillas se utilizarán para definir las distintas propiedades pertenecientes al objeto gestionado. En GDMO se utiliza una descomposición descendente, la clase está compuesta por paquetes y el paquete a su vez estará compuesto por atributos, notificaciones, comportamientos y parámetros.

En la recomendación X.721 se define un cierto número de clases de objetos gestionados, que pueden ser utilizadas en las diferentes funciones de gestión de sistemas, o utilizando la herencia, servir como superclases para la definición de nuevas clases de objetos. Existen ocasiones es las que será necesario definir nuestros propios tipos de datos.

En la siguiente figura se muestran las relaciones entre las distintas plantillas que componen el estándar GDMO.

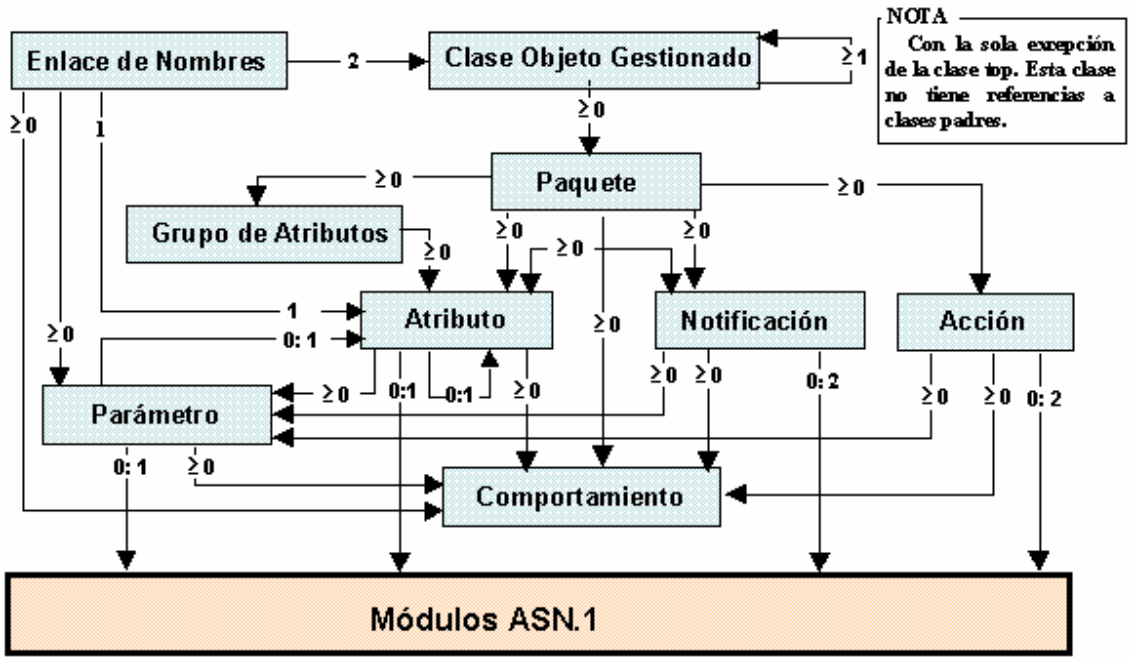

Figura 4.3 Referencias de la Plantilla Clase de Objeto Gestionado

Cada definición de clase de objeto gestionado, excepto la clase *top* debe derivar de otra clase de objeto o en su defecto de la clase principal *top,* es decir tendrá referencias a una o más clases padres. Además, cada una de las clases puede hacer referencia a cero o más paquetes. Finalmente la figura muestra que cada definición de Enlace de Nombres contendrá referencias a exactamente dos definiciones de clases de objetos gestionados.

4.3.1 Estructura de la Plantilla.

Representan un papel clave en GDMO, las clases de objetos gestionados definen las relaciones de herencia que permiten la reutilización de características provenientes de otras Clases de Objetos Gestionados existentes. Pueden contener además sus propios paquetes, grupos de atributos, acciones, comportamientos y notificaciones.

La definición de una Clase de Objeto gestionado se hace de forma uniforme en una plantilla estándar. De esta manera se elimina la confusión que puede resultar de personas diferentes, definiendo objetos de maneras distintas. Con esto se asegura, que las clases definidas en un lugar A, puedan ser interpretadas fácilmente en un lugar B. Además, tiene la gran ventaja de la reutilización de las clases que previamente se hayan definido utilizando la norma.

La plantilla para una clase de objeto gestionado tiene la siguiente estructura

```
<etiqueta-clase> MANAGED OBJECT CLASS
 [DERIVED FROM          <etiqueta-clase>     [,<etiqueta-clase>]*];
 [CHARACTERIZED BY      <etiqueta-paquete>   [,<etiqueta-paquete>]*];
 [CONDITIONAL PACKAGES <etiqueta-paquete>  PRESENT IF condición;
                      [,<etiqueta-paquete>] PRESENT IF condición]*];
REGISTERED AS           identificador-objeto;

Definiciones soportadas   condición -> cadena delimitadora
```

Definición 4.1 Estructura de la Plantilla Clase de Objeto Gestionado

4.3.1.1 Constructores y Cláusulas.

Esta plantilla dispone de las siguienes cláusulas y constructores:

- `<etiqueta-clase>`: Especifica el nombre de la clase de objeto gestionado GDMO.

- `MANAGED OBJECT CLASS`: Esta cláusula identifica al tipo de plantilla para la definición de Clases de Objetos Gestionados.

- `DERIVED FROM`: es una de las palabras claves más importantes, indica que la clase que se define heredará características de una o más superclases. Como ya se ha dicho la superclase más alta es *top,* en otras palabras, todos los Objetos Gestionados derivan de esta.

  No se permite la exclusión de propiedades en la herencia, es decir, las características de la superclase se heredarán todas y ninguna de ellas puede ser excluida. Lo que sí está permitido es la agregación de nuevas características mediante la inclusión de paquetes obligatorios o paquetes condicionales.

- `CHARACTERIZED BY`: Palabra clave que indica la presencia de paquetes obligatorios. Los paquetes pueden contener comportamientos, atributos, grupos de atributos, operaciones y notificaciones.

- `<etiqueta-paquete>`: identifica un paquete GDMO.

  - `CONDITIONAL PACKAGES`: Palabra clave que identificada la presencia de paquetes condicionales. La condición de aparición o no, viene dada por un texto que describe la condición a tener en cuenta.

  - `PRESENT IF condición`: Especifica la condición que debe cumplirse para que el paquete condicional sea incluido en la instancia de objeto gestionado.

- `REGISTERED AS identificador de objeto`: Define la identidad única del tipo de objeto gestionado definido en la clase, es usado por el protocolo para identificar esta Clase particular de Objeto Gestionado. Es una cláusula obligatoria.

  El identificador puede ser utilizado por el protocolo de gestión, para referenciar de forma única a la clase de objeto gestionado correspondiente. Es el parámetro usado para identificar de forma única el enlace de nombre GDMO.

4.3.1.1.1 Ejemplo.

Un ejemplo de definición de clase:

```
sistemaExterno                MANAGED OBJECT CLASS
       DERIVED FROM            sistema;
       CHARACTERIZED BY        sistemaExternoComunPackage;
       CONDITIONAL PACKAGE     sistemaExternoEstacionPackage
       PRESENT IF "el elemento corresponde a una estación del sistema";
 REGISTERED AS         {1 3 5 8 9 2};
```

En el ejemplo anterior se aprecia la definición de la clase de objeto gestionado *sistemaExterno*, que tiene características heredadas de la superclase *sistema*, que a su vez puede poseer superclases de las que heredará características.

La clase definida posee un paquete *sistemaExternoComunPackage,* en el que se definen atributos, acciones, comportamientos, grupos de atributos y notificaciones, que recoge aspectos que serán comunes a un sistema, subsistema o conjunto de equipos externo al centro de control NOMOS [2]. Tal y como se aprecia, la clase también tiene un paquete condicional el *sistemaExternoEstaciónPackage*, que estará presente sólo en el caso de que corresponda a una estación del sistema. Este paquete contendrá características particulares de este modelo de estación, por ello se deben incluir sólo cuando se trate de un recurso de este tipo.

Para finalizar se define un identificador único para la clase, utilizando para ello la cláusula REGISTED AS.

4.3.2 Herencia entre Clases.

En este apartado se van ha tratar aspectos relacionados con la herencia entre clases y el constructor DERIVED FROM, utilizado para la importación de todas las características procedentes de las superclases.

[2] Proyecto en el que se construye un Sistema Integrado Experto para la Gestión de la Red de Comunicaciones en una Compañía Eléctrica y en el que para la definición de los distintos objetos de gestión se utilizan los estándares y recomendaciones GDMO.

La ubicación de una clase en la jerarquía de herencias de las Clases de Objetos Gestionados, viene dada por medio de las referencias a las superclases a partir de las cuales se heredan las propiedades.

Un caso particular, la herencia múltiple puede producir que un mismo elemento importe características duplicadas, como podría pasar por ejemplo, si dos definiciones de superclases incluyen el mismo atributo, se asume que las subclases contendrán una sola copia de la definición involucrada.

Estas propiedades heredadas pueden sufrir modificaciones, utilizando para ello los constructores CHARACTERIZED BY y CONDITIONAL PACKAGES, los denominados *"paquetes condicionales*" que podrán incluirse o no en una clase determinada, en función de ciertos factores condicionales.

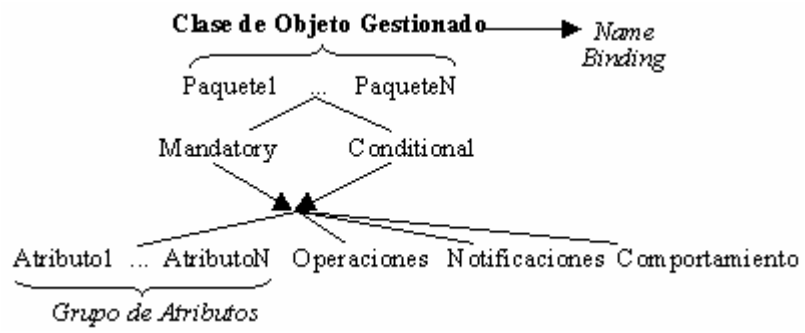

Figura 4.4 Definición de Clase de Objeto Gestionado

La incorporación de variantes de ciertas propiedades y funciones en una "instancia" de un Objeto Gestionado, se decide por medio de condiciones *"if"*, colocadas en la definición de la Clase de Objeto Gestionado. Para saber si un paquete condicional formará parte integral del objeto gestionado (de una instancia de dicha clase), el condicionante colocado en la plantilla de definición de la clase específica debe de cumplirse, es decir la condición debe evaluar a verdadero.

Esto hace más fácil el mapeo de un Objeto Gestionado a un recurso real, los paquetes condicionales ofrecen un alto nivel de flexibilidad, permitiendo una "asociación tardía" con el recurso real.

Desde el punto de vista de administración, los recursos reales pueden ser elementales o compuestos.

En el siguiente ejemplo se observa la definición de la clase de objeto básica denominada *grupoElectrógeno.*

```
grupoElectrogeno MANAGED OBJECT CLASS
       DERIVED FROM ISO/IEC 10165-2: top;
       CHARACTERIZED BY
       BEHAVIOUR comportamientoGrupoElectrogeno;
       ATTRIBUTES
                    MedidaTension           GET,
                    FalloArranque           GET,
                    BajaPrecionAceite       GET,
                    TemperaturaMotor        GET,
                    Sobreintensidad         GET,
                    TensionInsuficiente     GET,
                    NivelBajoAceite         GET,
                    TensionBajaBateria      GET,
                    FalloVentilador         GET,
                    FalloDisparoAutomatico  GET,
                    NivelBajoD50            GET,
                    CargadorMantenimiento   GET,
                    CargadorBateriaMotor    GET,
                    SobrevelocidadMotor     GET,
                    NivelBajoD100           GET;


          ACTIONS
                    GrupoDisponible,
                    GrupoServicio;
          NOTIFICATIONS
          AlarmaFalloArranque, AlarmaBajaPrecionAceite,
          AlarmaTemperaturaMotor,AlarmaSobreintensidad,
          AlarmaTensionInsuficiente, AlarmaNivelBajoAceite,
                    AlarmaTensionBajaBateria,
                    AlarmaFalloVentilador,
                    AlarmaFalloDisparoAutomatico,
                    AlarmaNivelBajoD50,
                    AlarmaCargadorMantenimiento,
                    AlarmaCargadorBateriaMotor,
                    AlarmaSobrevelocidadMotor,
                    AlarmaNivelBajoD100;
   REGISTERED AS             {1 3 5 8 9 0};
```

A continuación se explican los distintos aspectos que se pueden definir en un paquete cualquiera:

- BEHAVIOUR: Los paquetes incluidos en una subclase, por extensión heredarán el comportamiento de superclase.

- ATTRIBUTES: Los atributos se especifican en los paquetes incluidos en la definición de la subclase.

- ACTIONS: Las acciones se pueden incluir en las definiciones de subclases, pudiendo ser estas a su vez añadidas a las heredadas desde las superclases, o incluir parámetros útiles para las acciones heredadas. Los parámetros relacionados con una acción dada, serán la unión de aquellos que se definen en la plantilla de acción y todos los asociados con la acción, que puedan presentarse en los paquetes asociados.

- NOTIFICATIONS: Las notificaciones se pueden incluir en las definiciones de subclases, pudiéndose también heredar notificaciones desde las superclases, o incluir parámetros útiles para las notificaciones heredadas. Los parámetros relacionados con una acción dada, serán la unión de aquellos parámetros definidos en la plantilla de notificación correspondiente, mas todos aquellos parámetros que se encuentran en los paquetes y que se asocian a la notificación.

4.4 Plantilla de Paquetes.

La plantilla de paquete se sitúa jerárquicamente un nivel por debajo de las plantillas de Clases de Objeto Gestionado. Esta plantilla se usa para definir un paquete que agrupa combinaciones de caraterísticas, que podrán ser posteriormente incluidos en una o varias plantillas de Clase de Objetos Gestionados. En la figura anterior 4.3 se aprecia que puede referenciar a uno o más comportamientos, atributos, grupos de atributos, operaciones y notificaciones. Un paquete puede ser referenciado por más de una de Clase de Objeto Gestionado, asimismo una clase de objeto gestionado puede contener más de un paquete.

Las plantillas de paquetes se incluyen en las plantillas de Clases de Objetos, mediante los constructores CHARACTERIZED BY o CONDITIONAL PACKAGES, según sean *obligatorias* o *condicionales* respectivamente. Para los paquetes condicionales además, hay que indicar la condición que se debe cumplir para que aparezcan.

Los paquetes obligatorios, como su nombre indica, estarán siempre presentes en la clase. Por el contrario, los paquetes condicionales deberán presentarse dependiendo de las condiciones definidas.

Por ejemplo, sea la Clase de Objetos Gestionados que define las redes locales *LAN*, el paquete *LANEthernetPackage* deberá aparecer si el sistema es una red local tipo ethernet. Estos paquetes constan de atributos visibles a nuestro límite conceptual, operaciones sobre un Objeto Gestionado, comportamientos de un Objeto Gestionado y notificaciones emitidas por una Clase de Objetos Gestionados.

4.4.1 Aspectos sobre el comportamiento de los paquetes.

En esta plantilla se deben considerar los siguientes puntos claves:

- Las reglas concernientes a la creación y borrado de Objetos Gestionados deben definirse fuera, detallarán las relaciones entre las distintas instancias de una Clase de Objeto. Si existen relaciones con instancias de otra Clase, también habrá que incluirla, además de delimitar los valores de los IVMO (valores iniciales de los Objetos Gestionados).

- Incluir los atributos y operaciones que se pueden efectuar sobre estos atributos.

- Cada atributo tiene su propio conjunto de operaciones permitidas. Para cada una de estas operaciones se pueden también indicar detalles como los valores por defecto, valores iniciales, valores permitidos y requeridos. Por ejemplo, podemos darle al atributo *estadoUso*, el valor por defecto "0" (desocupado), si es el valor normal en el estado de un sistema. Asimismo se le puede indicar que los valores correctos para este atributo oscilan entre 0 y 3 (0=desocupado, 1=activo, 2=saturado y 3=desconocido).

- En esta plantilla hay que especificar las operaciones y las notificaciones de las instancias de una Clase de Objeto Gestionado. Tales como las operaciones realizadas sobre atributos: *Get, Replace Attribute Values, Replace With Default Value, Add Member y Remove Member*.

Sin embargo las que se realizan sobre una Clase de Objeto Gestionado: *Create* y *Delete* formarán parte de la plantilla de Enlace de Nombre. Además las acciones se definen en las plantillas de acción.

4.4.2 Estructura de la Plantilla de Paquete.

A continuación se muestra la estructura correspondiente:

```
<etiqueta-paquete> PACKAGE
  [BEHAVIOUR   <etiqueta-definición-comportamiento>
          [,<etiqueta-definición-comportamiento>]*;]
  [ATTRIBUTES <etiqueta-atributo> propertylist [<etiqueta-parámetro>]*
           [,<etiqueta-atributo> propertylist[<etiqueta-parámetro>]*]*;]
  [ATTRIBUTE GROUPS <etiqueta-grupo> [<etiqueta-atributo>]*
                                  [,<etiqueta-grupo> [<etiqueta-atributo>]*]* ;]
  [ACTIONS        <etiqueta-acción> [<etiqueta-parámetro>]*
                  [,<etiqueta-acción> [<etiqueta-parámetro>]*]*;]
  [NOTIFICATIONS   <etiqueta-notificación> [<etiqueta-parámetro>]*
                  [,<etiqueta-notificación> [<etiqueta-parámetro>]*]*;]
[REGISTERED AS    identificador-objeto];
```

Donde *property list* puede ser:

REPLACE-WITH-DEFAULT
DEFAULT VALUE valor-especificado
INITIAL VALUE valor-especificado
PERMITTED VALUES tipo-requerido
REQUIRED-VALUES tipo-requerido
GET | REPLACE |GET-REPLACE
ADD |REMOVE |ADD-REMOVE.

Y el valor-especificado: referencia-valor|DERIVATION RULE <etiqueta-definición-comportamiento>

Definición 4.2 Estructura de la Plantilla de Paquete

Los elementos principales de la plantilla:

● `BEHAVIOUR`: Información no procesable (expresado en formato texto libre), que identifica el comportamiento del paquete, que se definirá en la plantilla de comportamiento correspondiente. Debe enunciar por ejemplo la reacción de un Objeto gestionado cuando es estimulado con un evento. Incluye además las operaciones de gestión como:

- Los efectos de las operaciones de gestión en los objetos
- Las circunstancias que disparan la generación de notificaciones
- Los criterios para seleccionar un Valor Inicial de Objeto Gestionado IVMO.
- Las relaciones con otros objetos gestionados.
- etc.

● `ATTRIBUTE`: Lista todos los atributos que forman parte del paquete. Asociado con cada atributo se definen las propiedades y los parámetros.

● `ATTRIBUTE GROUP`: Permite identificar un conjunto de atributos como parte del paquete. Para el caso de grupos de atributos extensibles, la definición original del grupo de atributos puede extenderse añadiendo nuevas etiquetas de atributos.

● `ACTIONS`: Identifica el conjunto de definiciones de acciones que se incluyen en el paquete. El comportamiento que describirá el efecto que estas acciones producen sobre los objetos gestionados, debe ser definido mediante la plantilla de comportamiento.

Cada una de las acciones puede tener asociados uno o más parámetros, que se pueden usar para indicar valores de respuesta o por ejemplo posibles errores

específicos asociados a acción. La sintaxis de cada uno de los parámetros se define con la plantilla de parámetro correspondiente.

- `NOTIFICATIONS`: Incluye cualquier notificación aplicable al paquete. El comportamiento especificará la circunstancia bajo las cuales un objeto gestionado generará estas notificaciones.

Al igual que en el caso de las acciones, cada notificación puede tener uno o más parámetros, aunque en la práctica es raro su utilización.

4.4.2.1 Ejemplo.

A continuación se puede observar un ejemplo típico de definición de paquete, en el que se aprecian distintos elementos: comportamiento, atributos, grupo de atributos, acciones y notificaciones, que forman la definicón del tipo de objeto gestionado *sistemaExterno.*

```
SistemaExternoComun   PACKAGE
       BEHAVIOUR             sistemaExternoComun
      ATTRIBUTES             identificador          PERMITTED VALUES 0000 - XXXX,
                             estadoOperativo        INITIAL VALUE 0;
                             estadoUso              REPLACE WITH DEFAULT,
                             estadoAdministrativo   REPLACE WITH DEFAULT,
                             estadoGestion          REPLACE WITH DEFAULT,
                             responsable            GET;
       ATTRIBUTE GROUPS      estadoSistema;
       ACTION                modificarEstado;
       NOTIFICATION          crearObjeto;
                             destruirObjeto;
                             cambioEstado;
REGISTERED {XXX?}

sistemaExternoComun   BEHAVIOUR
   DEFINED AS "Sistema, subsistema o conjunto de equipos externos al centro de
       control NOMOS";
```

4.5 Plantilla de Atributo.

Los atributos son las propiedades más importantes de cuantas se definen en las clases, caracterizan las cualidades y el estado en que se encuentra un Objeto Gestionado. Se pueden agrupar formando uniones de atributos sobre los que se puede operar de forma conjunta. Los tipos de atributos utilizados, dependerán del objeto que está siendo modelado. Pueden estar asociados con tipos ASN.1 o puede derivarse de un atributo genérico como counters, gauges, thresholds, names, timers, o de tipos más complejos definidos en ISO 101165-2 ó en X.721.

En la figura inicial 4.3, se observa la relación que la plantilla atributo mantiene con el resto del plantillas de GDMO. Se observa que un atributo puede hacer referencia a uno o más parámetros, comportamientos y atributos. Asimismo también se observa que la plantilla atributo es referenciada cero o más veces por los paquetes, grupo de atributos, notificación y una referencia simple a sí misma. El Enlace de Nombre referencia a una definición de atributo. La definición de atributo hace referencia a un tipo ASN.1 (sintaxis de atributo) o alternativamente a la definición de un atributo (atributo padre).

Un atributo tiene un valor asociado que es visible a efectos de consulta, puede tener una estructura simple o compleja. Los valores y operaciones permitidas se concretan para cada atributo, estando permitido por ejemplo aplicar restricciones de protección contra escritura o limitaciones de valores de entrada/salida.

Cada atributo necesita tener un valor, así por ejemplo la definición de una clase de objetos *sistemaExterno*, puede tener un atributo *estadoAdministrativo = 0.* que indica que el valor inicial asignado al atributo es de 0. En este caso el valor 0 asociado al atributo *estadoAdministrativo* indica que el sistema está bloqueado.

Los atributos pueden tener valores simples o conjunto de valores, en el caso anterior podemos modelar el ancho de banda para permitir varios valores: *estadoAdministrativo = 0,1,* que significa que el sistema está bloqueado y desconectado. Cuando existen varios valores, el orden de estos no será relevante y no estarán permitidos los valores repetidos.

Los valores de los atributos pueden ser tratados utilizando para ello las operaciones oportunas. Aplicando la orden GET a un atributo determinado devuelve el valor de actual del atributo, con SET se sustituirá el valor del atributo por un nuevo valor.

Para que un objeto pueda ser identificado, la Clase de Objeto de Gestión necesitará tener al menos un atributo para su nombramiento, que identifique al Objeto Gestionado de forma única, este atributo no podrá ser modificado y será por tanto de sólo lectura.

4.5.1 Operaciones y Tipos de Atributos.

Existen dos categorías de atributos:

• *Singled Valued:* Gestionado como un sólo dato con respecto a las modificaciones. Aunque no es obvio, un atributo se considera de este tipo si es posible asociarle un único valor o un conjunto de valores ordenados.

• *Set Valued:* Es una colección de valores que no están ordenado, se define utilizando la construcción "*set of*".

Las operaciones orientadas a atributo son las siguientes:

- *Get*: lee el valor de un atributo.
- *Replace*: escribe el valor de un atributo.
- *Replace-with-default.*
- *Add*: agrega un valor a atributos set-valued.
- *Remove:* borra un valor de atributo de cierto set-valued.
- *Set-by-create:* asigna un valor, sólo cuando se crea el Objeto.
- *Not-modify:* evita que el atributo sea redefinido al crear subclases.

4.5.2 Estructura de la Plantilla

A continuación se muestra la platilla para la definición de atributos:

```
<etiqueta-atributo> ATTRIBUTE
  DERIVED FROM <etiqueta-atributo> | WITH ATTRIBUTE SYNTAX <tipo-requerido>;
  [MATCHES FOR qualifier [,qualifier]*;]
  [BEHAVIOUR   <etiqueta-definición-comportamiento>
             [,<etiqueta-definición-comportamiento>]*;]

  [PARAMETERS  <etiqueta-parámetro> [,<etiqueta-parámetro>]*;]
REGISTERED AS  identificador-objeto];

 donde qualifier puede ser: EQUALITY | ORDERING | SUBSTRINGS | SET-COMPARISON
                                    | SET-INTERSECTION
```

Definición 4.3 Estructura de la Plantilla de Atributos

Observemos los aspectos más interesantes:

- `ATTRIBUTE`: Palabra clave asociada a cada nombre de atributo.

- `DERIVED FROM`: Indica que el atributo se deriva de una definición existente, por lo tanto:

  Las reglas de búsqueda en MATCHES FOR se aplican a esta definición y a la definición desde la cual fue derivado el atributo, es decir un OR lógico de todas las reglas de búsqueda. Todo ello permitirá la adicción de nuevas reglas, así como la definición de nuevos atributos, basados en las definiciones de atributos que ya existen.

  El texto introducido en BEHAVIOUR y los parámetros en PARAMETER se consideran como extensiones de los que se heredan.

- `WHIT ATTRIBUTE SYNTAX`: Presente cuando el constructor anterior no aparezca, indicará un tipo de dato ASN.1 para el atributo.

- `MATCHES FOR`: Define tests que pueden ser ejecutados sobre valores de atributos por operación de filtrado. Si esto no se indica, entonces los tests sobre los valores no puede ser realizado y no se definen. Cualquier atributo específico caracteriza o indica cómo los atributos proceden bajo reglas de comparación provistas por MATCHES FOR u otro comportamiento. El comportamiento que es específico de una clase de objeto gestionado, se define en la plantilla de definición de la clase. Los calificadores que se pueden utilizar para realizar una búsqueda pueden ser:

  - EQUALITY, que lo hace por igualdad.

  - ORDENING, que permite hacer la comparación mayor o menor.

  - SUBTRING, comprueba que una determinada cadena forma parte del valor de un atributo, etc.

- `BEHAVIOUR`: Permite la definición del comportamiento particular de un atributo. No debe confundirse con el comportamiento específico de una Clase de Objeto Gestionado, definido en un paquete.

- `PARAMETERS`: Esta cláusula define los parámetros asociados con el comportamiento del atributo, permitiendo por ejemplo especificar errores asociados con el atributo.

- `REGISTRED AS`: La definición de atributo termina con este constructor, que identificar al atributo de forma única.

4.5.2.1 Ejemplos.

A manera de ejemplo, la siguiente plantilla define el atributo *fechaAlarma*:

```
fechaAlarma ATTRIBUTE
       WITH ATTRIBUTE SYNTAX Attribute-ASN1Module.EventTime;
       MATCHES FOR EQUALITY, ORDERING;
       BEHAVIOUR
       comparaFecha BEHAVIOUR
        DEFINED AS "los campos año, mes, día, hora, minutos y segundos se
        comparan con el valor de este atributo para determinar si es mayor o
        menor. Comienza comparando el valor del año, en caso de ser iguales
       se siguen comparando en orden, de mes, día, hora, minutos y
       segundos.";
REGISTERED AS {smi2AttributeID 13};
```

Se observa que hereda la sintaxis del atributo EventTime de ASN.1. Los constructores DERIVED FROM y WITH ATTRIBUTE SINTAX, no están presentes al mismo tiempo.

Un segundo ejemplo de plantilla, que define los atributos *estadoComunicaciones* e *idSistemaExterno* pertenecientes a la Clase de Objeto Gestionado *sistemaExterno*:

```
estadoComunicaciones        ::= SET OF INTEGER
idSistemaExterno            ::= direccionComunicaciones
direccionComunicaciones     ::= OCTET STRING SIZE (4)

estadoComunicaciones        ATTRIBUTE
     WITH ATTRIBUTE SINTAX SET OF INTEGER;
REGISTERED AS               {1 3 5 8 9 3};

idSistemaExterno    ATTRIBUTE
     WITH ATTRIBUTE SINTAX direccionComunicaciones;
REGISTERED AS               {1 3 5 8 9 4};
```

En la definición precedente de atributos observamos los siguientes puntos interesantes:

- Para la definición de los atributos, se usa la notación ASN.1.

- Notar que se ha definido el atributo *idSistemaExterno* mediante otro tipo de dato, *direccionComunicaciones.*

- REGISTERED AS es un *constructor* y {1 3 5 8 9 2} es su argumento.

Un nuevo ejemplo de plantilla de atributo:

```
contador     ATTRIBUTE
      WITH ATTRIBUTE SINTAX INTEGER;
      MATCHES FOR EQUALITY, ORDERING;;
REGISTERED AS       {1 3 5 8 9 5};
```

Se aprecia como la cláusula MATCHES FOR define un test que puede ser ejecutado sobre valores de atributos, utilizando una operación de filtrado. Si no se especifica este u otro calificador, entonces no será posible realizar una búsqueda sobre los atributos.

Un ejemplo de definición de un atributo *numeroEntradaSistemar* que se deriva de otro ya existente *counter1*, que puede haber sido definido en un lugar diferente.

```
numeroEntradaSistema        ATTRIBUTE
      DERIVED FROM   counter1;
REGISTERED AS        {1 3 5 8 9 6};
```

En la definición precedente del atributo *numeroEntradaSistema*, las características del atributo son derivadas de otro atributo, counter1, el cual es definido en un lugar diferente. DERIVED FROM permite hacer uso de definiciones de atributos existentes. Mediante la definición de nuevas reglas, se puede extender o restringir las definiciones derivadas de otro atributo.

Un nuevo ejemplo muestra nuevas características:

```
Counter2   ATTRIBUTE
  WITH     ATTRIBUTE SINTAX INTEGER;
  MATCHES  FOR EQUALITY, ORDERING;
  BEHAVIOUR counterBehaviour  BEHAVIOUR
           DEFINED AS "Testea por igual o mayor que los valores permitidos"
REGISTERED AS                {1 3 5 8 9 7};
```

DEFINED AS se utiliza junto con BEHAVIOUR, la cláusula DEFINED AS irá seguida por una cadena y usa como delimitadores de texto (“ ”). Proporciona mayor sentido o significado a la definición del comportamiento del atributo, análogamente puede ser usado en las plantillas de enlace de nombre, parámetros, atributos, acciones o notificaciones de la Clase de Objeto Gestionado.

Sea un nuevo ejemplo:

```
counter3   ATTRIBUTE
  WITH     ATTRIBUTE SINTAX INTEGER;
  MATCHES  FOR EQUALITY, ORDERING;
  BEHAVIOUR counterBehaviour  BEHAVIOUR
           DEFINED AS "Testea por igual o mayor que los valores permitidos"
           PARAMETER counterThresholdDetails;
REGISTERED AS               {1 3 5 8 9 8};
```

Se aprecia como se utiliza la palabra clave PARAMETER, para la definición de errores o fallas de procesamiento.

Un atributo puede estar compuesto por un único valor simple o por un conjunto de valores, para lo cual es necesario usar el tipo de dato SET OF. Existen dos tipos de atributos compuestos: SEQUENCE y SEQUENCE OF.

Los valores de atributo pueden ser un conjunto de valores del mismo tipo, restringiéndose estos a un *conjunto permitido*. El conjunto de valores permitidos recoge los valores que un atributo puede tomar, llamados también *valores permitidos*. Cuando se modifica un atributo, el nuevo valor de este no puede exceder los límites impuestos por el conjunto de valores permitidos.

Cuando una Clase de Objetos Gestionados tiene muchos atributos, es conveniente dividirlos en clases subordinadas que provocarán mayor eficiencia a la hora de efectuar operaciones sobre ellos.

4.6 Plantilla de Grupo de Atributos.

Esta plantilla se usa para definir uno o más atributos que pueden ser tratados de forma conjunta. Ver las referencias correspondientes en la figura 4.3. Estas agrupaciones son útiles en situaciones en las que es deseable tratar con una colección de atributos que tienen características comunes. En este caso podemos operar sobre el grupo de forma conjunta y no independientemente sobre cada uno de ellos. Por conveniencia y para fácil operación, podemos combinar atributos en un grupo de atributos. Sin embargo, esto restringe las operaciones que podemos ejecutar sobre él. Un grupo de atributos no tiene valor y sólo aquellas operaciones que no requieren valores pueden ser ejecutadas sobre este.

Una definición de Clase de Objeto Gestionado puede incluir todos los atributos de un grupo, haciendo alusión a éste, en lugar de a cada atributo individualmente. Un grupo de atributos puede estar incluido en una o más Clases de Objeto Gestionado.

Los atributos pueden formar parte de diferentes grupos de atributos.

4.6.1 Tipos de Grupos.

Existen dos modalidades de grupos:

- *Fijo:* En los cuales una colección de atributos es definida y más atributos no pueden ser agregados.

- *Expansible:* En donde nuevos atributos pueden ser agregados. Los atributos extensibles son definidos en paquetes obligatorios o condicionales.

4.6.2 Estructura de la Plantilla.

A continuación se muestran las cláusulas y constructores que conforman la plantilla en estudio:

```
<etiqueta-GrupoAtributo> ATTRIBUTE GROUP
    [GROUP ELEMENTS   <etiqueta-atributo> [,<etiqueta-atributo>]*;]
    [FIXED;]
    [DESCRIPTION      Cadena demilata;
[REGISTERED AS        identificador-objeto;
```

Definición 4.4 Estructura de la Plantilla de Grupo de Atributos

Algunos aspectos importantes sobre los elementos que la componen:

• `GROUP ELEMENTS:` Este constructor define el conjunto de etiquetas de atributos que identifican de forma individual a cada uno de los atributos que forman el grupo y que estarán presentes en todas sus instancias. Cada uno de los atributos referidos en el grupo, deberán ser definidos por medio de las plantilla de atributo correspondiente.

• `FIXED:` Cuando este constructor está presente, viene a indicar que el número de atributos miembros del grupo se define de forma fija. La cláusula GROUP ELEMENTS debe de estar presente en este caso en la definición de la plantilla. En el caso en que no aparezca esta cláusula, el grupo de atributos se considera expansible y GROUP ELEMENTS puede estar presente o no.

• `REGISTERED AS:` Suministra un identificador global único del grupo de objetos definidos en la plantilla.

4.6.2.1 Ejemplo.

En el siguiente ejemplo se muestra la definición de un grupo de atributos *grupoEstados,* formado por los atributos *estadoOperativo, estadoUso, estadoAdministrativo, estadoGestion.* Estos atributos a su vez, pueden formar parte de otros grupos de atributos además del grupo g*rupoEstados.*

```
grupoEstados ATTRIBUTE GROUP
   GROUP ELEMENTS   estadoOperativo,
                    estadoUso,
                    estadoAdministrativo,
                    estadoGestion;
   FIXED;
        DESCRIPTION "Engloba aquellos atributos que definen el estado
                     de un  objeto.";
REGISTED AS  {1 3 5 8 9 6};
```

En la definición anterior, cada uno de los atributos *estadoOperativo, estadoUso, estadoAdministrativo, estadoGestion* incorporados al grupo de estados, pueden ser a su vez de valor simple o conjunto de valores.

Debido a la cláusula FIXED, debemos tener en cuenta que no podremos agregar nuevos atributos a este grupo. En caso de no aparecer el constructor FIXED, se asume que es extensible, es decir por defecto se pueden agregar más atributos.

4.7 Plantilla de Acción.

La definición de una Acción especificará la funcionalidad, en términos de los efectos que esta produce en las Clases de Objetos gestionados sobre las que actúa. La plantilla ACCIÓN se usa para definir la conducta y la sintaxis asociada con un tipo particular de acción, que puede ser realizada por un tipo de objeto definido en una Clase de Objeto Gestionado. Se especifican acciones que se realizan usando el servicio M-ACTION de CMIS y no incluye acciones o comportamientos definidos para otros servicios de CMIS, como pueden ser M-GET, M-SET, etc.

En la figura 4.3 se observa que una acción contendrá referencias a cero o más comportamientos y parámetros. Cada definición de paquete puede contener cero o más referencias a la definición de acción. Además cada definición de acción puede tener cero, una o dos referencias a tipos ASN.1.

Una acción puede ser aplicada a más de una Clase, limitándose la descripción del comportamiento en este caso a los rasgos de conducta comunes a todas las clases de objetos gestionados sobre las que opera. Los rasgos específicos de cada una de las clases se describirán como partes de la definición de las propias clases.

Un conjunto de acciones, operaciones complejas que afectan al Objeto Gestionado, pueden definirse para que sean específicas a ese Objeto. Es decir, se puede detallar libremente operaciones particulares al Objeto Gestionado. Por ejemplo un "*reset"* del objeto.

Hay dos operaciones predefinidas que siempre afectan en su totalidad al objeto gestionado (instancia dinámica): *"create"* y "*delete*" objeto. Estas dos acciones se consideran

operaciones orientadas a objetos y diferentes a las anteriores que estaban orientadas a atributos.

4.7.1 Estructura de la Plantilla.

A continuación se muestra la platilla para la definición de acciones:

```
<etiqueta-atributo> ACTION
    [BEHAVIOUR <etiq-definición-comportamiento>
            [,<etiq-definición-comportamiento>]*;]
    [MODE CONFIRMED;]
    [PARAMETERS             <etiqueta-parámetro> [,<etiqueta-parámetro>*;]
    [WITH INFORMATION SYNTAX tipo-requerido;]
    [WITH REPLY SYNTAX       tipo-requerido;]
REGISTERED AS  identificador-objeto;
```

Definición 4.5 Plantilla de Acción

Los constructores que la componen:

• BEHAVIOUR: Especifica la forma de actuar, los parámetros asociados, los resultados que la acción puede generar y sus significados. Los comportamietos que aparecen en la etiquetas deberán incluirse dentro de las plantillas Behaviour oportunas.

• MODE CONFIRMED: La palabra clave indica que la acción tendrá un mensaje de respuesta o confirmación cuando se ejecuta. Si no se indica, entonces la confirmación o no de la ACTION, será decidido por el administrador.

• PARAMETERS: Sirve para indicar información o parámetros de información de la acción correspondiente, así como los fallos de procesamiento asociados a ésta.

• WITH INFORMATION SINTAX: Proporciona detalles de la información generada por la acción. Se identifica el tipo de dato ASN.1, que describe la estructura de la información manejada por el protocolo de gestión. Si la cláusula no aparece, implica que no existe información asociada con la invocación de la acción.

• WITH REPLY SINTAX: Ofrece detalles referentes a la información de respuesta como resultado de un ACTION. Si esta palabra clave se encuentra ausente, entonces no existe respuesta asociada con el ACTION.

• REGISTERED AS: Finalmente se registra el tipo de acción con un identificador global único.

4.7.1.1 Ejemplo.

En el siguiente ejemplo se define una acción nominada *modificarEstado.* Los detalles referentes a la conducta se pueden ver en la plantilla de comportamiento anexa *modificarEstadoConducta,* esta acción es útil para aquellos sistemas que permitan producir cambios de estados en sus estaciones de trabajo.

```
modificarEstado ACTION
    BEHAVIOUR                modificarEstadoConducta;
    MODE CONFIRMED;
    PARAMETERS               codigoActuacion;
    WITH ATTRIBUTE SYNTAX    CHARACTER STRING SIZE (128);
    WITH REPLY SYNTAX        CHARACTER STRING SIZE (128);
REGISTERED AS         {1 3 5 8 9 6};
```

```
modificarEstadoConducta BEHAVIOUR
      DEFINED AS "Esta acción permite modificar el estado de
      comunicaciones con un sistema externo al centro de control de
      NOMOS.";
REGISTERED AS         {1 3 5 8 9 6};
```

La definición precedente proporciona algunos detalles interesantes, *modificarEstado* es el nombre de la acción definida. Para detalles del comportamiento de esta acción, debemos recurrir a la plantilla de comportamiento etiquetada *modificarEstadoConducta*. Por tener la palabra clave MODE CONFIRMED, la acción debe tener un mensaje de confirmación o de respuesta después del envío de ACTION. Si MODE CONFIRMED no se indica, entonces la confirmación o no de Acción es decidido por la estación de administración.

WITH INFORMATION SINTAX proporciona detalles de la información transportada por la acción, indica que portará información referente a la acción a realizar y la información resultante, en un formato de cadenas de hasta 128 octetos. La palabra clave WITH REPLY SINTAX entrega detalles de la información de respuesta como resultado de la acción. Si esta palabra clave se encuentra ausente, entonces no existe respuesta asociada con la acción correspondiente. Finalmente, REGISTERED AS es el usual identificador de la plantilla de acción.

4.8 Plantilla de Notificaciones.

Un Objeto Gestionado, también puede ser un recurso autónomo en el pueden ocurrir eventos asincrónicos, cuando ciertos eventos internos o externos ocurren. Las notificaciones son los mecanismos utilizados por los objetos gestionados para informar sobre eventos que pueden ser iniciados por sí mismo, sin necesidad de que el sistema de administración lo solicite. Esta plantilla se usa para definir la conducta y la sintasis de un tipo particular de notificación emitida por una Clase de Objeto Gestionada. Una notificación puede ser referenciada por cero o más definiciones de paquetes. A su vez podrá referenciar a cero o más definiciones de parámetros, atributos y comportamientos. También puede referenciar a cero, uno o dos tipos ASN.1.

Esta plantilla se utiliza para describir las notificaciones que pueden (no necesariamente), especificarse en un Objeto Gestionado. Las notificaciones son particulares a cada objeto, y emitidas por estos cuando ocurre un evento interno o externo. El tipo de notificación es transportada en los parámetros Información del Evento o Respuesta del Evento de las primitivas CMIS M-EVENT-REPORT, definidas en el CMIP (CCITT X.710 / ISO 9595).

Para que la notificación sea útil deberá contener cierta información a ser usada. Cada definición incluye:

- La estructura de los datos intercambiados por el protocolo de gestión, que corresponden a la notificación.

- El comportamiento (*behaviour*) de la notificación.

- Por último la estructura de la información resultante de la notificación, que intercambia el protocolo de gestión.

- La asignación de un valor de identificador de objeto.

Las notificaciones serán internas o enviadas externamente dependiendo de los *Discriminadores de Eventos Enviados* (EFDs). Los EFDs determinan también si estas notificaciones generan reportes de eventos confirmados o no confirmados.

4.8.1 Estructura de la Plantilla.

La plantilla para las notificaciones consta de la siguiente estructura:

```
<etiqueta-notificación> NOTIFICATION
 [BEHAVIOUR <etiqueta-definición-comportamiento>
         [,<etiqueta-definición-comportamiento>]*;]
 [PARAMETERS                 <etiqueta-parámetro> [,<etiqueta-parámetro>*;]
 [WITH INFORMATION SYNTAX    tipo-requerido
       [AND ATTRIBUTE IDs    <nombre-campo(s)>  <etiqueta-atributo>
                             [,<nombre-campo(s)>  <etiqueta-atributo>]*];]
  [WITH REPLY SYNTAX         tipo-requerido;]
REGISTERED AS  identificador-objeto;
```

Definición 4.6 Plantilla de Notificación

Los distintos elementos que la componen:

● BEHAVIOUR: esta cláusula define el comportamiento de la notificación, el dato que se especificará con la notificación, el resultado que la notificación puede generar y su significado, primitiva M-EVENT-REPORT del servicio CMIS.

● PARAMETERS: Identifica la información, los parámetros respuesta (parámetros *event-information, event-reply)* y los fallos de proceso asociados con el tipo de notificación especificado.

● WITH INFORMATION SINTAX: Identifica el tipo de dato ASN.1 que describe la estructura de la información asociada a la notificación que transporta el protocolo de gestión. Si la cláusula no aparece, implica la no existencia de información asociada con la notificación.

● WITH REPLY SINTAX: Identifica el tipo de dato ASN.1 que describe la estructura de la información de notificación respuesta, que transporta el protocolo de gestión. Si la cláusula no aparece, específica la no existencia de la notificación específica asociada.

● REGISTERED AS: Como en las demás plantilla estudiadas, proporciona un idenficador, que será utilizado por el protocolo de gestión para refererirse a un tipo de notificación en concreto.

4.8.2.1 Ejemplo.

Un ejemplo sencillo de notificación que no contiene parámetros, no necesita información de entrada y no genera respuesta:

```
alarmaComunicaciones NOTIFICATION
REGISTERED AS {1 3 4 8 9 7};
```

NOMOS [Leon00] define cierto tipo de notificaciones aplicables a una gran variedad de clases de objetos gestionados, una notificación de este tipo es *cambioEstado.* A continuación se muestra su definición:

```
cambioEstado NOTIFICATION
 BEHAVIOUR  cambioEstadoCompoBehaviour;
 WITH INFORMATION SYNTAX Notification-ASN1Module.AttributeValueChangeInfo
       AND ATTRIBUTE IDS
       estadoOperativo
       estadoUso
       estadoAdministrativo
REGISTERED AS {1 3 4 8 9 3};

CambioEstadoBehaviour BEHAVIOUR
              DEFINED AS "Esta notificación permite conocer el cambio de
                          estado de un objeto. ";
```

Se observan los parámetros asociados a la notificación: *estadoOperativo, estadoUso* y *estadoAdministrativo*. Estos parámetros además siguen una sintaxis *AttributeValueChangeInfo* que se define siguiendo la sintaxis *Notification-ASN1Module*.

4.9 Plantilla de Parametros.

Permiten el traspaso de información entre los distintos elementos gestionados. Un parámetro puede definir fallas de procesamiento CMIS, consultas y respuestas de notificaciones, o consultas y respuestas de acciones. La plantilla correspondiente es utilizada para definir la sintaxis del parámetro que se incluirá en una plantilla de paquete, atributo, acción o notificación. Un parámetro puede ser incluido más de una vez en distintas plantillas.

Observar las referencias en la figura 4.3, una definición de parámetro referencia a cero o una definición de atributo o alternativamente a un tipo ASN.1 en un módulo ASN.1 y a cero o más definiciones de comportamientos. También cero o más definiciones de parámetros pueden ser refenciadas por definiciones de atributos, notificaciones, acciones y enlace de nombres.

Los parámetros se pueden asociar a operaciones y notificaciones, permitiendo definir fallos de procesamiento CMIS, peticiones y notificaciones o acciones.

4.9.1 Estructura de la plantilla.

Presentará los siguientes elementos:

```
<etiqueta-parámetro> PARAMETER
   CONTEXT     tipo-contexto;
     Sintaxis-o-sintaxis-opción;
   [BEHAVIOUR   <etiqueta-definición-comportamiento>
          [,<etiqueta-definición-comportamiento>]*;]
[REGISTERED AS identificador-objeto]

Donde
  - tipo-contexto puede ser:
            Palabraclave-contexto | ACTION-INFO | ACTION-REPLY | EVENT-INFO |
                                    EVENT-REPLY | SPECIFIC-ERROR

  y Palabraclave-contexto -> referencia-tipo.<identificador>

  - Sintaxis-o-sintaxis-opción
          WITH SYSNTAX referencia-tipo | ATTRIBUTE  <etiqueta-atributo>
```

Definición 4.7 Plantilla de Parámetro

Sus principales elementos:

• `CONTEXT`: Este constructor define el contexto en el cual se puede aplicar el parámetro. Entre otros contextos, tenemos el *context-keyword* que hará referencia a un contexto definido exteriormente a la plantilla. ACTION-INFO se define como un parámetro que se puede comunicar al CMIP como respuesta a alguna acción, habrá que definirlo en el constructor WITH REPLY SYSNTAX de la plantilla ACTION correspondiente.

• `WITH SYNTAX`: Si presente, identifica el tipo de dato ASN.1 que tiene el parámetro.

• `ATTRIBUTE`: Hace referencia a un tipo de atributo definido en una plantilla de atributo. La sintasis e identicador de objeto de la plantilla atributo sirve como sintasis e identificador de objeto de nuestra plantilla de parámetro.

• `BEHAVIOUR`: Permite la definición de la conducta o la semántica asociada con el parámetro. Para el caso en el que se usa el constructor ATTRIBUTE, el comportamiento no sufre modificación alguna por parte del comportamiento del atributo.

• `REGISTERED AS`: Como en todos los casos anteriores, suminisitra un identificador global único, usado por el protocolo de gestión para identificar al parámetro.

4.9.1.1 Ejemplo.

Sea la siguiente plantilla muestra la definición de un tipo de parámetro.

```
codigoEstadoComunicaciones    PARAMETER
       CONTEXT                ACTION-REPLY;
       WITH SYNTAX            INTEGER;
REGISTERED AS          {1 3 5 8 9 14};
```

En el ejemplo anterior, CONTEXT referencia condiciones definidas externamente para esta plantilla de parámetro. Este parámetro actua sobre el estado de las comunicaciones, pudiendo tener los valores 0 ó 1 (desactiva y activa las comunicaciones, respectivamente). El valor asignado por dedefecto es el "1", es decir comunicación activa.

4.10 Plantilla de Enlace de Nombre.

Los recursos modelados por los Objetos Gestionados deben reflejar todas las posibles características existentes en ellos. Sea por ejemplo un multiplexor que puede tener más de una interfase, que a su vez puede tener varios puertos. Para reflejar esta peculiaridad es necesario utilizar las plantillas de Enlaces de Nombres, que permiten crear jerarquías de contenido. Antes de que una Clase de Objeto Gestionado pueda ser instanciable, es necesario definir uno o más nombres de enlaces, que permitirán que la instancia del objeto gestionado, sea situada jerárquicamente en el lugar correspondiente dentro del sistema de gestión. Los objetos compuestos, contienen otros objetos, los primeros reciben el nombre de Objeto Gestionados Superiores, los últimos Objeto Gestionados Subordinados.

La figura 4.3 ilustra las relaciones lde la plantilla de enlace Nombre con las otras plantillas de GDMO, tienen un areferencia simple a la definición de atributo, exactamente dos a definiciones de clases de objetos gestionados y cero o más a definciones de parámetros y comportaientos.

En GDMO, para que un objeto gestionado pueda existir es necesario que tenga un nombre. Este nombre es utilizado por el protocolo CMIP, para identificar una instancia concreta de una Clase del Objeto Gestionado. La plantilla de Enlace de Nombre, especifica los medios por los cuales nombramos las distintas instancias de las clases de Objeto gestionados. Representa la relación de denominación entre una clase de objeto y las clases superiores de las que desciende, además de la relación existente entre esa misma clase y las clases subordinadas.

En general una instancia de objeto gestionado tendrá un nombre asociado que permite su identificación, por parte de otra instancia de objeto. Un atributo especialmente designado en cada objeto gestionado conocido como el atributo de denominación, sirve de identificación en el enlace de nombre.

La ubicación de una clase en la jerarquía de herencia de las Clases de Objetos Gestionados, se indica por referencia a las superclases, a partir de las cuales, se heredan las especificaciones para las propiedades del Objeto gestionado al que representa.

El enlace de nombre es útil para la definición del ciclo de vida de un objeto gestionado. Define las reglas para la creación, borrado, copiado y asginación de nombre a los objetos.

Cada instancia de objeto gestionado necesita tener un nombre único que se creará concadenando los *Nombre Distinguidos Relativos* (RDN), indicados en el árbol de nombre adecuado. Un RDN es único respecto a su superior y constará de un atributo de idenficación de la plantilla.

En la figura siguiente, la Clase de Objeto Gestionado *multiplexor* es superior, si se considera la clase de objeto gestionado *multiplexorPrimario* como referencia.

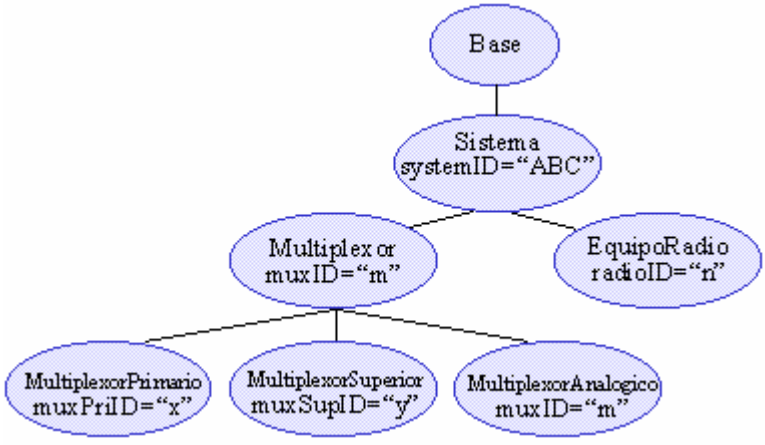

Figura 4.5 Árbol de Nombres

El nombre de la instancia de Objeto Gestionado, se obtiene concadenando los RDNs correspondiente. Es necesario disponer de un atributo conocido como atributo distinguido, que será único en la Clase de Objetos Gestionables *multiplexorPrimario* y es usado para distinguir cada instancia.

Un RDN será único respecto a sus superiores, constará del atributo de nombrado de identificación y un valor asociado que es usado por la plantilla de enlace de nombres.

4.10.1 Estructura.

La plantilla utilizada es la siguiente:

```
<etiqueta-nombre-enlazado> NAME BINDIGN
    SUBORDINATE OBJECT CLASS       <etiqueta-clase> [AND SUBCLASES];
    NAMED BY SUPERIOR OBJECT CLASS <etiqueta-clase> [AND SUBCLASES];
    WITH ATTRIBUTE                 <etiqueta-atributo>
    BEHAVIOUR   <etiqueta-definición-comportamiento>
             [,<etiqueta-definición-comportamiento>]*;]
    [CREATE modificador-crear [,<modificador-crear>]] [<etiqueta-parametro>]*;]
    [DELETE [modificador-borrar] [<etiqueta-parametro>]*;]
REGISTERED AS identificador-objeto;

Soportando producciones:
  Modificad or crear -> WITH-REFERENCE-OBJECT | WITH-AUTOMATIC-INSTANCE-NAMING
  Modificador Borrar -> ONLY-IF-NO-CONTAINED-OBJECTS | DELETES-CONTAINED-ONJECTS
```

Definición 4.8 Plantilla de Enlace de Nombre

El significado de cada cláusula o constructor se explica a continuación.

- `<etiqueta-nombre-enlazado>`: Especifica el enlace de nombre GDMO.

- `SUBORDINATE OBJECT CLASS <etiqueta-clase> [AND SUBCLASES]`: Identifica una Clase de Objeto Gestionado GDMO, cuyas instancias pueden ser nominadas usando un caso de la Clase de Objeto Gestionado especificada en el NAMED BY SUPERIOR OBJECT CLASS.

  La Subcláusula <etiqueta-clase>, identifica a la Clase subordinada AND SUBCLASES, si se incluye especifica que pueden nombrarse casos de una clase derivada de la clase subordinada. Para ello utiliza un caso de la clase especificada en el NAMED BY SUPERIOR OBJECT CLASS.

- `NAMED BY SUPERIOR OBJECT CLASS <etiqueta-clase>`: Esta cláusula recoge una clase cuyas instancias pueden ser utilizadas para identificar una instancia de la clase que aparece en SUBORDINATE OBJECT CLASS. La "etiqueta de clase" identifica la clase superior.

  - `AND SUBCLASSES`: Si se incluye, especifica que instancias de una clase derivada de una superior. Puede utilizarse para nominar una instancia de la clase especificada en SUBORDINATE OBJECT CLASS.

- `WITH ATTRIBUTE <etiqueta-atributo>`: Identifica el atributo que se usará para formar el nombre distinguido relativo (RDN), de un caso de la Clase de Objeto Gestionada indicada en SUBORDINATE OBJECT CLASS. La "etiqueta de atributo", identifica el atributo usado para formar el RDN, de una instancia de la clase especificada en el SUBORDINATE OBJECT CLASS.

- `BEHAVIOUR <etiqueta-definición-comportamiento>`: Constructor que sirve para especificar cualquier conducta resultante específicamente debido al uso del enlace de nombre.

- `CREATE modificador-crear <etiqueta-parametro>`: Especifica que se puede crear una instancia de la clase indicada en SUBORDINATE OBJECT CLASS, usando para ello la operación M-CREATE de CMIP.

El "modificador crear" se usa para describir las opciones permitidas en M-CREATE. Pueden ser:

– WITH–REFERENCE–OBJECT: Una referencia a objeto puede especificarse en la operación M-CREATE.

– WITH–AUTOMATIC–INSTANCE–NAMING: El nombre de instancia de objeto puede omitirse de la operación M-CREATE.

La "etiqueta de parámetro" se usa para detallar los parámetros de error, que se produzcan durante la operación de creación de un enlace de nombre.

• DELETE modificador–borrar <etiqueta–parametro>: Borrar una instancia de la clase SUBORDINATE OBJECT CLASS, usando para ello una operación M-DELETE del CMIP.

El "identificador de borrar" indica la conducta para el borrado de una instancia de objeto gestionado. Pueden ser:

– ONLY–IF–NO–CONTAINED–OBJECTS: Se producirá un error si se intenta borrar una instancia que contiene otras instancias.

– DELETES–CONTAINED–ONJECTS: Permitirá el borrado del objeto en cuestión y todos los contenidos asociado a él.

Se pueden hacer reseñas a otras pautas relacionadas con las operaciones de borrado de objetos, para ello habrá que describirlas en el constructor BEHAVIOUR.

- REGISTERED AS identificador–objeto: Se emplea para asociar un identificador único del objeto.

4.10.1.1 Ejemplo.

Sea el siguiente ejemplo:

```
multiplexorPrimarioNaming   NAME BINDING
      SUBORDINATE OBJECT CLASS multiplexorPrimario AND SUBCLASSES;
      SUPERIOR OBJECT CLASS  multiplexor;
      WITH ATTRIBUTE  multiplexorPrimarioId;
      BEHAVIOUR  multiplexorPrimarioConducta;
      CREATE   WITH-AUTOMATIC-INSTANCE-NAMING;
      DELETE  ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS               {1 3 5 8 9 20};
```

En la plantilla precedente existen aspectos interesantes.

El atributo distinguido *multiplexorPrimarioId,* se utiliza para la formación del RDN de la clase *multiplexor*.

SUBORDINATE OBJECT CLASS, contiene la etiqueta de la plantilla de la Clase de Objeto Gestionado, sobre la que se define el NAME BINDING.

AND SUBCLASSES establece que se puede utilizar el atributo *multiplexorPrimarioId,* para nombrar subclases de la clase *multiplexorPrimario.*

BEHAVIOUR especifica las distintas posibilidades a tener en cuenta, en el caso de que exista más de una relación de Enlace de Nombre.

Cuando se crea una instancia de un objeto, no es necesario incluir un nombre para la operación Create Object. A la operación Delete podemos asociarle restricciones del tipo ONLY-IF-NO-CONTAINED OBJECTS, ya que un Objeto Gestionado que esté siendo borrado puede contener cero o más objetos. Antes de realizar la operación Delete, debe realizarse primero el borrado de todos los objetos contenidos, de otra manera se producirá un error.

La opción DELETES-CONTAINED-OBJECTS, permite el borrado de un objeto junto con todos los contenidos en él. No es necesario el borrado previo de los objetos contenidos. Tener en cuenta que esta operación entraña un mayor riesgo, ya que se pueden borrar objetos.

A continuación se muestra otro ejemplo de plantilla de Enlace de Nombres, para una clase que contiene el objeto gestionable Hub (IEEE 802.3)

```
 HubName NAME-BINDING
  SUBORDINATE OBJECT CLASS  Hub;
  NAMED BY SUPERIOR OBJECT CLASS ISO/IEC 10165-2:System:
  WITH ATTRIBUTE HubID;
  BEHAVIOUR HubIDBehaviour,
REGISTERED AS {iso(1)std(0)iso8802(8802)csma(3)hubmgt(18)namebinding(3)hubname(X)};
```

4.11 Plantilla de Comportamiento.

Usada para describir el comportamiento esperado para una o más operaciones soportadas por una Clase de Objetos Gestionados. Cuando se define un objeto de gestión, es necesario especificar como se comportan los atributos, operaciones, notificaciones y nombres enlazados, todos estos detalles se especifican en esta plantilla. Parte de la definición de un objeto gestionado es la especificación del conjunto de operaciones de gestión que pueden realizarse en el mismo, asimismo el efecto que estas operaciones de gestión tienen sobre el objeto gestionado y sus atributos.

Así la ejecución de una operación de gestión puede estar condicionada al estado del objeto gestionado o a sus atributos. Los objetos gestionados pueden también emitir notificaciones, que contienen información relativa a la ocurrencia de un evento asociado al objeto gestionado.

Esta plantilla registra la semántica de los atributos, las operaciones y las notificaciones, e indica las relaciones, efectos laterales, etc. con otros Objetos Gestionados. El comportamiento generalmente se describe de manera informal utilizando lenguaje cotidiano.

También existen enfoques para describir el comportamiento, que incorporan formal *descriptions techniques* (FDTs), incluye un lenguaje de descripción y otro de especificación (SDL, ITU-T Z.100), que han sido agregados a la serie de estándares ISO 10165.

4.11.1 Estructura.

La Estructura es la siguiente:

```
<etiqueta-atributo>  BEHAVIOUR
   DEFINED AS "descripción";
```

Definición 4.9 Plantilla de Comportamiento

Sus elementos:

• `DEFINED AS:` El texto contenido dentro de una cadena de texto, expone los aspectos del comportamiento de la Clase de Objeto Gestionado o Enlaces de Nombre asociados, parámetros, atributos, acciones o notificaciones. La sintaxis será el lenguaje natural, aunque como ya hemos mencionado también está permitido el uso de alguna técnica formal de descripción. Además permite la inclusión de referencias a cláusulas incluidas en estos documentos de Recomendaciones o Normalizaciones Internacionales.

4.11.1.1 Ejemplo.

La definición del comportamiento vendrá especificada por un texto descriptivo tal y como se aprecia en la siguiente definición:

```
sistemaIdConducta    BEHAVIOUR
    WITH REPLY SYNTAX CHARACTER STRING SIZE (128);
    DEFINED AS "Identifica el sistema, normalmente un
código asignado por el fabricante, también puede ser un
número de inventario o código administrativo asginado
por el usuario.";
```

La plantilla de comportamiento es una de las que más se usan en la práctica. Esto se debe a que todas las demás plantillas GDMO hacen referencia a ésta, a excepción de la de Clase de Objeto Gestionado y la de Grupo de Atributo.

4.12 Revisiones y Registros de Clases de Objetos Gestionados.

El proceso de definición de las clases de objetos gestionados requiere de la asignación de un identificador global único, conocido como identificador de objeto, necesario para cada uno de los elementos de la clase de objeto gestionado, como es su nombre, tipos de atributos, reglas, etc. El valor de estos identificadores lo utiliza el protocolo de gestión, para identificar de forma única cada uno de los aspectos relacionados con el objeto gestionado, sus atributos, acciones y notificaciones.

Es por tanto un requisito indispensable para la definición de una nueva Clase de Objeto Gestionado, que exista un organismo responsable de identificar y establecer los mecanismos necesarios para que el registro de esta nueva clase pueda realizarse, emitiendo para ello unos valores identificadores del objeto, necesarios para su uso posterior.

El CCITT Rec. X.208 | ISO/IEC 8824 proporciona una estructura para los identificadores de objeto y los valores de los distintos nodos iniciales, de los cuales dependerán los nuevos identificadores que se definan. Para obtener más información acerca de los mecanismos de registros establecidos y las autoridades registradores, puede consultarse la recomendación X.660 de CCITT | ISO/IEC 9834-1.

Todos los valores de objetos identificados en las Recomendaciones de sistemas de gestión | International Standars se localizan debajo del nodo inicial:

{join-iso-ccitt ms(9)}

A continuación, se observa los distintos arcos situados debajo de este nodo definidos en la norma:

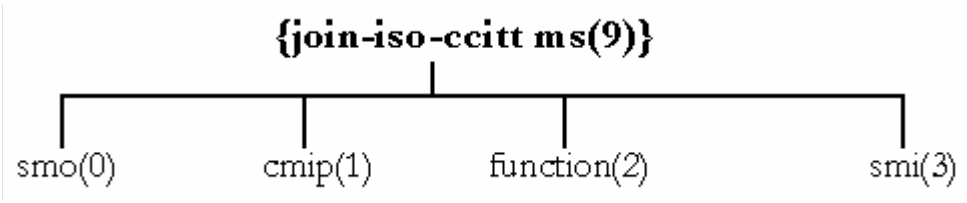

Figura 4.6 Nodos localizados debajo del principal

A continuación pasaremos a enumerar los distintos elementos ubicados en los niveles inferiores. Aquí se situarán los nodos futuros, surgido como consecuencia de la aparición de nuevos estándares de gestión. Cuando estos se produzcan habrá que hacer una nueva revisión de la norma.

4.12.1 Smo(0). Preámbulo de los Sistemas de Gestión.

ISO 10040/ITU-T X.701, System Management Overview. Proporciona una visión global de los estándares de sistemas de gestión y establece las bases para los sistemas de gestión distribuida. Introduce el concepto de dominio administrativo de gestión.

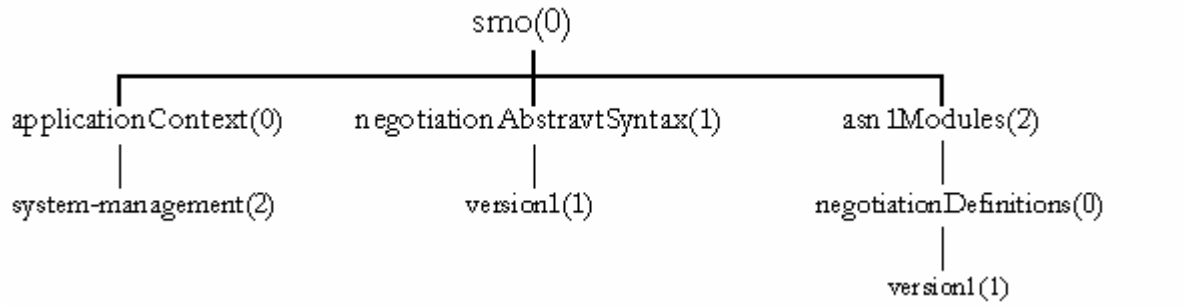

Figura 4.7 Subárbol Visión Global de los Estándares de Gestión

- **applicationContext(0),** este nodo registra identificadores de contexto para aplicaciones particulares. Debajo se sitúa *systems-management(2),* el identificador para el marco de aplicación de los Sistemas de Gestión.

- **negotationAbstractSyntax(1),** que identifica la primera versión de la negociación de sintaxis abstracta.

- **asn1Modules(2),** registra el modulo ASN.1 de identificadores particulares. Dentro se localiza *negotiationDefinitions(0),* para registrar versiones particulares del módulo ASN.1, tal y como consta en *versión(1).*

4.12.2 Cmip(1). Protocolo de Gestión de Información Común.

ISO 9596-1/ITU-T X.711, *Common Management Information Protocol (CMIP).* Recomendación que especifica el protocolo usado por las entidades de capa de aplicación, para el intercambio de la información de gestión.

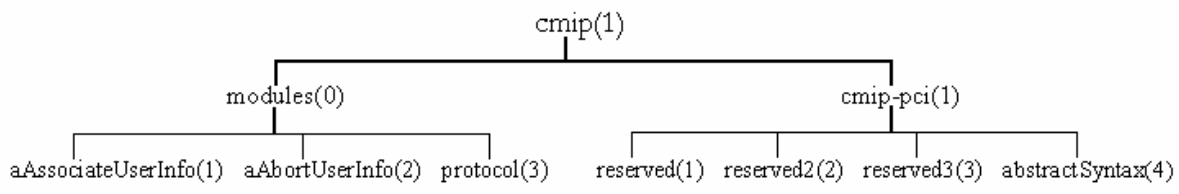

Figura 4.8 Subárbol correspondiente al Protocolo de Gestión

• **modules(0),** localiza los identificadores de objetos para los módulos ASN.1 de CMIP.

• **cmip-pci(1),** localiza los identificadores para la información de control del protocolo CMIP.

De forma general esta recomendación especifica:

- Los procedimientos para la transmisión de información de dirección entre las entidades de la aplicación.

- La sintaxis abstracta del Protocolo de Información de Gestión Común (CMIP) y las reglas de la codificación asociadas.

- Los procedimientos para la interpretación correcta de información de control;

4.12.3 Funtion(2). Funciones de Gestión de Sistemas.

Corresponden con ISO 10164-X|ITU-T X.7NN, donde X es la parte correspondiente al esquema de numeracíon en el estándar ISO, y X.7NN corresponde al número de la recomendación en ITU-T.

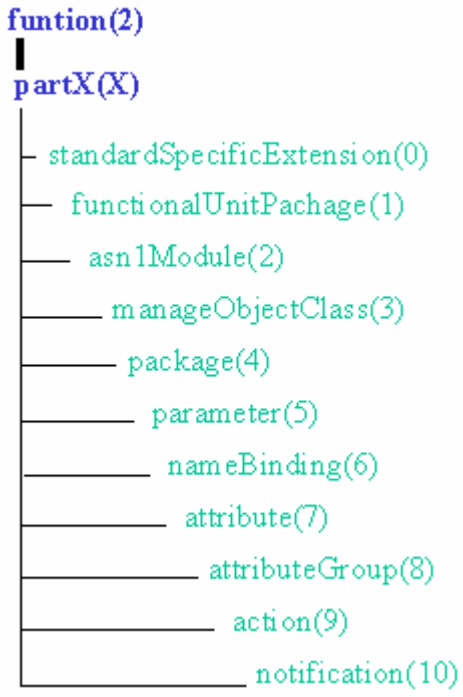

Figura 4.9 Funciones de Gestión

A continuación los enumeramos:

• **Norma ISO 10164-1/ITU-T X.730, Object Management Function,** Función de Gestión de Objeto. Esta norma define los objetos gestionados, describe las operaciones necesarias para su creación y configuración. Reglas de creación, borrado, cambio de nombre de los objetos, así como el borrado y cambio de valor de sus atributos.

• **ISO 10164-2/ITU-T X.732, State Management Function,** Función de Gestión de Estado. Identifica los distintos estados para los objetos gestionados. El estándar de gestión OSI permite dos clasificaciones de estados:

- *Operacional:* Para definir un objeto como ocupado, activo, enable o disable.

- *Administrativo:* Cerrado, abierto y protegido.

Esta especificación también indica las reglas para salir y entrar en estos estados.

• **ISO 10164-3/ITU-T X.733, Attributes for Representing Relationships,** Objetos y Atributos para Representar Relaciones. Define las relaciones entre los objetos gestionados, se establecen las siguientes:

- *Directa:* Parte de la información asociada a un objeto gestionado, que identifica a otro objeto.

- *Indirecta:* Una relación se deduce a través de dos objetos, concadenando dos o más relaciones directas.

- *Simétrica:* La iteración de reglas entre dos objetos es la misma.

- *Asimétrica:* La iteración de reglas entre dos objetos es distinta.

• **ISO 10164-4/ITU-T X.734, Alarm Reporting,** Función de Informe de Alarma. Define e identifica cinco categorías básicas de notificaciones:

- Comunicaciones.
- Calidad de Servicio.
- Procesamiento.
- Equipamiento.
- Entorno.

Además de estas cinco notificaciones, las alarmas proporcionan información referente a la posible causa de la alarma, gravedad de la misma, acciones a efectuar, etc.

• **ISO 10164-5/ITU-T X.735, Event Report Management Function,** Función de Gestión de Aviso de Informe. Establece las componentes para soportar el envío remoto de eventos y el procesado local de éstos, para ello se utiliza el concepto de discriminador.

• **ISO 10164-6/ITU-T X.736, Log Control Function,** Función de Control de Traza. Define las operaciones de control de registro histórico para un Sistema de Gestión de Red. Indica como se guarda la información correspondiente a las operaciones realizadas

sobre los objetos gestionados, se definen los criterios creación, borrado de ficheros de Log, etc.

- **ISO 10164-7/ITU-T X.737, Security Alarm Reporting Function,** Función de Notificación de Alarma de Seguridad. Define los procedimientos para la gestión de seguridad y comunicación de los distintos tipos de alarmas enviadas por los objetos gestionados. Se especifican los siguientes tipos de alarmas relacionadas con la seguridad:

  - Integridad.
  - Operacional.
  - Física.
  - Seguridad.
  - Time-to-Remain.

Posibles causas de emisión de alarmas pueden ser: que la información esperada no se ha recibido, modificación ilegal de la información recibida, etc.

- **ISO 10164-8/ITU-T X.740, Security Audit Trail Function,** Función de seguimiento de auditoria de seguridad. Análoga a la función de control de log, pero además ofrece la posibilidad de realizar auditorias sobre el fichero de históricos: contabilidad, seguridad, conexiones y desconexiones realizadas, etc.

- **ISO 10164-9/ITU-T X.741, Objects and Attributes for Access Control,** Objetos y atributos para el control de accesos. Permite al administrador de la red prevenir accesos no autorizados sobre los objetos gestionados. Define mecanismos de control de acceso basados en reconocimientos de perfiles de usuario y definen reglas para denegar el acceso a los elementos gestionados.

- **ISO 10164-10/ITU-T X.742, Accounting Meter Function.** Utilización de funciones de medida con propósitos contables. Esta norma define como se contabiliza, costo, registro e identificación del uso de la red de gestión. Además provee de umbrales de uso para determinados objetos gestionados, a los distintos usuarios del sistema.

- **ISO 10164-11/ITU-T X.739, Workload Monitoring Function.** Define un modelo de monitorización para un objeto gestionado. Establece procedimientos de regeneración, que serán usados en orden a aspectos relacionados con el funcionamiento de los objetos.

- **ISO 10164- Otras Funciones.** Tales como: Métrica de objetos y atributos, Función de gestión de diagnóstico, Función de planificación de sistemas de gestión, etc.

4.12.4 Smi(3). Modelo de Información de Gestión.

Define la base de información de gestión (MIB - *Management Information Base*), la cual contiene una representación de todos los objetos bajo el ambiente OSI que van a ser gestionados.

Corresponden con ISO 10165-X|ITU-T X.7NN, donde X es la parte del número correspondiente al esquema de numeracíon en el estándar IS, y X.7NN corresponde al número de recomendación en ITU-T.

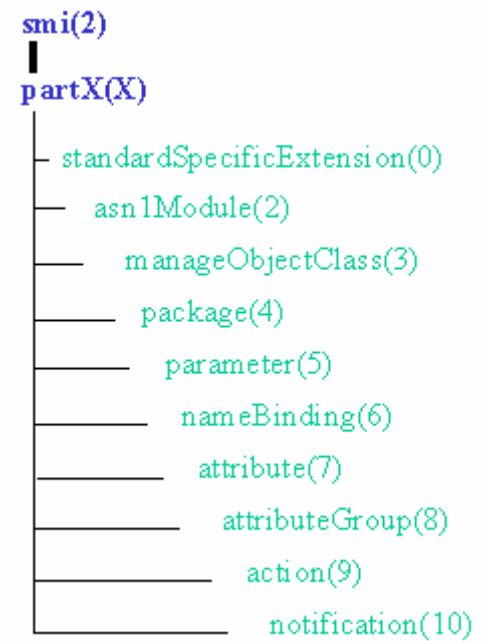

Figura 4.10 El Modelo de Información de Gestión

El estándar que trata la Estructura de Información de Gestión (SMI), se divide en cuatro partes:

- **ISO 10165-1/ITU-T X.720, Management Information Model,** Modelo de información de gestión. Esta norma presenta un modelo de información de gestión general para OSI, cuya finalidad es *"proporcionar una estructura a la información de gestión transportada externamente por los protocolos de gestión de sistemas y modelar los aspectos de gestión de los recursos conexos (p.e. una máquina de protocolo X.25)".*

- **ISO 10165-2/ITU-T X.721, Definitions of Support Objects,** Definición de la información de gestión. Esta recomendación define cierto número de clases de objetos gestionados, utilizadas en las diferentes funciones de gestión de sistemas. Sirven como superclases para la definición de nuevas clases de objetos mediante la herencia y que conforman la MIB.

- **ISO 10163-3, Definitions of management attributes.** Especifica los atributos usados por la gestión: cambios de estado, estado de error, estado de configuración, nombre distinguido de la clase de objeto, definiciones, etc.

- **ISO 10165-4/ITU-T X.722, Guidelines for the Definition of Managed Objects.** Guía para la definición de objetos gestionables. Realiza una amplia exposición de los principios básicos para la definición de los objetos gestionados. Sirven de orientación a quienes realizan la especificación de dichos objetos, así como favorecer la coherencia entre las definiciones que de ellos se establecen.

4.12.5 Valor del Identificador del Objeto.

En cada Recomendación, futuros nodos pueden ser añadidos debajo de este nivel (por ejemplo para localizar un identificador de un atributo particular), como requerimiento de la Recomendación.

El nuevo identificador se situaría en el siguiente nivel:

{join-iso-ccitt ms(9) smi(9) part4(4) managedOjectClass(3) actualClass(42)}

de esta forma, la definición de esta clase "*actual class"* quedaría incluida en el lugar indicado de la recomendación X.720 de CCITT | ISO 10165-1. Cuando este sea usado para especificar la clase de objeto gestionado a la que se le solicita una operación de servicio CMIS. El valor identificador de objeto indica que el destinatario de la operación de gestión del sistema, responda como un miembro de esta clase.

Capítulo 5

Integración de Reglas Expertas en el Modelo GDMO.

A lo largo de los capítulos anteriores, se ha visto que la complejidad y exigencias a las que están sujetas las redes de telecomunicaciones actuales, hacen que su gestión sea un aspecto muy importante. Para efectuar una administración eficiente de estos sistemas tan complejos, los estándares tradicionales de gestión no son suficientes y será necesario desarrollar nuevos modelos de control y supervisión que ofrezcan mayores posibilidades, una alternativa son los denominados *Sistemas de Gestión Experta Integrada.*

Este nuevo paradigma, contiene aspectos claramente diferenciadores a las técnicas actuales de gestión. En estas, se hace uso por separado de los Sistemas Expertos y de las plataformas de gestión. Existe una frontera definida entre el conocimiento aportado por el sistema experto y la plataforma encargada de la aplicación de dicho conocimiento en la administración y control de los Sistemas Telemáticos, pudiéndose considerar, incluso a cada uno por separado.

En esta tesis proponemos una nueva aproximación a la gestión de redes en la cual el sistema experto queda completamente integrado en la plataforma de gestión. Concretamente, las reglas expertas de gestión, que componen la base de conocimientos del Sistema Experto, se fusionan con la definición de los elementos que conforman la red de comunicaciones. Cada una de estas definiciones, contendrá las especificaciones correspondientes a los recursos que componen el sistema de información y el conjunto de reglas expertas que posibilitan el control y administración inteligente de los recursos que representan, objetivo principal del modelo.

Este nuevo modo de acometer la gestión de las redes de telecomunicaciones que presentamos y su implantación [León00], no está exento de las consiguientes dificultades. En este capítulo se abordan los aspectos principales de nuestra propuesta. Se expone la forma en que se realiza la integración de la base de conocimiento[1] del Sistema Experto, en la definición de los elementos del sistema a gestionar. Se utiliza para ello el estándar GDMO estudiado con detenimiento en el capítulo cuarto de esta tesis.

[1] La base de conocimientos, contiene todos los hechos, las reglas y los procedimientos del dominio de aplicación que son importantes para la solución del problema.

5.1 Gestión Experta Tradicional.

Comenzamos presentando el modelo que habitualmente se ha ido aplicando en la gestión de las redes de telecomunicaciones y que podemos denominar *"Modelo Tradicional"*. El sistema experto, es un elemento primordial que está presente en las plataformas de gestión denominadas inteligentes, Figura 5.1.

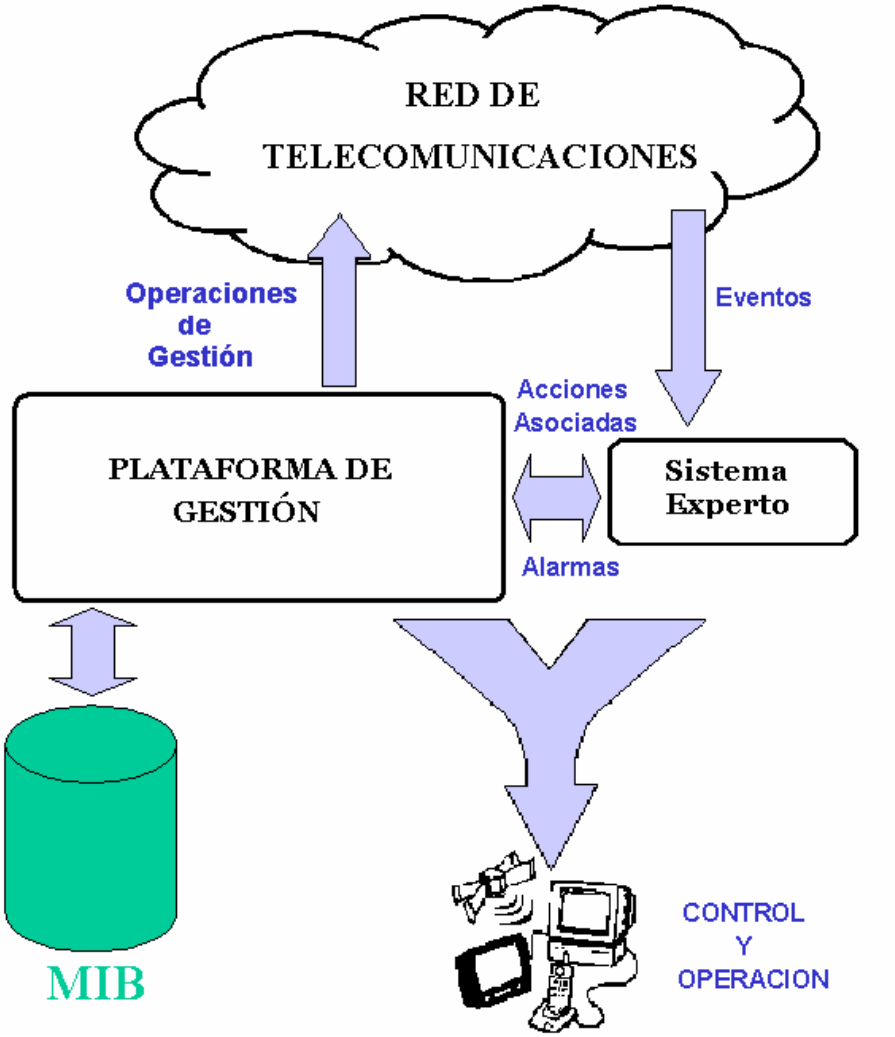

Figura 5.1 Arquitectura de la Gestión Experta tradicional de una Red

Los Sistemas Expertos utilizados en las primeras plataformas de gestión, se realizaron utilizando lenguajes de programación como LISP y PROLOG. A medida que el desarrollo y la complejidad de estos iban en aumento, la comunidad científica comenzó a buscar nuevas formas de concebir los sistemas, en menor tiempo y con menos esfuerzo.

Aparecieron en el mercado herramientas denominadas *Entornos de Desarrollo*, que junto con las opciones de representación del conocimiento, incorporaron además esquemas de inferencia y control, entre los que se encuentran: CLIPS, KEE, ART, etc. Estos entornos permiten crear sistemas expertos, programando el conocimiento experto en un área determinada. Asimismo, el propio entorno de desarrollo puede incluir elementos exclusivos de los sistemas expertos, como el motor de razonamiento y la interfaz de usuario.

Los Sistemas Expertos juegan un papel importante en el buen funcionamiento de los sistemas informáticos y se hace imprescindible su aplicación. Resuelven problemas de difícil solución tal y como lo harían expertos humanos. Pueden procesar grandes volúmenes de información, así como razonar de forma más rápida para la obtención y justificación de sus conclusiones. Examina la información proveniente de la plataforma de gestión: eventos, alarmas, etc. y realiza el razonamiento como lo haría un experto humano, hasta alcanzar un diagnóstico. Efectúa un análisis basado en inferencia lógica, para detectar las posibles situaciones de fallo. Una vez analizado el problema, indica las causas probables y propone las líneas de actuación para su solución inmediata [Giarratano01].

Una vez creado el Sistema Experto, es necesario proporcionar los engranajes necesarios para que la plataforma de gestión consiga interactuar con él. Esta interfaz de comunicación entre Sistema Experto y Plataforma de Gestión, dependen en gran medida de factores y características que tienen que ver con ambos elementos. Para cada combinación Sistema Experto/Plataforma, hay que desarrollar una interfaz única, en función de las posibilidades y los mecanismos que cada sistema posea para la comunicación externa con las aplicaciones: sockets, CORBA, etc.

El Sistema Experto puede ser tratado como una caja negra que recibe los eventos procedentes de los elementos que componen el sistema gestionado, los procesa y emite las acciones a efectuar. Si en cualquier momento necesitamos reemplazar el sistema experto, tendremos que realizar los cambios correspondientes en la interfaz de la plataforma de gestión, para que ambos elementos sigan interactuando. La situación es análoga para el caso en que sea necesario sustituir la plataforma de gestión. En resumen la realización de cambios en la plataforma de gestión o en el sistema experto, puede implicar que haya que verificar de nuevo la interconexión entre ambos elementos.

Es evidente que aunque el Sistema Experto y la Plataforma de Gestión, son elementos completamente distintos, en mayor o menor medida debe existir entre ellos un cierto nivel de compatibilidad e interoperatividad, que asegure el flujo de datos e información de control en ambos sentidos. Este trabajo puede ser arduo, y es el administrador del sistema de gestión el encargado de su correcta realización, Figura 5.2.

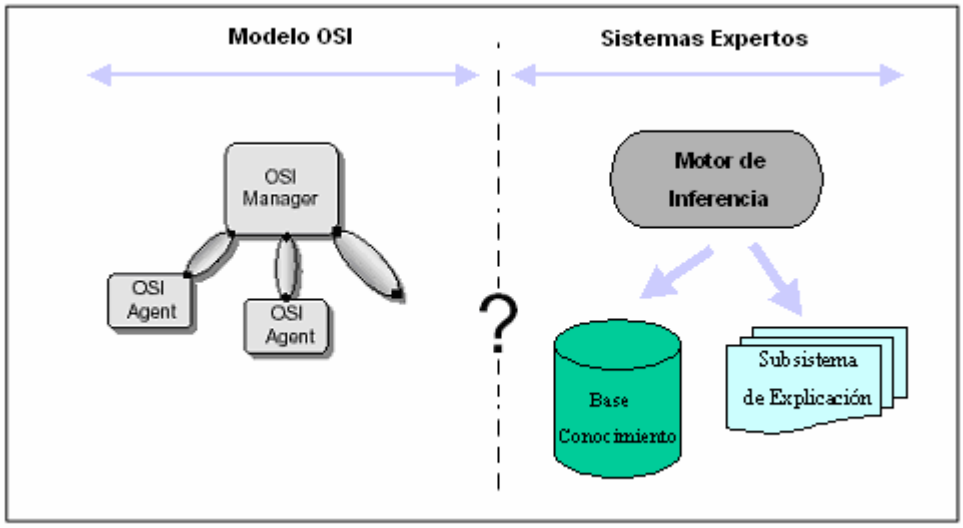

Figura 5.2 Independencia de los Objetos gestionados y la Gestión Experta.

Con la gestión inteligente integrada, pretendemos evitar este tipo de inconvenientes. La interconexión entre el *sistema experto* y la *plataforma de gestión* no es necesaria. Ambos

componentes se definen utilizando la misma sintaxis, *quedando los dos completamente integrados dentro de unas especificaciones únicas.*

5.2 GDMO y la Gestión Inteligente.

Mediante la sintaxis GDMO, se establecen las especificaciones correspondientes a los recursos gestionados. A su vez, independiente de las especificaciones GDMO y utilizando el software apropiado, se obtiene la base de conocimiento del Sistema Experto. Esta base de datos contiene las reglas expertas de gestión, aplicables a un dominio concreto de administración.

El sistema experto no tiene acceso a la MIB (Base de Información de Gestión), que contiene toda la información de gestión de los recursos de la red de telecomunicaciones [Stallings00]. La plataforma de gestión es el nexo que une las dos fuentes de información: la información de gestión y el conocimiento necesario para su correcta administración.

Como se aprecia en la siguiente Figura 5.3, en la arquitectura *tradicional* de los sistemas inteligente de gestión, se mantienen dos estructuras de información, que poseen tipos de especificaciones distintas. Cada una de estas fuentes de información, pueden usar lenguajes y métodos de programación distintos, completamente independientes, que pueden presentan distintos niveles de abstracción. Este tipo de arquitectura puede suponer un coste elevado en la administración y en el desarrollo de sistemas de gestión.

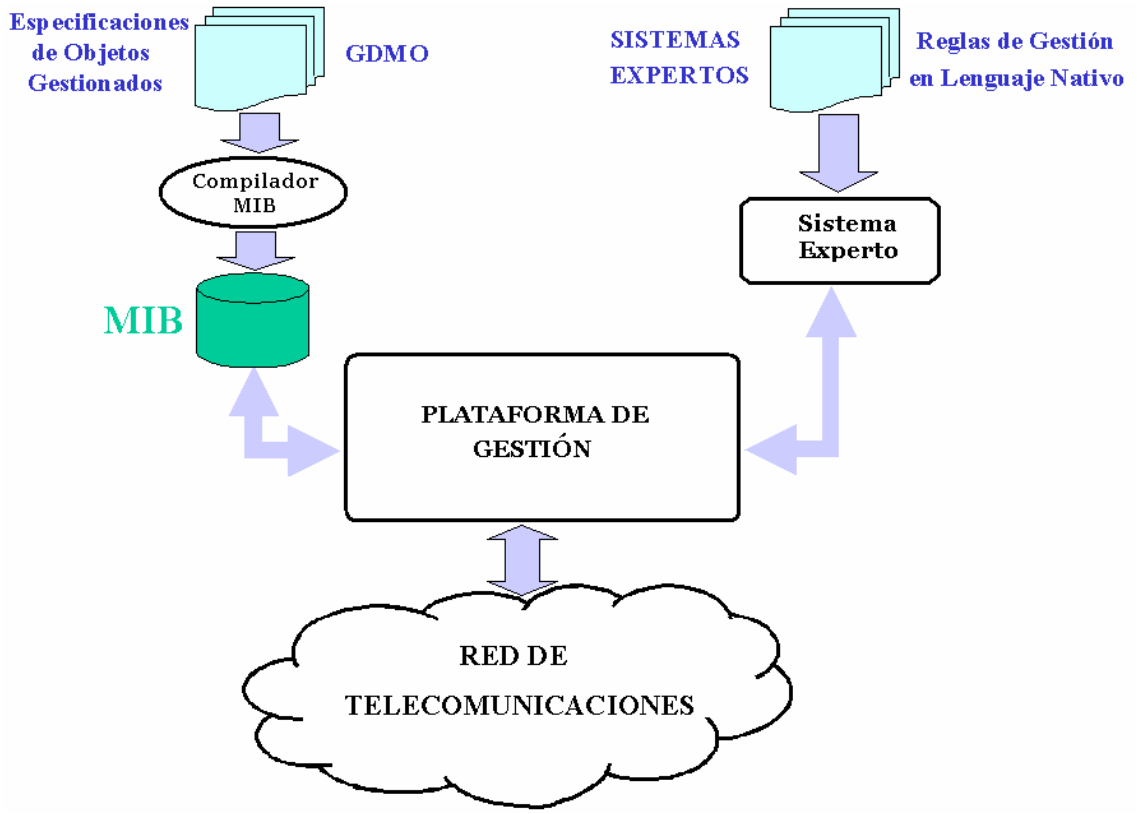

Figura 5.3 Arquitectura Tradicional de una Plataforma de Gestión Experta.

La definición de los objetos gestionados se realiza mediante un lenguaje de especificación de alto nivel, el estándar GDMO. Por otro lado las reglas expertas se componen en un lenguaje de programación propio del sistema experto, usualmente propietario del sistema y que puede ser incompatible con otros sistemas o lenguajes de programación. El administrador debe hacerse cargo de resolver problemas de bajo nivel,

como por ejemplo el establecer la correspondencia entre los eventos y las acciones relacionadas a estos. Estas operaciones son complejas y facilitan la aparición de errores.

5.3 Desventajas de la Gestión Experta Tradicional.

En el apartado anterior hemos visto que la gestión tradicional presenta carencias. A continuación se enumera una serie de deficiencias presentes en la gestión experta tradicional, que ponen de relieve los inconvenientes presentes en los sistemas actuales de gestión y como pueden ser resueltos utilizando el método de gestión experta integrada propuesto en nuestro trabajo.

• Desventajas derivadas de mantener dos elementos completamente independientes: la plataforma de gestión y el sistema experto. Hay que mantener unos vínculos, que permitan el establecer un dialogo permanente y continuo entre ellos.

• Las restricciones que surgen a la hora elegir una determinada plataforma. Se pueden presentar problemas de incompatibilidad con el sistema experto utilizado.

• La necesidad de utilizar redes heterogéneas, puede presentar dificultades a la hora de definir las reglas expertas de gestión de los recursos presentes.

El modelo de gestión OSI, ofrece un entorno favorable para nuestra proposición de diseño de una plataforma de gestión experta integrada. Contiene una arquitectura y unas características óptimas para el diseño de una plataforma de gestión que permita la integración de la base de conocimiento perteneciente al Sistema Experto, en las definiciones de los elementos que conforman el sistema gobernado.

5.4 Gestión Experta Integrada.

El objetivo y contribución principal de esta tesis es proveer de un método general y sistemático, que permita a un gestor OSI acceder a las reglas expertas de gestión integradas en el propio modelo y que representan la base de conocimiento del sistema experto. Estas reglas expertas, recogen las condiciones bajo las cuales se efectúan las distintas acciones de gestión de los elementos de la red. Cada dominio de aplicación, tendrá a su vez su propia base de reglas, que permitirá realizar las funciones de administración y control, relacionadas con el recurso de gestión donde se definen.

Se pondrá especial énfasis en el logro de los siguientes objetivos:

- En primer lugar, conocimiento de las necesidades requeridas por la Gestión Experta Integrada. Identificación de metas para la integración del Sistema Experto y la definición de los Objetos Gestionados.

- Proposición de medidas que mejoren la calidad global de GDMO y sus especificaciones.

- Plantear una nueva sintaxis de la norma GDMO que permita la composición de ambos factores, abriendo nuevas vías de propuestas para la mejora del estándar.

Se pretende eliminar la ruptura existente entre ambos componentes, sistema experto y plataforma de gestión. Del mismo modo, favorecer que el estándar de gestión OSI integre las bases de conocimientos aportadas por los Sistemas Expertos y que puede corresponder a distintos dominios de gestión [Hegering94].

En la siguiente Figura 5.4, se puede apreciar de forma gráfica el proceso de integración de las reglas expertas de gestión en las especificaciones de los recursos de un sistema de comunicaciones:

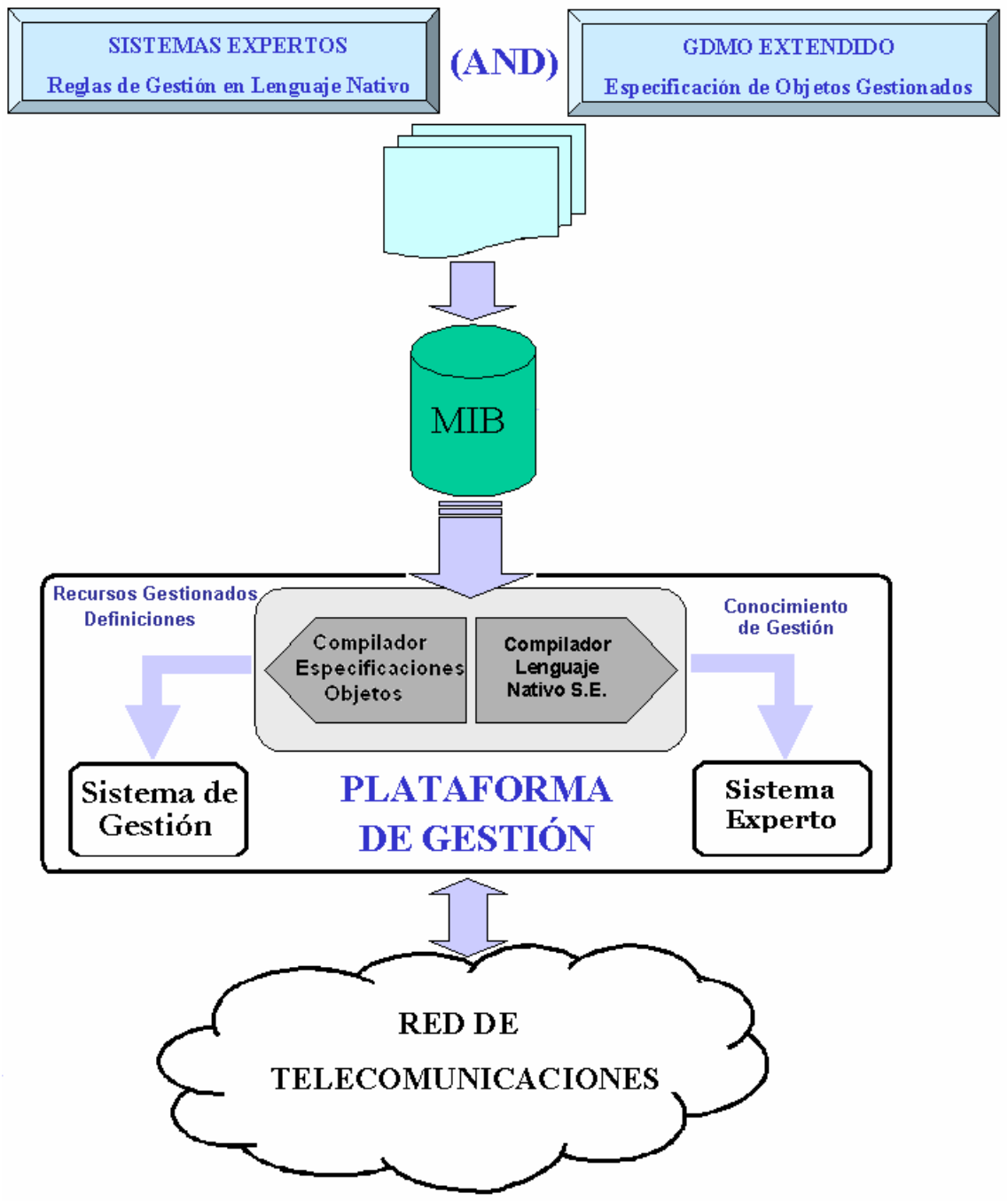

Figura 5.4 Integración de Reglas de Gestión en GDMO.

El estándar de gestión de sistemas OSI, propone una Base de Información de Gestión MIB, que contiene una colección de los datos necesarios para la gestión del sistema de telecomunicaciones. A cada una de las colecciones de objetos, referentes a un área de gestión común, se le conoce con el nombre de "*módulo MIB*". Conceptualmente la MIB representa una base de datos de "*parámetros de gestión*", que se almacenan siguiendo una estructura arborescente. Las entradas cercanas a la raíz del árbol definen las organizaciones de las cuales dependen los diferentes parámetros. Alejándonos de la raíz vamos

encontrando el emplazamiento concreto de los verdaderos parámetros de gestión (objetos MIB). Cada uno de los diferentes dispositivos proporcionará diferentes informaciones. Así un encaminador puede proporcionar a un gestor, su tabla de encaminamientos; un puente su tabla de filtrado, etc. En nuestro caso, pretendemos añadir nueva información a los objetos gestionados, la correspondiente al conocimiento aportado por los sistemas expertos y que se recogen en un conjunto de reglas expertas de gestión.

La definición de las distintas características de los objetos de la red y la forma en que los protocolos de gestión obtienen información de ellos, se declara por medio de una Estructura de la Información Gestión, SMI (*Structure of Management Information*). SMI identifica los tipos de los datos que pueden usarse, así como la nominación y representación de los recursos existentes dentro de la MIB.

Las especificaciones de objetos gestionados se realizan conforme a la Guía para la Definición de los Objetos Gestionados GDMO, que define la estructura y las relaciones existentes entre los distintos tipos de objetos gestionados. Está compuesto por nueve plantillas, es objeto de estudio y punto de partida de nuestra propuesta de integración del conocimiento de gestión, en la definición en los objetos gestionados.

La motivación principal, es proporcionar a los sistemas gestión un mayor grado de integración con las tecnologías de comunicaciones actuales, a la vez de disponer de todas las posibilidades y ventajas aportadas por la inteligencia artificial. Facilitando además una gestión mucho más eficiente de los recursos presentes en los grandes sistemas telemáticos.

Dependiendo de las herramientas y la metodología utilizada, hoy en día existen diferentes soluciones de integración. La gestión experta integrada obtiene una solución global, que permite a los administradores de redes utilizar la potencia aportada por la Inteligencia Artificial y en particular de los Sistemas Expertos, de una forma sencilla y transparente.

5.5 Ventajas de la Integración.

La presente sintaxis GDMO, no permite la integración de las reglas expertas de gestión, que sistemáticamente van a permitir facilitar la gestión de los recursos definidos en una red. El estándar actual carece de los constructores necesarios para su integración. En las distintas plantillas que lo conforman, no hay ninguna que manipule las operaciones de supervisión y control tan necesarias para la correcta administración de un sistema de comunicaciones, cuando se producen los fallos y averías.

La solución pasa por el desarrollo de una interfaz que admita la introducción de reglas expertas de gestión en la definición de los objetos de la MIB, de una manera simplificada e independiente de un entorno específico y con una implantación reglamentada. Esta solución será en general, más flexible que la realizada siguiendo los métodos tradicionales (lenguaje formal), en los cuales se crean las especificaciones correspondientes a las reglas expertas de gestión y a continuación se interconectan a la plataforma de gestión, considerando toda la problemática que esto conlleva [Hegering94].

Ventajas que se obtienen con la gestión Inteligente Integrada:

- ***Abstracción de los usuarios de la plataforma de gestión***. Respecto a la configuración y reglas expertas de gestión pertenecientes al sistema experto. No necesitan tener conocimientos de programación del Shell del Sistema Experto.

- ***Mayor rapidez en la resolución de problemas***. Al estar el sistema experto integrado en la plataforma de gestión formando una sola unidad modular, los retrasos y esperas producidas por la comunicación entre ambos elementos es inferior o desaparece.

- Compatibilidad entre las distintas plataformas de gestión. Integración del sistema experto en la norma *IS 10165-4/ITU-T X.722, Guidelines for the Definition of Managed Objects*[2][ITUT93]. Los estándares serán más rigurosos, fortaleciendo el grado de coexistencia y entendimiento entre las distintas plataformas y los modelos de gestión.

- ***Estandarización de las nuevas reglas expertas***. Por parte de un organismo de normalización como es ISO. Esto asegura la correcta integración de las definiciones de las reglas expertas de gestión, en las especificaciones de los elementos sobre los que actúan.

- Facilidad de reutilización de las definiciones y especificaciones existentes en el estándar. Mediante los mecanismos de herencia, propio de los lenguajes Orientados a Objetos, como es el caso de la sintaxis GDMO, proporciona un marco perfecto para la reutilización de las clases y las reglas expertas de gestión definidas en estas. El encapsulamiento y la Modularidad nos permiten utilizar una y otra vez las mismas clases en dominios de gestión distintos. Las relaciones existentes entre las distintas clases, hacen posible el añadir una nueva clase o un modulo nuevo (extensibilidad), sin afectar al resto de las especificaciones.

- ***Agilidad para la definición y realización de nuevas plataformas de gestión experta***. Se afianza la compatibilidad entre las distintas plataformas de gestión de sistemas telemáticos, a la vez que facilitan la comunicación y el trabajo conjunto entre ellas.

Todas estas ventajas al momento de codificar y producir, tienen en contrapartida un gran esfuerzo en el momento del diseño de las especificaciones de los objetos gestionados. Las nuevas especificaciones van a recoger detalles sobre las propiedades del recurso que representan y las declaraciones correspondientes al conocimiento que lo gestiona, lo que requiere de un exhaustivo trabajo de análisis y también de mucha disciplina.

5.6 Problemática de la Gestión Inteligente Integrada.

A medida que se profundiza en el tema, se constata que este nuevo argumento no está exento de objeciones y cuenta con dificultades propias.

Las cuestiones básicas que se plantean son las siguientes:

1. Cómo formular, describir, descomponer y asignar el conocimiento entre los distintos agentes inteligentes definidos en GDMO.

2. Cómo capacitar a los distintos objetos para que se comuniquen el conocimiento e interactúen, que lenguaje de comunicación o protocolos deben utilizarse, qué y cuando deben comunicarse, etc.

---

[2] *Guía para la definición de objetos gestionables.* Realiza una amplia exposición de los principios básicos para la definición de los objetos gestionados, que van a servir de orientación a quienes realizan la especificación de dichos objetos, así como favorecer la coherencia entre las definiciones que de ellos se establecen.

3. Cómo asegurar que los objetos actúan coherentemente al tomar decisiones o realizar acciones, acomodar los resultados de las decisiones locales y prevenir efectos no deseados.

4. Cómo capacitar a los objetos para representar y razonar sobre las acciones, planes y conocimiento de otros objetos, coordinación de resultados.

5. Cómo diseñar metodologías y plataformas de gestión utilizando técnicas de la Inteligencia Artificial, que permitan mayor control y gestión sobre los sistemas donde se aplican.

La integración de las reglas expertas en las definiciones de los objetos GDMO, ofrece solución a estas cuestiones y permite plantear un nuevo paradigma en la gestión experta de las redes de telecomunicaciones. En este sentido, los estudios realizados se centran en la integración de la base de conocimiento, de modo que las reglas expertas se definen junto con las especificaciones GDMO del objeto gestionado. Constituyendo lo que sería una base de conocimiento distribuida entre los distintos objetos que conforman el sistema gestor [Sloman95].

Para conseguir resolver las cuestiones enumeradas, debemos establecer los mecanismos oportunos que posibiliten la inclusión de las reglas expertas de gestión, dentro de las especificaciones de los objetos gestionados, que se definen utilizando la norma GDMO, Figura 5.5.

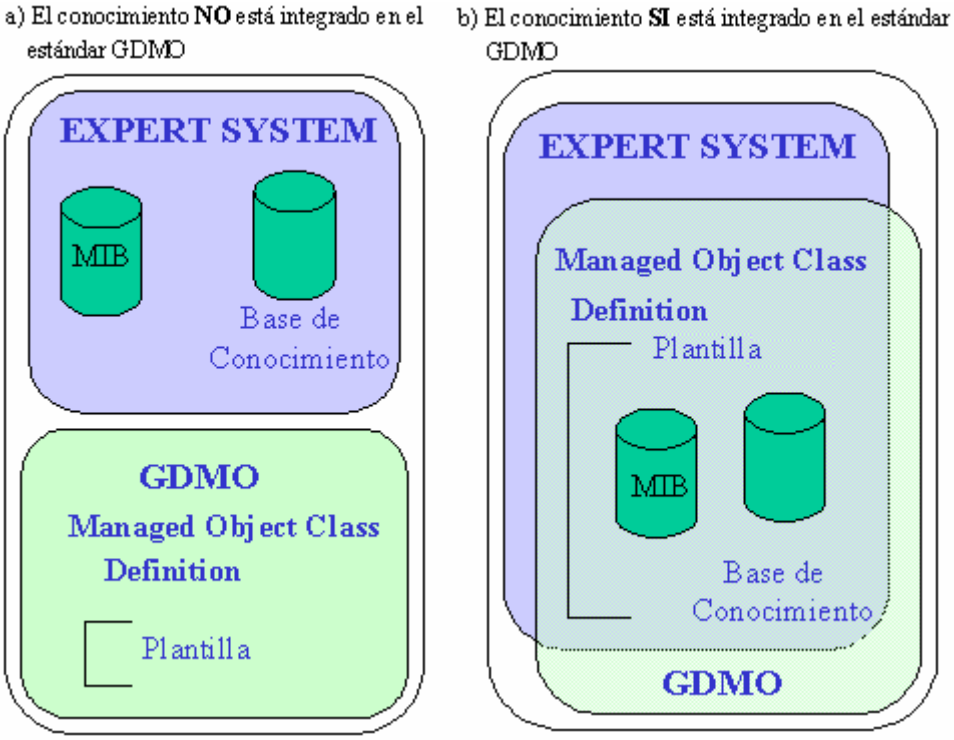

Figura 5.5 Integración de Conocimiento y Especificaciones de Objetos.

Para conseguir el objetivo representado en la figura anterior, adelantamos que es preciso efectuar algunos cambios en las plantillas del estándar GDMO. Destacamos, la incorporación de un nuevo elemento, que reunirá toda la información correspondiente a la base de conocimiento aportado por los expertos, en un dominio específico. Concretamente una nueva plantilla bautizada con el nombre *RULE* y su propiedad asociada *RULES*.

En los próximos apartados del capítulo, se contemplan todas las cuestiones técnicas referentes a la propuesta principal de la tesis. Se describen los mecanismos de integración, que van a permitir la inserción de la base de conocimiento del Sistema Experto en las especificaciones de los objetos. Esta integración proporcionaría una única base de datos, formada por las definiciones de los elementos que constituyen la red de telecomunicaciones a gestionar y las reglas expertas de gestión. Esto permite a su vez mayores posibilidades de tratamiento y administración.

5.7 Extensión de la Norma GDMO.

GDMO es un lenguaje de especificación semiformal basado en conceptos de diseño orientado a objetos. Creado para satisfacer las necesidades de gestión en distintos dominios específicos y que concuerda con la filosofía de los sistemas expertos actuales, desarrollados para un entorno variado de implantación y aplicación.

La experiencia práctica con GDMO, muestra que es un lenguaje de especificación que cuenta con grandes posibilidades, que pueden ser explotadas a la hora de crear definiciones [Black95]. Así, se plantea la incorporación de nuevos constructores en la norma para que permita incorporar nuevas características del objeto definido. Se sugiere una propiedad inédita que contendrá toda la información concerniente de la base de conocimiento del Sistema Experto. Esta nueva característica permitirá almacenar el conocimiento de gestión en la definición de los recursos que conforman el sistema a gestionar.

En la siguiente figura se pueden observar las nueve plantillas que constituyen el estándar GDMO actual. Dichas plantillas fueron debidamente estudiadas en el capitulo cuarto de la tesis.

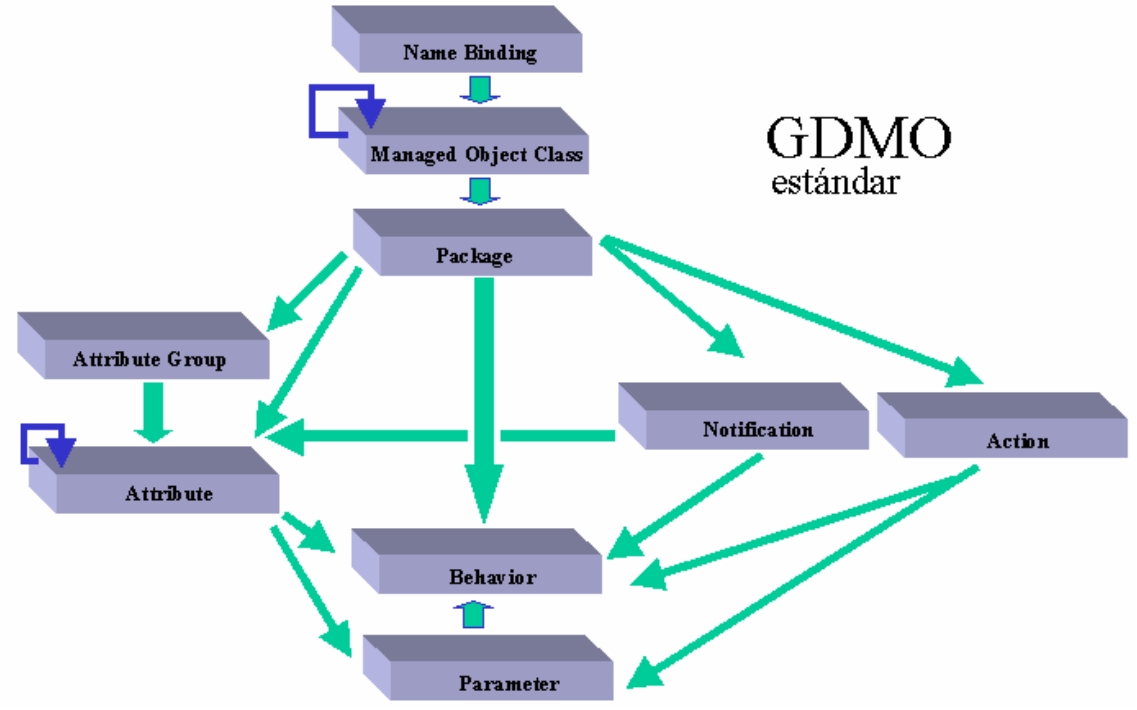

Figura 5.6 Conexiones entre plantilla de la norma GDMO.

La plantilla *Clase de Objeto Gestionado*, es de vital importancia en la norma y es fundamental para la construcción de las especificaciones de los recursos definidos. Sobre

esta plantilla, incurren todas las características y relaciones que permiten la reusabilidad de definiciones efectuadas entre los distintos tipos de objetos.

Una de sus grandes ventajas, es la posibilidad de reutilizar las especificaciones existentes. Utiliza para ello los mecanismos de enlaces de Nombres y la herencia entre clases.

Cada tipo de plantilla tiene una estructura determinada, que posibilita la definición de todas sus características asociadas. Así una plantilla de Clase de Objeto Gestionado, se compone a su vez de otros elementos, entre los que destacan los paquetes. Como ya se explicó, los *Paquetes* contienen las definiciones que se pueden de alguna manera coleccionar para formar conjunto de características asociadas a una clase: de atributos, operaciones, notificaciones, comportamiento y los parámetros.

Cada una de los elementos que componen el paquete, se definen con las plantillas correspondientes. Así para proveer la información referente a las características de un objeto, debemos emplear la plantilla *Atributo*. Esta plantilla contiene la sintaxis y las especificaciones para comprobar el valor, comportamientos, parámetros e identificador del atributo definido.

Hay ocasiones en las que es conveniente agrupar los atributos, para formar un grupo de atributos. La plantilla de *Grupo de Atributos* indica el conjunto de atributos que conforman el grupo y un valor para identificar al grupo.

La plantilla *Acción* define el comportamiento y la sintaxis de los tipos de acciones que porta la primitiva M_ACTION de CMIS.

Las *Notificaciones* enviadas por la primitiva M-EVENT-REPORT de CMIS, se definirán en la plantilla del mismo nombre.

Utilizamos unos valores numéricos en las definiciones de atributos, operaciones y notificaciones, se definen como *Parámetros*. Las especificaciones y la sintaxis de los parámetros se listan junto con los comportamientos.

El nombramiento y el manejo de los objetos, se especifican por medio de un atributo nombre que identifica al objeto superior, se utilizan para ello los *Enlaces de Nombres.*

Cuando se define un objeto gestionado, hay que especificar el comportamiento de sus atributos, operaciones sobre atributos, notificaciones, enlace de nombres y en nuestro caso las *reglas expertas* definidas. Estos detalles se explican en la plantilla de *Comportamiento.* Esta plantilla contendrá una extensión de los aspectos sobre la conducta de las distintas características de los objetos gestionados.

No nos adentramos más en el estudio de estos componentes, para un estudio más detallado puede consultarse el capítulo cuarto de esta tesis.

Se observa que ninguno de los elementos que actualmente conforman el estándar GDMO hace referencia a la base de conocimiento del sistema experto, que soporta la gestión de los recursos definidos. Debido a esto, *proponemos que la sintaxis experimente una modificación y se añade una nueva propiedad plantilla denominada* ***Rule.*** Figura 5.7.

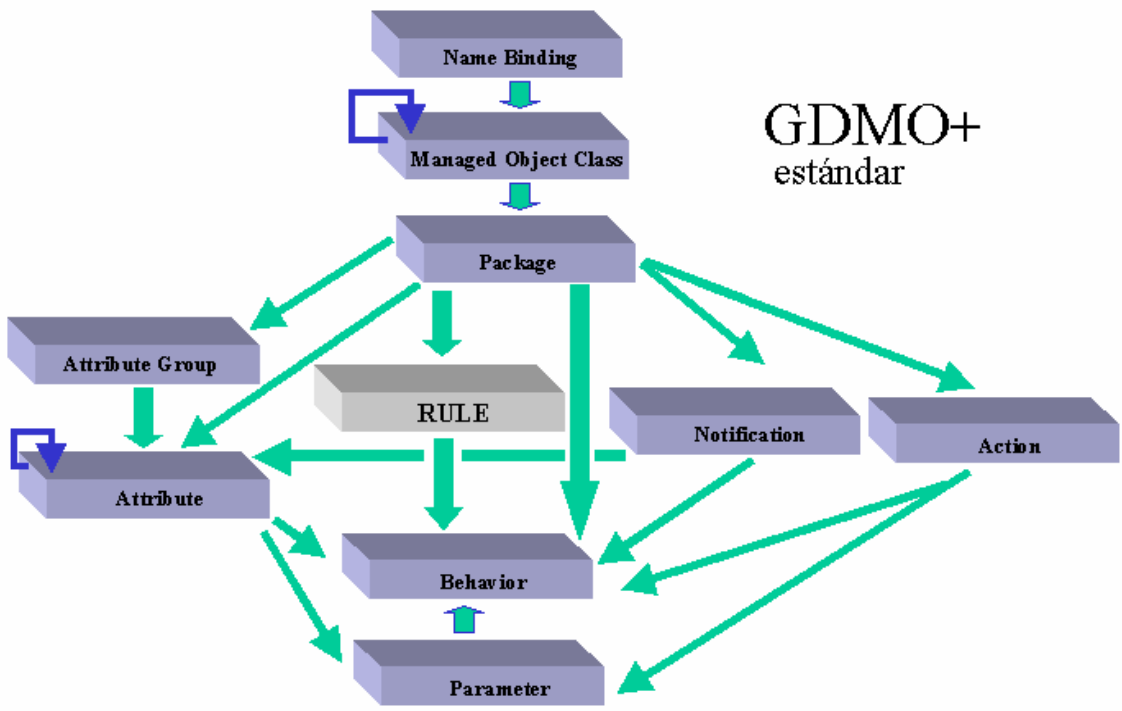

Figura 5.7 Relaciones entre las plantillas del estándar GDMO+ propuesto.

El estándar propuesto que surge como consecuencia de la incorporación de la singular plantilla *Rule* y las relaciones de esta nueva plantilla con las demás plantillas del estándar, sugerimos se denomine ***GDMO Extendido*** y lo denotaremos como ***GDMO+***.

5.7.1 Plantilla de Clase de Objeto Gestionado.

Para la inclusión del conocimiento del Sistema experto en esta nueva sintaxis, además de la nueva plantilla de regla, existen dos plantillas que jugarán un papel principal: la plantilla de *Clase de Objeto Gestionado* y la *Plantilla de Paquete*. En el estándar que planteamos ambas plantillas experimentan cambios y aunque conservan similitudes con la sintaxis de la norma original, contendrá nuevas características que permitirán la incorporación del conocimiento de gestión perteneciente al sistema experto.

La plantilla de clase de objeto gestionado, se utiliza para definir los distintos tipos de objetos existentes en un sistema. Tiene un papel primordial cuando se fijan las relaciones de herencia que permiten, la reutilización de características especificadas en otras Clases de Objetos Gestionados. Esta plantilla puede contener a su vez paquetes, paquetes condicionales, atributos, grupos de atributos, acciones, comportamientos, notificaciones y *las reglas expertas de gestión.*

La propiedad RULES es utilizada para hacer referencia a la plantilla *RULE*. Esta cláusula se incluye en la plantilla de paquete, e identificará una definición de regla experta en concreto que contiene las especificaciones propias de la regla experta de gestión definida, Figura 5.8.

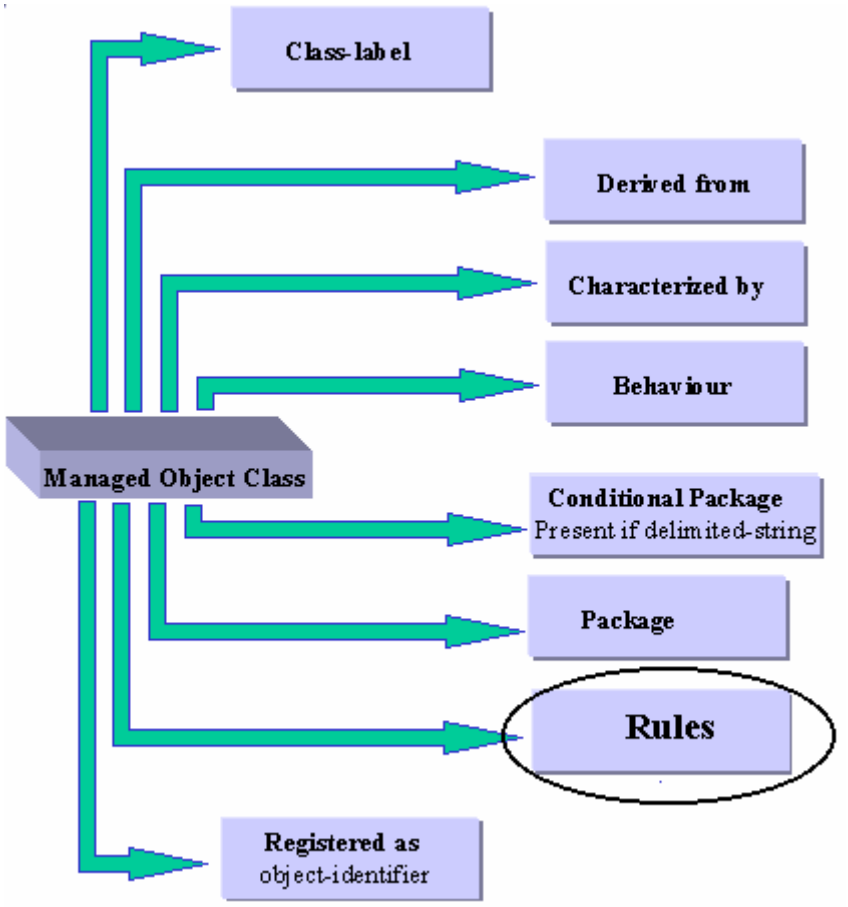

Figura 5.8 Constructores de la Plantilla de Clase de Objeto Gestionado

La definición de una Clase de Objeto gestionado, se hace de forma uniforme con la plantilla estándar, eliminando la confusión que puede resultar de personas diferentes, definiendo objetos de maneras distintas. Con esto se asegura, que las clases y las reglas expertas de gestión definidas en un lugar A, pueden ser interpretadas fácilmente en un lugar B. Además, una gran ventaja es la de poder reutilizar las clases de objetos y las reglas expertas de gestión previamente definidas.

La nueva plantilla de clase de objeto gestionado, tiene la siguiente estructura formal:

```
<etiqueta-clase> MANAGED OBJECT CLASS
   [DERIVED FROM              <etiqueta-clase>    [,<etiqueta-clase>]*];
   [BEHAVIOUR                 <etiqueta-clase>];
   [CHARACTERIZED BY          <etiqueta-paquete>  [,<etiqueta-paquete>]*];
   [CONDITIONAL PACKAGES      <etiqueta-paquete>  PRESENT IF condición;
                            [,<etiqueta-paquete>] PRESENT IF condición]*];
   [RULES  <etiqueta-regla> [,<etiqueta-regla>]*];
REGISTERED AS           identificador-objeto;
```

En la extensión del estándar GDMO que se propone, la constitución de esta plantilla es fundamental. Además del constructor RULES, posee los mismos elementos que en la norma original y sigue conservando todas las propiedades relacionadas con los lenguajes Orientados a Objetos, en los que se basa el estándar GDMO: herencia, especialización, contenido, etc. [Weiss98].

De otra forma la nueva sintaxis GDMO+ integra el conocimiento experto de gestión, dentro de las especificaciones de los recursos que se gestionan. Para lo que, según

se ha visto, se introduce una nueva propiedad denominada *RULES* y su plantilla asociada *RULE*. Esta nueva propiedad podrá ser incluida dentro de las clases de objetos gestionados o haciendo uso de los paquetes, como se verá en el próximo apartado.

5.7.2 Plantilla de Paquete.

La plantilla de paquete se sitúa jerárquicamente un nivel por debajo de la plantilla de Clases de Objeto Gestionado. Puede estar compuesta a su vez, por otros paquetes, atributos, comportamientos, *reglas*, etc.

Agrupa combinaciones de comportamientos, atributos, grupos de atributos, operaciones, notificaciones, parámetros y la inédita propiedad *RULES*. Todas estas propiedades definidas en los paquetes, serán posteriormente incluidas en la plantilla de Clase de Objetos Gestionados, donde se incorpore dicho paquete. Un mismo paquete, puede ser referenciado por más de una de Clase de Objeto Gestionado, lo que implica que una misma regla experta de gestión puede ser incluida en más de una clase de objeto gestionado.

Los paquetes siguen conservando los mismos aspectos de usabilidad e inclusión, que en la norma GDMO originaria, con la única salvedad de la incorporación de la nueva propiedad *Rules*. La descripción formal de la nueva plantilla de paquete en el estándar GDMO+, es la siguiente:

```
<etiqueta-paquete> PACKAGE
[BEHAVIOUR  <etiqueta-definición-comportamiento>
          [,<etiqueta-definición-comportamiento>]*;]
 [ATTRIBUTES <etiqueta-atributo> propertylist [<etiqueta-parámetro>]*
           [,<etiqueta-atributo> propertylist[<etiqueta-parámetro>]*]*;]
 [ATTRIBUTE GROUPS <etiqueta-grupo> [<etiqueta-atributo>]*
                [,<etiqueta-grupo> [<etiqueta-atributo>]*]* ;]
 [ACTIONS          <etiqueta-acción> [<etiqueta-parámetro>]*
                [,<etiqueta-acción> [<etiqueta-parámetro>]*]*;]
 [NOTIFICATIONS    <etiqueta-notificación> [<etiqueta-parámetro>]*
                [,<etiqueta-notificación> [<etiqueta-parámetro>]*]*;]
 [RULES <etiqueta-regla> [,<etiqueta-regla> ]*;]
[REGISTERED AS    identificador-objeto];
```

Donde *property list* puede ser:

REPLACE-WITH-DEFAULT
DEFAULT VALUE valor-especificado
INITIAL VALUE valor-especificado
PERMITTED VALUES tipo-requerido
REQUIRED-VALUES tipo-requerido
GET | REPLACE |GET-REPLACE
ADD |REMOVE |ADD-REMOVE.

Y el valor-especificado: referencia-valor|DERIVATION RULE <etiqueta-definición-comportamiento>

La definición contiene la nueva cláusula RULES, podemos observarlo de forma gráfica en la siguiente figura 5.9. Como el resto de propiedades definidas en un paquete, *RULES* posee una plantilla asociada que permitirá definir cada una de las reglas contenidas en el paquete. Además, dentro de un mismo paquete, puede existir una o varias reglas.

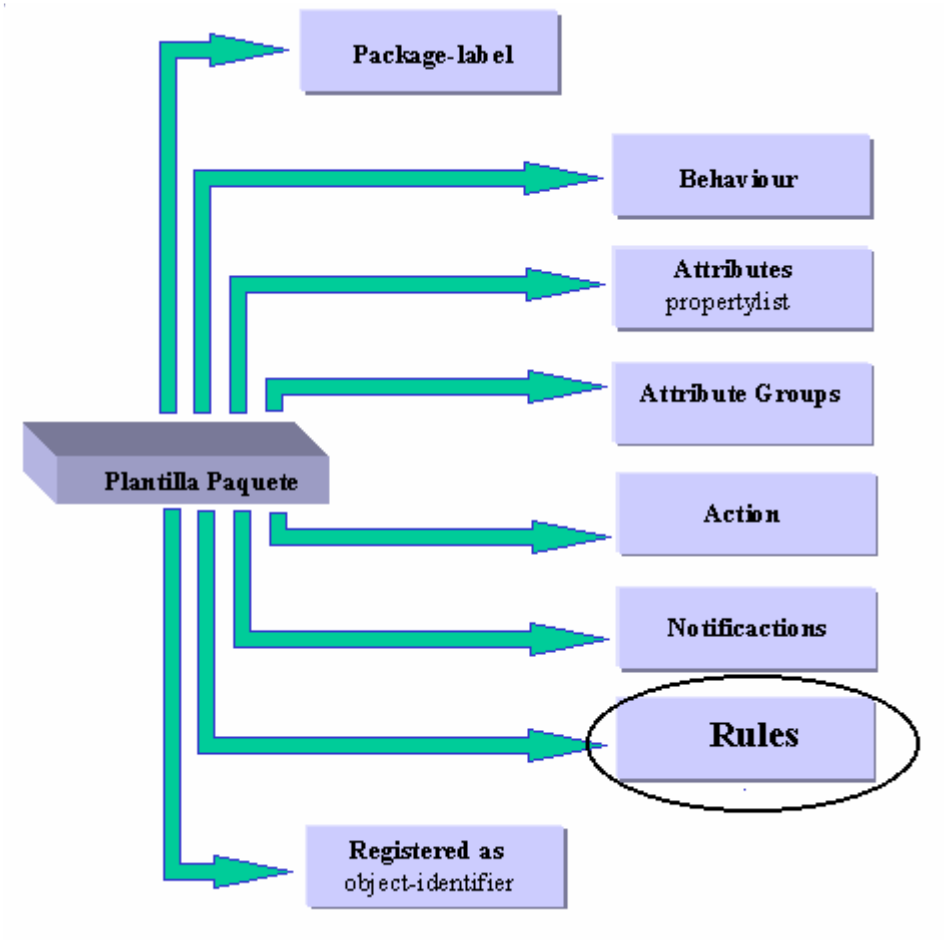

Figura 5.9 Nueva plantilla de Paquete en GDMO+.

Cada regla experta de gestión puede tener asociados uno o varios:

- *Parámetros:* valores que se pueden asociar a operaciones y notificaciones, permitiendo definir fallos de procesamiento, peticiones y notificaciones o acciones.

- *Alarmas*: Notificaciones particulares a cada objeto y que son emitidas por estos cuando ocurre un evento interno o externo. Para que la notificación sea útil debe contener una cierta información [Malheiros98].

Todo evento definido en GDMO+ mediante la plantilla NOTIFICATION, posee tres parámetros implícitos que pueden ser referenciados en los antecedentes de las reglas. Sus nombres se consideran palabras reservadas de la extensión de la norma que aquí estamos tratando. Son los siguientes:

- SOURCE-OBJECT: nombre distinguido del objeto que emite el evento.
- DATE: fecha en la que se ha emitido el evento.
- TIME: instante de tiempo en el que se ha emitido el evento.

- *Acciones:* Un conjunto de operaciones de gestión, que afectan al Objeto Gestionado. Una acción puede ser aplicada a más de una Clase, aunque también pueden ser definidas para que sean específicas a un Objeto determinado.

Estos elementos son importantes, para el correcto funcionamiento del motor de inferencia del sistema experto. Serán útiles para efectuar la activación y el disparo de una regla experta de gestión determinada. Cada uno de los parámetros, notificaciones y acciones utilizadas, deben ser definidos haciendo uso de la plantilla compañera de la norma GDMO Extendida.

A continuación se explica en mayor detalle la plantilla RULE, que recogerá todos los aspectos relacionados con la base de conocimiento de gestión.

5.8 La Plantilla de Regla.

Esta nueva plantilla es la principal novedad y pieza clave de nuestra propuesta para obtener una extensión del estándar GDMO. Dota a las clases de objetos gestionados, de las propiedades correspondientes para proveer a los objetos gestionados del conocimiento específico a un dominio concreto de gestión.

En la siguiente figura se aprecian las partes que conforman la plantilla RULE:

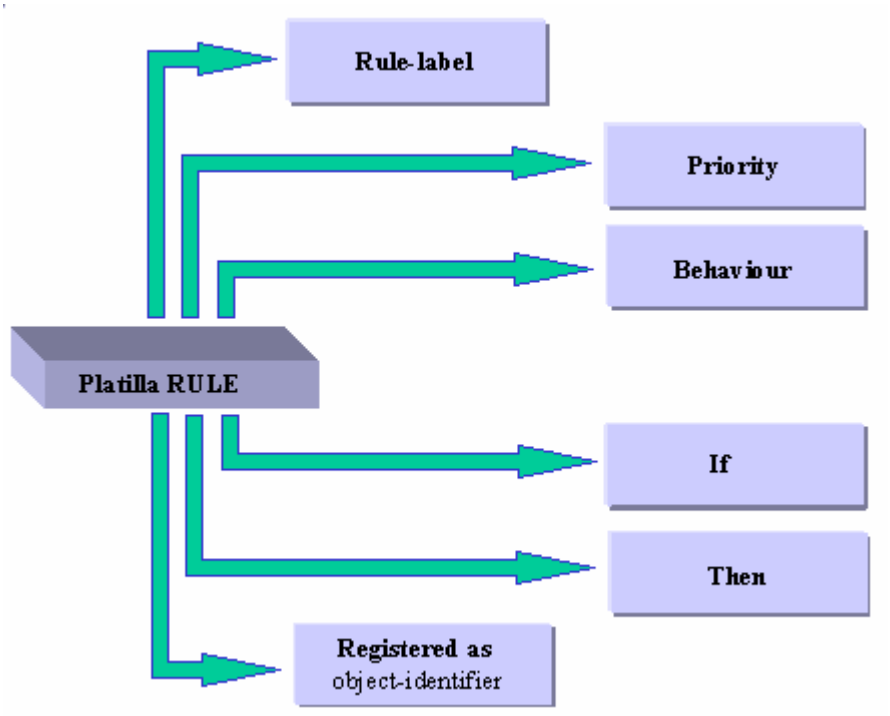

Figura 5.10 La Plantilla RULE en el estándar GDMO+.

La estructura formal de la plantilla *RULE* es la siguiente:

```
<rule-label>  RULE
  [BEHAVIOUR   <behaviour-definition-label>
                [,<behaviour-definition-label>]*;]
   [PRIORITY     <priority> ;]
   [IF           ocured-event-pattern [,ocured-event-pattern]*]
   [THEN         sentence [, sentence]*] ;
REGISTERED AS  objetc-identifier;
```

Veamos las distintas partes que conforman la característica plantilla de regla:

● El primer elemento que se observa en la definición de la plantilla es el encabezado. Este constructor está compuesto a por dos apartados:

- <Rule-Label>: nombre asignado a la regla experta de gestión. La definición de la regla debe tener un nombre único que la caracterice.

- *RULE*: palabra clave que denota el tipo de plantilla. En nuestro caso, plantilla la definición de especificaciones de las reglas expertas de gestión.

Hay que ser cautos a la hora de crear nuevas reglas en las clases. Cuando se introduce una nueva regla en una clase, se le asigna un nombre. Si este nombre ha sido previamente asignado a otra regla experta definida en la misma Clase de Objeto Gestionado o alguna de sus superclases, la nueva regla de gestión reemplaza a la existente. Es decir cuando existen dos definiciones de una misma regla, prevalece la realizada en último lugar.

Después del encabezado, aparecen los elementos que componen la definición de la regla. Algunos de estos elementos son comunes a las otras plantillas de la norma:

● *BEHAVIOUR:* Define la conducta de cada una de las reglas expertas. Cada regla puede tener asociado más de una definición de comportamiento y cada uno de éstos, especificados por medio de la plantilla *behaviour* correspondiente.

También existen elementos que son específicos de la propia plantilla RULE:

● *PRIORITY:* cada regla puede tener asociada una prioridad o ninguna.

● *IF:* las condiciones impuestas para que las acciones se ejecuten. Pueden existir cero o más elementos condicionales.

● *THEN:* A continuación las conclusiones emitidas por el motor de inferencia del sistema experto y la lista de operaciones de gestión que se ejecutará cuando la regla se dispare. No es obligatorio que una regla tenga acciones asociadas.

● *REGISTERED* AS *identificador de objeto*: Una única identidad que se usa en el protocolo, para identificar esta regla experta de gestión. El identificador de regla además es obligatorio.

En los siguientes apartados se estudia detenidamente cada uno de los elementos, las cláusulas y los constructores que conforman la nueva plantilla RULE del estándar GDMO Extendido.

5.8.1 BEHAVIOUR. Descripción del Comportamiento.

Las etiquetas <*behaviour-definition-label*> como en el resto de las plantillas de la norma, identifican el comportamiento asociado a la plantilla regla. Cada descripción de comportamiento se formaliza utilizando la plantilla BEHAVIOUR.

Esta plantilla no sufre modificaciones respecto a la existente en el estándar GDMO y su estructura se corresponde con la siguiente:

```
[BEHAVIOUR <behaviour-definition-label>
        [, <behaviour-definition-label>]*;]
```

En el caso de que aparezca esta propiedad en la descripción de una regla, deberá acompañarla al menos una etiqueta que identifique la definición o definiciones de plantillas de comportamiento relacionadas, en el caso de existir más de una.

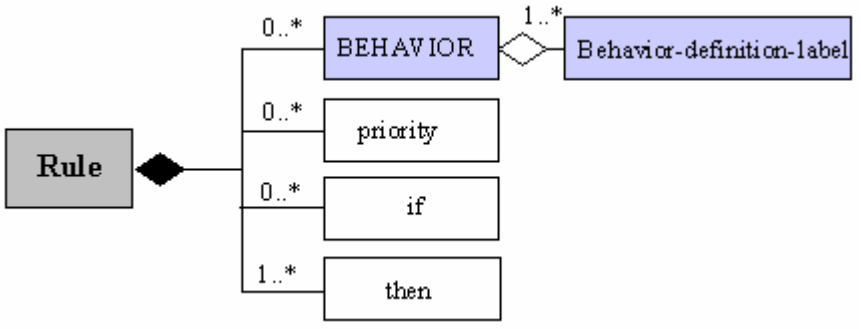

Figura 5.11 El comportamiento en la plantilla de Regla

A continuación se presentan dos ejemplos de definiciones de comportamiento asociados a dos clases de objetos gestionados, las denominadas *damageFrecuentBehaviour* y *errorTransmitionBehaviour*.

```
damageFrecuentBehaviour BEHAVIOUR
       DEFINED AS
              "Esta regla permite detectar una situación de alarma en el estado
              de la alimentación eléctrica del dispositivo."

errorTransmitionBehaviour BEHAVIOUR
       DEFINED AS
              "Esta regla permite detectar un fallo en el módulo de transmisión
              del Transceptor de Radio de una estación determinada."
```

Como se aprecia en las descripciones anteriores, la plantilla BEHAVIOUR obedece al objetivo de especificación del comportamiento de las reglas expertas de gestión. Por tanto, no es necesario introducir cambio alguno y sigue siendo análoga a la que originariamente posee el estándar GDMO.

5.8.2 PRIORITY. Prioridad de Disparo de la Regla Experta de Gestión.

Cuando las precondiciones de una determinada regla se satisfacen, casan o encajan con algunos hechos de la memoria de trabajo, la regla se activa y pasa a estar contenida en la agenda[3] del sistema.

En el caso de que haya varias reglas en la agenda (reglas activas), se puede presentar el problema de no saber cual de ellas es la que debe ser disparada. Por esto hay que seguir una ***estrategia de resolución de conflictos*****,** un conjunto de criterios que permitan determinar el orden de disparo de las reglas que están activas en un momento determinado (Anexo D).

La ***prioridad*** es un valor numérico asociado a una regla. Indica la importancia de la regla y por tanto la prioridad con que debe ejecutarse. La asignación se realiza poniendo la siguiente expresión entre el nombre de la regla y el primer elemento condicional de esta:

```
[PRIORITY <priority>;]
```

[3] Se conoce por agenda a la parte del sistema que contiene una lista de las reglas activadas.

Esta cláusula es un entero positivo o negativo entre unos valores que pueden venir fijados por el lenguaje de programación utilizado o por el programador. Especifica la prioridad de la regla, de tal forma que cuanto más positivo sea, mayor es la prioridad de la regla. En la mayoría de los lenguajes de programación, la prioridad por defecto es 0.

En la siguiente figura se muestra la representación UML de la plantilla de RULE. La prioridad no es un elemento obligatorio, puede aparecer o no. En el caso que la prioridad no aparezca se le asigna un valor por defecto. Por tanto todas las reglas tendrán una prioridad asociada. El valor de la prioridad por defecto puede ser fijado por el administrador del sistema y se realiza siguiendo las indicaciones propias del entorno de programación utilizado.

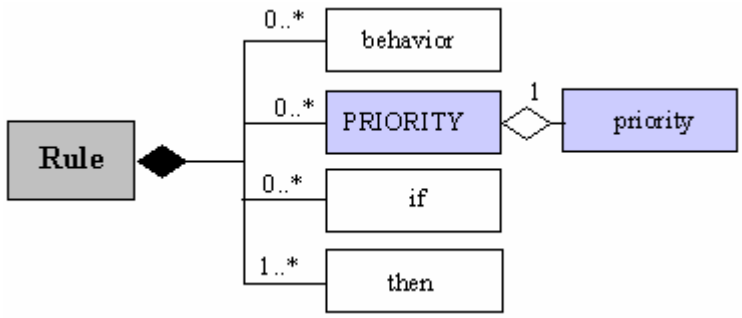

Figura 5.12 La Prioridad en la plantilla de Regla

La nueva característica *RULE*, recoge todas las especificaciones de las reglas expertas de gestión que actúan sobres las instancias[4] de una clase de objeto gestionado. Por tanto, todas las instancias de una misma regla de gestión, que se incluyen en una clase de objeto gestionada tendrán la misma prioridad.

Supongamos una clase de objeto gestionado, donde se define un recurso denominado *"puerto"* que pertenece a un determinado equipo de comunicaciones. Dentro de esta clase, se define una regla experta que efectúa operaciones de gestión sobre dichos objetos:

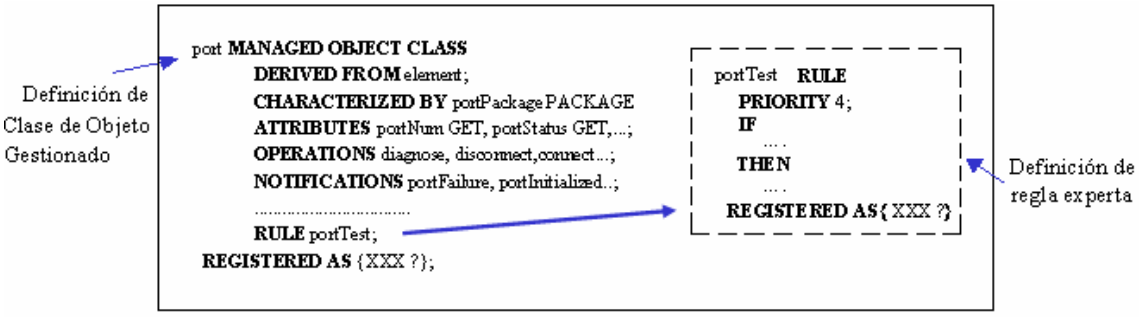

Figura 5.13 Definición de regla experta de gestión

La prioridad "**4**" asignada a la regla experta de gestión *portTest* de la clase de objeto gestionado *port,* es aplicable a la totalidad de las instancias de dicha regla.

[4] Es un caso particular de un objeto definido mediante la plantilla de clase de objeto gestionado, en el estándar GDMO.

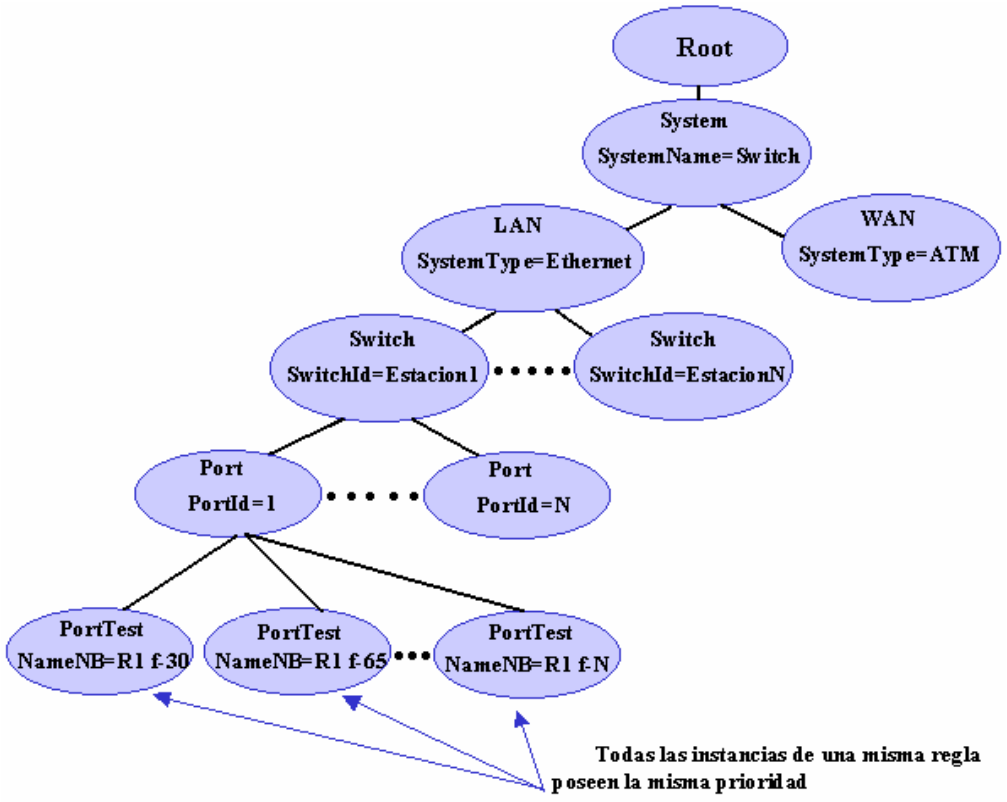

Figura 5.14 Instancias de una Regla y Prioridades de Ejecución.

Todas las entidades de la clase que contiene la regla experta de gestión *portTest*, tienen la misma prioridad fijada en la definición de la regla.

En cada momento se selecciona la mejor regla atendiendo a la estrategia de resolución de conflictos y a las prioridades de las reglas. En el siguiente ejemplo se observan las reglas *damageFrecuent* y *errorTransmission*. Suponiendo que ambas se encuentran en la agenda, lo cual implica que están en disposición de disparo (ejecución). La regla etiquetada como *errorTransmission* se ejecutaría antes que la regla *damageFrecuent*, ya que su prioridad es mayor.

```
damageFrecuent RULE
  BEHAVIOUR damageFrecuentBehaviour;
  PRIORITY 3;
  …
Registered {..};

errorTransmission  RULE
  BEHAVIOUR errorTransmissionBehaviour
  PRIORITY 4;
  …
Registered {...};
```

5.8.3 IF. Eventos Necesarios para la Activación de las Reglas.

Sabemos que una regla es una expresión del tipo

"Si el *antecedente* es cierto para los hechos almacenados en la lista de hechos, entonces pueden realizarse las acciones especificadas en el consecuente".

Para todo conjunto de acciones asociadas a una regla, existen unos requisitos previos a la ejecución que deben ser cumplidos. En este apartado se estudia la manera en que una determinada regla, indica las restricciones que deben darse para que la acción o acciones asociadas tengan lugar.

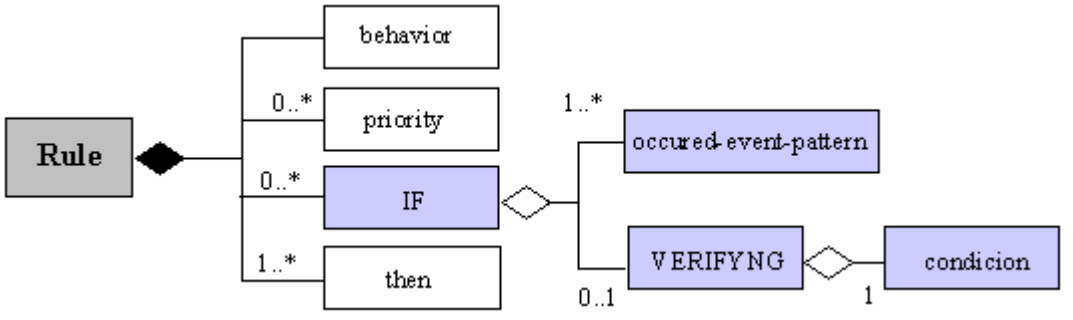

Figura 5.15 La cláusula IF en la plantilla de Regla

Para el establecimiento de las condiciones de disparo, hay que seguir el siguiente formato:

```
[IF <patrón> [, <patrón>]*
    <condición> [,condición]*;]
```

El antecedente de una regla consta de cero o más elementos condicionales. Procedamos al estudio de cada uno de estos.

5.8.3.1 Las Variables.

Dentro de la plantilla RULE, se permite utilizar variables globales. Conceptualmente, son similares a las variables globales que pueden encontrarse en lenguajes procedurales como Java, C, etc. Es decir, son variables cuyo valor es independiente de los constructores y pueden accederse desde cualquiera de los elementos de una regla experta de gestión.

En el estándar GDMO+ propuesto, existen las *Variables Locales*. Son aquellas que se crean en el ambiente de una regla experta de gestión. Normalmente se definen en el antecedente de una regla y suelen utilizarse en el proceso de reconocimiento de patrones para capturar algún valor concreto del campo de un hecho.

Las variables tienen la sintaxis siguiente:

```
?<símbolo-de-la variable>
```

El valor de un campo puede capturarse mediante una variable. En el caso de que aparezca la misma variable en más de un elemento condicional, la primera ocurrencia de la variable lo que hace es capturar el valor, en las restantes ocurrencias se sustituye el valor capturado por la variable.

El siguiente ejemplo muestra el uso de las variables:

```
damageTerminal RULE
 BEHAVIOUR damageTerminalBehaviour;
 PRIORITY 3;
 IF
   (?fecha ?hora ?estacion-1 ?alarma ?estacion-2 ALARMA)
THEN
  ("Datos de la Alarma: " ?fecha" "?hora),
  ("Tipo: " ?alarma),
  ( "Fallo general entre sistemas" ?estacion-1 "y" ?estacion-2),
        Ejecución de Acciones ...
REGISTER AS {rule 1 3 6 2};
```

Tal y como se observa, la regla experta de gestión se activa cuando se produce una alarma en el sistema gestionado. En el momento del disparo de la regla, las variables *?fecha, ?hora, ?estacion-1, ?alarma* y *?estacion-2,* se encargan de almacenar la información correspondiente. Estos datos pueden ser utilizados más tarde en el consecuente de dicha regla, haciendo uso para ello de las variables declaradas.

5.8.3.1.1 Comodín.

Constituye cualquier valor que se encuentre almacenado en una casilla, viene representado por el símbolo "**?".** Se utiliza para registrar cualquier valor y exactamente uno, que se encuentre almacenado en un hecho.

A continuación se muestra un ejemplo muy simple del comodín "?". En esta ocasión la regla experta de gestión se activa cada vez que se produzca una alarma, independiente del tipo de alarma producida.

```
damageTerminal RULE
 BEHAVIOUR damageTerminalBehaviour;
 PRIORITY 3;
 IF
       (? ? ? ? ALARMA)
 THEN
       Ejecución de Acciones ...
REGISTER AS {rule 1 3 6 2};
```

En este caso la regla experta se activa cada vez que se produce una alarma y no ofrece información alguna sobre ésta. La información aportada por la alarma, no se almacena y por tanto no podrán ser recuperados.

Existe una diferencia sustancial entre el uso del comodín "?" y una variable ?x. Con un comodín se hace la sustitución de un valor, por el contrario cuando se utiliza una variable, dicho valor se asocia a una zona de memoria donde se almacena y posibilita con ello su uso posterior.

5.8.3.2 Los Patrones.

El elemento condicional más simple es el patrón `<pattern>`, consta de una o varias restricciones sobre el valor de algunos atributos de un hecho. Las posibles restricciones son:

```
pattern ->
       [notification [,notification]*]
       [attribute [,attribute]*]
notification -> <notification-label> | ?
attribute -> attribute-nane [attribute-value]
attribute-name  -> SOURCE-OBJECT | DATE | TIME | <attribute-label>
attribute-value -> <literal> | <attribute-value-label> | ?
```

Se aprecia que un elemento patrón puede estar compuesto a su vez por otras partes:

5.8.3.2.1 Notificación.

Elemento condicional formado por un evento. Incluye la etiqueta de la notificación <notification-label>, previamente definida por medio de la plantilla de la norma GDMO+. Puede incluir el símbolo "?" estudiado, que constituye el elemento comodín unicampo.

En general, los parámetros de las alarmas que no sean de interés para la activación de una regla, no tienen por qué ser enumerados explícitamente. La existencia de estos atributos se realiza por medio del carácter comodín "?".

5.8.3.2.2 Atributos.

Un patrón puede ser un atributo, que incluye el nombre del atributo seguido del valor asociado. Un atributo también podrá ser un tipo de parámetro implícito en el estándar GDMO+:

- SOURCE-OBJECT: nombre distinguido del objeto que emite el evento.

- DATE: fecha en la que se ha emitido el evento.

- TIME: instante de tiempo en el que se ha emitido el evento.

Para la definición de los atributos se utiliza la plantilla ATTRIBUTE del estándar GDMO+. Podemos tener referencia a éstos por medio de la etiqueta de atributo, `<attribute-label>`.

5.8.3.2.3 Literales.

La restricción más básica que puede utilizarse como elemento condicional, es aquella en la que se exige un valor exacto para una casilla de un hecho. Estas restricciones reciben el nombre de restricciones literales y se representa de la forma siguiente:

```
<literal-value>.
```

Los valores literales se expresan sintácticamente con la notación ASN.1 correspondiente al tipo de parámetro en cuestión.

```
errorSistema RULE
 BEHAVIOUR errorSistemaBehaviour;
 PRIORITY 1;
 IF
   (? ? ? "ERROR_FATAL" ? ALARMA)
THEN
  ("Se produjo fallo general en el sistema"),
  ("Acciones: REINICIAR EL SISTEMA ASOCIADO");
REGISTER AS {rule 1 3 6 2};
```

En el ejemplo tenemos el literal "ERROR_FATAL", que se incluye dentro de un elemento condicional en el antecedente de la regla experta de gestión.

5.8.3.3 Condiciones Lógicas Asociadas.

Los elementos condicionales anteriores se basaban principalmente en determinar hechos que casaran con un cierto patrón simple. Dentro del constructor IF existe un

elemento condicional que permite evaluar expresiones en el antecedente de la regla experta de gestión.

Con este nuevo elemento, se faculta al constructor IF para evaluar expresiones que pueden llegar a ser todo lo complejas que estimemos, permitiendo tolerar una amplia gama de requisitos previos a la ejecución de las acciones relacionadas a la regla experta. Una expresión puede evaluar a verdadero o falso.

Su sintaxis es la siguiente:

```
condition ->
        | expression operator expression
        | NOT expression
        | expression
        | TRUE
        | FALSE
```

5.8.3.4 Expresiones.

Las expresiones condicionales contienen operadores, notificaciones y atributos que pueden ser valores numéricos o literales. Para poderlos utilizar deben ser definidos con las plantillas de NOTIFICATION y ATTRIBUTE del estándar GDMO+.

En una expresión condicional, se pueden usar paréntesis para indicar el orden de evaluación de las distintas partes que la conforman. Las expresiones condicionales pueden contener distintos operadores: aritméticos, relacionales y lógicos.

- *Operadores Aritméticos:* Según el orden de preferencia son:

```
operator -> + | - | * | /
```

- *Operadores Lógicos:* En orden de preferencia son los siguientes:

```
logical-operator -> NOT| AND | OR
```

- *Operadores Relacionales:* Por orden de preferencia son los siguientes:

```
operator ->   | IS-SUBSTRING-0F
              | IS-SUBSET-OP
              | IS-SUPERSET-OF
              | NULL-INTERSECTS-WITH
extended-relational-operator ->
        = | >= | <= | IS-MEMBER-OF
```

La evaluación de una expresión puede ser no-FALSE o FALSE.

a) ***Si es no-FALSE***, la regla se activa si además se satisface el resto de los elementos condicionales de la regla. Todos los elementos condicionales deben evaluar a no-FALSE.

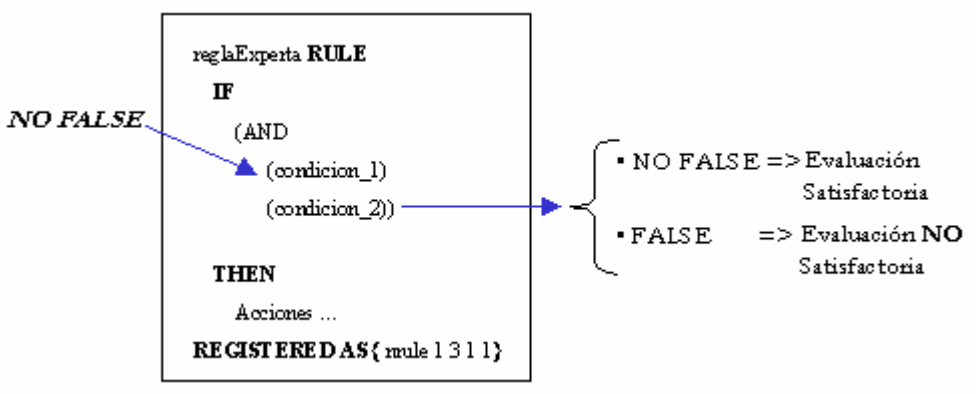

Figura 5.16 Evaluación IF con expresiones NO-FALSE.

En el ejemplo anterior se aprecia una primera expresión *"condicion_1"* que evalúa a no-FALSE. La activación de la regla experta de gestión depende directamente de la evaluación de la segunda expresión *"condicion_2"*. Para que lleguen a realizarse las acciones contenidas en THEN, deben evaluar ambas expresiones a no-FALSE, ya que se relacionan mediante un operador "**y**".

b) ***Si es FALSE***, la regla nunca se activaría, salvo que existan elementos condicionales **"conectivos"** (véase apartado siguiente), que hagan que el antecedente se satisfaga.

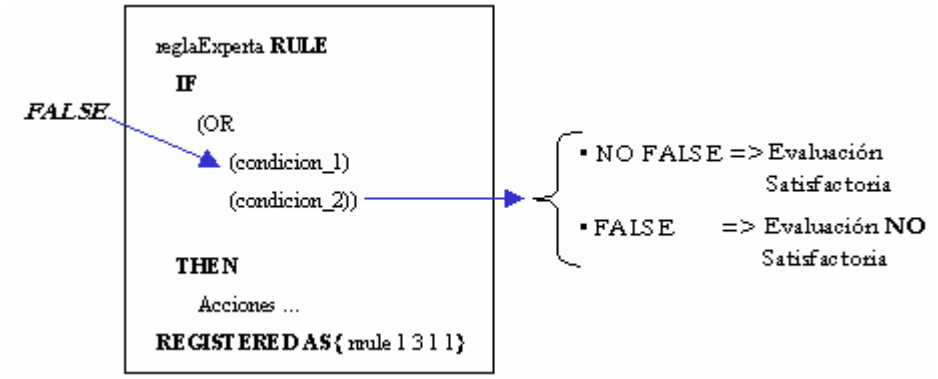

Figura 5.17 Evaluación IF con expresiones FALSE.

En este caso, la "*condicion_1*" evalúa a FALSE, sin embargo se relaciona con una segunda condición mediante el operador "**o"**, lo cual posibilita que la regla experta se activa en el caso de que la "*condicion_2*" evalúe a NO-FALSE.

Se observa que la función de los operadores conectivos es crucial para la correcta evaluación del conjunto de expresiones contenidas en el elemento condicional IF. A continuación procedemos a un estudio más exhaustivo de éstos.

5.8.3.5 Conectivas entre Elementos Condicionales.

Las restricciones conectivas establecen condiciones del tipo "**no"**, "**y"** y "**o"** en los valores de una variable. Esta idea puede extenderse dando lugar a la conectividad entre los distintos elementos condicionales. Por defecto, todos los elementos condicionales de una regla se encuentran enlazados por la conectiva "**y"**.

Sea la regla siguiente:

```
damageTerminal RULE
 BEHAVIOUR damageTerminalBehaviour;
 PRIORITY 3;
 IF
   ((?fecha1 ?hora1 ?equipo CTR190/7_TX_C1 ?puerto ALARMA)
    (?fecha2 ?hora2 CTR190/7_PRX1 ?puerto ALARMA))
THEN
   ("Se produjo una avería asociada: FALLO EN EL OSCILADOR LOCAL DE
    ESTACIÓN "?equipo y "FALLO EN MÓDULO DE TRASMINSIÓN EN EL PUERTO
    "?puerto …);
REGISTER AS {rule 1 3 6 2};
```

```
Ante los hechos siguientes:

      ...
      (04/12 1039.2882 HUB-1 CTR90/7_TX_C1 SALIDA-8)          n
      (04/12 1055.2705 HUB-1 CTR90/7_PRX1 SALIDA-8)           n+1
      ...
```

Se establece implícitamente la conectiva "***y***" entre los elementos condicionales `(HUB-1 CTR190/7_TX_C1 SALIDA-8) y (HUB-1 CTR190/7_PRX1 SALIDA-8).`

Las conectivas entre elementos condicionales reciben el nombre de operadores condicionales. Los más relevantes son:

- ***Elemento condicional "o"***. Presenta la sintaxis:

```
(OR <elementos-condicionales>+)
```

Siguiendo con el ejemplo anterior, tendríamos:

```
damageTerminal RULE
 BEHAVIOUR damageTerminalBehaviour;
 PRIORITY 3;
 IF
   (OR(?fecha ?hora1?equipo CTR190/7_TX_C1 ?puerto ALARMA)
   (?fecha ?hora2 ?equipo CTR190/7_PRX1 ?puerto ALARMA))
THEN
   ("Se produjo una avería asociada a FALLO EN EL OSCILADOR
       LOCAL DE ESTACIÓN "?equipo O "FALLO EN MÓDULO DE TRASMINSIÓN EN
       EL PUERTO "?puerto ...);
REGISTER AS {rule 1 3 6 2};
```

Para que la regla experta de gestión se active, debe de verificarse al menos una de las dos condiciones.

- ***Elemento condicional "y"***. Para que la regla experta de gestión se active, se deben cumplir todas las condiciones presentes. Sigue la siguiente sintaxis:

```
(and <elementos-condicionales>+)
```

Ejemplo:

```
DamageTerminal RULE
 BEHAVIOUR damageTerminalBehaviour;
 PRIORITY 3;
 IF
   (AND(?fecha ?hora1 ?equipo CTR190/7_TX_C1 ?puerto ALARMA)
   (?fecha ?hora2 ?equipo CTR190/7_PRX1 ?puerto ALARMA))
THEN
    ("Se produjo una avería asociada a FALLO EN EL OSCILADOR
          LOCAL DE ESTACIÓN "?equipo Y "FALLO EN MÓDULO DE TRASMINSIÓN EN
          EL PUERTO "?puerto ...);
REGISTER AS {rule 1 3 6 2};
```

En el caso anterior debe producirse las alarmas `CTR190/7_TX_C1` y `CTR190/7_PRX1`.

• ***Elemento condicional "no***". Presenta la sintaxis:

```
(not <elemento-condicional>)
```

Para que la regla se active, no debe producirse ninguno de los elementos condicionales a los que afecte el operador NOT. Sea el ejemplo siguiente:

```
nivelrecepcion RULE
 BEHAVIOUR nivelRecepcionBehaviour;
 PRIORITY 3;
 IF
   (NOT(?fecha ?h1 ?local RT21_PTX2 ¿destino ALARMA))
 THEN
    ("No hay BAJADA DE NIVEL EN SEÑAL DE RECEPCIÓN " ?local);
REGISTER AS {rule 1 3 6 5};
```

En un instante dado, si no se ha producido la alarma `RT21_PTX2`, la regla *`nivelRecepcion`* pasa a formar parte de la agenda del sistema experto de gestión.

5.8.3.6 Ejemplo.

A continuación podemos percibir como operan los elementos estudiados, a través de un paradigma de regla experta de gestión:

```
filtroAlarmas RULE
 BEHAVIOUR filtroAlarmasBehaviour;
 PRIORITY 2;
 IF
   (?fecha ?h1 ?remota ?alarma ALARMA)
   (?fecha ?h1 ?remota ?alarma DESAPARECE-ALARMA)
       & : (<(ABS(- ?h1 ?h2)) 1.00))
 THEN
       Ejecución de Acciones...
 REGISTER AS {rule 1 3 6 2};
```

El caso anterior muestra un ejemplo de empleo de variables que aparecen en más de una ocasión, en distintos elementos condicionales. Se trata de una regla experta de gestión responsable de filtrar eventos pocos relevantes, producidos en un sistema de gestión. Elimina aquellas alarmas que surgen y desaparecen de forma casi instantánea (en un intervalo de tiempo inferior a 1 sg.), y que no requieren de tratamiento alguno.

La primera ocurrencia de las variables ?fecha, ?remota y ?alarma, se encarga capturan los valores proporcionados por el hecho. En las restantes ocurrencias de dichas variables, lo que se hace es sustituir los valores que contienen.

El sistema de gestión en su funcionamiento normal, genera distintas notificaciones, acompañadas de parámetros que determinan distintos aspectos del mensaje emitido.

```
Alarma ocasionadas cuando se provoca una incidencia:

(31/01 1100.2000 ESTACION-1 CTR190/7_RX ESTACION-2 PR_LIMITE_INF)          1
(04/12 1034.1688 MULTIPLEX_C MP31_EXT_FONIA MAD ALARMA)                    2
(04/12 1034.1692 MULTIPLEX_C MP31_EXT_FONIA MAD DESAPARECE_ALARMA)         3
(12/02 1034.1220 ESTACION-4 CCA-34_C_C1/C2 ESTACION-7 CANAL_2_LOCAL)       4
(04/12 1034.1355 TRASNCEPTOR_1 SPU1_BER_1 BER ALARMA)                      5
(04/12 1034.1464 TRASNCEPTOR_1 SPU1_BER_1 BER DESAPARECE_ALARMA)           6
...
```

Los hechos 2 y 3 casan con los elementos condicionantes de la regla *filtroAlarmas*, que se activa y se dispone a la ejecución de las acciones coligadas. Lo mismo ocurre con los hechos 5 y 6. En ambos casos el tiempo transcurrido entre la aparición y la desaparición de la alarma es inferior a 1 segundo. Este tipo de reglas como se ha descrito anteriormente, resultan muy útiles a la hora de detectar alarmas que aparecen y desaparecen de forma instantánea y puede servir para la eliminación de los hechos relacionados o la toma de acciones preventivas oportunas.

Cada alarma contiene información correspondiente a la ocurrencia o circunstancias que provocan la incidencia. De esta manera, tomando como ejemplo el primer hecho[5] de la relación anterior, se obtienen los datos siguientes:

- La fecha en que se produce: 31/01.
- La hora: 1100.2000.
- Tipo de alarma: CTR190/7_RX1, "Fallo en recepción".
- Equipos Implicados en la incidencia:
    - Origen: ESTACION-1.
    - Destino: ESTACION-2.

de manera análoga se adquiere la información en los hechos restantes.

5.8.4 THEN. Ejecución de Acciones.

La sección THEN permite enumerar una serie de operaciones, instrucciones, desarrollos, etc., que se realizan en caso de disparo de la regla. Es decir, las *acciones* son las respuestas de gestión que se emprenderán cuando los eventos (notificaciones) emitidos por los recursos del sistema gestionado, verifican los requerimientos impuestos por la regla de gestión asociada.

Estas acciones conforman las operaciones y diagnósticos que la plataforma de gestión integrada, ofrece como respuesta a las alarmas que se producen en el sistema de telecomunicaciones gestionado.

En la siguiente Figura 5.18, la representación UML muestra la composición de la cláusula THEN de la plantilla RULE en el estándar GDMO+.

[5] Representa un trozo de información que se almacena en la denominada lista de hechos. Cada hecho tiene asignando un identificador (*fact identifier*), un índice asociado a ese hecho en la lista.

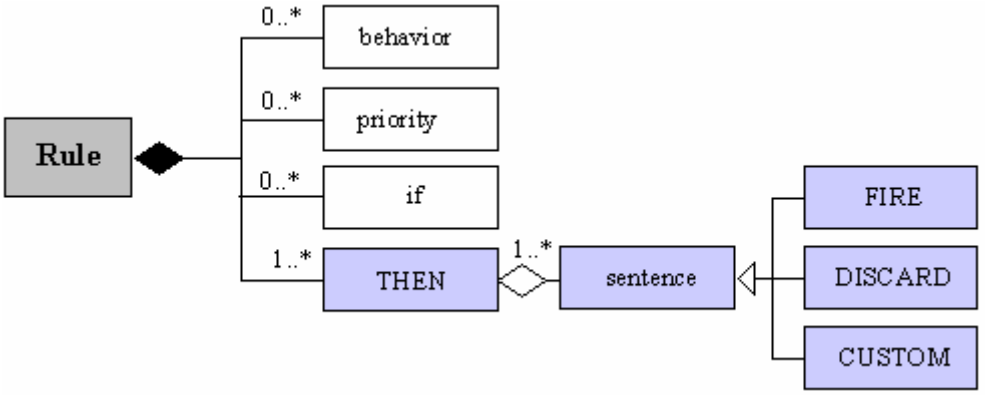

Figura 5.18 Las Acciones en la plantilla de Regla.

Su estructura formal:

```
[THEN <sentence> [,<sentence>]*;]
```

El elemento *sentence,* a su vez puede estar formado por otros subelementos, tal y como se detalla a continuación:

```
sentence ->
      FIRE   <hecho> [(assignment [,assignment]*)]
                                    [AND FORWARD];
      |DISCARD  <hecho-index>;
      |CUSTOM   <delimited-string>;
      |expression
```

5.8.4.1 Anular y Confirmar Hechos.

El conjunto de todos los hechos conocidos se almacena en la lista de hechos. Es posible agregar y eliminar hechos que representan información de gestión. Esto puede ser muy importante para la correcta administración de los recursos que componen el sistema de telecomunicaciones a administrar. El procedimiento se explica a continuación.

Hay ocasiones en las que resulta útil poder realizar modificaciones, duplicaciones o eliminaciones de hechos (e instancias) en el consecuente de una regla. Para ello resulta imprescindible poder detectar el índice (dirección) que la plataforma de gestión asoció al hecho cuando éste se afirmó, se denomina ***Captura de Direcciones de Hechos.***

La captura de direcciones se realiza en el antecedente de la regla, su sintaxis es la siguiente.

```
?<símbolo-variable> <- <Patrón-Elemento-Condicional>
```

En próximos apartados se muestran ejemplos de cómo puede utilizarse la captura de direcciones para modificar y borrar hechos.

- ***FIRE***.

Permite crear un nuevo hecho a partir de una serie de parámetros y su posterior envío a la plataforma de gestión. Se utiliza principalmente para generar eventos resultantes del proceso de inferencia experta. Eventos de alto nivel que identifican la posible causa de fallo y sus posibles soluciones.

Su sintaxis es:

```
FIRE <hecho>;
```

Cuando se añade un hecho, se devuelve la dirección de hecho del evento agregado.

Los parámetros que componen el hecho, pueden coincidir con parámetros de los eventos de la sección IF, estar prefijados con valores literales o resultar de algún proceso de cálculo (expresiones). Cuando se incorpora el modificador AND FORWARD, el evento se reenvía a la plataforma de gestión, en caso contrario sólo se actualiza el sistema experto. Esto permite generar eventos internos que sirven para mantener estados, cuando es necesario realizar un análisis complejo. En la siguiente regla un ejemplo de utilización:

```
grupoElectrógeno RULE
 PRIORITY 3;
 IF
  ?fecha ?hora ?estacion-1 ZAK_D/I_FALLO_INT ?estacion-2 ALARMA)
 THEN
  fire (?fecha ?hora ?estacion-1 AVERIA_EQUIPO_ZAK30/5 ?estacion-2 ALARMA));
REGISTER AS {rule 1 3 6 2};
```

Introduce un nuevo hecho en la lista de hechos. Este hecho a su vez, puede servir para la activación de otras reglas expertas de gestión que estén a la espera de que éste se produzca. El ejemplo anterior activa las reglas con una condición de generación de alarma `ZAK_D/I_FALLO_INT`.

- ***DISCARD***.

Al igual que es posible agregar hechos a la lista, también se pueden eliminar uno o más eventos de la base de hechos. Estos hechos, previamente deben de estar en la Base de Hechos del sistema experto asociado a la plataforma de gestión. Su sintaxis es la siguiente:

```
DISCARD <hecho-índice>
```

Se incluye el índice del hecho que hay que retirar de la lista. Sea la siguiente definición de regla siguiendo el estándar GDMO+:

```
filtroAlarma RULE
 PRIORITY 2;
  IF
   ?A<-(? ?h1 ?host ?alarma ALARMA)
   ?B<-(? ?h2 ?host ?alarma DESAPARECE_ALARMA &:(<(ABS(- ?h1 ?h2)) 1.00))
  THEN
   (discard ?A),
   (discard ?B),
   ("FALSA ALARMA " ?host);
REGISTER AS {rule 1 3 6 1};
```

Sea la definición de regla experta anterior en el estándar GDMO+, paradigma de utilidad para la eliminación de falsas alarmas. Si una regla determinada aparece y desaparece con un intervalo de tiempo inferior a un segundo, esto genera dos eventos, que activan la regla "*filtroAlarma*" y su ejecución. De esta forma se elimina el conjunto de hechos relacionados a dichas alarmas de la lista de hechos, que carecen de utilidad para la gestión del sistema.

Sean los hechos siguientes:

```
...
(04/12 1039.2782 UNIDAD1 TENSION_PASA_LIM_INF ALARMA)                    n
(04/12 1039.2805 UNIDAD1 TENSION_PASA_LIM_INF DESAPARECE_ALARMA)         n+1
...
```

La regla *filtroAlarma* se activa en el momento de su ejecución eliminando loshechos n y n+1 de la base de hechos.

Para la correcta eliminación de los hechos, las variables ?A y ?B recogen sus índices. En el caso de que se satisfagan las condiciones de la regla experta, los índices son necesarios para su posterior borrado de la lista de hechos. En este caso se eliminan las todas las alarmas que aparecen y desaparecen en un tiempo inferior a 1 segundo. Esta regla es muy útil para la eliminación de falsos avisos de averías.

- ***CUSTOM.***

Existe un tercer tipo denominado CUSTOM, que permite realizar otras acciones, incluyendo código nativo del sistema experto residente en la plataforma de gestión. Posibilita incorporar en el estándar GDMO+, comandos específicos a ciertos lenguajes, así como las mejoras existentes en los diferentes entornos de programación de sistemas expertos.

Transfiere al sistema experto la acción indicada. La operación a ejecutar, obligatoriamente debe ir expresada en el lenguaje nativo del Sistema Experto utilizado por la plataforma.

Sea el siguiente ejemplo de regla experta definida con la plantilla RULE del estándar GDMO+:

```
moduloTransmision RULE
 PRIORITY 3;
 IF
  (?fecha ?h1 ?estacion CTR190/7_TX_C1 ?destino ALARMA)
THEN
   ("Averia Asociada: FALLO EN MODULO DE TRANSMISION CTR190/7."),
   ("¿Desea reiniciar el equipo" ?estacion "(y/n)?:"),
   (custom ((set-attribute-value ?reinicio =(read)));
REGISTER AS {rule 1 3 6 1};
```

Se coloca una instrucción en ART*Enterprise 3.0[6]. El comando que aparece en al línea CUSTOM, permite la asignación de un valor a una variable dentro de la zona de ejecución de la regla experta de gestión.

```
(custom ((set-attribute-value ?reinicio =(read)))
```

Cuando se produce la alarma `CTR190/7_TX_C1`, la regla se activa.

```
...
(04/12 1039.2782 Estacion1 CTR190/7_TX_C1 Estacion2 ALARMA)
...
```

Posteriormente la regla se dispara y se presenta un mensaje al usuario de la plataforma de gestión, requiriéndole una información determinada. En este caso:

```
Avería Asociada: FALLO EN MODULO DE TRANSMISION CTR190/7.
¿Desea reiniciar el equipo Estacion1 (y/n)?:
```

El usuario proporciona la respuesta deseada "yes/no" y se almacena en la variable `?reinicio`. Esta información puede ser utilizada a continuación para enviar una señal de

[6] Entorno de desarrollo integrado, que cuenta con una potente interfaz gráfica. Basado en el lenguaje de programación C++ es utilizado para construir sistemas expertos. Soporta razonamiento basado en casos y reglas (ver Anexo D).

reinicio o para realizar las operaciones de control correspondientes sobre el equipo en cuestión.

Se ha comprobado que en la plantilla RULE del estándar GDMO+ propuesto, se pueden introducir comandos propios de cualquiera de los lenguajes de programación existentes, en nuestro caso se ha implantado el comando del *ART*Entrerprise 3.0*:

```
set-attribute-value ?variable =(valor)
```

Análogamente está permitido introducir expresiones o comandos propios de CLIPS, Prolog, etc.

Sea el comando refresh de Clips, que vuelve a introducir en la agenda las activaciones disparadas de una regla experta. Su formato es el siguiente:

```
(refresh <rule name>)
```

Un ejemplo de ejecución del comando Clips dentro del estándar GDMO+:

```
moduloTransmision RULE
  PRIORITY 3;
  (?fecha ?h1 ?remota CTR190/7_TX_C1 ?destino ALARMA)
THEN
  ("Averia Asociada: FALLO EN MODULO DE TRANSMISION CTR190/7"),
  (custom (refresh moduloTransmision);
REGISTER AS {rule 1 3 6 1};
```

De esta forma se consigue que en la agenda del sistema siempre exista una instancia de la regla `moduloTransmision`, que podrá ser disparada de nuevo, siempre que se siga satisfaciendo el antecedente para los hechos nuevos que se han derivado en la aplicación previa de reglas.

5.8.5 Activación de las Reglas Expertas de Gestión.

Se ha especificado el conocimiento de un problema mediante un conjunto de reglas expertas de gestión. En el sistema gestionado se producen una serie de acontecimientos que conforman una lista de hechos. Las reglas expertas de gestión se han integrado junto con las especificaciones de las clases de objetos gestionados, mediante la plantilla RULE del estándar GDMO ampliado. En este punto, se plantea una duda ***¿cómo se aplican las reglas?*** Para entenderlo, es necesario conocer algunos conceptos previos.

Hemos visto que una regla es una expresión del tipo:

Si *antecedente* **entonces** *consecuente.*

Su lectura es la siguiente:

Si el *antecedente* es cierto para los hechos almacenados en la lista de hechos, entonces pueden realizarse las acciones especificadas en el *consecuente.*

Se dice que una regla se activa cuando toda sus precondiciones se satisface. Es decir, cuando encajan con algunos de los hechos de la memoria de trabajo.

Para que las reglas expertas de gestión definidas con la plantilla RULE se activen, es necesario y suficiente que acontezcan los eventos enumerados en los patrones de la sección

IF. Deben verificarse todas las restricciones que pudieran implicar dichos patrones, es decir que existan los eventos y se equiparen a los patrones.

Los objetos gestionados emiten alarmas, estás van casando con las distintas condiciones de las reglas definidas en las clases de objetos gestionados. Cuando se verifican todas las condiciones de una regla experta de gestión, ésta se incorpora a la agenda del sistema. En función de las reglas que haya en ese instante en la agenda, la regla pasará a su activación y la ejecución de las operaciones de gestión vinculadas.

De forma breve y considerando que un evento equipara a un patrón cuando coinciden sus formatos y se respetan sus restricciones, ***una regla de gestión se activa cuando en el sistema de gestión hay eventos que se equiparan con todos los patrones IF.***

En lenguaje natural podría leerse la regla como:

```
SI HAN OCURRIDO eventos
       QUE EQUIPAREN <patrones-if>...<patrones-if>
       TALES QUE <condición>
ENTONCES
       HACER <acción_de_gestion>...<acción_de_gestion>
```

Advertir que una misma regla puede ser activada por distintos conjuntos de hechos. Cada una de las posibles activaciones recibe el nombre de *instancia de una regla.* Se dice que una regla está desactivada si no está activada, no existe ninguna instancia de dicha regla en la agenda del sistema.

5.8.6 Ejecución de las Reglas de Gestión.

Cuando una regla de gestión se dispara, se efectúan las acciones (consecuentes) asociadas a una regla determinada. Previamente la regla debe de haber pasado a estar activa. Para que una regla pueda ser disparada, es un prerrequisito indispensable que esté activa, o dicho de otra forma que resida en la agenda del sistema.

Todas las instancias de una regla de gestión se almacenan en la agenda. Cuando una instancia de una regla se dispara, la regla se suprime de la agenda y no vuelve a activarse[7].

Como se ha comentado, la agenda del sistema de gestión contiene las reglas activadas candidatas a ser disparadas. Cada módulo tiene su propia agenda y cada una de ellas funciona como una pila. La regla que se encuentra en el tope de la pila, es la primera en ser ejecutada. Cuando una regla de gestión se añade a la agenda, ocupará una posición que viene dada en función de la prioridad:

1.- La nueva regla experta de gestión, se colocará por debajo de todas las reglas con una prioridad mayor a la suya y por encima de aquellas reglas con una prioridad inferior.

2.- Para la reordenación de reglas expertas con igual prioridad, se utiliza las denominadas estrategias de resolución de conflictos. Ver anexo D.

3.- Si tras aplicar los pasos anteriores sigue habiendo igualdad de prioridades, las reglas expertas de gestión podrán ser ordenadas arbitrariamente atendiendo a otros aspectos. Normalmente por el orden de definición de las reglas, en cuyo caso las reglas de

[7] Excepto cuando se utiliza el comando *fire* visto en el 5.8.4.1 del presente tema.

gestión que se definen en una superclase, se disparan antes que las definidas en una de sus subclases. Figura 5.19.

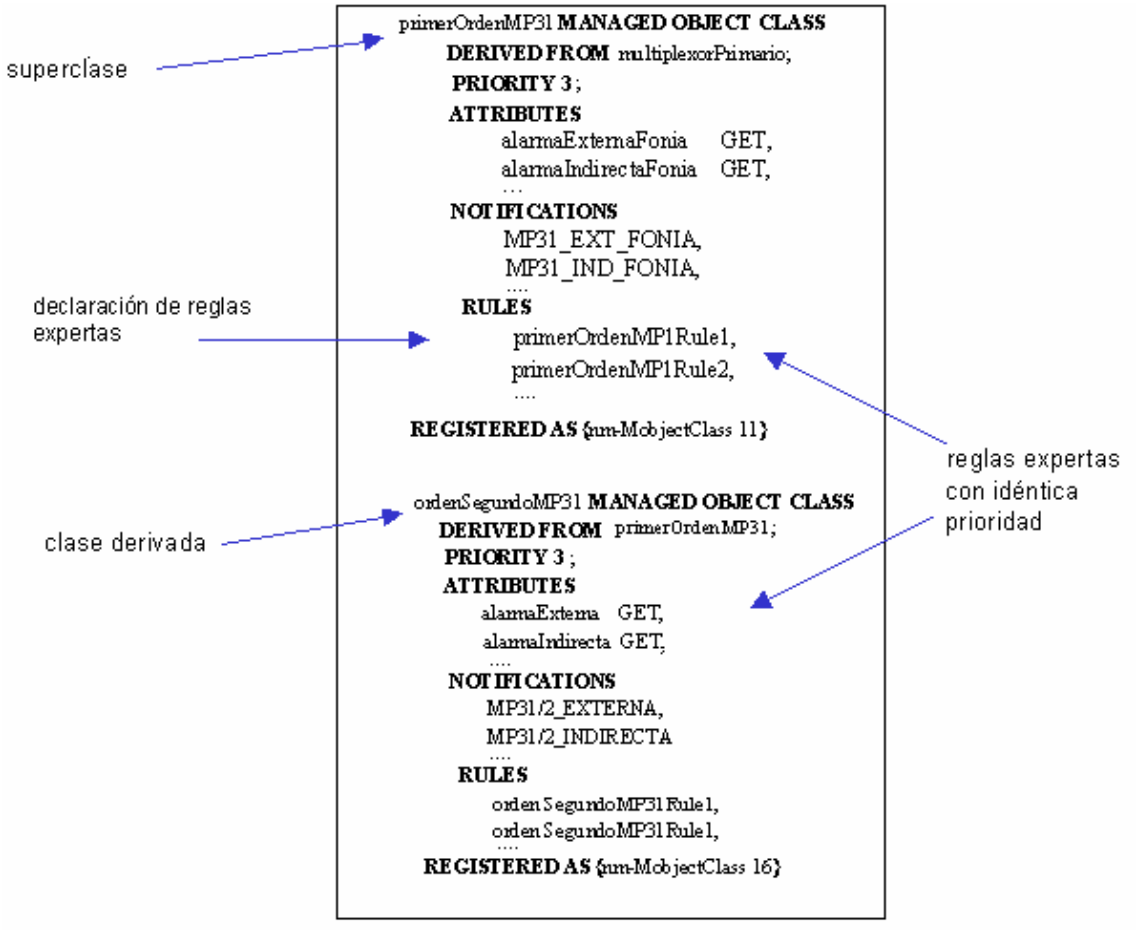

Figura 5.19 Resolución del orden de ejecución de reglas expertas de gestión

En el ejemplo anterior se aprecian la definiciones de dos clases de objetos gestionados `primerOdenMP31` y `ordenSegundoMP31`, correspondientes a recursos de comunicaciones denominados multiplexores[8]. Se observa que la clase `ordenSegundoMP31` es derivada de la superclase `primerOdenMP31`. Si se presentara el caso de activación de las reglas de ambas clases de objetos gestionados y tuviesen igual prioridad, se aplicarían las estrategias de resolución de conflictos, enumeradas en el anexo D.

Si aún así, sigue existiendo la incógnita de saber que regla experta de gestión se ejecutaría primero, se atiende al método establecido en este apartado. Se ejecuta en primer lugar las reglas de gestión `primerOrdenMP31Rule1`, `primerOrdenMP31Rule2`, ..., puesto que son las reglas expertas de gestión pertenecientes a la superclase de objeto gestionado.

Todo lo anteriormente expuesto, muestra que la asignación de prioridades a las reglas de gestión es una labor sumamente importante. En un sistema de gestión de redes de telecomunicaciones, donde existen servicios con un alto nivel de impacto sobre la comunidad de usuarios, hay que ser cautos a la hora de establecer la importancia de las alarmas a las que hay que prestar mayor importancia.

[8] Dispositivo que mezcla la información de varias fuentes, para ser enviadas a través de una única línea de comunicaciones.

Es conveniente establecer valores de prioridades, de modo que las alarmas de los recursos más importantes del sistema gestionado, reciban respuesta antes que otras alarmas menos importantes. Por todo ello, la asignación de prioridades a las reglas expertas, es un factor a tener en cuenta a la hora de definir las reglas expertas de gestión.

Por ejemplo, si en un momento dado se recibe una alarma de un multiplexor tipo `primerOdenMP31` y de un multiplexor tipo `ordenSegundoMP31`, la caída de servicio en el primero de ellos afectará a un número mucho mayor de usuarios y equipos terminales, que el caso de caída del segundo y por tanto habrá que prestarle una atención más notable.

5.9 Herencia de Reglas Expertas de Gestión entre Clases de Objetos Gestionados.

En este apartado se tratan los aspectos relacionados con la herencia de reglas de gestión, entre distintas clases de objetos gestionados. Las plantillas de clases en el estándar GDMO extendido, utilizan el constructor DERIVED FROM para la incorporación de características propias de una superclase, en las subclases de objetos gestionados.

La derivación entre clases proporciona la reutilización efectiva de las especificaciones de la clase base determinada. Si se tiene una clase base totalmente depurada, la herencia ayuda a reutilizar las definiciones en una nueva clase. No es obligatorio comprender las construcciones de la clase original, únicamente es necesario conocer el recurso que define. El objeto gestionado será tratado como una caja negra, de la que se heredan todas sus propiedades. No es imprescindible conocer las especificaciones que las definen, únicamente entender su cometido. La clase que hereda las propiedades de la clase base, puede definir sus propias características de tipo de objeto, entre ellas la incorporación de nuevas reglas expertas de gestión.

La ubicación de una clase en la jerarquía de herencias de las Clases de Objetos Gestionados, viene dada por medio de las referencias a las superclases a partir de las cuales se heredan las propiedades.

Cada una de las clases de objetos gestionados definidas se posiciona en un ***Árbol Jerárquico de Herencia.*** De esta forma la clase en cuestión hereda todas las propiedades de su padre y este a su vez hereda todas las propiedades de sus padres. *Así podemos resumir el proceso de creación de una nueva clase de objeto gestionado, simplemente en posicionarla en un lugar concreto dentro del árbol jerárquico de herencia y añadir las nuevas propiedades exclusivas de la clase de objeto gestionado en declaración* [Litman02].

5.9.1 Necesidades de Herencia.

Dada la multiplicidad de los sistemas de gestión actuales y los sistemas telemáticos donde se aplican, no es de extrañar que las descripciones asociadas a los recursos que lo conforman sean cada vez más prolijas. De forma general, una única clase de objeto gestionado no será suficiente por sí sola, para soportar el diseño de las especificaciones correspondiente a sistemas tan complejos. En consecuencia, es preciso recurrir a una amplia colección de Clases de Objetos Gestionados y de interrelaciones, que permitan definir todos los aspectos referentes a las especificaciones sobre los recursos que conforman la red a gestionar y el conocimiento de gestión, ambos aspectos imprescindibles para la correcta administración de los sistemas [Barba99].

En dominios de aplicación tan complejos, es primordial que las características propias de una clase de objeto gestionado sean reutilizadas por otras. Se hace fundamental establecer los instrumentos que permitan la reutilización de las especificaciones existentes.

Se necesitan tácticas que autoricen las relaciones entre clases, un mecanismo que proporcione los medios para crear las jerarquías de clases de objetos gestionados. Ordenamientos que posibiliten que una primera clase X, sea afín a una segunda clase Y, pero con la posibilidad de poder añadirle características propias al tipo de recurso definido. A este mecanismo se le denomina ***Herencia***, y es propia de los lenguajes Orientados a Objetos sobre los que se fundamenta el estándar GDMO y su extensión propuesta. La definición de herencia aplicada al estándar GDMO+ sería la siguiente:

Es el proceso por el que se pueden crear nuevas clases de objetos gestionados, denominadas clases de objetos gestionados derivadas, a partir de una clase base.

En ocasiones en el paradigma de los lenguajes Orientados a Objetos, se utilizan las denominadas *Clases Anidadas*. Consistentes en definir nuevas clases dentro de otras clases de objetos existentes. En la sintaxis del estándar GDMO, esta característica no es compatible. Tal y como se puede apreciar en la siguiente definición, la estructura de la plantilla de clase de objeto gestionado, carece de los constructores necesarios que permitan la creación de una nueva clase de objetos gestionado dentro de una clase existente.

```
<etiqueta-clase> MANAGED OBJECT CLASS
   [DERIVED FROM  ...
   [CHARACTERIZED BY  ...
   [CONDITIONAL PACKAGES ...
REGISTERED AS ...
```

La herencia es uno de los paradigmas más poderosos con que cuentan la sintaxis GDMO+. Facilita el desarrollo de las especificaciones, de los cada vez más complejos y heterogéneos Sistemas de Información actuales [Liebowitz98]. Sobre todo cuando se trata de grandes sistemas, donde la herencia es fundamental para la creación de definiciones cortas, compactas y sobre todo sin código redundante.

Por tanto, haciendo uso de las posibilidades ofrecidas por la herencia, las reglas expertas de gestión definidas para una clase de objeto gestionado, pueden ser aplicadas en otras clases de objetos gestionados, sin la obligación de realizar nuevas definiciones.

En GDMO+, la herencia se manifiesta con la creación de un tipo de objeto gestionado definido por el usuario, una clase. Esta nueva clase puede heredar las características de otras clases de objetos gestionados existentes, una superclase. También puede derivar sus propiedades a otras clases de objetos, las subclases. Cuando se hereda, las subclases de objetos derivadas reciben las propiedades: acciones, notificaciones, *reglas*, etc. de la clase original. A la vez que pueden añadir nuevas características o modificar las características heredadas. Se hace realmente una copia de la clase base en la nueva clase derivada y permite al diseñador añadir o modificar propiedades, sin alterar el código de la clase de objeto gestionado base.

Cuando en una subclase se redefine una regla experta heredada de una superclase, la especificación que prevalece será la que se efectúe en último lugar.

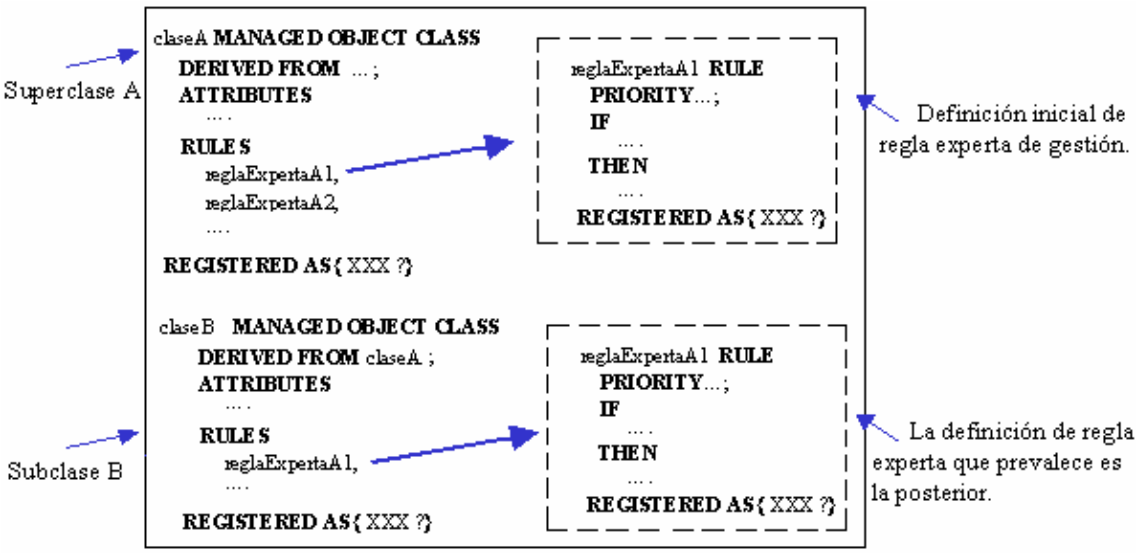

Figura 5.20 Redefinición de Reglas Expertas de Gestión.

En la figura anterior se aprecia que la regla experta de gestión *reglaExpertaA1* perteneciente a la clase *claseA,* es heredada por la subclase *claseB*. En la subclase la regla experta es redefinida para añadir nuevas condiciones en el antecedente o acciones en el consecuente. Cuando se aplique la *reglaExperta1* a las instancias de la *claseB*, se utiliza la definición realizada en esta clase y no en la superclase. Es decir la definición que prevalece es la realizada en la subclase *claseB*.

5.9.2 Clases Abstractas de Objetos Gestionados.

Son unas clases especiales que aparecen también en el estándar GDMO. Se utilizan para la creación y diseño de las jerarquías de clases objetos gestionados definidas.

Este tipo de clase es análoga a cualquier otra, pero con el propósito de servir como clase base de objeto gestionado o clase de partida a otras clases. Tienen ciertas restricciones, la más significativa es que nunca se va a crear una instancia de una clase abstracta, de ningún modo existirá un objeto del tipo definido en la clase. Se puede decir, que una clase abstracta presenta una interfase común para los objetos de las clases derivadas.

En la norma GDMO y por consiguiente también su extensión GDMO+, existe una primera clase denominada clase *"top"*, que se encuadra dentro de este tipo. Impera un solo requisito: todas las clases de GDMO deben heredar, generalmente de forma indirecta, al menos de la clase *"top"*.

Esta clase incluye algunos atributos estándar. De esta forma todas aquellas plantillas de clase de objeto gestionados que no heredan de otras plantillas deberán heredar así, de la clase *"top"*.

5.9.3 Instrumentos para la Herencia de Reglas Expertas.

Hemos comprobado que GDMO+, al igual que GDMO sigue una sintaxis especialmente Orientada a Objetos y que ofrece mecanismos propios para el desarrollo de especificaciones bajo esta metodología. En este apartado se exponen las posibilidades que este paradigma ofrece y la forma en que se implementa la herencia.

El estándar GDMO y su nueva extensión GDMO+, basan toda su potencia en la noción de Plantillas. Una plantilla puede percibirse como una definición formal, con una serie de elementos que permiten la definición de los objetos gestionados. Cada una de las plantillas se combina a su vez con otras, construyendo las especificaciones completas de clases de objetos gestionados e incluyendo todas las relaciones de herencia y contención existentes entre ellas.

La Plantilla de Clase de Objeto Gestionado, es el núcleo de toda definición de tipo de objeto gestionado. Contiene las referencias a las otras plantillas que forman parte de la descripción de la clase. Toda definición de clase de objeto gestionado, debe poseer los siguientes elementos:

- *Un nombre leíble:* la etiqueta de la clase.
- *Un OID*: el nombre de la clase.
- *Un orden*: que especifica su posición dentro de la jerarquía de herencia.

De los cinco constructores que conforman la plantilla de clase de objeto gestionado, es la cláusula *"**DERIVED FROM**"* la encargada de posicionar la clase de objeto gestionado definida, dentro de la jerarquía de herencia [Udupa99].

En el siguiente ejemplo, la clase *LAN* se sitúa por debajo de la superclase *System*:

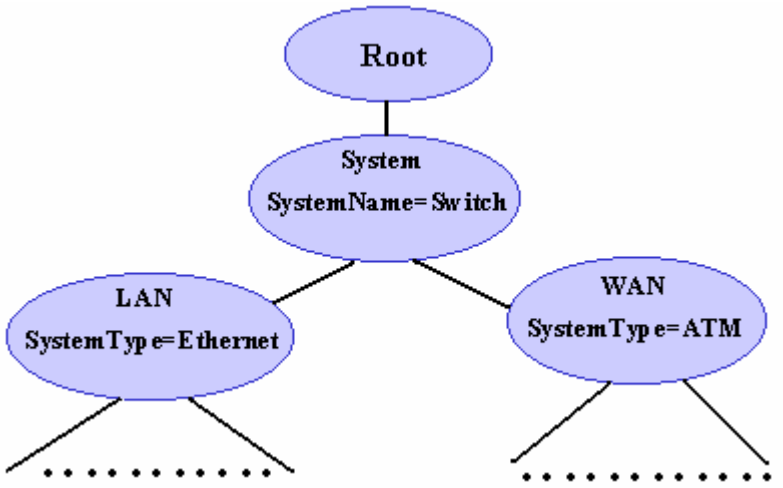

Figura 5.21 Árbol de jerarquía de clases de objetos gestionados

De forma general una clase heredará las propiedades definidas en las clases situadas en los niveles superiores a la misma. Así, las clases de objetos gestionados de niveles inferiores a la clase *System* heredan todas sus propiedades, entre ellas las reglas expertas de gestión.

Con esto se consigue que ***una misma regla experta de gestión, actúe sobre varias clases de objetos gestionados***. Para conseguir el objetivo, hay que establecer una metodología apropiada de herencia de reglas expertas de gestión entre clases, en próximos apartados se presentan las técnicas para su logro.

5.9.4 Especificación de Reglas Expertas de Gestión en Superclases.

El primero de los métodos que se presenta para heredar las reglas expertas de gestión se resuelve de la siguiente manera:

"Se definen las reglas expertas que se pretenden reutilizar en una clase de objeto gestionado denominada superclase. A partir de esta superclase, se definen las subclases de objetos gestionados que heredarán las reglas expertas de gestión".

Las reglas expertas heredadas, se definen únicamente en la superclase. Las subclases reutilizan todas las propiedades de la superclase junto con las reglas expertas de gestión.

Sea el siguiente ejemplo:

```
equipoRadio MANAGED OBJECT CLASS
    DERIVED FROM sistema;
    CHARACTERIZED BY equipoRadioPkg PACKAGE
    RULES equipoRadioRule1, equipoRadioRule2;
REGISTERED AS {xxx ?};

transceptorRT21 MANAGED OBJECT CLASS
    DERIVED FROM equipoRadio;
    CHARACTERIZED BY transceptorRT21Pkg PACKAGE
    RULES RT21Rule;
REGISTERED AS {xxx ?};
```

Se ha definido una subclase de objeto gestionado *transceptorRT21*, que hereda las características de la superclase de objeto gestionado *equipoRadio*. Todas las subclases definidas a partir de ésta, contendrán las propiedades especificadas en el paquete *equipoRadioPkg* y las reglas expertas *equipoRadioRule1* y *equipoRadioRule2* de la superclase. Cada una de estas subclases puede tener sus propias reglas expertas de gestión, para lo cual hay que incluirlas en la clase correspondiente junto con las demás propiedades individuales. Tal es el caso de la regla *RT21Rule,* que se aplicará exclusivamente al tipo de objetos gestionados y definidos por la subclase *transceptorRT21*.

Concluyendo, las reglas expertas de gestión comunes a un conjunto de clases, se definen en la superclase correspondiente y las reglas expertas de gestión exclusivas a las subclases, se definen en cada una de ellas de forma individual.

5.9.5 Utilización de Paquetes.

Las propiedades heredadas pueden sufrir modificaciones, para ello se emplean los siguientes constructores:

- CHARACTERIZED BY: Para incorporar paquetes obligatorios.

- CONDITIONAL PACKAGES: Para los denominados *"paquetes condicionales"* que podrán incluirse o no en una clase determinada, en función de ciertos factores condicionales.

La incorporación de variantes de una regla de gestión determinada en una "*instancia*" de un Objeto Gestionado, se decidirá por medio de condiciones. Estas expresiones condicionales se incluyen en la definición de Clase de Objeto Gestionado, donde el paquete aparecerá o no en función de su cumplimiento. Para saber si un paquete condicional formará parte integral de una clase específica de objeto gestionado, deberán cumplirse todas las condiciones relacionadas.

En el siguiente ejemplo de clase de objeto gestionado definida por medio del estándar GDMO+, se aprecia que la subclase *tokenRing* hereda todas las propiedades definidas para la superclase *lanNet,* que contiene las características generales a la las redes locales. Se especifica un paquete obligatorio *tokenRingLan,* que contiene las nuevas peculiaridades de redes locales con método de acceso al medio en *token ring* [Parenll97]. Asimismo se define un conjunto de caracteres particulares de enrutamiento en un paquete condicional, para el caso en que se disponga de una conexión a un troncal de red con tecnología de alta velocidad FDDI, *Fiber Distribute Data Interface* [Parenll97]. El paquete contiene todas las propiedades características de este modo de conexión.

```
tokenRing MANAGED OBJECT CLASS
  DERIVED FROM lanNet;
  CHARACTERIZED BY tokenRingLan;
  CONDICIONAL PACKAGE tokenRingRouter PRESENT IF "connected to
                      an FDDI backbone LAN";
REGISTER AS {1 3 6 1 9 2};
```

5.9.5.1 Inclusión de Reglas Expertas de Gestión en un Paquete.

Las reglas expertas siguen un tratamiento análogo a las restantes propiedades de una clase de objeto gestionado y por tanto pueden ser incluidas dentro de los paquetes. El método a seguir es el siguiente:

"Las reglas expertas de gestión, junto con todas las propiedades comunes a un conjunto de clases se agregan en un paquete. A continuación, el paquete definido se incluye en todas las clases de objetos gestionados, donde debe aparecer."

En el paquete además de las reglas expertas, se definen todas las propiedades correspondientes al recurso especificado en la clase: atributos, notificaciones, acciones, parámetros, etc.

Observémoslo de forma más clara mediante un ejemplo, se define la superclase de objeto gestionado *equipoRadio*, que contiene un paquete *equipoRadioPkg*. El paquete incluye las propiedades aplicables al tipo de recurso definido en la clase. Distinguir las dos reglas expertas de gestión *equipoRadioRule1* y *equipoRadioRule2*. Se define a continuación dos clases de objetos gestionados *tranceptorRT21* y *tranceptorCRT190*, subclases de la primera y que contienen ambas reglas expertas de gestión.

Definición de clases.

```
equipoRadio MANAGED OBJECT CLASS
    DERIVED FROM sistema;
    CHARACTERIZED BY equipoRadioPkg PACKAGE
REGISTERED AS {xxx ?};

transceptorRT21 MANAGED OBJECT CLASS
    DERIVED FROM equipoRadio;
    CHARACTERIZED BY tranceptorRT21Pkg PACKAGE
REGISTERED AS {xxx ?};

transceptorCRT190 MANAGED OBJECT CLASS
    DERIVED FROM equipoRadio;
    CHARACTERIZED BY tranceptorCRT190Pkg PACKAGE
REGISTERED AS {xxx ?};
```

Definición de paquetes.

```
equipoRadioPkg PACKAGE
    RULES equipoRadioRule1, equipoRadioRule2;
REGISTERED AS {xxx ?};

TransceptorRT21Pkg PACKAGE
    RULES RT21Rule;
REGISTERED AS {xxx ?};

transceptorCRT190Pkg PACKAGE
    RULES CRT190Rule1, CRT190Rule2;
REGISTERED AS {xxx ?};
```

Las subclases de objetos gestionados *transceptorRT21* y *transceptorCRT190* heredan el paquete *equipoRadioPkg* definido para la superclase *equipoRadio*. El paquete contiene todas las propiedades específicas para la superclase, entre las que se encuentran las reglas expertas de gestión *equipoRadioRule1* y *equipoRadioRule2*.

Para la incorporación de reglas expertas de gestión en las subclases, se pueden utilizar nuevamente los paquetes. Tal es el caso del paquete *transceptorRT21Pkg* que contiene la regla experta de gestión *RT21Rule*, definida de forma exclusiva para la subclase *transceptorRT21*. Análogamente las reglas expertas de gestión *CRT190Rule1* y *CRT190Rule2* definidas en el paquete *transceptorCTR190Pkg*, son específicas de la subclase *transceptorCRT190*. Estas reglas pueden a su vez ser heredadas a las subclases definidas a partir de ésta.

Este procedimiento mejora el mostrado en el apartado anterior, en cuanto aprovecha todas las ventajas que resultan de la utilización de los paquetes estudiadas en el capítulo 4.

5.9.6 Herencia Múltiple.

La norma GDMO+ admite que una clase pueda reutilizar características provenientes desde varias superclases de objetos gestionados. Una clase derivada puede tener más de una clase base origen. Esta forma de herencia es muy adecuada para la definición de objetos que son compuestos por naturaleza, como por ejemplo un recurso *Conmutador ATM/Ethernet*, que puede heredar características desde varias clases de objetos gestionados.

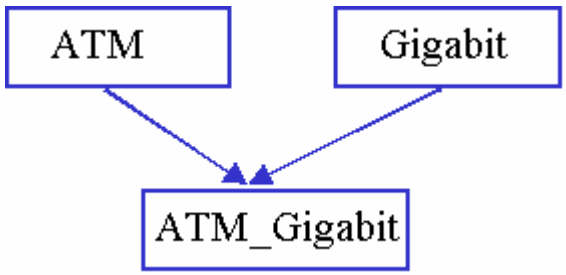

Figura 5.22 Herencia Múltiple.

Para este caso se definen dos clases independientes que recogen todas las características correspondientes a conmutadores tipo ATM y tipo Ethernet. Una tercera clase denominada ATM/Ethernet, heredará propiedades desde ambas superclases.

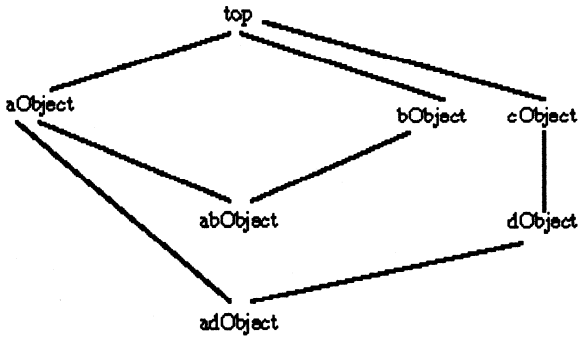

Figura 5.23 Subclase con Relación de Herencia Múltiple

Se trata de una extensión de la sintaxis de la clase derivada. Introduce una cierta complejidad en las especificaciones de los objetos gestionados, pero proporciona grandes beneficios. En la siguiente definición de clases se muestra como se realiza la herencia múltiple desde dos superclases:

```
switchATM MANAGED OBJECT CLASS
  DERIVED FROM switch;
  CHARACTERIZED BY switchATMPkg PACKAGE
  RULES switchATMRule1, switchATMRule2;
REGISTERED AS {xxx ?};

switchGigabit MANAGED OBJECT CLASS
  DERIVED FROM switchEthernet;
  CHARACTERIZED BY switchGigabitPkg PACKAGE
  RULES switchGigabitRule1, switchGigabitRule2;
REGISTERED AS {xxx ?};

switchATMGigabit MANAGED OBJECT CLASS
   DERIVED FROM switchATM, switchGigabit;
   CHARACTERIZED BY switchATMGigabitPkg1 PACKAGE
   RULES switchATMGigabitRule;
REGISTERED AS {xxx ?};
```

Se definen dos clases que recogen las especificaciones de los tipos conmutadores ATM y Gibgabit [Cisco99]. Cada clase contiene las propiedades del modelo específico, contienen reglas expertas de gestión propias. Para el caso del conmutador ATM existen las reglas *switchATMRule1*, *switchATMRule2*. Lo mismo ocurre para la clase conmutador tipo Gigabit que posee las reglas *switchGigabitRule1*, *switchGigabitRule2*.

Seguidamente se detalla una subclase *switchATMGigabit* que heredará todas las propiedades de las dos clases de conmutadores existentes. Observar que el constructor DERIVED FROM en la subclase, contiene las clases *switchATM* y *switchGigabit*. La nueva clase definida *switchATMGigabit,* recoge las especificaciones para un tipo de conmutador que dispone de puertos con tecnología ATM y Gigabit. Incluye todas las propiedades definidas en ambas clases, entre las que se encuentran las reglas expertas de gestión: *switchATMRule1*, *switchATMRule2, switchGigabitRule1* y *switchGigabitRule2*.

Igualmente, se observa que la subclase incluye una nueva regla experta de gestión propia del tipo definido denominada *switchATMGigabitRule,* aplicable única y exclusivamente a las instancias de esta clase de objeto *switchATMEthernet*.

5.9.6.1 Anfibologías de la Herencia Múltiple.

La aplicación de varias superclases o clases bases múltiples, puede provocar que se produzcan un conjunto de ambigüedades en las especificaciones GDMO+.

Un caso particular de la herencia múltiple, puede producir que un mismo elemento importe características duplicadas. Sucede cuando dos definiciones de superclases incluyen definiciones correspondientes a una misma regla experta de gestión. En el caso que una regla experta estuviese incluida en ambas superclases, es obvio que la subclase únicamente contendría una definición de ésta. Se asume que la subclase contendrá sólo una copia de las definiciones de reglas involucradas.

Planteamos la siguiente duda: ***¿Cómo acceden los objetos de la clase derivada a la propiedad Regla Experta correcta de la clase base?*** El nombre de la propiedad no es suficiente, no se podrá deducir cuál de las dos especificaciones es la invocada. Este tipo de duplicidad se denomina ***Colisión de Nombre***.

Si un objeto hereda propiedades de más de una clase de objeto gestionado, puede haber una colisión en alguna de las características heredadas. En concreto, una regla experta de gestión puede estar presente en dos o más de las superclases.

Sea una clase de objeto gestionado *swicthFastEthernet,* que deriva de dos clases base *swicthEthernet* y *wicthFast*. Ambas clases contienen una regla experta de gestión con el nombre *ethernetRule*. Cuando se define un objeto de la clase *swicthFastEthernet*, hay que resolver la ambigüedad surgida entre las definiciones de la regla experta de gestión existentes en ambas superclases. Para ello se utilizará un mecanismo denominado de ***Resolución de Ámbito.***

```
switchEthernet MANAGED OBJECT CLASS
  DERIVED FROM sistema;
  CHARACTERIZED BY switchEthernetPkg PACKAGE
  RULES ethernetRule;
REGISTERED AS {xxx ?};

switchFast MANAGED OBJECT CLASS
  DERIVED FROM sistema;
  CHARACTERIZED BY switchFastPkg PACKAGE
  RULES ethernetRule, switchFastRule;
REGISTERED AS {xxx ?};

switchFastEthernet MANAGED OBJECT CLASS
   DERIVED FROM switchEthernet, switchFast;
   CHARACTERIZED BY switchFastEthernetPkg1 PACKAGE
   RULES switchFastEthernetRule;
REGISTERED AS {xxx ?};
```

Vamos a exponer el método seguido para la resolución del conflicto aplicándolo sobre la definición de clase de objeto gestionado *switchFastEthernet*:

1. Se aplica la primera de las superclases del árbol de herencia. El objeto definido perteneciente a la clase *swicthFastEthernet,* hereda todas las características definidas para la primera de la superclases *swicthEthernet*. Esta primera superclase incluye la propiedad regla experta de gestión *EthernetRule,* que es común a ambas superclases.

2. Acto seguido, se aplica la segunda de las superclases *swicthFast* y se heredan todas sus características, excepto la regla experta *EthernetRule* que ha sido previamente incluida desde la primera de las superclases *swicthFastEthernet*.

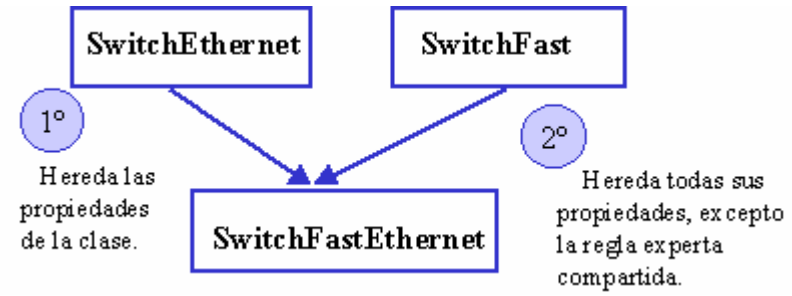

Figura 5.24 Herencia múltiple, ambigüedades.

De esta forma tan solo se incluye la definición de la regla experta de gestión existente en la primera de las superclases. Esta técnica es extensible a aquellos casos en que una subclase herede desde más de dos superclases, y éstas a su vez compartan la definición de una misma regla experta de gestión. Las instancias de la subclase, tomarán como válida únicamente la definición de la regla experta existentes en la primera de las superclases que aparezca en el oportuno árbol de herencias.

5.9.6.2 Herencia Repetida.

Un caso especial que puede llegar a darse en la herencia múltiple es la denominada herencia repetida. Una condición de la herencia múltiple, es que una clase derivada no puede heredar más de una vez de una sola clase, al menos no directamente.

Sin embargo, puede suceder que una clase de objeto gestionado pueda derivar dos o más veces por caminos distintos, de modo que se pueda repetir una misma superclase. La peculiaridad de recibir por herencia desde misma clase más de una vez se conoce como herencia repetida.

Cuando se produce la herencia repetida, una subclase puede heredar una misma regla experta de gestión por dos vías distintas. GDMO+ dispone de dos mecanismos para solventar este tipo de ambigüedades.

- ***Compartición:*** Una característica que se hereda de un ascendiente de forma repetida, bajo el mismo nombre. Aparece cuando se hereda la misma regla experta de gestión con el mismo nombre, por distintos caminos.

  Esto caso se resuelve simplemente dando origen a una única característica en el descendiente, se origina una *Compartición.*

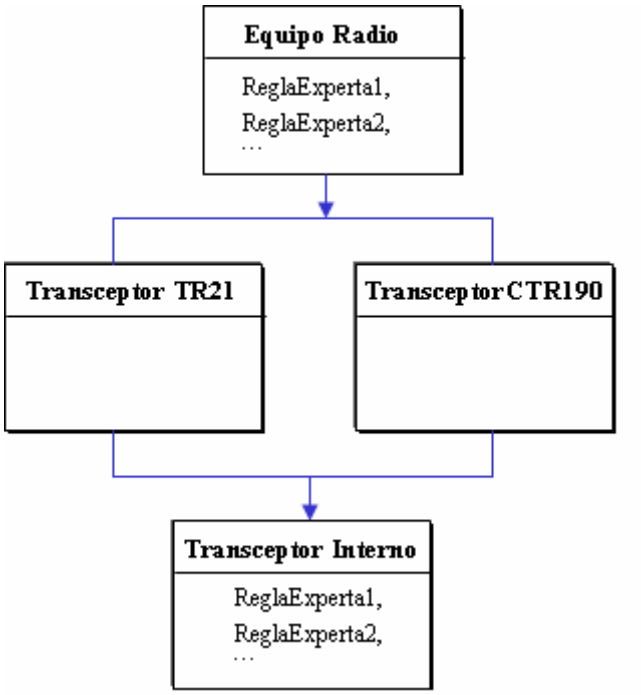

Figura 5.25 Herencia Repetida y Compartición.

• ***Replicación:*** Sin embargo, puede ocurrir que la regla experta de gestión se redefina en uno de los caminos de herencia. Esto provoca que aunque ambas reglas tienen el mismo nombre, tengan distinta definición, son distintas y por tanto hay que conservar ambas reglas expertas.

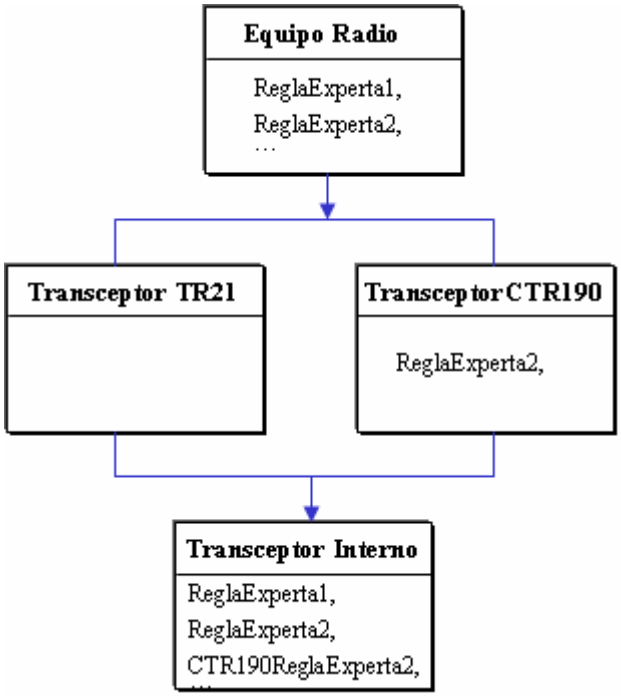

Figura 5.26 Herencia Repetida y Replicación.

En el ejemplo de la figura 3.26, la regla experta de gestión *ReglaExperta2* sufre una modificación, para adecuarla a la gestión efectiva del transceptor de radio tipo CTR190. La clase de objeto gestionado *TransceptorInterno* precisará mantener las dos copias de la misma propiedad heredada por caminos distintos. Cuando un

atributo u operación se hereda en forma repetida bajo diferentes nombres, el resultado de la herencia es la *replicación* del atributo u operación en la clase descendiente. Se hereda la regla experta de gestión original y la regla experta de gestión redefinida, a la que se asigna un nombre distinto.

Aunque hemos visto distintos métodos de resolución de conflictos surgidos con la herencia repetida, en general lo ideal es revisar nuestro diseño y examinar las posibles alternativas que eviten esta situación.

5.9.7 Clases de Objetos Gestionados Virtuales.

Las reglas expertas de gestión definidas para una clase de objeto gestionado serán aplicables a cada una de las entidades propias de la clase. De igual forma afectarían a subclases que hereden estas reglas expertas, desde las superclases de objetos gestionados.

Existen casos en los que es conveniente que una misma regla experta de gestión sea aplicable simultáneamente a objetos de distintas clases. Tal es el caso de reglas que dependen directamente de las características incluidas en varias clases de objetos gestionados definidas. Se plantea la siguiente cuestión:

¿Qué ocurre cuando una misma regla experta de gestión depende de aspectos y propiedades recogidas en más de una clase de objeto gestionado y es necesario el manejo simultáneo de ambas?

Para resolverlo se crean unas clases intermedias que recogen todos aspectos de las distintas clases de objetos implicadas, las *Clases de Objetos Gestionados Virtuales*. Estas clases contienen las propiedades necesarias para que el motor de inferencia pueda contrastar los hechos y proceda a la activación de la regla experta de gestión, en que caso de que estos se satisfagan. Efectúa el reconocimiento de los patrones y los procesa como si de una única clase de objeto gestionado se tratase.

En la siguiente figura se aprecia de forma gráfica el concepto de clase virtual. La clase de objeto gestionado *claseAB* contiene la regla experta de gestión *reglaExpertaAB,* que tiene dependencia de hechos relacionados con los tipos de objetos definidos en las clases *claseA* y *claseB*:

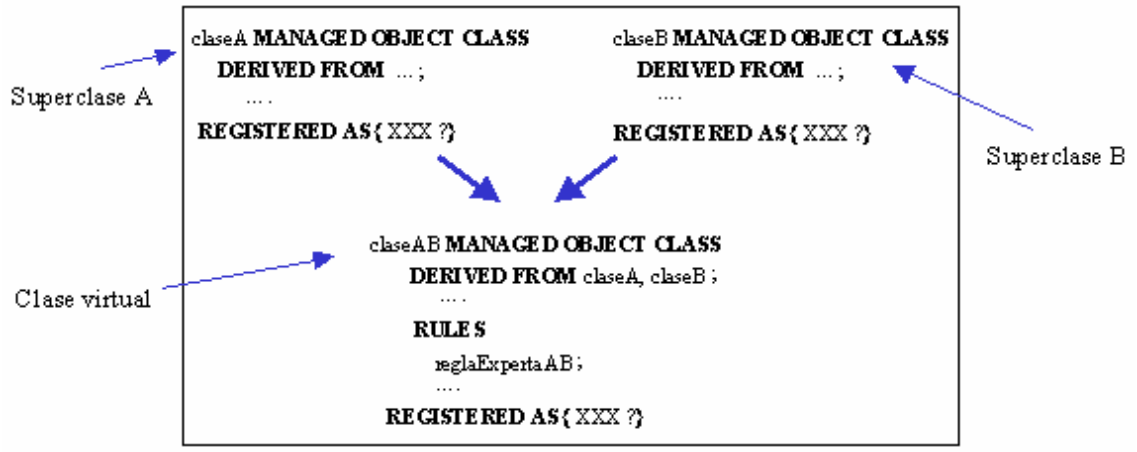

Figura 5.27 Clase de Objeto Gestionado Virtual.

Para captar mejor el objetivo de las clases virtuales, supongamos un ejemplo donde se presente una conexión entre un transceptor de radio tipo CTR190 y un multiplexor tipo MP31. Sea una regla experta de gestión que detecte anomalías y proceda a realizar una valoración y posterior diagnosis de la conexión. La regla experta de gestión necesita conocer detalles referentes a alarmas o eventos inherentes a ambas clases de objetos gestionados implicadas.

Puede resultar por tanto, que los dos tipos de recursos involucrados tengan reglas expertas de gestión definidas para ser aplicadas sobre las entidades propias al objeto definido y por otro lado reglas expertas de gestión que se aplicarán de forma común a ambas clases de objeto.

La siguiente definición de clase de objeto gestionado corresponde al multiplexor de primer orden MP31:

```
multiplexorMP31 MANAGED OBJECT CLASS
  DERIVED FROM multiplexorSuperior;
  CHARACTERIZED BY tranceptorMP31Pkg PACKAGE
  RULES MP31EntradaRule, …;
REGISTERED AS {xxx ?};
```

Se observa que la clase de objeto definida, contiene reglas expertas de gestión peculiares del recurso que se define. En este caso la regla *MP31EntradaRule*, detecta fallo en la señal de transmisión en el multiplexor de primer orden tipo MP31.

La alarma `MP31_EXT_FONIA` es característica de multiplexor MP31 y por tanto aplicable a esta clase.

```
MP31EntradaRule RULE
  PRIORITY 3;
  (?fecha ?h1 ?remota MP31_EXT_FONIA ?destino ALARMA)
 THEN
  ("Grado de Severidad: 3"),
  ("Averia Asociada:
       FALLO EN SEÑAL DE ENTRADA A MP31, ESTACION " ?remota);
REGISTERED AS {rule 1 3 6 9};
```

De igual forma se definen reglas expertas de gestión que afectan a los objetos definidos en la clase *transceptorCTR190*, que representa al recurso del tipo equipo de radio CTR190.

```
transceptorCTR190 MANAGED OBJECT CLASS
  DERIVED FROM equipoRadio;
  CHARACTERIZED BY tranceptorCTR190Pkg PACKAGE
  RULES CTR190EnlaceRule, …;
REGISTERED AS {xxx ?};
```

La regla experta CTR190*EnlaceRule,* se emplea para la gestión del recurso definido en la clase *transceptorCTR190* y trata alarmas pertenecientes a este tipo de objeto.

```
CTR190EnlaceRule RULE
  PRIORITY 3;
   (OR
    ((?fecha ?h1 ?remota CTR190/7_RX_C1 ?destino ALARMA)
     (NOT(?fecha ?h2 ?remota CTR190/7_RX_C2 ?destino ALARMA & :
              (<(ABS(? ?h1 ?h2)) 1.00))))
  THEN
  ("Grado de Severidad: 3"),
  ("Avería Asociada: FALLO EN RECEPCION, ESTACION " ?remota),
  ("Recomendacion: REVISAR EL RECEPTOR CTR190/7.");
REGISTERED AS {rule 1 3 6 9};
```

Las alarmas `CTR190/7_RX_C1` y `CTR190/7_RX_C2` son exclusivas del transceptor CTR190 y por tanto aplicables únicamente a este tipo de objeto.

Como se ha indicado, pueden existir relaciones entre distintas clases de objetos gestionados, que a su vez deben quedar manifiestas en las reglas expertas de gestión. Hay que tener en cuenta las alarmas y eventos que se producen en los tipos de recursos involucrados y especificados en clases de objetos distintas. Para ello se introduce una nueva clase de objeto denominada *"virtual"*, que recoge la definición de regla experta de gestión, donde se advierten los aspectos necesarios para su activación y que pueden corresponder a clases de objetos diferentas.

Siguiendo con el ejemplo anterior, en la siguiente regla experta de gestión se aprecian elementos que pertenecen a las dos superclases de objetos, y se utilizan de forma simultánea para la evaluación de la regla experta:

```
CTR190_MP31Rule RULE
PRIORITY 3;
IF
(?fecha ?h1 ?remota MP31_EXT_FONIA ?destino ALARMA)
(OR(?fecha ?h2 ?remota CMF?38_BER_1 ?destino ALARMA
       & : (<(ABS(? ?h1 ?h2)) 1.00))
   (?fecha ?h3 ?remota CMF?38_BER_2 ?destino ALARMA
       & : (<(ABS(? ?h1 ?h3)) 1.00))
   (?fecha ?h4 ?remota CMF?38_F_AL_TR_C1 ?destino ALARMA
       & : (<(ABS(? ?h1 ?h4)) 1.00))
   (?fecha ?h5 ?remota CMF?38_F_AL_TR_C2 ?destino ALARMA
       & : (<(ABS(? ?h1 ?h5)) 1.00)))
  THEN
   ("Grado de Severidad: 3"),
   ("Avería Asociada:
      FALLO EN EQUIPO CONECTADO AL MP31, ESTACION " ?remota ),
   ("Recomendacion: REVISAR EQUIPOS CONECTADOS A MULTIPLEXOR");
REGISTERED AS {rule 1 3 6 9};
```

Se determinan propiedades definidas en las dos clases de objetos, la alarma `MP31_EXT_FONIA` es propia de la clase *MP31Entrada*, mientras que las alarmas `CMF?38_F_AL_TR_C1`, `CMF?38_F_AL_TR_C2` y `CMF?38_BER_2` pertenecen al tipo de objeto especificado en la clase de objeto *transceptorCTR190*.

La clase de objeto gestionado virtual que contiene esta regla experta y por consiguiente conjuntamente aspectos de ambas clases de objetos: *transceptorCTR190* y *multiplexorMP31,* tiene la siguiente estructura:

```
CTR190_MP31 MANAGED OBJECT CLASS
 DERIVED FROM transceptorCTR190, multiplexorMP31;
 CHARACTERIZED BY CTR190_MP31Pkg PACKAGE;
 RULES CTR190_MP31Rule, …;
REGISTERED AS {xxx ?};
```

Se ha establecido una regla experta de gestión `CTR190_MP31Rule`, que reúne aspectos presentes en las clases de objetos gestionados involucradas: *transceptorCTR190* y *multiplexorMP31*.

Técnicamente una clase virtual define por lo menos una propiedad, en nuestro caso una regla experta de gestión. Esta propiedad regla puede ser de nueva creación o una redefinición de alguna de las reglas heredadas de las superclases.

Se deduce que las clases virtuales son de gran utilidad. Con este modelo de definición se consigue heredar propiedades de las clases de objetos gestionadas y reutilizarlas de forma conjunta. Permite a la clase virtual acoger las reglas expertas procedentes de las superclases, e incorporar nuevas condiciones o acciones a la regla experta reutilizada.

Una definición virtual puede venir con un prototipo de la propiedad (regla experta), la cual no contiene implementación determinada de la misma, y puede ser posteriormente redefinida en alguna de sus subclases.

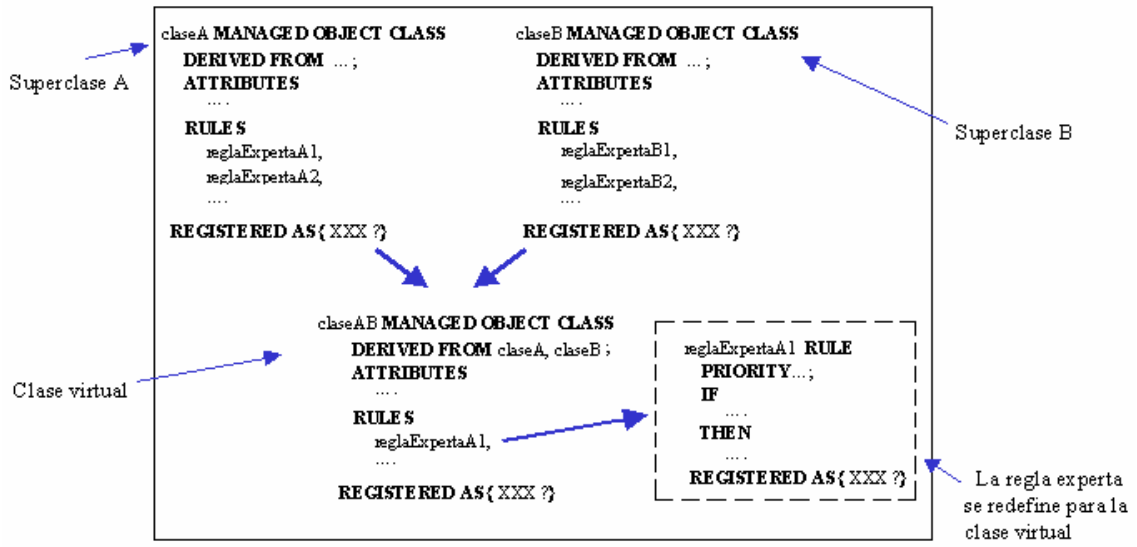

Figura 5.28 Clase virtual con redefinición de regla experta de gestión.

5.10 Un Ejemplo de Clase de Objeto Gestionado.

A continuación se muestra un paradigma de integración de reglas expertas de gestión, tomando para ello la definición de un objeto gestionado transceptor de radio CTR190 perteneciente a la red de telecomunicaciones privada que actualmente da servicio a una compañía eléctrica.

En el ejemplo siguiente se observa una Clase de Objeto Gestionado *radioTransceptorCTR190*, donde se definen las propiedades del tipo de objeto real denominado transceptor de radio modelo CTR190.

```
radioTrasnceptorCTR190 MANAGED OBJECT CLASS
      DERIVED FROM "rec.X721":top;
      CHARACTERIZED BY radioTransceptorCTR190Package;
REGISTERED AS {nm-MobjectClass 1};
```

La clase incluye el paquete obligatorio *radioTransceptorCTR190Package*, que contiene todas las especificaciones correspondientes a dicho dispositivo. Las características propias de la clase definida, son incorporadas a través del paquete *transceptorPackage:*

```
transceptorPackage PACKAGE
    ATTRIBUTES
        reception Power    GET,
        sense              GET,
        speedTransmission  GET,
         …
     NOTIFICATIONS
         damageFeeding,
         inferiorLimit,
         repairAction;
     RULES damageFrecuent,
           errorTransmission,
            …
REGISTERED AS {nm-package 1};
```

Entre las distintas propiedades, merece una mención especial las dos reglas expertas que han sido asociadas a la clase definida por medio de la cláusula RULES. Para la descripción de las reglas se emplea la plantilla RULE de la norma GDMO+ propuesta. Esta propiedad permite incorporar el conocimiento correspondiente al Sistema Experto que gestionará los tipos de objetos gestionados *radioTrasnceptorCTR190* que define las propiedades del dispositivo que representa el recurso real presente en la red.

Ambas reglas detectan anomalías o defectos de funcionamientos producidos en el transceptor y sugieren las medidas necesarias para resolver la excepción.

Concretamente la primera de las reglas *damageFrecuent,* se encarga de detectar casos en los que se producen averías en la alimentación eléctrica del dispositivo.

```
damageFrecuent RULE
   BEHAVIOUR damageFrecuentBehaviour;
   PRIORITY 4;
   IF
     (?fecha ? ?remota CTR190/7_F_ALIM_2 ?destino ALARMA)
     (NOT (?fecha ? ?remota CCA?34_AIS_DE_BB ?destino ALARMA))
   THEN
     ("Grado de Severidad: 4"),
     ("Avería Asociada: Avería en fuente de alimentación 2 de CRT190/7,
       estación: " ?remota " o avería en alimentación de la fuente 2"),
     ("Recomendación: Revisar la fuente y/o su alimentación.");
REGISTERED AS {rule 1 3 6 5};
```

La segunda regla *transmissionError* detectar anomalías producidas en el módulo de transmisión del equipo y ofrece las recomendaciones necesarias para su inmediata reparación.

```
transmissionError RULE
  BEHAVIOUR transmissionErrorBehaviour;
  PRIORITY 3;
  IF
    (?fecha ?h1 ?remota CTR190/7_TX_C2 ?destino ALARMA)
    (?fecha ?h2 ?destino CTR190/7_RX_C2 ?remota ALARMA
      & : (<(ABS(? ?h1 ?h2)) 1.00))
  THEN
    ("Grado de Severidad: 3"),
    ("Avería Asociada: Fallo en Canal2 del Módulo de transmisión,
      estación " ?remota),
    ("Recomendación: Revisar el transmisor CTR190/7.");
REGISTERED AS {rule 1 3 6 9};
```

5.11 Identificación de una Nueva Regla de Gestión.

Con la integración del conocimiento de gestión en la propuesta GDMO+, la definición de las reglas pasará a ser dependiente de las especificaciones de los objetos gestionados. Esto puede originar problemas de normalización de las reglas aplicadas, que habrá que solucionar ideando nuevas estrategias.

El CCITT Rec. X.208 | ISO/IEC 8824 [ISO88], proporciona una estructura para los identificadores de objeto y los valores de los distintos nodos iniciales, de los cuales dependerán los nuevos identificadores que se definan. Para obtener más información acerca de los mecanismos de registros establecidos y las autoridades registradores, puede consultarse la recomendación X.660 de CCITT | ISO/IEC 9834-1[ISO96]**.**

Indicar que una vez que el organismo correspondiente decide asignar un valor identificador objeto a un elemento de información de gestión, es necesario que cualquier revisión que se haga de la definición del objeto gestionado no altere la semántica de la información. En la práctica, esto significa que se pueden realizar cambios de edición en la definición de la información de gestión registrada, pero no aceptará cambios que puedan afectar a la información desde el punto de vista del protocolo, esto será extensible también para la propiedad RULES [ISO92A].

Para el estándar GDMO Extendido propuesto en la tesis, deberemos tener en cuenta todos los aspectos relacionados con la nueva propiedad denominada *"RULES"*, así como con su plantilla asociada *"RULE"*.

Para que esto sea posible, las **Funciones de Gestión de Sistemas, funtion(2)** y el **Modelo de Información de Gestión, smi(3),** deben soportar una ampliación para que estas recomendaciones recojan las funciones y mecanismos de definición que permitan contener la nueva propiedad RULES. Para ello se proponen nuevas versiones de dichas recomendaciones.

Tal y como se puede observar en las figuras siguientes, el nodo etiquetado como **rule(11)** reunirá los elementos necesarios para definir el conocimiento asociado al sistema experto en:

Como el resto de propiedades del estándar GDMO+ que se incluyen dentro de la definición de una Clase de Objeto Gestionado: clases, atributos, acciones, etc., las reglas expertas de gestión estarán sujeto a y mecanismos de registro y criterios de revisión análogos. En la siguiente figura se muestra la ampliación que experimentarían los nodos funtion(2) y Smi(3).

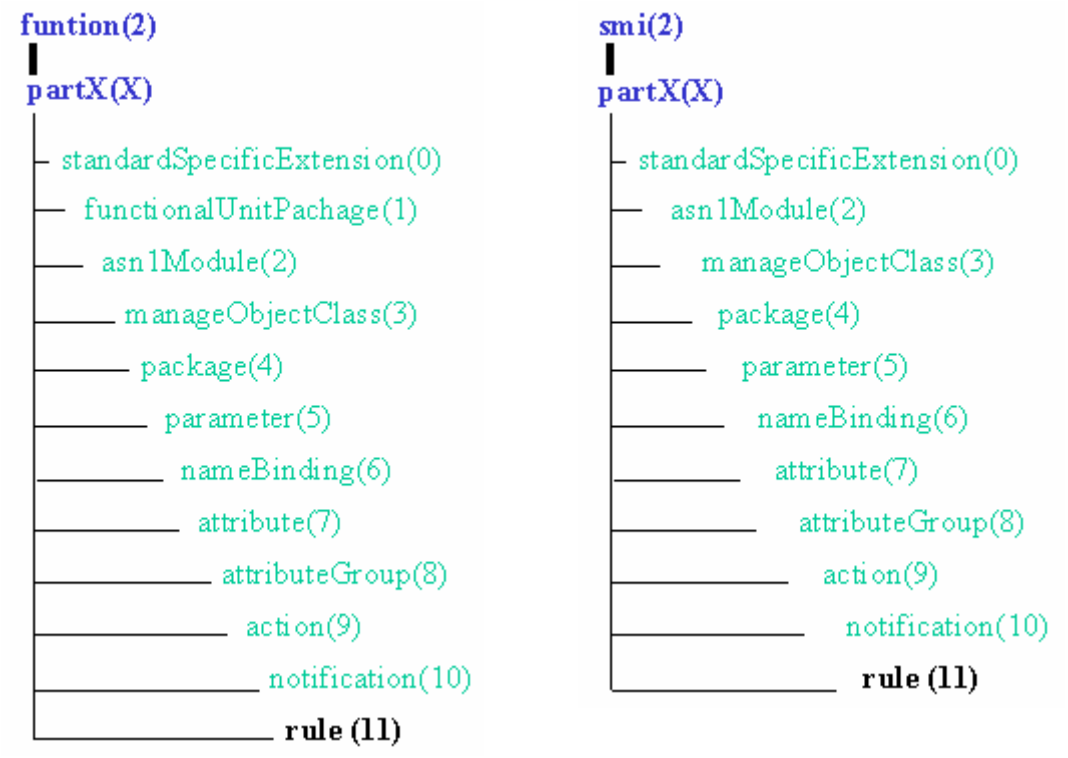

Figura 5.29 Ampliación del Modelo de Información y las Funciones de Gestión.

En este caso para añadir una nueva regla de gestión en la propuesta del estándar GDMO Extendido, se haría en el nivel siguiente:

{join-iso-ccitt ms(9) smi(9) part4(4) rule(11) actualRule(45)}

De esta forma, la definición de esta regla "*actual rule"* quedaría incluida en el lugar indicado de la recomendación X.720 de CCITT | ISO 10165-1 [ISO92B].

Capítulo 6

Aplicación del Estándar GDMO+ a una Red de Telecomunicaciones Privada del Sector Eléctrico.

En este capítulo se elabora una aplicación práctica de la proposición de norma GDMO+. Se procede a la integración de un sistema experto dentro de las especificaciones del Sistema de información administrado. Para ello, se utiliza el conjunto de plantillas y la metodología estudiada en los capítulos anteriores de esta tesis. Se procede al estudio y modelado de un Sistema Experto de gestión, de modo que la base de conocimientos correspondiente se integra dentro de las especificaciones de los recursos gestionados, para ello utilizaremos las prescripciones planteadas en el estándar GDMO+.

El sistema de comunicaciones empleado para efectuar el paradigma de plataforma de gestión inteligente integrada pertenece a una Empresa Eléctrica, en particular la Compañía Sevillana Endesa (CSE). La administración y control actual de dicha red, recae sobre un Sistema experto desarrollado por el Departamento de Tecnología Electrónica de la Universidad de Sevilla [Leon99], denominado NOMOS. El conocimiento aportado por este sistema experto, servirá como base de integración en la notación GDMO+.

A continuación se hace una breve descripción de los elementos que forman parte del Sistema de Administración y Control, así como la filosofía del Sistema Experto empleado. Se describe el proceso de desarrollo e integración de ambos aspectos. Los resultados finales obtenidos se incluyen en el anexo C, que comprende el listado de especificaciones GDMO+ obtenidas. Contiene las definiciones correspondientes a los recursos de comunicaciones e incorporan el conjunto de reglas perteneciente a la base de conocimientos del Sistema Experto NOMOS.

6.1 Introducción.

La extensión y complejidad de operación de las redes de servicio, y en particular de las redes eléctricas, ha llevado a la generalización y perfeccionamiento de los sistemas de control que automatizan su explotación, siendo concebidos como elementos que mejoran la calidad del servicio, a la vez que disminuyen los costes de su administración.

El transporte y distribución de energía eléctrica desde los puntos de generación hasta los de consumo, se basa en un conjunto de técnicas en constante evolución. Las líneas únicas, han ido dejando paso a una situación en la que se garantiza el suministro de la energía a un consumidor a través de rutas alternativas, multienlazadas entre sí, a fin de garantizar la continuidad del servicio.

La Compañía Sevillana Endesa, dispone de una red de control distribuido denominada *Sistema de Supervisión de Comunicaciones* (SSC). Este sistema realiza la adquisición de grandes volúmenes de información, efectúa su tratamiento en el centro de supervisión y control, actuando en tiempo real sobre el proceso en cuestión. Puede disponer a su vez de otros módulos de software para la resolución de problemas específicos, como pueden ser el cálculo de rendimientos, cálculo de consumos, etc.

Todo esto, provoca una notable complejidad en la planificación, gestión y explotación de la red eléctrica y requiere de una mayor supervisión y control de los sistemas de información. La Gestión Experta Integrada, dispone de los medios necesarios para realizar la administración de una forma óptima y eficiente.

6.2 El Sistema de Supervisión de Comunicaciones de la Red de Radioenlaces.

Se trata de un sistema abierto que permite la integración de equipos de varios fabricantes, encargados de realizar operaciones específicas. Hace funciones de canalizador de todos los datos recogidos para, a través de líneas de comunicaciones de alta velocidad, ponerlos a disposición de los administradores y operadores.

El sistema está especialmente recomendado para llevar a cabo la supervisión en la red de información de la compañía eléctrica. Efectúa el control de los diferentes procesos que en ellas se desarrollan, permitiendo a los usuarios disponer de la información procedente de distintos puntos del Sistema de Información Gestionado.

A continuación se hace una breve exposición de los elementos que conforman el Sistema de telecomunicaciones en estudio.

6.2.1 Descripción del Sistema de Información.

La subred de Radioenlaces está constituida básicamente por equipos de transmisión/recepción digital vía radio en las bandas de UHF y SHF. Los equipos empleados son multiplexores digitales de tercero, segundo y primer orden. Estos últimos empleando la técnica MIC y un conjunto de equipos auxiliares instalados en una serie de 39 subestaciones.

La gestión de operación de los equipos de transmisión de la subred de radioenlaces, se realiza actualmente mediante el Sistema de Supervisión de Comunicaciones (SSC). Dicho sistema monitoriza, en tiempo real, los principales parámetros de la red con objeto de conseguir una operación óptima de la misma.

Los equipos de comunicaciones de la subred de radioenlaces, por sí solos tienen muy limitadas sus capacidades de gestión remota; la mayoría de estos equipos suministran, localmente, información de su estado tanto con indicadores luminosos, como con salidas digitales. Por ello, se ha requerido instalar un sistema SCADA[1], constituido por un centro de control, instalado en la planta ático del edificio central de la compañía y un conjunto de remotas (RTUs: "remote terminal units"), instaladas en las subestaciones repetidoras, que

[1] "Supervisory Control And Data Acquisition", Adquisición de Datos y Control de Supervisión . Se trata de una aplicación software especialmente diseñada para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos de campo (controladores autónomos, autómatas programables, etc.) y controlando el proceso de forma automática desde la pantalla del ordenador. Provee de toda la información que se genera en el proceso productivo a diversos usuarios.

registran los parámetros de funcionamiento de los distintos equipos y que envían dicha información al centro de control, por requerimiento de éste.

La comunicación entre el centro de control y las RTU's se realizan utilizando el protocolo TELETRANSA. El centro de control solicitará las medidas analógicas, estados digitales e incidencias registradas por las distintas estaciones remotas. Así mismo, las Unidades Remotas podrán enviar al centro de control mensajes de órdenes, útiles para la gestión de los distintos recursos.

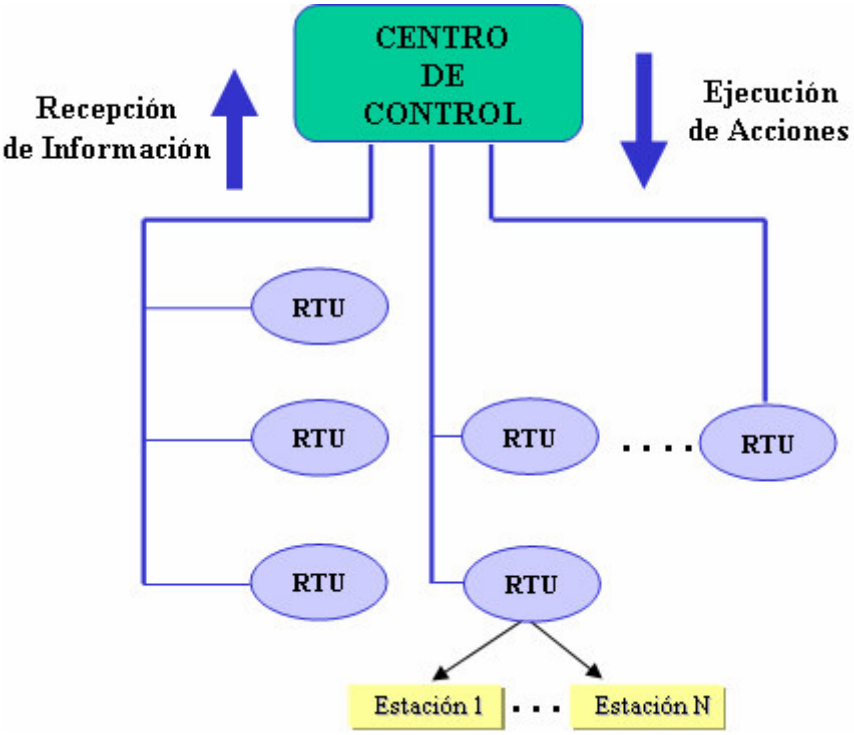

Figura 6.1 Jerarquía del Sistema de Control de Comunicaciones.

6.3 Subred de Radioenlaces. Equipos Digitales de Transmisión.

El carácter geográficamente disperso de la red, con gran número de subestaciones y centrales, obliga a mantener una red de recursos que aportan la información necesaria para ofrecer una administración eficiente.

En este apartado se presenta una descripción formal y las principales capacidades de gestión de los equipos de los tipos de objetos, detectados en la subred de Radioenlaces de la Compañía Sevillana Endesa (*CSE*).

La subred de radioenlaces de la CSE está constituida, fundamentalmente por equipos de transmisión digitales que responden a los distintos niveles de multiplexión de las recomendaciones del ***CCITT***, así como por equipos transceptores de radio en la banda de microondas. Forman también parte de la red algunos equipos de línea de fibra óptica, así como un conjunto de equipos auxiliares instalados en la mayoría de las subestaciones.

6.3.1 Equipos de Multiplexión.

Un multiplexor es una Unidad funcional que permite a varias fuentes de información utilizar simultáneamente medios comunes de transmisión, según criterios de frecuencia, tiempo y longitud de onda, asegurando en todo momento a cada fuente su propia vía independiente. La Recomendación G.703 del CCITT, prevé 4 niveles para el multiplexado temporal entre 64 Kbit/s y 140 Mbit/s.

Cada nivel, da nombre a los distintos sistemas de multiplexión. Se diferencian dos niveles: Multiplexores de primer orden y Multiplexores de orden superior.

• *Multiplexores de Primer Orden.* Realizan la multiplexión de 30 canales vocales o de datos (64 Kbit/s) en un canal de 2 Mbit/s. En el caso de la subred de radioenlaces analizada, esta función es realizada por tres equipos diferentes:

- Multiplexor de primer orden MP31.
- Multiplexor de primario ZAK 30/5.
- Multiplexor de primario ZAK D/I.

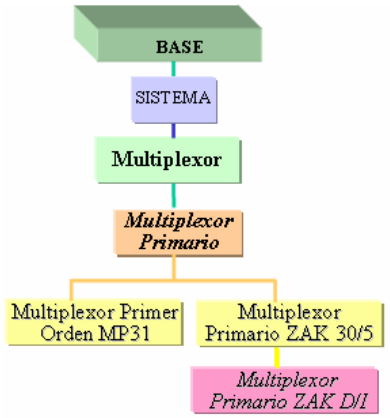

Figura 6.2 Mux. Orden Primario

• *Multiplexores de Orden Superior:* Los multiplexores de orden superior realizan, respectivamente, el multiplexado de 2 a 8 Mbit/s (segundo orden), de 8 a 34 Mbit/s (tercer orden) y de 34 a 140 Mbit/s (cuarto orden). En todos los casos, las señales afluentes (tributarias) se multiplexan por entrelazado cíclico de los elementos binarios, con justificación definida según las Recomendaciones G.742 y G.751 del CCITT. En este nivel tenemos los siguientes equipos:

- Multiplexor de segundo orden MP31/2.
- Multiplexor de segundo orden MDM 8/2.
- Multiplexor de tercer orden MP31/X.3.
- Multiplexor de tercer orden MDM 34/8.

La multiplexión de 2 a 34 Mbit/s, puede realizarse agrupando en un mismo bastidor 4 multiplexores MDM8 y un multiplexor MDM34.

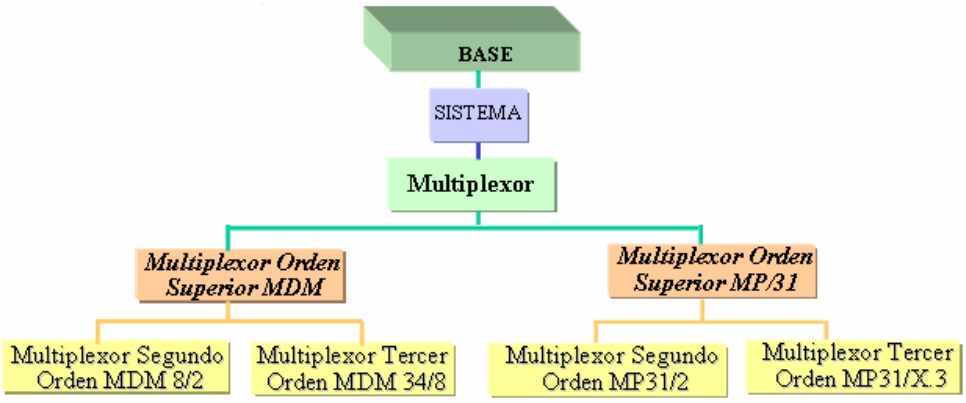

Figura 6.3 Multiplexores de orden superior.

Procedamos al estudio de cada uno de estos equipos.

6.3.1.1 Multiplexores de primer orden MP31.

De GTE Telecomunicazioni [GTE-81], el equipo Multiplexor MP31 representa la primera etapa de la jerarquía del multiplexor digital europeo. El multiplexor MP31 opera como unidad común que se combina con unidades adaptadoras de canales de datos o de voz, que adaptan el equipo al entorno de trabajo.

La principal forma de utilización del equipo MP31, es como multiplexor PCM con diversas variaciones de señalización. El MP31 puede transmitir los datos de señalización de la central, usando tanto la señalización asociada al canal como señalización por canal común. El equipo MP31 puede satisfacer flexiblemente necesidades de transmisión de datos muy distintas. Parte o todos los intervalos de tiempo que habitualmente se utilizan para transmisión de voz, se pueden reservar para datos. Asimismo, los bits libres del formato de trama del equipo, quedan disponibles para transmitir datos.

El equipo suministra señales digitales que permiten monitorizar determinadas anomalías en el funcionamiento del equipo. Recogemos aquellas que van a ser supervisadas por el SSC:

- Fallo de alimentación.
- Alarma externa de fonía.
- Alarma indirecta de fonía.
- Alarma interna de fonía.
- Tasa excesiva de errores de fonía.
- Pérdida de señal de 64K en el intervalo de tiempo 16.
- Fallo de sincronización.
- Alarma externa o interna de señalización.
- Alarma indirecta de señalización.

6.3.1.2 Multiplexores Primario ZAK 30/5.

De Ericsson [ERIC-91], las características funcionales de este multiplexor primario son similares a las del MP31, si bien, proporciona la monitorización de mayor número de indicadores de estados. A las alarmas del apartado anterior, hay que añadir las siguientes:

- Pérdida de la señal tributaria entrante de 2 Mbit/s.
- Tasa de errores excesiva (BER > 1E-3).
- Fallo interno.
- Pérdida de alineación de trama.
- Recepción de señal indicadora de alarma SIA.
- Fallo de alineamiento de multitrama.
- Fallo de alimentación.

6.3.1.3 Multiplexores de primer orden ZAK D/I.

Tipo de objeto de que deriva del Multiplexor primario ZAK 30/5, de Ericsson. [ERIC-91] y que posee sus mimas características.

6.3.1.4 Multiplexor de Segundo Orden MP31/2.

De Ericsson [ERIC-85], este equipo corresponde a la segunda etapa del multiplexor digital europeo. Multiplexa un máximo de 4 señales tributarias de 2 Mbit/s, en una señal de 8 Mbit/s.

Entre los fallos monitorizados por el multiplexor MP31/2 se encuentran los siguientes:

- Indicación de alarmas externas.
- Indicación de alarmas indirectas.
- Fallo en los tributarios.
- Estado de la alimentación.

6.3.1.5 Multiplexor de Tercer Orden MP31/X.3.

De Ericsson [ERIC-85], las características funcionales y las capacidades de gestión del multiplexor de tercer orden MP31/X.3, son básicamente las mismas que las del MP31/2. A diferencia de que éstas corresponden al tercer nivel de la jerarquía del multiplexor europeo. Multiplexa un máximo de 4 señales de 8 Mbit/s, en una señal principal de 34 Mbit/s.

6.3.1.6 Multiplexor de segundo orden MDM 8/2 y de Tercer orden MDM 34/8.

Los equipos MDM 8/2 y MDM 34/8, son respectivamente multiplexores digitales de segundo y tercer orden, con características funcionales similares a los expuestos anteriormente.

Entre los fallos monitorizados por estos equipos se observan los siguientes:

- Fallo de alineación de trama.
- Tasa excesiva de errores (BER >1E-3).
- Recepción de SIA.
- Pérdida de señal tributaria entrante.
- Fallo por desbordamiento de memoria.
- Fallo de alimentación local.

6.3.2 Equipos de Línea de Fibra Óptica.

Gestionados por el sistema de supervisión de comunicaciones de la CE. Los equipos terminales de línea óptica forman parte de la subred de radioenlaces.

- *Equipo de Línea de Fibra Óptica ETLO-34*: Es un repetidor terminal de línea óptica, con una velocidad de transmisión del canal principal de 34 Mbit/s.

Las anomalías supervisadas por el SSC son las siguientes:

- Fallo de alimentación local.
- Fallo de alimentación remota.
- Fallo de degradación de láser.
- Pérdida de señal entrante.
- Pérdida de señal saliente.
- Tasa de errores > 1E-3.
- Tasa de errores > 1E-6.

6.3.3 Equipos Transceptores de Radio.

En la subred de radioenlaces de CSE están instalados dos tipos de equipos de radio: el Transceptor de Radio RT21 y el Transceptor de Radio CTR190.

6.3.3.1 Transceptor de Radio RT21.

El equipo RT21 es un transceptor digital de radio con arquitectura redundante. Incorpora dos módulos idénticos, pudiendo conmutar de uno a otro en caso de fallo. Suministra dos canales full-duplex para velocidad de transmisión de 2 Mbit/s, [TELE-85].

Los parámetros analógicos supervisados en el RT21, son los siguientes:

- Potencia de salida del canal 1.
- Potencia de entrada al canal 1.
- Potencia de salida del canal 2.
- Potencia de entrada al canal 2.

Así mismo, suministra información digital sobre las siguientes anomalías:

- Pérdida de la señal digital de entrada al canal 1.
- Pérdida de la señal de entrada al canal 2.
- Fallo del módulo de transmisión del canal 1.
- Fallo del módulo de recepción del canal 1.
- Fallo del módulo de transmisión del canal 2.
- Fallo del módulo de recepción del canal 2.
- Fallo de la fuente de alimentación del canal 1.
- Fallo de la fuente de alimentación del canal 2.
- Recepción de SIA en el canal 1.
- Recepción de SIA en el canal 2.
- Envío de SIA por el canal 1.
- Envío de SIA por el canal 2.
- Fallo en transmisión multiplexor o en generador sincronismos canal 1.
- Fallo en transmisión multiplexor o en generador sincronismos canal 2.
- Fallo del demultiplexor de recepción del canal 1.
- Fallo del demultiplexor de recepción del canal 2.
- Tasa de errores excesiva en el canal 1.
- Tasa de errores excesiva en el canal 2.
- Tasa de errores alta en el canal 1.
- Tasa de errores alta en el canal 2.
- Fallo del aleatorizador del canal 1.
- Fallo del aleatorizador del canal 2.
- Fallo del desaleatorizador del canal 1.
- Fallo del desaleatorizador del canal 2.

Con objeto de igualar el envejecimiento de los dos canales de radio, el equipo puede operar en modo automático realizando por sí solo, la conmutación periódica de los canales 1 y 2. No obstante, también ofrece la posibilidad de realizar manualmente dicha conmutación tanto en transmisión, como en recepción.

6.3.3.2 Transceptor de Radio RT190.

El equipo CRT190 es un transceptor digital de radio con arquitectura redundante, y de características funcionales similares al equipo anterior. Este equipo opera en colaboración con un módulo de procesamiento de señales como el CCA-33 o el CCA-34. La utilización de un módulo u otro dependerá del tipo y número de indicaciones que se quieran supervisar. Suministra dos canales full-duplex para velocidades de hasta 34 Mbit/s [GTE-86].

Los parámetros analógicos supervisados en el CTR190, son:

- Potencia de entrada al canal 1.
- Potencia de entrada al canal 2.

El equipo CTR190, suministra información digital sobre las siguientes anomalías:

- Fallo del módulo de transmisión del canal 1.
- Fallo del módulo de recepción del canal 1.
- Fallo del módulo de transmisión del canal 2.
- Fallo del módulo de recepción del canal 2.
- Fallo de la fuente de alimentación del canal 1.
- Fallo de la fuente de alimentación del canal 2.
- Fallo de inserción de bits.
- Fallo de extracción de bits.
- Tasa de errores excesiva en el canal 1.
- Tasa de errores excesiva en el canal 2.
- Tasa de errores alta en el canal 1.
- Tasa de errores alta en el canal 2.
- Fallo de alineamiento de trama en el canal 1.
- Fallo de alineamiento de trama en el canal 2.

El módulo CCA-34 permite también monitorizar:

```
- Indicador de recepción de SIA.
- Indicador de canal 1 fuera de servicio.
- Indicador de canal 2 fuera de servicio.
```

Por su parte, el módulo CCA-33 permite monitorizar:

```
- Indicador de recepción de SIA.
- Indicador de canal 1 fuera de servicio.
- Indicador de canal 2 fuera de servicio.
- Fallo de alimentación del canal 1.
- Fallo de alimentación del canal 2.
- Pérdida de datos a la entrada del canal 1.
- Pérdida de datos a la entrada del canal 2.
```

Es posible también forzar la conmutación manual del canal 1 o del canal 2.

6.3.4 Minilink 8 MBIT/S.

Los indicadores supervisados en este equipo son:

```
- Pérdida de la señal digital de salida.
- Pérdida de la señal digital entrante.
```

6.3.5 Equipos Auxiliares.

Además de los equipos presentados en apartados anteriores, en la subred de radioenlaces están presentes otros equipos, muy importantes para el correcto funcionamiento de la subred:

- Grupos electrógenos para alimentación autónoma.
- Rectificadores de corriente.
- Presurizadores.
- Torres de medición de parámetros meteorológicos.
- Detectores de puertas abiertas.
- etc.

6.4 Proceso de Desarrollo.

La compañía eléctrica CSE dispone en la actualidad de un Sistema Experto, NOMOS desarrollado por el Departamento de Tecnología Electrónica y dedicado a la gestión de la red de radioenlaces existente. Nuestro objetivo se centra en la integración de dicho Sistema Experto, en las especificaciones GDMO+ de los recursos existentes en la red de radioenlaces.

Como paso previo a la integración de las reglas expertas de gestión de la base de conocimiento, en las especificaciones GDMO+ (anexo C), en esta sección se procede a explica brevemente el proceso de construcción del Sistema Experto.

Para el alcance de los objetivos establecidos anteriormente, fue necesario definir un esquema de desarrollo que comprendía aspectos tales como la técnica de adquisición a emplear, el modo de representar el conocimiento adquirido y los patrones de verificación de resultados. Se empleó una filosofía clásica de diseño inspirada en la técnica de Prototipado Rápido a lo largo de cinco fases fundamentales: Familiarización con el Dominio, Adquisición del Conocimiento, Representación del Conocimiento, Programación y Evaluación.

Es muy importante destacar que el proceso de desarrollo no se realiza de modo completamente secuencial, puesto que la verificación de los resultados del sistema incide directamente en consultas al experto, lo que puede dar lugar a cambios posteriores para adecuar el programa a sus sugerencias.

En los siguientes apartados se presenta un resumen de cada uno de los pasos del proceso y se justifica las decisiones acerca del diseño tomadas en cada uno de ellos.

6.4.1 Dominio del Sistema y Adquisición del Conocimiento.

El primer paso en el desarrollo del sistema experto, fue la familiarización con el dominio sobre el que se aplica el sistema, por parte del ingeniero del conocimiento. Tiene como objetivo permitirle manejar la terminología y tener una visión general del dominio, comprendiendo los conceptos básicos asociados al mismo.

En el caso del Sistema Experto para el SSC de la subred de Radioenlaces, las vías para lograr un conocimiento adecuado del dominio fueron las siguientes:

- Los documentos existentes sobre el sistema suministrados por la CSE, que permiten fijar los conceptos fundamentales.

- La interacción con el experto, que posibilita aclarar las dudas surgidas sobre aspectos concretos del sistema a través de conversaciones directas y de la observación de la operación real del SSC.

Al finalizar esta etapa, el ingeniero del conocimiento poseía un conocimiento lo suficientemente amplio del sistema como para pasar al siguiente hito del proceso de desarrollo. En cualquier caso, el conocimiento del dominio se fue incrementando paralelamente al desarrollo del sistema, debido al continuo contacto con el experto.

6.4.2 Representación del Conocimiento.

A partir de la información contenida en los Resúmenes de Reuniones para la Adquisición de Conocimiento, se procede a representar la información de un modo formalizado. En nuestro caso se ha utilizado la representación basada en reglas.

Este tipo de representación tiene grandes ventajas, ofrece *transparencia*, ya que el conocimiento se almacena en forma explícita y sin ambigüedades, haciendo fácil su revisión posterior, *eficiencia*, *flexibilidad* y la posibilidad de una *inferencia directa*, ya que las reglas permiten un mecanismo de inferencia directo al aplicarse sobre las alarmas. El esquema de representación basado en reglas de producción, según se ha visto es del tipo:

SI-ENTONCES.

El conocimiento declarativo se almacena en forma de hechos y el conocimiento procedimental en forma de reglas. Los hechos coinciden en el caso del SSC con las alarmas que llegan procedentes de las unidades remotas y las reglas contienen la información suministrada por el experto, en relación con cada alarma o conjunto de alarmas.

Toda la información extraída de las reuniones con el Experto, se codifica en forma de reglas que siguen la siguiente estructura común:

```
SSR1 A/B
G.S: 3
   SI:
     A CTR190/7 TX CX B
     B CTR190/7 PRXX A < - 60
   ENTONCES:
     A.A: FALLO EN MODULO DE TRANSMISION CTR190/7, CANAL X, ESTACION A
     RC: REVISION DEL TRANSCEPTOR CTR190/7
```

- *Nombre:* que la identifica claramente del resto de reglas del sistema experto. En algunas casos el nombre incluye una letra mayúscula (p.e SSR1 A/B) indicando diversas variantes de una misma regla.

- *Antecedentes:* conjunto de alarmas que deben presentarse para que la regla pueda disponerse.

- *Consecuencias:* Grado de Severidad (GS), Avería Asociada (AA) y Recomendaciones de Actuación (RC), frente al conjunto de alarmas asociadas al antecedente.

Es importante destacar los siguientes aspectos relacionados con las reglas del Sistema Experto:

- Las reglas son muy genéricas, se pretende que sean válidas para una misma alarma o conjunto de alarmas, independientemente de la estación remota donde estas se produzcan.

- Las alarmas que aparecen en los antecedentes de una regla presentan una relación temporal en base a criterios de simultaneidad, de modo que varias alarmas no se consideran relacionadas, si existe un intervalo temporal entre ellas mayor de una cierta cantidad, que puede ser fijado a voluntad.

- El conjunto de reglas del Sistema Experto no es exhaustivo en cuanto a que no contempla todas y cada una de las posibilidades que pueden presentarse en el SSR. La decisión de no llevar a cabo un desarrollo exhaustivo del Sistema Experto se tomó de acuerdo con el experto, las razones para ello fueron la gran cantidad de tiempo necesario para realizarlo y el hecho de que un tratamiento exhaustivo de las alarmas no hubiese aportado ninguna ventaja cualitativa al sistema, que puede ser completado fácilmente, debido a la gran modularidad que permite la técnica de representación empleada.

- Existen un conjunto de reglas particulares, cuyo nombre tiene el formato SSRFILX, cuya misión es realizar el filtrado de aquellas alarmas que aparecen y desaparecen reiteradamente en intervalos temporales cortos. Estas reglas permiten reducir el conjunto de hechos sobre los que se aplican las reglas del Sistema Experto, mejorando la velocidad y disminuyendo la cantidad de información redundante que llega al operador. Estas reglas tendrán mayor prioridad que ninguna otra, además para que actúen sobre todos los recursos del sistema gestionado, se incluyen en las clases de objetos gestionados de nivel superior: *"Base"*, y se heredan a todas las demás clases.

La razón de ello es lograr que las reglas responsables del filtrado de alarmas (SSRFILX) actúen en primer lugar, eliminando los datos redundantes, permitiendo que el resto de las reglas se aplique únicamente sobre los hechos significativos y reduciendo

enormemente el tiempo de proceso, así como también la cantidad de información redundante o no relevante presentada al operador.

De modo que aparecen cuatro niveles distintos de prioridad:

- Grado de Prioridad 2 ó Muy Alta Prioridad, que corresponde a las reglas SSRFILX, que se incluyen en la clase de objeto gestionado *base* (Anexo C).

- Grado de Prioridad 0 ó Prioridad Media, que es la que se considera por defecto y se aplica a la mayoría de las reglas. Las clases SSRFIL1, SSRFIL2 y SSRFIL3 son ejemplos de este tipo.

- Grado de Prioridad -1 ó Prioridad Baja, que aparece en aquellas reglas que eliminan hechos (por ejemplo SSR11 A/B, SSR48 A/B ó SSR59) como consecuencia de su actuación, y a las que se fuerza a ejecutarse en último lugar para evitar que perturben a otras reglas.

- Grado de Prioridad -2 ó Muy Baja Prioridad, que únicamente aparece en la regla SSR12 C/D, para lograr controlar parcialmente el flujo del programa, en el caso particular de la interacción entre alarmas excluidas del filtrado.

6.4.3 Evaluación.

El proceso de evaluación de un sistema experto tiene una serie de particularidades, entre ellas la inexistencia de procedimientos normalizados, que hacen necesario llevar a cabo dicha tarea de un modo cuidadoso. En lo que respecta al sistema experto para la red de Radioenlaces, se diseñó un procedimiento de evaluación, que comprendía las etapas siguientes:

1.- Validación de la Base de Conocimiento. Esta etapa se desarrolló paralelamente al proceso de Adquisición del Conocimiento. Cada vez que se concluía un ciclo de entrevista con el experto y validación de la información obtenida a través de los Resúmenes de las Sesiones, se procedió a programar cada una de las reglas extraídas. Una vez implementadas, se hacía interactuar dichos grupos de reglas sobre conjuntos escogidos de alarmas del SSC, lo que permitió comprobar la funcionalidad de las mismas y las posibles interacciones no esperadas entre distintas reglas. Los resultados obtenidos eran comunicados al experto y, de acuerdo con el mismo, se realizaron los cambios requeridos.

Al finalizar esta etapa se disponía de un prototipo completo del sistema, cuya Base de Conocimiento había sido validada por el experto.

2.- Verificación del Prototipo Final. El objetivo de la verificación es lograr un prototipo funcionalmente correcto, haciendo especial énfasis en la velocidad del mismo, el proceso de filtrado y el examen de las interacciones no esperadas entre múltiples reglas. Se trató de un proceso más complejo que el descrito en la etapa anterior, por lo que se requirió del empleo de importantes facilidades de depuración, suministradas por el Shell ART-IM.

3.- Validación del Prototipo Final. Última fase de la evaluación y consiste en la realización de un Protocolo de Pruebas y valorar el comportamiento del Sistema experto frente a incidencias provocadas en la Subred de Radioenlaces.

6.5 Objetivos del Sistema de Gestión.

En los apartados previos hemos repasado distintos aspectos relacionados con el Sistema Experto NOMOS, que va a ser utilizado para el desarrollo de nuestro prototipo basado en la notación GDMO+. La filosofía cambia respecto a las plataformas de gestión tradicionales. En nuestro caso el sistema experto se distribuye entre los distintos elementos que constituyen la red de radioenlaces.

A la hora de abordar el desarrollo de la plataforma de gestión integrada, es esencial definir claramente cuáles son los objetivos que debe cubrir y qué filosofía marcará su funcionamiento. En el caso de la subred de Radioenlaces, se pretende solucionar o paliar los siguientes inconvenientes:

1.- La gran cantidad de información que llega al operador procedente del sistema de supervisión, que en el caso de avería grave puede comprometer el rendimiento del sistema.

2.- La necesidad de que el operador del sistema posea un alto grado de cualificación y experiencia. El Sistema de Supervisión no suministra información directa sobre las incidencias o averías que pueden ocasionar cada alarma o conjunto de alarmas.

La gestión inteligente integrada, debe actuar facilitando la labor del operador que se encuentra en el Centro de Control, de modo que a partir de la información proveniente del control, realice una serie de tareas que permitan paliar los siguientes problemas:

- Filtrar alarmas redundantes que pueden saturar la capacidad del operador, en el caso del SSC se eliminan alarmas que aparecen y desaparecen reiteradamente en un corto intervalo de tiempo. Al centro de control tan sólo llega la información útil respecto a la alarma y las acciones a realizar por el operador, si así se requiere.

- Detectar las posibles incidencias o averías ocurridas en el sistema a partir de la información sobre las alarmas presentes en cada momento. En el caso de no necesitar la intervención de éste, el propio sistema efectuaría las acciones de gestión asociadas.

- Realizar una priorización de las incidencias o averías a través de la asignación a cada una de ellas de un determinado Grado de Severidad, establecido por el experto.

- Presentar al operador una serie de sugerencias de actuación, que le permitan corregir las incidencias o averías y mejorar la operación del sistema.

- Todos los objetivos anteriores han de ser cubiertos en un tiempo razonable, para permitir la operación en tiempo real.

La plataforma de gestión efectúa el análisis de los eventos producidos y realiza las operaciones de gestión necesarias para la resolución de la incidencia. Asimismo filtra los eventos y alarmas que no requieren acciones, eliminando información ineficaz para la gestión y evitando la sobrecarga del sistema gestionado.

6.6 Árbol de Contención.

El árbol de contención recoge la relación definida entre objetos gestionados, denominada contención. El modelo de gestión ISO permite la localización de los objetos gestionados mediante una estructura jerárquica de éstos. Esta estructura jerárquica se denomina Árbol de Información de Gestión *(Management Information Tree, MIT*), o árbol de contención y permite que los tipos inferiores "hereden" las características de los tipos superiores (ingeniería del software orientada a objetos).

La relación de contenencia no indica un tipo de objeto, sino en qué otros objetos está incluido en un objeto. No implica una contención física de unos recursos en otros, simplemente se utiliza para identificar de forma inequívoca la posición de un objeto gestionado dentro del conjunto de todos los que se hayan definido, es decir de qué otros objetos depende.

Esta relación de contención tiene un doble significado:

- Un objeto puede contener a otro objeto.

- Un objeto contenedor (o superior) puede, alternativamente, estar contenido en otro objeto gestionado.

De forma general, un objeto superior puede contener más de un objeto, pero un objeto contenido (o subordinado), puede ser contenido solamente en un objeto superior al mismo tiempo. Esta restricción fuerza una estructura arborescente en la jerarquía.

Las relaciones de contención definidas para objeto gestionado, se realizan a través de las plantillas *Name Bindings* disponibles en el estándar GDMO y en su extensión propuesta GDMO+. Esta plantilla se utiliza para denotar con un nombre de objeto a los distintos recursos gestionados, según hemos visto en los capítulos 4 y 5 de la tesis.

A continuación en la figura 6.4 se muestra el árbol de la contención de la red de telecomunicaciones en estudio. Tal y como se aprecia, recoge todas las relaciones de contención existentes entre los distintos recursos que componen dicha red.

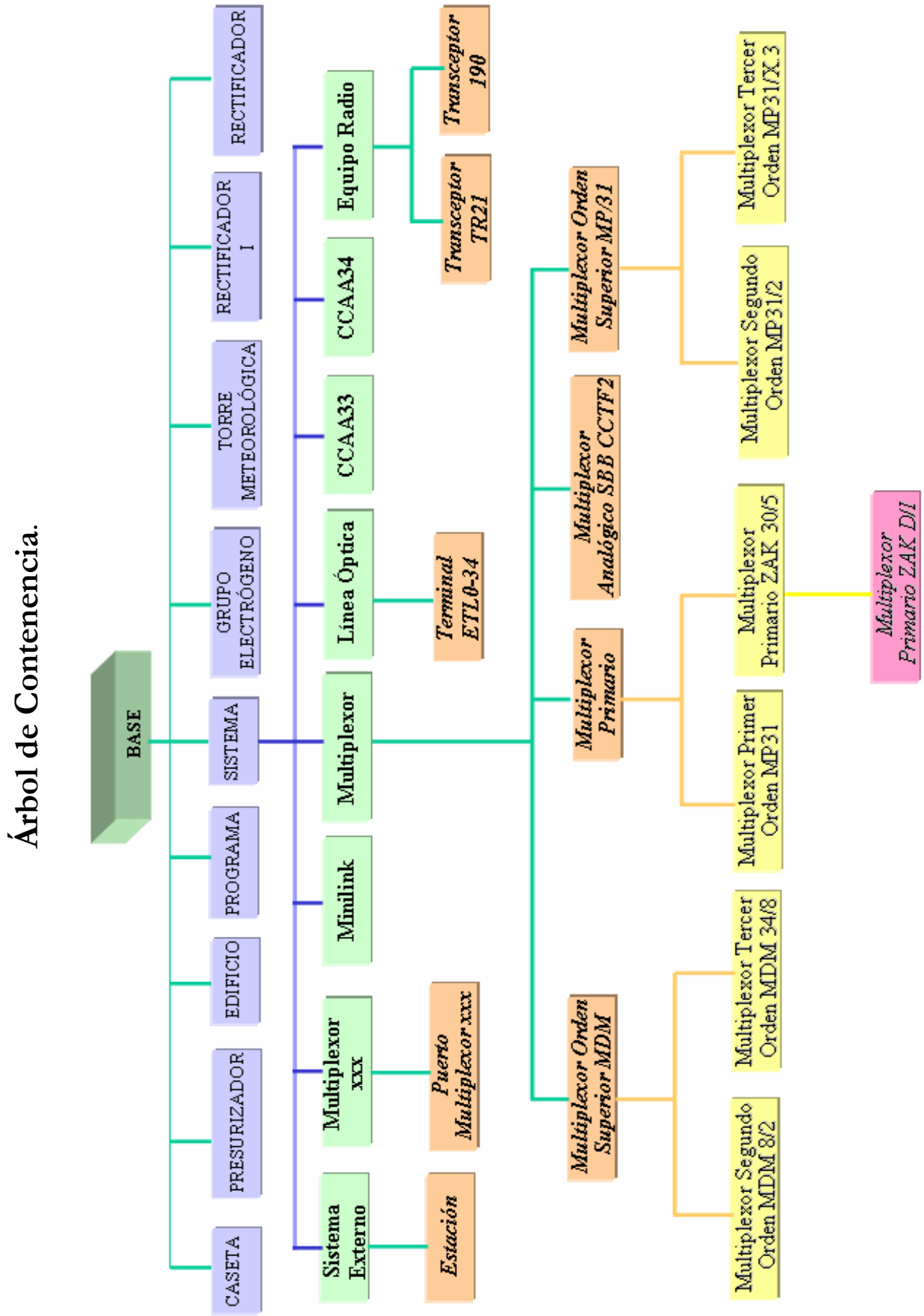

Figura 6.4. Árbol de Contención.

Al árbol de contenencia de la figura anterior, se añade un nuevo tipo de objeto, que surge de la unión de dos tipos de objetos existentes. Corresponden a las *Clases de Objetos Gestionados Virtuales* estudiadas en el apartado 5.7.9 del capítulo 5. Estos tipos de objetos, son necesarios para la fusión de las propiedades procedentes de dos o más clases de objetos gestionados. Se utilizan para la incorporación de reglas expertas de gestión, que requiere el tratamiento simultáneo de aspectos relacionados con más de una clase de objeto gestionado.

En la siguiente figura aparecen tres nuevos tipos de objetos que atienden a este arquetipo y que son necesarios para la construcción del sistema de gestión integrado correspondiente a la red de telecomunicaciones de CSE en estudio. Las clases a las que hacemos referencia son: *CTR190_MP31, CTR190_CCAA33/34 y CTR190_CCAA33/34.*

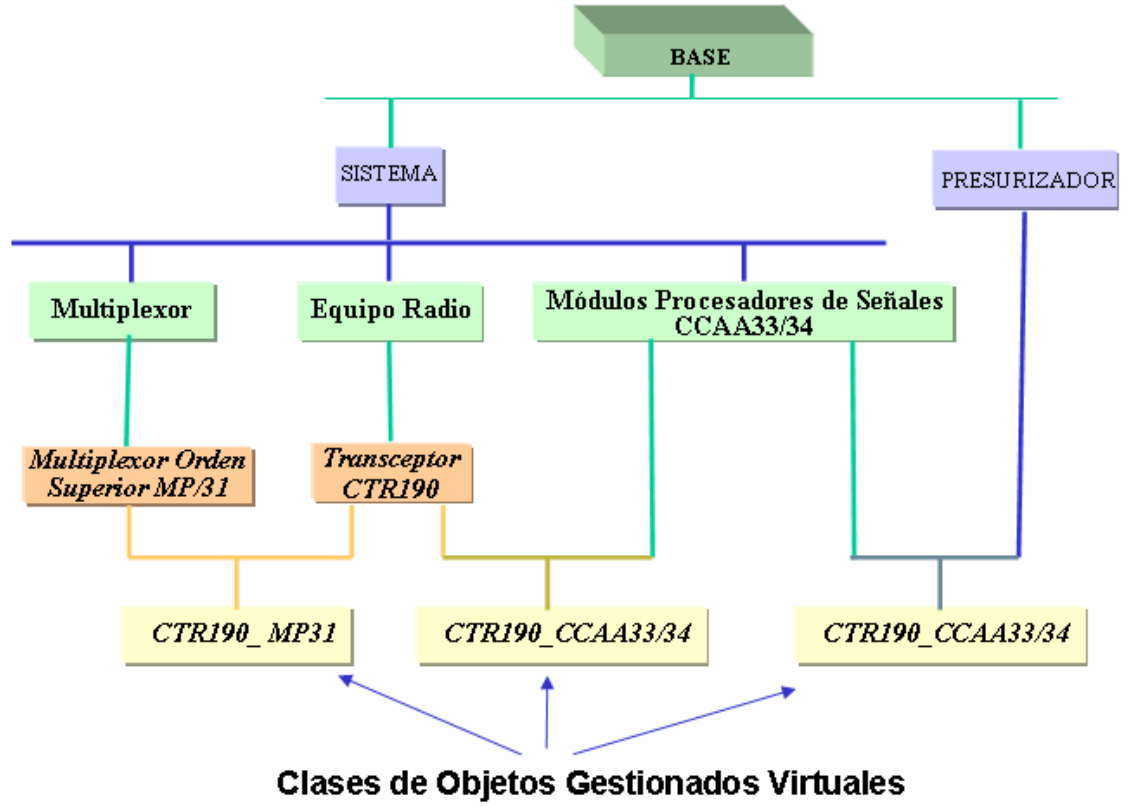

Figura 6.5. Árbol de Herencia de las clases de objetos gestionados Virtuales.

6.7 Prototipo de Gestión Utilizando la Notación GDMO+.

La gestión de alarmas es un aspecto muy importante a contemplar dentro de la administración y control de una red de Telecomunicaciones. Especialmente si ocurre como en nuestro caso, que dicha red se compone de un conjunto muy numeroso y heterogéneo de subredes que han de interoperar entre ellas de una forma fiable y eficiente.

La recogida y análisis de fallos en tiempo real puede influir de forma notable en la operatividad de la red. Gracias a la detección en tiempo real, se facilita la resolución de fallos y averías, a partir del momento en que éstas se producen. Como consecuencia de este proyecto, se realiza una gestión eficiente de las alarmas del sistema de radioenlaces de CSE, figura 6.6.

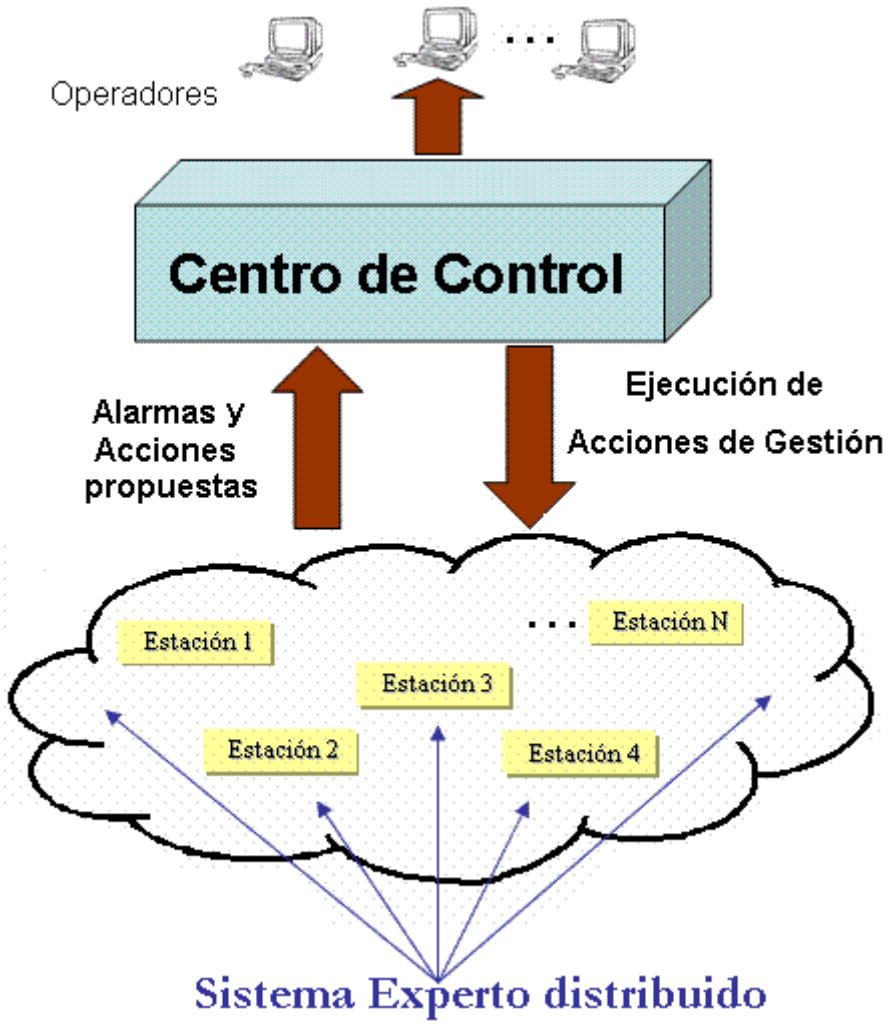

Figura 6.6 Sistema Experto Integrado para el SSC.

Se implementa una herramienta que permite monitorizar el estado de los componentes de una red a través de la observación de las alarmas que éstos emiten. Se emplea la información de interés, independientemente del protocolo o formato que se utilice para la emisión de las alarmas, ofreciendo una interfaz homogénea al operador. El prototipo desarrollado contiene además funciones complementarias, que posibilitan o facilitan la consecución óptima de los objetivos de administración y control.

Las especificaciones se efectúan utilizando las bases del estándar GDMO+ propuesto. La herramienta se implementa utilizando ART*Enterprise (Anexo D), un entorno de desarrollo integrado C++, que contiene una serie de funcionalidades avanzadas para la construcción de sistemas expertos. En los siguientes apartados se describe la estructura y los principales elementos que constituyen el prototipo, poniendo especial interés en los aspectos relacionados con la base de conocimiento.

6.7.1 Implementación.

La mayoría de tráfico interurbano de CSE, se controla con un sistema de comunicaciones constituido por radioenlaces, según se ha visto en los apartados anteriores del capítulo. La figura 6.5 ofrece una visión general.

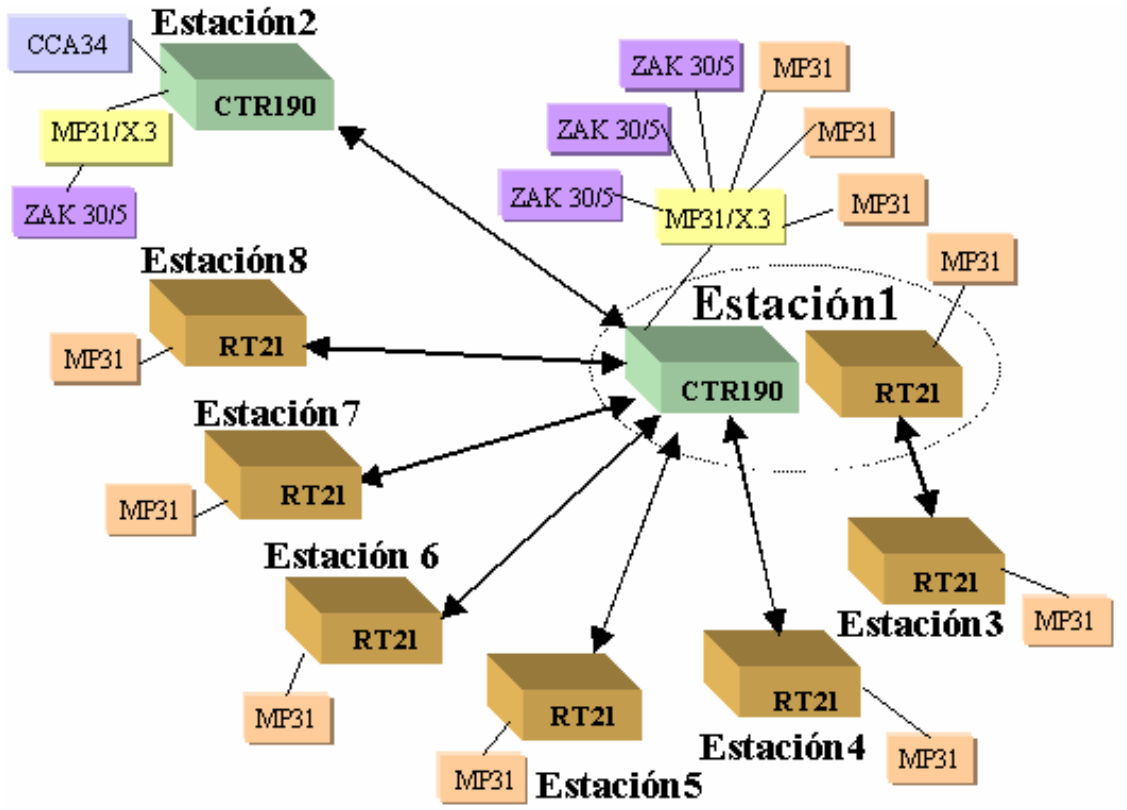

Fig. 6.7 Entorno de pruebas y arquitectura física de la red de gestión

En la actualidad la red se gestiona mediante el sistema experto NOMOS, que forma parte de la plataforma de gestión que lo administra. Nosotros proporcionamos una nueva versión denominada NOMOS+, que ofrece una visión distinta del sistema de gestión existente. NOMOS+, es el primer sistema experto integrado dentro de las especificaciones de los objetos gestionados y utiliza para ello la norma GDMO+ propuesta.

Las guías propuestas en GDMO+, permiten incluir todas las reglas expertas dentro de las definiciones de los recursos gestionados de la red de radioenlaces. El Anexo C, contiene listado completo de las especificaciones GDMO+ de nuestro prototipo. Dichas especificaciones, contienen la información correspondiente a los recursos que forman la red de telecomunicaciones de radioenlaces y el conocimiento de gestión perteneciente al sistema experto NOMOS que lo administra.

6.7.2 Arquitectura del sistema.

Describe las ideas sobre las que se desarrolla una arquitectura distribuida para la gestión de alarmas y define los servicios y elementos funcionales para llevarla a cabo.

La herramienta desarrollada presenta una estructura típica de los sistemas expertos, que se compone de tres elementos importantes: *La base de conocimiento,* que representa el conocimiento almacenado sobre el dominio del problema. *El motor de inferencia,* unidad procesadora que resuelve los problemas detectados en la red; hace uso de la inferencia lógica de los hechos de la lista de hechos y las reglas de la base de conocimiento. *La interfaz de usuario,* que controla el motor de inferencia y los mensajes de entrada/salida que puede ofrecer a los usuarios del sistema, incorpora un módulo que permite interoperar de forma fácil con la base de reglas del sistema.

Nuestro prototipo cuenta además con un módulo preprocesador. Esta etapa previa a la inferencia, está formada por una unidad procesadora-traductora a la que se aplica como

entrada un fichero que contiene las especificaciones GDMO+ procedentes del sistema gestionado. El Preprocesador efectúa el reconocimiento del contenido del fichero, y extrae el conocimiento aportado por el sistema experto. Es decir separa las reglas expertas de gestión, de las definiciones de las clases de objetos gestionados GDMO+, Figura 6.8.

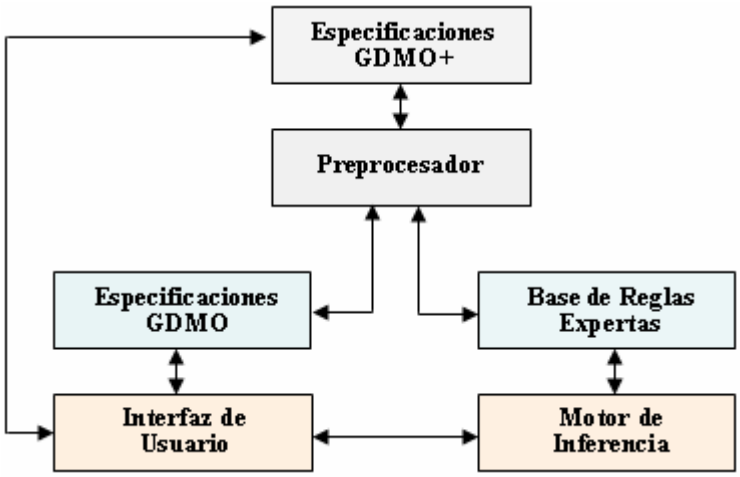

Fig. 6.8 Diagrama de Bloques del Sistema

Realicemos un estudio más detallado de cada uno de los componentes que componen el sistema de gestión.

6.7.2.1 El Módulo Preprocesador.

Actúa en la etapa inicial de proceso, realiza un tratamiento preparatorio de las especificaciones GDMO+. Analiza el fichero fuente que contiene las definiciones de los objetos gestionados y el conocimiento de gestión integrado. Para la realización de dichas definiciones se utiliza el conjunto de plantillas y pautas explicadas en el estándar GDMO+ propuesto, estudiado en el capítulo 5 de la tesis.

El un primer momento el preprocesador separa las especificaciones puramente GDMO, parte que corresponde a la definición de las clases de objetos gestionados, de las reglas expertas de gestión integradas, Figura 6.9.

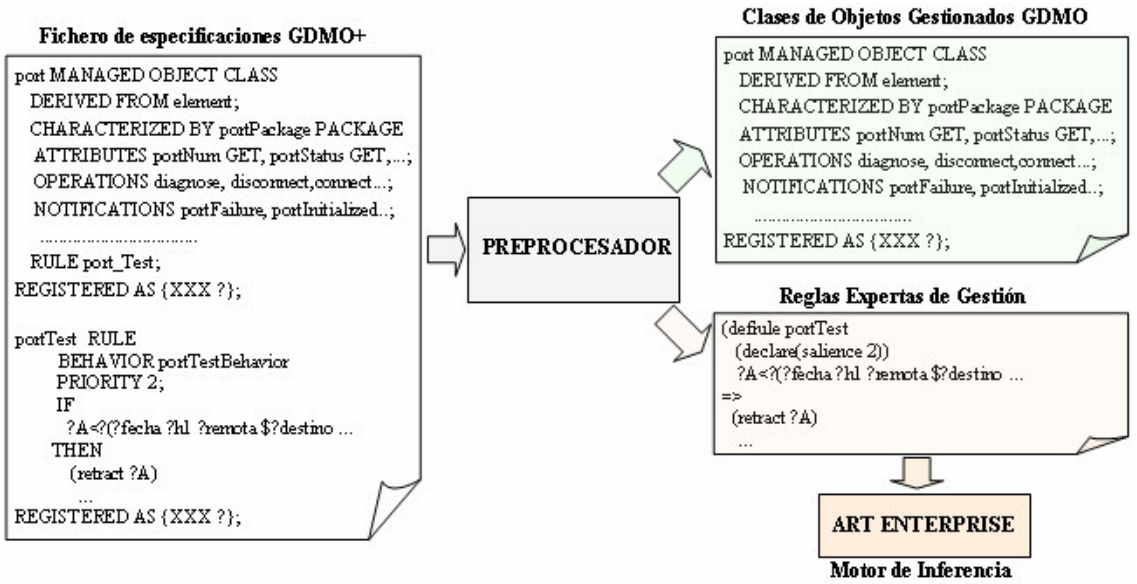

Fig. 6.9 Función del Preporcesador

Las reglas expertas extraídas no tienen la sintaxis apropiada para ajustarse al lenguaje nativo del motor de inferencia, en nuestro caso el ARTScript de Art Enterprise. Por tanto hay que traducir toda la base de reglas, para adecuarla al formato correcto. Véase el siguiente ejemplo:

DEFINICIÓN DE REGLA EXPERTA EN GDMO+

```
errorTransmision RULE
   BEHAVIOR errorTrasnmisionBehavior;
   PRIORITY 3;
   IF
      (?fecha ?h1 ?remota CTR190/7_TX_C1 ?destino ALARMA)
      (?fecha ?h2 ?destino CTR190/7_RX_C1 ?remota ALARMA  & : (<(ABS(- ?h1 ?h2)) 1.00))
   THEN
      ("Grado de Severidad: 3" )
      ("Avería: FALLO EN CANAL 1 DEL MODULO DE TRANSMISION, ESTACION " ?remota)
      ("Recomendación: REVISAR EL TRANSMISOR CTR190/7."  )
REGISTERED AS {nm-rule FIL2};
```

DEFINICIÓN DE REGLA EN SINTAXIS ARTSCRIPT DE ART ENTERPRISE

```
(defrule errorTransmision
   (salience 3)
   (?fecha ?h1 ?remota CTR190/7_TX_C1 ?destino ALARMA)
   (?fecha ?h2 ?destino CTR190/7_RX_C1 ?remota ALARMA  & : (<(ABS(- ?h1 ?h2)) 1.00))
=>
   (printout t "Grado de Severidad: 3" t)
   (printout t "Avería: FALLO EN CANAL 1 DEL MODULO DE TRANSMISION, ESTACION " ?remota t)
   (printout t "Recomendación: REVISAR EL TRANSMISOR CTR190/7." t))
```

Tal y como se observa, existen diferencias sustanciales entre la definición de la regla expresada en sintaxis GDMO+ y su réplica en ARTScript. El preprocesador es el encargado de realizar la trascripción de las reglas expertas de una sintaxis a otra.

El Módulo preprocesador se ha desarrollado utilizando el lenguaje de programación PHP. En una pasada recoge el contenido del fichero que contiene las definiciones y genera dos salidas: un primer fichero que contiene las Especificaciones GDMO de las clases de objetos gestionados y un segundo fichero que contiene el conocimiento correspondiente al Sistema Experto, que servirá como entrada al motor de inferencia, Figura 6.9.

Desde el punto de vista del administrador, la parte más interesante del comportamiento del preprocesador, es que está gobernado por una serie de pautas. Directivas que se siguen para reconocer el contenido del fichero con las especificaciones y realizar la separación de los tipos de información, además de realizar la traducción del conocimiento al lenguaje de programación utilizado. Por tanto estas pautas serán dependientes de la herramienta de desarrollo de sistemas expertos utilizada, en este caso ART*Enterprise 3.0.1.

Una vez obtenido el código fuente (preprocesado), éste se traslada al resto de las etapas del prototipo, en particular a la unidad de Inferencia de Art*Enterprise para su procesado.

Como se ha comentado el preprocesado del fichero inicial genera dos ficheros de salida en el directorio de trabajo. Estos ficheros tienen el mismo nombre que el fichero fuente, pero con extensiones serán: "***gdmo***" y "***art***". Sea por ejemplo el fichero "*ejemplo.ext*" que se utiliza como entrada en el preprocesador, se generan las salidas "*ejemplo.gdmo*" que contiene las especificaciones GDMO de las clases de objetos gestionados y "*ejemplo.art*" que contiene la base de reglas en código ARTScipt.

6.7.2.2 La Base de Conocimiento.

En el apartado anterior se ha visto como obtener la base de reglas, una colección de reglas expertas de gestión, que representa el conocimiento almacenado acerca del dominio de gestión, en concreto de red de radioenlaces de la compañía CSE. La base de conocimiento de nuestro sistema experto está compuesta aproximadamente por unas cien reglas expertas de gestión codificadas en ARTScript, el lenguaje de programación utilizado por ART*Enterprise. ARTScript posee las características apropiadas para permitir a un usuario neófito en el tema, codificar el conocimiento de forma fácil, sin necesidad de poseer conocimientos avanzados y utilizar mecanismos complicados de programación.

La base de conocimiento contiene información estática y dinámica, el conocimiento acerca de los protocolos y errores comunes que se pueden producir en la red. Por ejemplo, un error o caída de servicio producido en la conexión existente entre dos transceptores de radio. Para integrar en la MIB, dicha base de conocimiento, se utiliza la nueva propiedad "RULES" y su plantilla RULE de GDMO+.

El siguiente ejemplo muestra las especificaciones GDMO+ de una clase de objeto gestionado y las reglas expertas de gestión que contiene. Define un objeto tipo transceptor de radio CTR190 perteneciente a la red de radioenlaces y las reglas expertas *falloElectrico* y *errorTransmision*.

```
radioTrasnceptorCTR190 MANAGED OBJECT CLASS
        DERIVED FROM "rec.X721":top;
        CHARACTERIZED BY radioTransceptorCTR190Package;
REGISTERED AS {nm-MobjectClass 1};

radioTransceptorCTR190Package PACKAGE
      ATTRIBUTES
          reception Power     GET,
          sense                GET,
          speedTransmission GET,
          …
    NOTIFICATIONS
          damageFeeding,
          inferiorLimit,
          repairAction;
    RULE
          falloElectrico,
          errorTransmision;
REGISTERED AS {nm-package 1};


falloElectrico  RULE
 PRIORITY 3;
 BEHAVIOR falloElectricoBehavior;
 IF
   (?fecha ? ?Estacion1 CTR190/7_F_ALIM_2 ?Estacion2 ALARMA)
   (NOT(?fecha ? ?Estacion1 CCA?34_AIS_DE_BB ?Estacion2 ALARMA))
 THEN
   ("Severidad:" 3) ,
   ("Diagnóstico: Avería en Fuente de Alimentación 2 de CTR190, Estación:" ?estacion1"),
   ("o avería en alimentación de fuente 2"),
   ("Recomendación: Revisar la Fuente y/o Alimentación");
REGISTERED AS {nm-rule 1};
```

```
errorTransmision  RULE
 PRIORITY 4;
 BEHAVIOR errorTransmisionBehavior;
 IF
  (?fecha ?hora1 ?Estacion1 CTR190/7_TX_C2 ?Estacion2 ALARMA)
  (?fecha ?hora2 ?Estacion2 CTR190/7_RX_C2 ?Estacion1 ALARMA
     & : (<(ABS(? ?hora1 ?hora2)) 1.00))
THEN
    ("Severidad:" 4) ,
    ("Diagnóstico: Fallo en Canal2 del Módulo de Transmisión, estación:" ?estacion1),
    ("Recomendación:" Revisar el transmisor CTR190/7");
REGISTERED AS {nm-rule 2);
```

La clase de objeto gestionado *radioTransceptorCTR190*, incluye el paquete obligatorio *radioTransceptorCTR190Package* que contiene todas las propiedades procedentes del recurso real representado, el transceptor de radio modelo CTR190. Observar las dos propiedades que se definen mediante la cláusula RULES: *falloElectrico* y *errorTransmision,* dos reglas expertas de gestión exclusivas del recurso transceptor de radio CTR190. Estas reglas representan el conocimiento de gestión definido para este tipo de recurso y se añaden mediante la propiedad RULES, que aparece en la *Managed Object Class.* Asociada a cada propiedad RULES, aparece las definición de regla expertas de gestión, realizada con la plantilla RULE: *falloElectrico RULE y errorTransmision RULE*.

Cuando se producen alarmas en la red de enlaces, el sistema experto integrado realiza un estudio de los eventos producidos. Después de un análisis realiza las operaciones de gestión apropiadas. El fin de las reglas expertas anteriores, es la detección de las anomalías o fallos producidos en el transceptor, a la vez que ofrecen las acciones destinadas a resolver las incidencias asociados a las alarmas. La primera regla *falloElectrico*, detecta fallos en la alimentación eléctrica del dispositivo CRT190. La segunda regla *errorTransmision*, se dedica a la detección de errores en el módulo de transmisión de datos del transceptor CTR190.

A continuación, un ejemplo de las alarmas que se producen en el enlace existente entre dos transceptores de radio CTR190, estación1 y estación5. Se observan las respuestas emitidas por el sistema de gestión, en este caso el sistema experto integrado.

Alarmas generadas cuando ocurre alguna incidencia:

F1 (31/01 1115.1836 estacion1 CTR190/7_TX_C2 estacion5 ALARM)
F2 (31/01 1117.2134 estacion5 CTR190/7_RX_C2 estacion1 ALARM)
F3 (31/01 1114.0002 estacion1 CTR190/7_F_ALIM_2 estacion5 ALARM)
…

El sistema de gestión realiza el análisis siguiente:

FIRE 1: errorTrasmisión f-1f-2
Severidad: 4 ,
Diagnóstico: Fallo en Canal2 del Módulo de Transmisión, estación: estacion5
Recomendación: revisar el transmisor CTR190/7

FIRE 2: falloElectrico f-3
Severidad: 3
Diagnóstico: Avería en Fuente de Alimentación 2 de CTR190, Estación: estacion5
o avería en alimentación de fuente 2
Recomendación: Revisar la Fuente y/o Alimentación

2 rules fired.
Run time is 0.074 seconds, 27.0270 Rules/Sec.

Cuando se cumplen las condiciones del antecedente de la regla, ésta se incorpora en la agenda del sistema a la espera de su disparo posterior. La regla experta ejecuta las operaciones de gestión *falloElectrico* y *errorTrasmision*, que tratan de resolver y ofrecer un diagnóstico a las distintas incidencias producidas en los recursos tipo *radioTrasnceptorCTR190*

Señalar que las reglas expertas de gestión que componen nuestra base de conocimiento, pueden sufrir alteraciones: agregación de nuevas reglas, modificación o borrado de reglas expertas existentes. Estas operaciones se realizan mediante el componente interfaz de usuario estudiado en un apartado próximo.

6.7.2.3 El Motor de Inferencia.

El mecanismo de inferencia es la unidad lógica con la que se extraen conclusiones de la base de conocimientos según un método fijo de solución, que se configura imitando el procedimiento humano. En nuestro prototipo el motor de inferencia es el ART*Enterprise, una shell de programación para construcción de sistemas expertos (Anexo D). Utilizando una herramienta de propósito general como es ART*Enterprise, evita complicaciones innecesarias en la implementación del sistema experto y permite centrar nuestro esfuerzo en la resolución de los problemas de análisis y diseño del sistema de gestión. Además proporciona un producto final, con un grado aceptable de estandarización, de calidad y de portabilidad probada entre distintas plataformas de gestión.

El primer paso que realiza el Sistema cuando ocurre una falla, es seleccionar sólo los eventos que resulten relevantes para el análisis del fenómeno ocurrido. Se eliminan las alarmas que aparecen y desaparecen en un tiempo inferior a un segundo, que de forma general no requieren de tratamiento alguno. Una vez obtenido los eventos relevantes, el Sistema Experto integrado debe analizar el comportamiento del sistema de transmisión durante la ocurrencia de las fallas, emitir diagnósticos y aportar recomendaciones. Posterior a la resolución de los problemas, el sistema de gestión debe proseguir con la observación del comportamiento del sistema.

El procedimiento de inferencia en NOMOS+, consiste en buscar una regla que nos permita confirmar la hipótesis buscada. Para que el Sistema Experto integrado pueda analizar los fallos producidos y emitir un diagnóstico favorable, necesitará conocer los siguientes datos:

- Tipo de alarmas generadas en el Sistema de Telecomunicaciones.
- El tiempo expresado en hora, minuto, segundo y mseg. de los eventos.
- El equipo o equipos implicados.

Una vez obtenido estos datos, el Sistema Experto integrado debe:

- Identificar el tipo de falla.
- Identificar el probable origen de la falla.
- Analizar y diagnosticar el comportamiento de las estaciones afectadas.
- Realizar las Acciones y Recomendaciones para la resolución de la incidencia.

6.7.2.4 La interfaz de usuario.

El interfaz de usuario de nuestra herramienta, contiene un módulo para analizar, definir y modificar los archivos de especificaciones GDMO+. Dispone de un conjunto de rutinas de entrada/salida y una interfaz de comandos que lo gestiona. Estos elementos

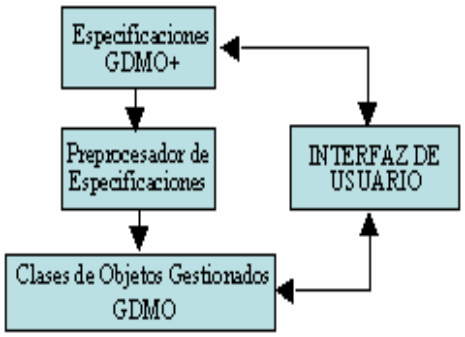

Figura 6.10 Interfaz de Usuario

permiten administrar e inspeccionar el sistema de forma interactiva. El administrador podrá examinar las definiciones GDMO+ de las clases de objetos gestionados, las reglas expertas y las propiedades que lo componen, de una forma fácil y rápida.

La figura 6.10 siguiente muestra las interrelaciones entre la Interfaz y los distintos elementos que conformar el prototipo.

El preprocesador realiza el análisis y tratamiento de los archivos de especificaciones de las clases GDMO+. Para la programación de este módulo se utiliza el lenguaje PHP.

6.7.2.4.1 Funciones de la Interfaz de Usuario.

Operaciones implementadas para reconocer y solucionar problemas comunes de configuración en las especificaciones GDMO+. Se pueden clasificar en dos categorías:

- Gestión de clases de objetos gestionados.
- Gestión de reglas expertas de gestión.

Veamos las principales operaciones permitidas.

6.7.2.4.1.1 Carga de Especificaciones GDMO+.

Cuando iniciamos la herramienta, el administrador en primer lugar deberá proceder a la carga de las definiciones de clases GDMO+. Para ello se realiza una llamada al fichero que contiene las especificaciones correspondientes. En el ejemplo siguiente se realiza la lectura del fichero *"especificaciones.ext"*[2]. El comando utilizado es el siguiente:

```
>read "nombre _ fichero"
```

Las instrucciones mecanografiados por el usuario se muestran en negrilla.

```
➢ read "especificaciones.ext"
  Read 33 managed object class
  Read 101 management expert rules
➢
```

La interfaz realiza el reconocimiento del contenido del fichero de entrada, a continuación muestra como resultado la cantidad de clases de objetos gestionados y reglas expertas, existentes en el fichero.

6.7.2.4.1.2 Análisis del Contenido de Especificaciones GDMO+.

Podemos analizar el contenido del fichero y mostrar las denominaciones de las distintas definiciones de clases de objetos gestionados, definidas en el sistema. El comando empleado tiene la siguiente sintaxis:

```
> Show classes "clases"
```

```
➢ show classes
  1 base MANAGED OBJECT CLASS
  2 sistema MANAGED OBJECT CLASS
  3 sistemaExterno MANAGED OBJECT CLASS
  4 programa MANAGED OBJECT CLASS
  5 multiplexor MANAGED OBJECT CLASS
  6 multiplexorSegundoOrdenMP31 MANAGED OBJECT CLASS
➢ Clases de Objetos Gestionadas: 6
```

[2] La extensión del fichero es .ext, por corresponder su contenido con especificaciones ***ext***endidas GDMO+.

En caso de no indicar el parámetro *"nombre_clases"*, como en el ejemplo anterior, se muestran todas las clases existentes en el fichero de especificaciones GDMO+. También permite la utilización del carácter comodín *, que posibilita la búsqueda de clases de objetos, cuyos nombre cumplan un patrón establecido. Véase el siguiente ejemplo:

```
➢ Show classes multiplex*
  1 multiplexorMP31 MANAGED OBJECT CLASS
  2 multiplexorSegundoOrdenMP31 MANAGED OBJECT CLASS
  3 multiplexTercerOrderMP31 MANAGED OBJECT CLASS
  ….
```

6.7.2.4.1.3 Acceso a las Definiciones de Clases de Objetos Gestionados GDMO+.

Existe la posibilidad de analizar la definición completa de una clase de objeto gestionado determinada. La sintaxis del comando:

```
>show class "nombre_clase"
```

El siguiente ejemplo muestra las especificaciones GDMO+, de los tipos de objetos definidos por la clase *multiplexorMP31*.

```
➢ show class muxtiplexorMP31
 multiplexorMP31 MANAGED OBJECT CLASS
    DERIVED FROM multiplexor;
    CHARACTERIZED BY multiplexorMP31Package;
 REGISTERED AS {nm-MobjectClass 15}

 multiplexorMP31NB  NAME BINDING
    SUBORDINATE OBJECT CLASS multiplexorMP31
                AND SUBCLASSES;
    NAMED BY SUPERIOR OBJECT CLASS multiplexor
                AND SUBCLASSES;
    WHIT ATTRIBUTE multiplexorMP31Id;
 REGISTERED AS {nm-nb 15}

multiplexorMP31Package PACKAGE
    ATTRIBUTES
          alarmaExterna              GET,
          alarmaIndirecta            GET,
          …
    NOTIFICATIONS
          MP31/X.3_EXTERNA,      -- alarma externa
          MP31/X.3_INDIRECTA     -- alarma indirecta
          ...
    RULES
          multiplexorAveriaMP31,
          falloSeñalEntradaMP31,
          …
 REGISTERED AS {nm-package 9}
➢
```

Por ser elementos imprescindibles a la hora de definir una clase de objeto gestionado, se presentan también las plantillas NAME BINDING y PACKAGE asociadas a la clase *multiplexorMP31*.

6.7.2.4.1.4 Acceso a las de Definiciones de Reglas expertas de Gestión

Por tratarse las reglas expertas de propiedades especiales y objeto de estudio de la tesis, se ofrece la opción de acceder de forma individual y directa al conocimiento definido en las distintas clases de objetos gestionados. Resultará de utilidad a la hora de conocer el

conocimiento de gestión vinculado a cada recurso de la red. La sintaxis del comando asociado es la siguiente:

```
>show rules for class "nombre_clase"
```

En el siguiente ejemplo el administrador del sistema, realiza una consulta que muestra todas las reglas expertas definidas para el recurso multiplexor MP31, clase de objeto gestionado *multiplexorMP31*.

```
➢ show rules for class multiplexorMP31
        1  multiplexorAveriaMP31     RULE
        2  falloSeñalEntradaMP31     RULE
        3  falloTerminaLejanoMP31     RULE
        4  falloAlimentacionMP31     RULE
➢
```

6.7.2.4.1.5 Operaciones sobre el conocimiento definido en una clase GDMO+.

Despliega la definición detallada de una regla experta definida para un tipo de objeto concreto:

```
>show rule "nombre_regla_experta" in class "nombre_clase"
```

Es forzoso indicar la clase de objeto gestionado de la que se quiere obtener la definición de regla experta de gestión, ya que como hemos estudiado en capítulos anteriores, pueden existir reglas expertas de gestión con igual nombre, vinculadas a distintas clases de objetos gestionados.

En el ejemplo siguiente se muestra la definición de la regla experta de gestión *falloAlimentacionMP31,* que está en la clase *multiplexorMP31*.

```
➢ show rule falloAlimentacionMP31 in class multiplexorMP31
  falloalimentacionMP31 RULE
     BEHAVIOR falloalimentacionMP31Behavior;
     PRIORITY 3;
     IF
        (? ? ?estacion1 MP31/X.3_OR_ALIM ?estacion2 ALARMA)
    THEN
       ("Grado de Severidad: 4" )
       ("Avería Asociada: Fallo en la alimentación eléctrica de MUX MP31, estación" ?estacion1)
       ("Recomendación: Revisar la fuente de alimentación" )
  REGISTERED AS {nm-rule 36};
➢
```

Hasta ahora hemos visto operaciones de gran utilidad para el administrador del sistema. La herramienta también ofrece la posibilidad de agregar conocimiento de gestión, es decir nuevas reglas expertas. Permite extender la base de conocimiento de nuestro sistema, o bien borrar y modificar el que ya existe.

Para agregar nuevas reglas a la base de conocimiento utilizaremos el siguiente comando:

```
>add rule "regla_experta" to "clase_objeto_gestionado"
```

A continuación del comando *add,* se introduce la definición de la nueva regla experta de gestión que se quiere añadir a la clase. Para finalizar pulsar <INTRO> y la regla quedaría incorporada en la clase de objeto gestionado indicada. En el siguiente ejemplo se agrega la nueva regla experta *falloTributario* a la clase *multiplexorMP31*.

```
➢ add rule falloTributariosMP31 to class multiplexorMP31 <INTRO>
   falloTributariosMP31 RULE
      BEHAVIOR falloTributariosMP31Behavior;
       PRIORITY 3;
       IF
         (? ? ?estacion1 MP31/X.3_TRIBUTARIOS ?estacion2 ALARMA)
       THEN
          ("Grado de Severidad: 3" )
          ("Avería Asociada: Falta señal tributarios MP31/X.3, estación:" ?estacion1)
          ("Recomendación: Revisar tributarios equipos asociados" )
REGISTERED AS {nm-rule 30}; <INTRO>

➢ show rules for class multiplexorMP31
          1  multiplexorAveriaMP31     RULE
          2  falloSeñalEntrada     RULE
          3  falloTerminaLejanoMP31     RULE
          4  falloAlimentacionMP31     RULE
          5  falloTributariosMP31     RULE
➢ 5 Reglas expertas de Gestión Listadas
```

El conocimiento añadido en forma de regla experta puede ser utilizado como base, para diseñar nuevas operaciones de gestión. Así una nueva regla experta de gestión *falloTributarioMP31* ha sido agregada a la clase de objeto gestionado *multiplexorMP31*, en el caso de producirse las alarmas que aparecen en el antecedente de la regla, se ejecutarían las correspondientes acciones del precedente.

De forma análoga la interfaz permite modificar y borrar una regla experta de gestión existente. En el siguiente ejemplo modificamos la prioridad de la regla *falloTributariosMP31* del valor 3 a 4.

```
➢ Update expert rule falloTributarioMP31 in class multiplexorMP31 <INTRO>
   errorStationRemoteMP31 RULE
      BEHAVIOR errorStationRemoteMP31Behavior;
       PRIORITY 4;
       IF  (? ? ?estacion1 MP31/X.3_TRIBUTARIOS ?estacion2 ALARMA)
       THEN
          ("Grado de Severidad: 3" )
          ("Avería Asociada: Falta señal tributarios MP31/X.3, estación:" ?estacion1)
          ("Recomendación: Revisar tributarios equipos asociados" )
REGISTERED AS {nm-rule 30}; <INTRO>
➢
```

En el siguiente ejemplo muestra como se elimina la regla experta *multiplexorAveriaMP31* en la clase *multiplexorMP31*.

```
➢ Delete rule multiplexorAveriaMP31 in class multiplexorMP31

➢ ¿Desea realmente borrar la regla? S/N: s

➢ show  rules for class multiplexorMP31
          1  falloSeñalEntrada     RULE
          2  falloTerminaLejanoMP31     RULE
          3  falloAlimentacionMP31     RULE
          4  falloTributariosMP31     RULE
➢
```

Concluir que nuestro prototipo aporta la ventaja de utilizar una programación lógica (PHP) y un motor de inferencia de uso común como es Art*Enterprise. La programación lógica realizada para la realización de la interfaz de usuario facilita la administración de la base de conocimientos, de forma cómoda. El Art*Enterprise, asimismo reduce el proceso de implementación del sistema experto. Comentar además que nuestro prototipo está todavía en la fase experimental y por ello ofrece la posibilidad de refinar las funcionalidades existentes, así como la agregación de alguna nueva.

Capítulo 7

Conclusiones y Trabajos Futuros.

Tras el desarrollo teórico y metodológico realizado a lo largo de los distintos apartados de la tesis, este capítulo presenta las conclusiones finales sobre el trabajo efectuado. Asimismo propone las futuras líneas de continuación, para una posible ampliación de las conclusiones obtenidas.

7.1 Resumen.

Se ha visto que los modelos de gestión existentes, no son capaces de resolver las cuestiones planteadas en los apartados iniciales del capítulo cinco. Hasta la fecha sólo se han aplicado soluciones individuales a dominios concretos de gestión, haciendo uso de plataformas de gestión específicas. En este contexto, ***proponemos soluciones que permiten mejorar la interoperabilidad en lo que se refiere a los modelos de información de gestión y los Sistemas Expertos, analizando los requisitos necesarios para llevar a cabo la integración de ambos aspectos***. Para ello, se analizan los requisitos necesarios para realizar la integración de la base de conocimiento, en el estándar GDMO.

Las consideraciones que se plantean son las siguientes:

1. Formular, describir, descomponer y asignar el conocimiento entre los distintos agentes inteligentes definidos en GDMO.

2. Capacitar a los distintos objetos para que se comuniquen el conocimiento e interactúen, utilizar el lenguaje de comunicación o protocolos correspondientes, qué y cuando deben comunicarse, etc.

3. Asegurar que los objetos actúen coherentemente al tomar decisiones o realizar acciones, acomodar los resultados de las decisiones locales y prevenir efectos no deseados.

4. Facultar a los objetos para representar y razonar sobre las acciones, planes y conocimiento de otros objetos, coordinación de resultados, etc.

5. Diseñar metodologías y plataformas de gestión utilizando técnicas de la Inteligencia Artificial, que permitan mayor control y gestión sobre los sistemas donde se aplican.

Se trata de resolver el problema existente para llevar a cabo una gestión inteligente integrada. Hasta ahora los objetos gestionados no son capaces de utilizar el conocimiento que aporta la base de conocimientos que recoge las operaciones de administración y control propias a un dominio de gestión. Se requiere de una interfaz que sirva de nexo de

unión entre las especificaciones propias del recurso y las del sistema experto. En la presente tesis, hemos aportado una contribución original para incluir las reglas expertas, dentro de las especificaciones de los elementos de la red, para lo que planteamos un nuevo estándar denominado GDMO Extendido o simplemente GDMO+.

7.2 Metodología y Resultados.

A lo largo de los capítulos de la tesis, se describen los instrumentos de integración, que permiten la inserción de la base de conocimiento del Sistema Experto, en las especificaciones de los objetos. Se resumen en los siguientes apartados.

- En un primer momento, se plantean las ***motivaciones que conducen a la utilización de la gestión inteligente***. Exponemos las ventajas de llevar a cabo una gestión de red asistida por un sistema experto.

- A continuación se realiza un ***Análisis de los modelos de gestión***. Se efectúa un primer estudio de los principales modelos de información y se analizan las posibilidades de integración aportadas por los lenguajes de información de gestión. Se presta una atención especial al modelo propuesto por OSI y se estudian sus aspectos más importantes.

- En un nuevo apartado realizamos una ***Evaluación de los Sistemas Expertos*** y su ***Aplicación a la gestión de redes de comunicaciones.*** Se presentan las distintas posibilidades que la Inteligencia Artificial ofrece para una mejor administración y control de los sistemas de telecomunicaciones actuales. Igualmente se establecen mecanismos de interoperabilidad entre los Sistemas Expertos y los sistemas de gestión. Se muestran algunas de las aproximaciones existentes que proporcionan mecanismos de interoperabilidad entre ambos aspectos, comprobando las limitaciones de dichas propuestas.

  De este estudio se puede concluir que los actuales métodos, no son suficientes para llevar a cabo una gestión inteligente integrada en dominios distintos de gestión.

- ***Obtención de información teórica necesaria,*** acerca del lenguaje de definición de información de gestión GDMO. Estudio de la sintaxis de GDMO y sus conceptos principales. El estándar es la base sobre la que se construye la nueva sintaxis. La extensión propuesta aporta la posibilidad de incorporar las reglas expertas de gestión en las especificaciones de objetos gestionados definidos.

- ***Necesidad de incorporar en la Gestión, las posibilidades que ofrecen los Sistemas Expertos.*** Se indican las mejorías que conlleva unificar la especificación de los objetos gestionados y las reglas que lo gestionan. Adquiere especial relevancia GDMO, lenguaje de definición de información y se toman como punto de partida las construcciones que ya posee. Se observa la necesidad de definir nuevas estructuras para aquellos casos en que sea necesario expresar el conocimiento.

- ***Formalizar la propuesta principal de la tesis, el estándar denominado GDMO Extendido,*** para la incorporación de reglas. Se presenta un método de representación del conocimiento que se apoya en una ampliación del estándar

GDMO, que tiene en cuenta la capacidad de gestión aportada por los sistemas expertos. Se realiza un análisis de la semántica de lenguaje de definición de información de gestión y se proponen los engranajes válidos para la inclusión de aspectos referentes al conocimiento aportado por los Sistemas Expertos.

En lo que se refiere al proceso de fusión se adaptan las plantillas existentes en la norma GDMO, a la que añadimos una nueva propiedad denominada *"RULES"* y su plantilla asociada *"RULE"*.

- ***Diseño conceptual de una arquitectura de gestión basada en el uso de GDMO Extendido,*** análisis de su funcionamiento. Se utiliza para ello la red de telecomunicaciones perteneciente a una compañía del Sector Eléctrico. Debe cumplir con los requisitos y especificaciones del estándar propuesto.

- ***Desarrollo y prueba de un prototipo,*** para las especificaciones realizadas con la nueva sintaxis GDMO+. Define los objetos gestionados y las reglas expertas pertenecientes al Sistema Experto que gobierna una red de telecomunicaciones.

7.3 Líneas Futuras de Investigación.

Es necesario señalar que alguno de los aspectos de la contribución propuesta en esta tesis doctoral, pueden ser mejorados y marcar las líneas de trabajo futuro.

El trabajo desarrollado, constituye el punto de partida para nuevas líneas de investigación relacionadas con la integración del conocimiento de gestión. Pueden emplearse como mejora a las contribuciones propuestas y servir como apertura a nuevos campos de aplicación de la inteligencia artificial:

- **Aplicación de la integración de regla expertas de gestión en el Modelo de gestión Internet**. Es un modelo de gestión muy extendido, que posibilita su implementación en grandes redes. Casi todos los fabricantes de dispositivos como puentes y encaminadores diseñan sus productos para soportar SNMP. Cuenta las ventajas de simplicidad de diseño de su protocolo SNMP, que hace que la información de gestión que se necesita intercambiar en este modelo ocupa pocos recursos de la red y la posibilidad de expansión debido a la sencillez del protocolo SNMP, que lo hace fácil de actualizar.

El protocolo SNMP también tiene desventajas que hay que tener en cuenta a la hora de realizar un diseño de gestión integrado. SNMP es un protocolo que se considera tan simple, que la información está poco organizada, lo que le hace no muy acertada para gestionar las grandes redes existentes en la actualidad. Además tiene grandes fallos de seguridad. Estos problemas se resuelven en la nueva versión SNMPv2.

- ***Estudio de la portabilidad a otras plataformas y extensión del ámbito de actividad de los Sistema Expertos Integrados***. Aplicación a sistemas de telecomunicaciones de diferente naturaleza al radioenlace utilizado en nuestro estudio, tales como redes conmutadas ATM, Ethernet, Gigabit Ethernet, etc.

- ***Ampliación del tratamiento de alarmas y eventos producidos en el sistema gestionado.*** Incorporar características adicionales del modelo de gestión OSI a otras áreas funcionales, que no se han contemplado en esta primera versión: gestión de configuración, contabilidad, seguridad, etc.

• ***Integración de una base de datos suplementaria***, que permitiría manejar toda la información adicional que se necesita para el manejo de alarmas, así como el trabajo en torno a la realización de correlación de eventos.

• ***Mejora de la interfaz de usuario***, optimizando la ya existente e incorporando nuevas características como pueden ser las posibilidades de personalizar el entorno de usuario, tener distintas vistas de la red por categoría de operador, mostrar representaciones gráficas de históricos de alarmas, etc. Asimismo, inclusión de capacidades de proceso de informes de históricos que permitan extraer información adicional de los datos recibidos.

• ***Aplicación de otro tipo de representación del conocimiento y razonamiento, distinto a las reglas.*** Obtener nuevos compromisos entre expresividad y eficiencia, a la hora de escoger el método más apropiado. Así por ejemplo cabe la posibilidad de utilizar un método de representación híbrida u otros posibles métodos de representación alternativos como las Redes Semánticas, Redes Neuronales, Marcos, etc.

Anexo A.

Estándares de Gestión de Red.

En los sistemas de gestión existen una serie de estándares, que ayudan a la definición de los protocolos, servicios de gestión, base de datos, etc. Cada modelo de gestión, tanto OSI como INTERNET tienen su propio procedimiento de normalización.

Por su complejidad empezaremos estudiando los estándares de gestión para OSI.

A.1 Los estándares de OSI.

La ITU-T/OSI (*International Telecommunications Union / Open System Interconnection*), antes CCITT, ha creado varias normas, que junto con las normas expedidas inicialmente por la ISO (*International Organization for Standarization*) constituyen la definición de los protocolos y los servicios necesarios para la gestión de red en entornos abiertos (OSI). Dichos servicios están plasmados en el estándar de Servicio Genérico de Información de Gestión (CMIS - *Common Management Information Service*), y los protocolos los define la norma Protocolo Genérico de Información de Gestión (CMIP - *Common Management Information Protocol*). El conjunto de estándares generados como resultado del trabajo conjunto de la ISO y la ITU en este tema, son los denominados Gestión de Sistemas OSI [Kauffels92]

En la siguiente tabla A.1 se ofrece la relación entre las distintas normalizaciones que podemos encontrarnos. Como se observa en la tabla los estándares se dividen en cinco categorías:

1. *Introducción y Estructura de Gestión OSI:* Provee una introducción de carácter general acerca de los conceptos de gestión.
2. *CMIS/CMIP:* Define el servicio genérico de información de gestión (CMIS - *Common Management Information Service*), el cual provee los diferentes servicios necesarios para gestionar las aplicaciones. Define también el protocolo genérico de información de gestión (CMIP - *Common Management Information Protocol*), que es el encargado de proporcionar el intercambio de información, actuando como un soporte para CMIS.
3. *Funciones de Gestión de Sistemas*: Define las funciones específicas que debe desempeñar un sistema de gestión.
4. *Modelo de Información de Gestión:* Define la base de información de gestión (MIB - *Management Information Base*), la cual contiene una representación de todos los objetos que se van a ser gestionados bajo el ambiente OSI.

5. Gestión de capas: Define la información, servicios y funciones de gestión relacionadas con las capas específicas del modelo OSI.

Contenido	ISO	CCITT
Visión General del Modelo de Gestión OSI		
OSI Basic Reference Model Part 4: Management Framework	7498-4	X.700
Systems Management Overview	10040	X.701
CMIS/CMIP		
Common Management Information Service Definition	9595	X.710
Amendment 4: Access control	9595 DAM 4	X.710
Amendment X: Allomorphism	9595 PDAM X	X.710
CMIP Specification Part 1: Specification	9596-1	X.711
Amendment X: Allomorphism	9596 PDAM X	X.711
Part 2: Protocol Implementation Conformance Statement (PICS) Proforma	9596-2	X.712
Funciones de gestión de sistemas		
Part 1: Object Management Function	10164-1	X.730
Part 2: State Management Function	10164-2	X.732
Part 3: Attributes for Representing Relationships	10164-3	X.733
Part 4: Alarm Reporting Function	10164-4	X.734
Part 5: Event Report Management Function	10164-5	X.735
Part 6: Log Control Function	10164-6	X.736
Part 7: Security Alarm Reporting Function	10164-7	X.737
Part 8: Security Audit Trail Function	10164-8	X.740
Part 9: Objects and Attributes for Access Control	10164-9	X.741
Part 10: Accounting Meter Function	10164-10	X.742
Part 11: Workload Monitoring Function	10164-11	X.739
Part 12: Test Management Function	10164-12	
Part 13: Summarization Function	10164-13	
Part 14: Confidence and Diagnostic Test Categories	10164-14	
Accounting Management	SC 21 N 4971	
Part s: Schedulling Function	SC 21 N 6021	
OSI Software Management	SC 21 N 6040	
General Relationship Model	SC 21 N 6041	
Management Domains	SC 21 N 6047	
Management Knowledge	SC 21 N 6048	
Synchronization	SC 21 N 6049	
Performance Management	SC 21 N 6306	
Modelo de Información de Gestión		
Part 1: Management Information Model	10165-1	X.720
Part 2: Definition of Management Information	10165-2	X.721
Part 4: Guidelines for the Definition of Managed Objects	10165-4	X.722
Part 5: Generic Managed Information	10165-5	
Requirements for Implementation Conformance Statement Proformas	10165-6	
Gestión de capas		
Elements of Management Information Related to OSI *Network Layer Standards*	10733	
Transport Layer Management	10737	

Tabla A.1 Estándares de Normalización OSI.

Vamos a realizar un breve estudio de cada una de las categorías.

A.2. Visión General de la Gestión OSI.

El primero de los apartados del conjunto de normas, ofrece una introducción a la Gestión de red se divide en las siguientes normas:

ISO 7498-4/ITU-T X.700, Management Framework.

Modelo básico de referencia-Parte 4: *Arquitectura de Gestión*. Proporciona definiciones, conceptos básicos y objetivos acerca de la gestión OSI, así como las denominadas cinco áreas funcionales: gestión de configuración, gestión de prestaciones, gestión de seguridad, gestión de fallos y gestión de contabilidad.

A la hora de definir estas funcionalidades básicas que ha de soportar un sistema de gestión, son multitud las clasificaciones que tradicionalmente se han venido haciendo atendiendo a criterios de diversa naturaleza. Sin embargo, la ISO (International Standarization Organization, Organización Internacional de Estandarización) realizó una clasificación de las tareas de los sistemas de gestión, en estas cinco áreas funcionales.

Esta clasificación fue originariamente aplicada a los sistemas de gestión de la torre de protocolos OSI (Open Systems Interconnection, Interconexión de sistemas abiertos), y más tarde aplicada directa o indirectamente a la práctica totalidad de los ámbitos de gestión.

ISO 10040/ITU-T X.701, System Management Overview.

Preámbulo de los sistemas de gestión. Proporciona una apreciación global de las estándares de sistemas de gestión y establece las bases para los sistemas de gestión distribuida. Además introduce en el concepto de dominio administrativo de gestión.

A.3 El Protocolo CMIP y los servicios CMIS.

ISO 9595/ITU-T X.710, Common Management Information Service Element (CMISE).

Definición del Servicio Genérico de Información (MIS) y del Elemento de Servicio Genérico de Información de Gestión (CMISE), usados para el intercambio de información y comandos, con el propósito de gestionar un sistema.

Esta recomendación define:

- Un conjunto de primitivas que constituyen los elementos de servicio de aplicación.
- Los parámetros que se pasan.

ISO 9596/ITU-T X.711, Common Management Information Protocol (CMIP).

Esta Recomendación especifica el *Protocolo de Información de Gestión Común,* protocolo usado por lasa entidades de capa de aplicación, para intercambiar la información de gestión.

Esta Recomendación especifica:

- Los distintos procedimientos para la transmisión de información de gestión entre las entidades de aplicación implicadas.

- La sintaxis abstracta del (CMIP) y las reglas de la codificación asociadas.

- Los procedimientos para la interpretación correcta de información de control.

A.4 Funciones de gestión de sistemas.

ISO 10164-1/ITU-T X.730,Object Management Function.

Función de Gestión de Objeto, esta norma define los objetos gestionados y describe las operaciones necesarias para su creación y configuración. Reglas de creación, borrado, cambio de nombre y listado de objetos, así como borrado y cambio de valor de sus atributos.

ISO 10164-2/ITU-T X.732, State Management Function.

Función de Gestión de Estado, identifica los distintos estados para los objetos gestionados. El estándar de gestión OSI permite dos clasificaciones de estados:

- *Operacional:* Para definir un objeto como ocupado, activo, enable o disable.
- *Administrativo:* Cerrado, abierto y protegido.

Esta especificación también indica las reglas para salir y entrar en cada uno de estos estados.

ISO 10164-3/ITU-T X.733, Attributes for Representing Relationships.

Objetos y Atributos para Representar Relaciones, define las relaciones entre los objetos gestionados. Se establecen las siguientes:

- Direct: Parte de la información asociada a un objeto gestionado que identifica a otro objeto.
- Indirecta: Una relación se deduce a través de dos objetos, concadenando dos o más relaciones directas.

- Simetrica: La iteración de reglas entre dos objetos es la misma.

- Asimétrica: La iteración de reglas entre dos objetos es distinta.

ISO 10164-4/ITU-T X.734, Alarm Reporting.

Función de Aviso de Alarma, define e identifica cinco categorías básicas de errores, notificaciones:

- Comunicaciones
- Calidad de Servicio
- Procesamiento
- Equipamiento
- Entorno

Además de estas cinco notificaciones, las alarmas proporcionan información referente a la posible causa de la alarma, gravedad de la misma, acciones a efectuar, etc.

ISO 10164-5/ITU-T X.735, Event Report Management Function.

Función de Gestión de Aviso de Evento. Establece las componentes para soportar el envío remoto de eventos y el procesado local de éstos, utilizando para ello el concepto de discriminador.

ISO 10164-6/ITU-T X.736, Log Control Function.

Función de Control de Traza. Define las operaciones de control de registro histórico, para un Sistema de Gestión de Red. Indica como se guarda la información de las operaciones realizadas sobre los objetos gestionados, se definen los criterios de creación y borrado de ficheros de Log, etc.

ISO 10164-7/ITU-T X.737, Security Alarm Reporting Function.

Función de Aviso de Alarma de Seguridad. Define los procedimientos para la gestión de seguridad y la recepción de los distintos tipos de alarmas, enviadas por los objetos gestionados. Se especifican los siguientes tipos de alarmas relacionadas con la seguridad:

- Integridad.
- Operacional.
- Física.
- Seguridad.
- Time-to-Remain.

Pueden ser distintas las causas por las que pueden ocurrir las alarmas pueden ser: información esperada no se ha recibido, modificación ilegal de la información recibida, etc.

ISO 10164-8/ITU-T X.740, Security Audit Trail Function.

Función de Seguimiento de Auditoria de Seguridad. Análoga a la función de control de log, pero además ofrece la posibilidad de realizar auditorias sobre el fichero de históricos, tales como: contabilidad, seguridad, conexiones y desconexiones realizadas, etc.

ISO 10164-9/ITU-T X.741, Objects and Attributes for Access Control.

Objetos y Atributos para el Control de Accesos. Permite al administrador de la red prevenir accesos no autorizados sobre los objetos gestionados. Define mecanismos de control de acceso, basados en reconocimientos de perfiles de usuario, definiendo reglas para denegar el acceso a los elementos gestionados correspondientes.

ISO 10164-10/ITU-T X.742, Accounting Meter Function.

Utilización de funciones de medida con propósitos contables. Esta norma define como se contabiliza, el costo y registro e identificación del uso de la red de gestión. Además provee de umbrales de uso para determinados objetos gestionados, por parte de los usuarios de los recursos.

ISO 10164-11/ITU-T X.739, Workload Monitoring Function.

Define un modelo de monitorización para un objeto gestionado. Establece procedimientos de regeneración, que serán usados en orden a aspectos relacionados con el funcionamiento de los objetos.

ISO 10164- Otras Funciones.

Ofrece funcionalidades alternativas tales como: Métrica de objetos y atributos, Función de gestión de diagnóstico, Función de planificación de sistemas de gestión, etc.

A.5 Modelo de Información de Gestión.

Parte del estándar que trata la Estructura de Información de Gestión (SMI). Se divide en cuatro partes:

ISO 10165-1/ITU-T X.720, Management Information Model.

Modelo de Información de Gestión. Esta norma presenta un modelo de información de gestión general para OSI, cuya finalidad es "*proporcionar una estructura a la información de gestión transportada externamente por los protocolos de gestión de sistemas y modelar los aspectos de gestión de los recursos conexos (p.e. un conmutador Ethernet)*".

ISO 10165-2/ITU-T X.721, Definitions of Support Objects.

Definición de la Información de Gestión. Esta recomendación define cierto número de clases de objetos gestionados que conforman la MIB, son utilizadas en las diferentes funciones de gestión de sistemas. Estas clases sirven como superclases para la definición de nuevas clases de objetos, por medio de la herencia de clases.

ISO 10163-3, Definitions of management attributes.

Especifica los atributos usados por la gestión: cambios de estado, estado de error, estado de configuración, nombre distinguido de la clase de objeto, definiciones, etc.

ISO 10165-4/ITU-T X.722, Guidelines for the Definition of Managed Objects.

Guía para la definición de objetos gestionables. Realiza una amplia exposición de los principios básicos para la definición de los objetos gestionados. Esta norma va a servir de orientación a quienes realizan la especificación de dichos objetos, así como favorecer la coherencia entre las definiciones que de ellos se establecen.

Estándar utilizado en nuestras investigaciones para realizar la integración del conocimiento de gestión en las especificaciones de los recursos de red.

A.6 RFCs Internet.

El IAB (Internet Architecture Board), organismo dependiente directamente de la ISOC (Internet Society), es el encargado de aprobar las recomendaciones y estándares de Internet, a través de una serie de documentos denominados RFC's, *Request for Comments*. Los RFC's son descripciones de protocolos y servicios, describen el funcionamiento interno de Internet, servicios de la red, protocolos y sus aplicaciones. Ofrecen detalles de los procedimientos y formatos para su implementación. Nacen como resultado de estudios o sumarios del trabajo de comités técnicos o sesiones de trabajo y pueden ir desde el tratamiento de temas muy técnicos, hasta la definición de conceptos más generales. Todos los RFC's son considerados de dominio público a menos que se explicite algo en contra de esta norma general. A continuación se describen algunos de los RFC's más significativos emitidos por el IAB:

- RFC 1052. *IAB Recommendations for the Development of Internet Network Management Standards*. Recomendaciones del IAB para el desarrollo de estándares de gestión de red para Internet, abril 1988.

- RFC 1066. *Management Information Base for Network Management of TCP/IP-based internets*. Base de información de gestión para redes basadas en TCP/IP, agosto 1988.

-RFC 1085. *ISO Presentation Services on top of TCP/IP-based internets*. Servicios ISO de presentación sobre redes basadas en TCP/IP, diciembre 1988.

- RFC 1155. *SMI, Structure and Identification of Management Information*. Mayo 1990. Estructura e Identificación de la Información de gestión para redes basadas en TCP/IP. Describe cómo se definen los objetos gestionados contenidos en el MIB.

- RFC 1156. *MIB, Management Information Base for Network Management of TCP/IP-based internets*. Base de Información de Gestión para la gestión de redes basadas en TCP/IP, mayo 1990.

- RFC 1157. *A Simple Network Management Protocol (SNMP)*. Protocolo SNMP, mayo 1990.

- RFC 1212. *Concise MIB Definitions*. Definición concisa de MIB, Marzo 1991.

- RFC 1213. *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*. Base de información de la información de gestión para redes basadas en TCP/IP: MIB-II, Marzo 1991.

- RFC 1215. *A Convention for Defining Traps for use with the SNMP*. Definición de Traps para uso en SNMP, marzo 1991.

- RFC 1227. *SNMP MUX Protocol and MIB*. Protocolo SNMP MUX y MIB, marzo 1991.

- RFC 1228. *SNMP-DPI Simple Network Management Protocol Distributed Program Interface,* marzo 1991.

- RFC 1239. *Reassignment of Experimental MIBs to Standard MIBs*. Reasignación de MIBs experimentales a MIBs estándares, junio 1991.

- RFC 1351. *SNMP Administrative Model*. Modelo administrativo de SNMP, julio 1992.

- RFC 1352. *SNMP Security Protocols*. Protocolos de seguridad SNMP, julio 1992.

A.7 Núcleo TMN.

Normas que definen cómo funcionan las redes de telecomunicaciones. Las Recomendaciones del ITUT no tienen carácter vinculante, aunque generalmente se aplican por su gran calidad y porque garantizan la interconectividad de las redes, a la vez que permiten la

prestación de servicios de telecomunicaciones a escala mundial. Las más interesantes relacionadas con el modelo TMN, son las siguientes:

- ITU-T M.3000: Introducción a las recomendaciones TMN.

- ITU-T M.3010: Recomendación que define los principios de las redes de gestión de comunicaciones, 1993.

- ETR 037[1]: TMN: Principios, objetivos y conceptos.

- ITU-T M.3020: Recomendación que define la Metodología de especificación de mensajes TMN, 1993.

- ETR 046: Guía para modelado de la gestión de comunicaciones.

- ITU-T M.3100: Especifica el Modelo genérico de información de gestión, 1993.

- I-ETS 300 293: Objetos genéricos de gestión.

- ITU-T M.3180: Define el catálogo de información de gestión, 1993.

- I-ETS 300 653: Librería para objetos genéricos de gestión en el nivel de red, 1996.

- ETR 088: Clases de objetos para soporte de funciones de planificación de dependencia temporal.

- ITU-T M.3200: Introducción a los servicios de gestión de red, 1993.

- ETR 047: Servicios de gestión TMN, 1992.

- ETR 048: Descripción de los servicios de gestión TMN, 1992.

- **ITU-T M.3400**: Funciones de gestión TMN.
- **ITU-T M.3300**: Facilidades de gestión TMN en el interfaz F.
- **ITU-T Q.811**: Especificación Q3: capa superior.
- **ITU-T Q.812**: Especificación Q3: capa inferior
- **ITU-T Q.821**: Especificación Q3: Vigilancia de alarmas
- **ITU-T Q.822**: Especificación Q3: Gestión de rendimiento

[1] Los informes técnicos del European Telecommunications Standards Institute (ETSI) o Instituto de Estándares de Telecomunicación Europeos es una organización de estandarización de la industria de las telecomunicaciones (fabricantes de equipos y operadores de redes) de Europa, con proyección mundial. No contienen especificaciones técnicas, sino información complementaria sobre el entorno técnico relacionado con las cuestiones de normalización. Un ETR no está sometido a los mismos procedimientos de aprobación que las normas del ETSI y se publica una vez aprobado por el correspondiente comité técnico.

Anexo B.

Sintaxis Abstracta y Sintaxis de Transferencia ASN.1.

Uno de los problemas fundamentales que aparecen en la comunicación entre dos sistemas, es el traslado eficaz de los datos, es decir, que los datos que se reciban son los que se han enviado (Integridad). El problema de cómo comunicar dos sistemas diferentes es similar al problema que se presenta entre dos personas a la hora de comunicarse, siendo por ejemplo una inglesa y la otra de nacionalidad alemana.

Imaginemos que en lugar de dos sistemas tenemos un número mucho mayor, que a su vez utilizan distintos leguajes, la comunicación entre las máquinas se presentará complicada. La solución consiste en utilizar una vía de comunicación, usando una misma sintaxis y semántica, esta va a ser común y va utilizarse para la representación de datos y estructuras de datos.

En el nivel de aplicación vamos a utilizar *Sintaxis Abstracta*, en particular la *Abstrac Syntax Notation One (ASN.1),* describe la estructura genérica de los datos independientemente de cualquier codificación técnica para la representación de los datos, permite definir tipos de datos y especificar valores para estos datos.

Por otro lado entre el nivel de aplicación y el nivel de presentación, habrá un conjunto de reglas para transformar los datos, la sintaxis de los datos transferidos entre los distintos niveles de presentación deberá ser entendida por ambos extremos. Esta sintaxis se le conoce como S*intaxis de Transferencia.* La sintaxis abstracta y la sintaxis de transferencia deben ser determinadas en el comienzo y durante el tiempo de comunicación entre los sistemas.

La sintaxis de transferencia que vamos a estudiar es la *Basic Encoding Rules (BER),* que indica la forma en la cual los datos son representados en términos de patrones de bits para ser transferidos entre entidades.

Hay que tener en cuenta que la sintaxis local de cada sistema puede depender del protocolo utilizado. En la siguiente figura se ilustran los conceptos de sintaxis abstracta y sintaxis de transferencia.

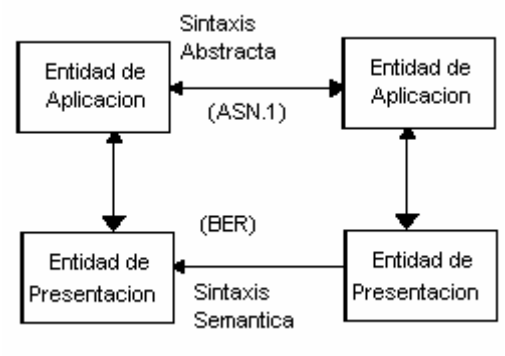

Figura B.1. Sintaxis Abstracta y Sintaxis de Transferencia

Los estándares relevantes de ISO son:

- 8824 Specification of Abstract Syntax Notation One (ASN.1).
- 8825 Specification of Basic Encoding Rules for ASN.1 (BER).

Los estándares relevantes de CCITT son:

- X.208 Specification of Abstract Syntax Notation One
- X.209 Specification of Basic Encoding Rules for ASN.1

B.1 La Notación Sintáctica Abstracta Uno (ASN.1).

El aprendizaje de ASN.1 es similar al de cualquier lenguaje de programación de alto nivel. ASN.1 nos proporciona una herramienta fundamental para el uso de aplicaciones, para la descripción de la información de una forma común e independiente de cualquier implementación particular.

ASN.1 es un lenguaje formal desarrollado y estandarizado por CCITT(X:208) e ISO(8824). ASN.1. Su importancia radica en lo siguiente:

- Es usado para la definición de sintaxis abstractas de aplicaciones de datos.
- Es usado para definir la estructura de la aplicación y presentación de las unidades de protocolos de datos (PDUs).
- Es usado para definir la gestión de la información base por los dos sistemas de gestión más importantes: SNMP y OSI.

ASN.1 es una notación flexible que permite definir varios tipos de datos, desde tipos simples como enteros, hasta tipos estructurados como conjuntos y secuencias, además de tipos complejos definidos en función de otros tipos.

Esta sintaxis abstracta es usada para intercambiar información entre componentes de aplicación de diferentes sistemas. El intercambio consiste en PDUs del nivel de aplicación, los cuales contienen información de protocolos de control y datos de usuario. En un sistema, la información representada usando una sintaxis abstracta debe ser mapeada ("traducida"), de alguna forma para su presentación al usuario humano. Similarmente, esta sintaxis abstracta debe ser mapeada en algún formato local al sistema para su almacenamiento. Es importante destacar que tal "traducción" es usada en el caso de la información base de gestión: la MIB. Sin embargo, los elementos en la MIB son definidos usando la sintaxis abstracta. De este modo, la notación de la sintaxis abstracta es empleada

por un usuario para definir la MIB; la aplicación debe entonces convertir esta definición de forma conveniente para su almacenamiento local.

La componente debe también traducir la información de la sintaxis abstracta de la aplicación, a la sintaxis de transferencia que describe los valores de los datos en formato binario, conveniente para la iteración con la componente de transferencia de datos. Por ejemplo, una sintaxis abstracta puede incluir un tipo de dato carácter; la sintaxis de transferencia podría especificar código ASCII o EBCDIC.

Este enfoque para el intercambio de datos de aplicación soluciona los dos problemas que se presentan en un entorno distribuido heterogéneo en la representación de los datos.

- Hay una representación común para el intercambio de datos entre diferentes sistemas.
- Internamente a un sistema, una aplicación usa alguna representación particular. El esquema sintaxis abstracta/transferencia soluciona automáticamente diferencias en la representación entre entidades cooperantes en una aplicación.

B.2 Reglas Básicas.

En primer lugar vamos a conocer los conceptos básicos, comenzando con los tipos y continuando con la definición de los módulos. En ASN.1, el punto de partida es el *valor*. Una instancia de un objeto puede ser un valor, una colección de todos estos valores es un *tipo*. Estos tipos serán similares a los tipos de datos que se pueden encontrar en los lenguajes de programación.

ASN.1 es una notación flexible que permite definir una variedad de tipos de datos, desde *tipos* simples como son enteros y cadenas, hasta *tipos* estructurados y complejos definidos en términos de otros.

La notación básica del tipo se representa con la siguiente sintaxis:

NombreDeTipo ::= TIPO

La notación básica de un valor:

NombreDeValor NombreDeTipo ::= VALOR.

Los tipos permitidos en ASN.1 son los siguientes:

- Tipos Simples
- Tipos Estructurados.
- Tipos Etiquetados
- Subtipos.

Al igual que cualquier lenguaje de programación, ANS.1 tiene sus propias reglas:

- La disposición no es significativa; espacios múltiples y líneas en blanco pueden ser consideradas como espacios simples.
- Los comentarios empiezan por 2 guiones "—" y terminan del mismo modo o por final de línea.

- Los identificadores del lenguaje empiezan con una letra, pueden contener letras, dígitos y guiones, pero no pueden terminar con un guión o contener dos consecutivos.
- El identificador para un tipo debe empezar con una letra mayúscula.
- El identificador para un valor debe empezar con una letra minúscula
- Las palabras reservadas están todas en letras mayúsculas.
- Una referencia de tipo o el nombre de un módulo empiezan por mayúsculas.

Vamos a explicar cada uno de los tipos a través de ejemplos.

B.3 Tipo Simple.

Los tipos simples son aquellos que no contienen componentes, son tipos "atómicos". Algunos de estos tipos utilizados en gestión:

- INTEGER: tipo de dato que toma como valor un número entero.

```
EthernetNumCollisions ::= INTEGER
```

Otro caso más interesante:

```
EthernetNumCollisionsRange ::= INTEGER {min(0), max(1000)}
```

sería análogo a decir que min = 0 y max = 3.

- OCTET STRING: Puede ser usado para definir datos binarios, utilizando uno o más octetos.

```
EhernetAdapterNumber ::=OCTET STRING
```

- OBJECT IDENTIFIER: es un identificador único de una información particular de un objeto. Es una secuencia de componentes enteros que identifican un objeto como un algoritmo o un tipo atributo.

- NULL: Toma el valor nulo.

B.4 Tipo Estructurado.

Algunos tipos de datos estructurados se explican a continuación:

- SÉQUENSE: Tipo usado para definir una variable que tiene cero o más elementos, y donde a su vez el orden de los elementos es importante (una lista). Puede tener diferentes tipos de elementos.

- SÉQUENSE OF sólo se le permite un tipo de elemento.

```
EthernetCollisionsCounter ::= SEQUENSE OF { highValue INTEGER,
                                            LowValue INTEGER}
```

Está permitido crear nuevos tipos a partir de los ya existentes. Un ejemplo:

```
LanSimpleCounterLimits ::= SEQUENSE
      {Counter1 COMPONETS OF EthernetCollisionsCounter,
      {Counter2 EthernetAdapterNumber}
```

En la definición anterior notar que se usa COMPONENTS OF, esto indica que todos los elementos de la secuencia EthernetAdapterNumber, deben ser incluidos.

- CHOICE, ofrece la posibilidad de incluir calores nulos.

Sea la definición de un adaptador Ehernet y queremos tener en cuenta el caso en el que no tenga un número de identificador. Debemos de modelarlo de la siguiente manera:

```
EhernetAdapterNumber ::= CHOICE {NULL, OCTET STRING}
```

De esta manera usando CHOICE, el adaptador de red podrá tener un número de identificación o no. Más tarde volveremos sobre el uso de CHOICE.

B.5 Tipos Etiquetados.

Estos tipos son usados para modelar variables que eliminan ambigüedades, en estructuras de tipos de datos definidas y donde existen posibilidades de confusión de cómo se reciben o interpretan los datos.

Las etiquetas pueden ser EXPLICIT o IMPLICIT (explicitas e implícitas). Cuando se usa una etiqueta IMPLICIT, no es necesario transmitir el tipo de dato durante la transferencia al otro extremo, mientras que en el caso de EXPLICIT, la transferencia del tipo de datos es necesaria. Cuando no se especifica nada, se toma por defecto el tipo EXPLICIT.

La definición de etiqueta tendrá una clase y un número dentro de corchetes. A continuación se listan los tipos de etiquetas disponibles:

- UNIVERSAL: Usada para los tipos de datos definidos en la norma X.208, es decir los tipos básicos.

- CONTEXT-SPECIFIC: En los tipos estructurados, hay problemas de distinción entre los distintos elementos, especialmente cuando se usa CHOISE. A menos que el emisor especifique los elementos, será muy laborioso para el receptor llegar a entender que elemento está recibiendo en cada momento.

```
BeaconingCounter ::=SEQUENCE OF
            {counterName [0] IMPLICIT INTEGER,
            counterNumber [1] IMPLICIT INTEGER}
```

En esta producción, counterName tiene contexto especifico [0] e indica que es el primer elemento. De forma análoga counterNumber tiene contexto específico [1] y es el segundo elemento.

- PRIVATE: Utilizado para identificar un tipo de dato incluido dentro de una organización o país.

B.6 Definición de Módulos.

Los módulos son usados principalmente para agrupar definiciones ASN.1. Estos también permiten utilizar definiciones de tipos externo que han sido declaradas en otros sitios (módulos), mediante el uso del mecanismo IMPORT y EXPORT. Se puede decir que los módulos son similares a las funciones del lenguaje C o las subrutinas del Pascal.

Podemos encontrar definiciones de módulos incluidas en paquetes normalizados de clases de objetos de gestión. Veamos una definición de módulo en el siguiente ejemplo:

```
LanNetworkModule
DEFINIONS EXPLICIT TAGS ::= BEGIN
--A continuación comienza el cuerpo del modulo
IMPORTS
       RelativeDistinguishedName FROM InformationFramework
       {joint-iso-ccitt ds (5) modules (1)
       InformationFramework (1)}
-- Fin de IMPORTS
EXPORTS
       LanNetworkName::= SEQUENCE of RelativeDistinguishedName

-- Fin EXPORTS
MacAddresses::= OCTET STRING (SIZE (6))
LanWorkstationSerialNumbers::= OCTET STRING (SIZE (32))
LanSegment ::= SET OF LanWorkstationSerialNumbers
EthernetNetworks ::= SET OF MacAddresses
TokenRingnetworks ::= SET OF LanSegment
LanNetworks ::= SET {ethernet [0] IMPLICIT EthernetNetworks,
                     tokenNet [1] IMPLICIT TokenRingNetworks}
END
```

En la definición anterior, el tipo *RelativeDistinguishedName* está definido en el módulo externo InformationFramework (1), va ha ser usado en este módulo utilizando el constructor IMPORTS. Además se observa que LanNetWorkName se va a utilizar en otro módulo, por esta razón está definido como EXPORTS. Tal y como se puede apreciar, aunque no es obligatorio el uso de IMPORT y EXPORT, es recomendable su utilización a fin de facilitar el trabajo con las definiciones de tipos.

B.7 Subtipos.

Un subtipo como el propio nombre indica, utiliza tipos ya definidos (figura B.2). Cuando se deriva un subtipo, es necesario tener un valor.

Los subtipos son utilizados para los siguientes casos:

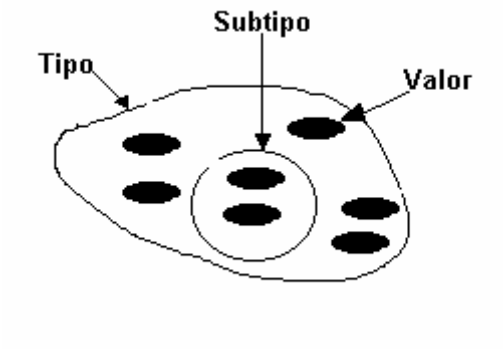

Figura B.2. Tipos, Valores y Subtipos

- Cuando dos o más tipos tienen características comunes, es recomendable hacer una clase padre con todas las características comunes y a partir de ella obtener los distintos subtipos con sus características individuales.

- Si tenemos la necesidad de limitar el tamaño o valores de los tipos existentes.

- Cuando por su interés se desea explicar un subconjunto de valores con más detalle.

B.8 Basic Encoding Rules (BER).

Es utilizado en el nivel de presentación, antes de transferir los valores ASN.1, tal y como muestra la Figura B.1 de la introducción. El dato se transfiere como un flujo de octetos. Al proceso de representación del dato como una secuencia de octetos se le conoce con el nombre de *Codificación*. Cuando se transmite un dato en sintaxis abstracta, la codificación se realiza en un extremo al ser enviado y la decodificación al ser recibido, en el otro de los extremos.

Como ya se ha mencionado, la norma X.208 de CCITT define las reglas de codificación, consistentes en un conjunto de procedimientos para codificar los datos.

La estructura de BER puede ser de la forma Identificador-Longitud-Valor, ***I****dentifier-****L****ength-****C****ontents* (ICL), (ISO 8825-1), así llamada porque la base de la codificación es una estructura formada por estas tres partes. Cuando la longitud y el contenido es conocido. El identificador indica el tipo de dato ASN.1, la longitud indica la longitud del contenido que sigue al campo longitud, y el contenido contiene le valor ASN.1 a transferir.

En la literatura es posible que encuentre otros nombres para TLV, incluyendo Etiqueta-Longitud-Valor (***T****ag-****L****ength-****V****alue*). La codificación de cualquier valor, como por ejemplo una PDU, se construye en forma TLV.

Vamos a ver cuáles son las reglas que hay que seguir a la hora de codificar estructuras ASN.1. En realidad, existe libertad para fijar esas reglas, pero las recomendadas más extendidas son las BER.

Cuando el tipo es primitivo (simple), el contenido es una serie de octetos, tal cual se ve en la siguiente, Figura B.3.

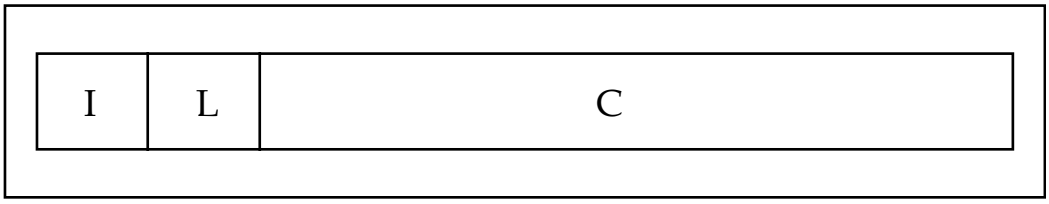

Figura B.3 Estructura con tipo simple.

Cuando el tipo es estructurado, los contenidos son una serie de codificaciones anidadas, cada una siguiendo la forma ILV. Esto es lo que trata de representarse en la figura B.4 siguiente.

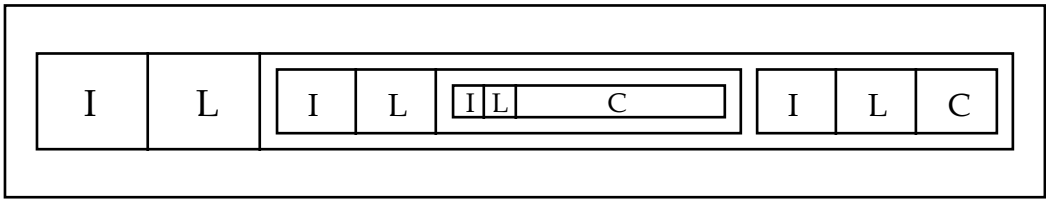

Figura B.4 Estructura con tipo estructurado.

B.8.1 El Campo Identificador

No tiene un nº máximo de octetos para codificarlo, pero se ha de usar siempre el mínimo nº de octetos posible. Para tipos con números (etiquetas) entre 0 y 30 se usa un único octeto, tal y como aparece en la figura B.5:

Figura B.5 Campo Identificador

Los dos bits CC representan la *clase* de la etiqueta (*UNIVERSAL*, *APPLICATION-WIDE*, *CONTEXT-SPECIFIC* o *PRIVATE*), el bit F la *forma* del tipo (simple o estructurado) y los cinco bits TTTTT el nº de la etiqueta. Si estos últimos cinco bits están todos a uno, eso indica que se están utilizando varios octetos para codificar el tipo. En el caso del SNMP, todos los tipos utilizados tienen etiquetas menores que 30, por lo que sólo se utiliza un octeto.

Los valores de los bits para cada clase y cada forma aparecen en la figura B.6:

bit 8 7	CLASE	bit 6	FORMA
0 0	universal	0	primitivo
0 1	application-wide	1	estructurado
1 0	context-specific		
1 1	private-use		

Figura B.6 Codificación de clases y formas.

B.8.2 El Campo Longitud

La misión de este campo es permitir encontrar el final de los contenidos. Su codificación admite tres formatos: *corto*, *largo* e *indefinido*. Los dos primeros codifican la longitud en octetos de los contenidos. El indefinido indica el final de los contenidos mediante una marca especial.

El formato corto puede emplearse cuando la longitud es menor o igual que 127 octetos. Su forma es la que se muestra en la figura B.7:

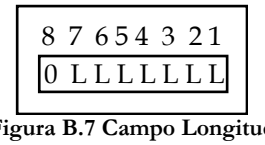

Figura B.7 Campo Longitud

El campo LLLLLLL representa la longitud de los contenidos.

El formato largo puede representar cualquier longitud, figura B.8:

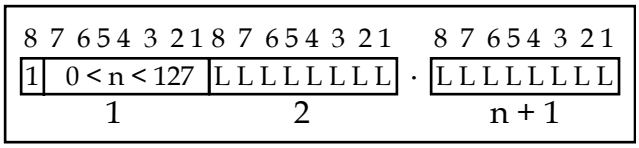

Figura B.8 Formato Largo.

El campo que antes contenía la longitud ahora contiene la longitud de la longitud, *n*. La longitud en sí misma está en los octetos siguientes a *n*.

El formato indefinido también puede codificar cualquier longitud. Este formato no se usa en SNMP, pero el agente lo tiene añadido a su funcionalidad. El primer octeto en este formato es cómo el de la figura B.9:

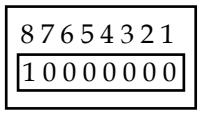

Figura B.9 Formato Indefinido

Este octeto en el campo longitud indica que el marcador de final de contenidos está inmediatamente después de éstos. El marcador son dos octetos consecutivos a cero. El formato indefinido sólo puede usarse con tipos estructurados.

B.8.3 El Campo de Contenido de Datos

Los contenidos codifican un valor del tipo. Su estructura, por tanto, refleja el conjunto de valores del tipo. A continuación se ve con cierto detalle la forma de codificar los valores de cada uno de los tipos que utiliza SNMP.

INTEGER.

Los octetos del contenido representan el valor usando una representación binaria en complemento-a2. Veamos algunos ejemplos con 16 bits:

```
1 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 -32768
0 1 0 0 0 0 0 0  0 0 0 0 0 0 0 0  16384
0 0 1 0 0 0 0 0  0 0 0 0 0 0 0 0   8192
1 0 0 1 0 1 1 0  0 1 0 0 0 1 1 0 -27066
0 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  32767
```

Se debe usar siempre en la codificación el mínimo nº de octetos posible: los nueve bits más significativos no deben ser iguales.

La etiqueta del tipo INTEGER es 2.

NULL

Se codifica sin octetos de contenido: su etiqueta es 5 y la longitud 0.

OCTET STRING.

Los octetos representan los elementos de la string. Su etiqueta es 4.

Hay otra forma de codificarlo, que parte de considerar el OCTET STRING como un tipo estructurado, pero esa forma no se emplea en SNMP.

OBJECT IDENTIFIER.

Los contenidos consisten en una secuencia de números. Cada nº (excepto los dos primeros, que se combinan en uno) se representa como una serie de octetos, usando los 7 bits menos significativos de cada octeto y poniendo el bit más significativo a 1, excepto en el último octeto, en el que se pone a 0. Se debe usar el mínimo nº de octetos posible.

Los contenidos están formados por la concatenación en orden, de varias series de octetos.

Los dos primeros números, digamos *m* y *n*, se combinan formando el nº 40*m* + *n*.

Veamos un ejemplo. El valor:

números OBJECT IDENTIFIER ::= { 2 6 6 247 1 }

se codifica, incluyendo tipo y longitud, como:

06 05 5606817701

Como se ve, la etiqueta de este tipo es 6.

SEQUENCE.

Los valores de los componentes deben aparecer en el orden de su definición. Por ejemplo, dada la siguiente definición:

```
Lecturas ::= SEQUENCE OF INTEGER (-100..60)
números Lecturas ::=
                { -5, -4, 4, 15, 20, 24, 29, 28, 20, 15, 3, -1 }
```

el valor de números se codificaría (incluyendo tipo y longitud):

```
10 24
02 01 FB  02 01 FC  02 01 04  02 01 0F  02 01 14  02 01 18
02 01 1D  02 01 1C  02 01 14  02 01 0F  02 01 03  02 01 FE
```

La única variación en la codificación que podríamos encontrar en este ejemplo es en el formato de la longitud.

Anexo C

Listados de Especificaciones GDMO+, para la gestión de una Red de Telecomunicaciones Privada del Sector Eléctrico.

En este anexo se establecen las descripciones realizadas para un sistema de información siguiendo la propuesta de norma GDMO+. Se utiliza para ello las definiciones de tipos de objetos pertenecientes a la red de Radioenlaces de la Compañía eléctrica Sevillana-Endesa.

En las siguientes especificaciones se incorpora la base de conocimiento referente al sistema experto gestor, que queda dentro de cada una de las especificaciones de los tipos de objetos gestionados definidos para el sistema.

C.1 Especificaciones GDMO+.

Se relacionan los tipos de objetos gestionados descritos de acuerdo con la estructura de la norma GDMO+. El propósito no es establecer una lista exhaustiva de todos los elementos que componen los distintos tipos de objetos del sistema de información: atributos, acciones, parámetros, etc. Por el contrario, se pretende exponer de forma intuitiva todas las definiciones que componen la base de conocimiento del Sistema Experto y sus reglas expertas de gestión integradas.

Procederemos pues al estudio de los distintos recursos del sistema gestionado, representados en los árboles de contención de las figuras 6.4 y 6.5 del capítulo 6.

C.1.1 Base.

Tipo que identifica a la base del árbol de tipos de objetos. Es el tipo de objeto más alto dentro de la jerarquía de objetos.

C.1.1.1 Definición de Clase de Objeto Gestionado.

```
base MANAGED OBJECT CLASS
        DERIVED FROM "Rec. X.721 | ISO/IEC 10165-2: 1992" : top;
        CHARACTERIZED BY  basePackage;
          -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 1}
```

C.1.1.2 Definición de Enlace de Nombre.

```
baseNB NAME BINDING
        SUBORDINATE OBJECT CLASS base
            AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS top
            AND SUBCLASSES;
        WHIT ATTRIBUTE baseId;
REGISTERED AS {nm-nb 1}
```

C.1.1.3 Definición de Paquete.

```
basePackage PACKAGE
        ATTRIBUTES
                tipo                    GET,
                nombre                  GET-REPLACE,
                descripcion             GET,
                dependencia             GET,
                localizacion            GET-REPLACE,
                graficoAsociado         GET,
                ayuda                   GET;
        RULES
                SSRFIL1, SSRFIL2, SSRFIL3, SSRFIL4;
REGISTERED AS {nm-package 1}
```

C.1.1.4 Definición de Atributos.

```
tipo ATTRIBUTE
        WITH ATTRIBUTE SYNTAX tipo;
        MATCHES FOR EQUALITY;
        BEHAVIOUR tipoBehaviour
        BEHAVIOUR DEFINED AS
            ”Atributo que identifica de forma única el tipo de un objeto determinado del sistema”;
REGISTERED AS {nm-attribute 1}

nombre ATTRIBUTE
        DERIVED FROM ” iimcIIMCIMIBTRANS”:displayString;
        BEHAVIOUR nombreBehaviour
        BEHAVIOUR DEFINED AS ”Atributo que identifica de forma única un objeto del sistema, utilizando para ello
        una descripción que tiene el árbol de dependencias de objetos”;
REGISTERED AS {nm-attribute 2}

descripcion ATTRIBUTE
        DERIVED FROM ” iimcIIMCIMIBTRANS”:displayString;
        BEHAVIOUR descripcionBehaviour
        BEHAVIOUR DEFINED AS  “Breve descripción del objeto. Se utiliza únicamente a título informativo y no
                sirve para identificar al objeto del sistema”;
REGISTERED AS {nm-attribute 3}

dependencia ATTRIBUTE
        DERIVED FROM ” iimcIIMCIMIBTRANS”:displayStrig;
        BEHAVIOUR dependenciaBehaviour
        BEHAVIOUR DEFINED AS
            ”Indica el objeto superior del cual depende, dentro del árbol de dependencias de los objetos”;
REGISTERED AS {nm-attribute 4}

localizacion ATTRIBUTE
        DERIVED FROM ” IIMCRFC12131354ASN.Integer;
        MATCHES FOR EQUALITY, ORDERING;
        BEHAVIOUR localizacionBehaviour
        BEHAVIOUR DEFINED AS ”Determina si el objeto es local o es remoto. Información de gran valor para
        distinguir el modo de procesamiento de las acciones y las notificaciones del objeto”;
REGISTERED AS {nm-attribute 5}

graficoAsociado ATTRIBUTE
        DERIVED FROM ” IIMCRFC12131354ASN.Integer128;
        MATCHES FOR EQUALITY, ORDERING;
        BEHAVIOUR graficoAsociadoBehaviour
        BEHAVIOUR DEFINED AS
            ”Reconoce el gráfico asociado a un objeto, que será presentado al operador cuando se solicite  el detalle
        gráfico de un objeto”;
REGISTERED AS {nm-attribute  6}

ayuda ATTRIBUTE
        DERIVED FROM ” IIMCRFC12131354ASN.displayString;
        BEHAVIOUR ayudaBehaviour
        BEHAVIOUR DEFINED AS
                ”Descripción del texto de ayuda asociado a un objeto”;
REGISTERED AS {nm-attribute 7}
```

C.1.1.5 Definición de Reglas Expertas de Gestión.

```
SSRFIL1 RULE
    BEHAVIOR SSRFIL1Behavior;
    PRIORITY 2;
    IF
        ?A<-(?fecha ?h1 ?remota ?alarma ?destino ALARMA)
        ?B<-(?fecha ?h2 ?remota ?alarma ?destino DESAPARECE_ALARMA & : (<(ABS(- ?h1 ?h2)) 1.00))
        (TEST(NOT(equal ?alarma CMF-38_PREBER_1)))
        (TEST(NOT(equal ?alarma CMF-38_PREBER_2)))
        (TEST(NOT(equal ?alarma SPU_1_PREBER_1)))
        (TEST(NOT(equal ?alarma SPU_1_PREBER_2)))
        (TEST(NOT(equal ?alarma CMF-38_BER_1)))
        (TEST(NOT(equal ?alarma CMF-38_BER_2)))
        (TEST(NOT(equal ?alarma SPU_1_BER_1)))
        (TEST(NOT(equal ?alarma SPU_1_BER_2)))
    THEN
        (retract ?A),
        (retract ?B);
REGISTERED AS {nm-rule FIL1};

SSRFIL2 RULE
    BEHAVIOR SSRFIL2Behavior;
    PRIORITY 2;
    IF
        ?A<-(?fecha ?h1 ?remota ?alarma ALARMA)
        ?B<-(?fecha ?h2 ?remota ?alarma DESAPARECE_ALARMA & : (<(ABS(- ?h1 ?h2)) 1.00))
    THEN
        (retract ?A),
        (retract ?B);
REGISTERED AS {nm-rule FIL2};

SSRFIL3 RULE
    BEHAVIOR SSRFIL3Behavior;
    PRIORITY 2;
    IF
        ?A<-(?fecha ?h1 ?remota $?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_SUPERIOR ?num)
        ?B<-(?fecha ?h2 ?remota $?destino MEDIDA_EN_ALARMA_LIM.SUP_PASA_A_NORMAL ?num
                & : (<(ABS(- ?h1 ?h2)) 1.00))
    THEN
        (retract ?A),
        (retract ?B);
REGISTERED AS {nm-rule FIL3};

SSRFIL4 RULE
    BEHAVIOR SSRFIL4Behavior;
    PRIORITY 2;
    IF
        ?A<-(?fecha ?h1 ?remota $?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num)
        ?B<-(?fecha ?h2 ?remota $?destino MEDIDA_EN_ALARMA_LIM.INF_PASA_A_NORMAL ?num
                & : (<(ABS(- ?h1 ?h2)) 1.00))
    THEN
        (retract ?A),
        (retract ?B);
REGISTERED AS {nm-rule FIL4};
```

C.1.2 Sistema.

Tipo que identifica a un sistema, subsistema o conjunto de equipos.

C.1.2.1 Definición de Clase de Objeto Gestionado.

```
sistema MANAGED OBJECT CLASS
        DERIVED FROM  base;
        CHARACTERIZED BY  sistemaPackage;
         -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 2}
```

C.1.2.2 Definición de Enlace de Nombre.

```
sistemaNB NAME BINDING
        SUBORDINATE OBJECT CLASS sistema
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS base
                AND SUBCLASSES;
        WHIT ATTRIBUTE baseId;
REGISTERED AS {nm-nb 2}
```

C.1.2.3 Definición de Paquete.

```
sistemaPackage PACKAGE
        ATTRIBUTES
                IdentificationSistema                   GET,
                "Rec. X.721":OperationalState           GET,
                estadoUso                               GET,
                estadoAdminisrtativo                    GET,
                estadoGestion                           GET,
                responsable                             GET;
        ATTRIBUTESGROUP estado;
        NOTIFICATIONS
                crear, borrar, modificar;
        PARAMETERS
                identificadorOrigen, causaProbable;
REGISTERED AS {nm-package 2}
```

C.1.2.4 Definición de Atributos.

```
identificationSistema  ATTRIBUTE
        DERIVED FROM " IIMCRFC12131354ASN.displayString;
        BEHAVIOUR ayudaBehaviour
        BEHAVIOUR DEFINED AS
        "Descripción del texto de ayuda asociado a un objeto";
REGISTERED AS {nm-attribute 8}

estadoUso  ATTRIBUTE
        DERIVED FROM " IIMCRFC12131354ASN.displayString;
        BEHAVIOUR ayudaBehaviour
        BEHAVIOUR DEFINED AS
        "Descripción del texto de ayuda asociado a un objeto";
REGISTERED AS {nm-attribute 9}

estadoAdminisrtativo  ATTRIBUTE
        DERIVED FROM " IIMCRFC12131354ASN.displayString;
        BEHAVIOUR ayudaBehaviour
        BEHAVIOUR DEFINED AS
        "Descripción del texto de ayuda asociado a un objeto";
REGISTERED AS {nm-attribute 10}

estadoGestion  ATTRIBUTE
        DERIVED FROM " IIMCRFC12131354ASN.displayString;
        BEHAVIOUR ayudaBehaviour
        BEHAVIOUR DEFINED AS
        "Descripción del texto de ayuda asociado a un objeto";
REGISTERED AS {nm-attribute 11}

responsable  ATTRIBUTE
        DERIVED FROM " IIMCRFC12131354ASN.displayString;
        BEHAVIOUR ayudaBehaviour
        BEHAVIOUR DEFINED AS
        "Descripción del texto de ayuda asociado a un objeto";
REGISTERED AS {nm-attribute 12}
```

C.1.3 Sistema Externo.

Tipo que identifica a un sistema, subsistema o conjunto de equipos externo al centro de control NOMOS. La principal característica de un sistema externo es que mantiene comunicaciones con el centro de control NOMOS.

C.1.3.1 Definición de Clase de Objeto Gestionado.

```
sistemaExterno MANAGED OBJECT CLASS
    DERIVED FROM  sistema;
   CHARACTERIZED BY sistemaExternoPackage;
   -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 3}
```

C.1.3.2 Definición de Enlace de Nombre.

```
sistemaExternoNB NAME BINDING
        SUBORDINATE OBJECT CLASS  sistemaExterno
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS sistema
                AND SUBCLASSES;
        WHIT ATTRIBUTE sistemaExternoId;
 REGISTERED AS {nm-nb 3}
```

C.1.3.3 Definición de Paquete.

```
sistemaExternoPackage PACKAGE
        ATTRIBUTES
                        direccionComunicaciones     GET,
                        estadoComunicaciones        GET;
        ACTIONS         modicaComunicaciones;
        NOTIFICATIONS   alarmaComunicaciones;
        PARAMETERS      actuacionComunicaciones;
REGISTERED AS {nm-package 3}
```

C.1.4 Programa.

Tipo que identifica a un programa o conjunto de programas del sistema.

C.1.4.1 Definición de Clase de Objeto Gestionado.

```
programa MANAGED OBJECT CLASS
   DERIVED FROM  base;
   ATTRIBUTES
        identificacionPrograma          GET,
        versionPrograma                 GET,
        administrador                   GET;
 -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 4}
```

C.1.4.2 Definición de Enlace de Nombre.

```
sistemaNB NAME BINDING
        SUBORDINATE OBJECT CLASS "base"
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS …
                AND SUBCLASSES;
        WHIT ATTRIBUTE baseId;
REGISTERED AS {nm-nb 4}
```

C.1.5 Estación.

Tipo que identifica a una estación del sistema.

C.1.5.1 Definición de Clase de Objeto Gestionado.

```
estacion MANAGED OBJECT CLASS
        DERIVED FROM  sistemaExterno;
        ATTRIBUTES
           SituacionGeografica          GET;
REGISTERED AS {nm-MobjectClass 5}
```

C.1.5.2 Definición de Enlace de Nombre.

```
estacionNB NAME BINDING
        SUBORDINATE OBJECT CLASS estacion
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS sistemaExterno
                AND SUBCLASSES;
        WHIT ATTRIBUTE estacionId;
REGISTERED AS {nm-nb 5}
```

C.1.6 Edificio.

Tipo que identifica a un edificio o parte de un edificio.

C.1.6.1 Definición de Clase de Objeto Gestionado.

```
edificio MANAGED OBJECT CLASS
        DERIVED FROM  base;
        ATTRIBUTES
                responsable             GET;
REGISTERED AS {nm-MobjectClass 6}
```

C.1.6.2 Definición de Enlace de Nombre.

```
edificioNB NAME BINDING
        SUBORDINATE OBJECT CLASS edificio
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS base
                AND SUBCLASSES;
        WHIT ATTRIBUTE edificioId;
REGISTERED AS {nm-nb 6}
```

C.1.7 MultiplexorXXX.

Tipo que identifica a un multiplexor modelo xxx.

C.1.7.1 Definición de Clase de Objeto Gestionado.

```
multiplexorXXX MANAGED OBJECT CLASS
        DERIVED FROM  sistema;
         ATTRIBUTES
                numeroPuertos                   GET;
REGISTERED AS {nm-MobjectClass 7}
```

C.1.7.2 Definición de Enlace de Nombre.

```
multiplexorXXXNB NAME BINDING
        SUBORDINATE OBJECT CLASS multiplexor
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS sistema
                AND SUBCLASSES;
        WHIT ATTRIBUTE multiplexorId;
REGISTERED AS {nm-nb 7}
```

C.1.8 puertoMultiplexorXXX.

Tipo que identifica a una entrada a un multiplexor modelo xxx.

C.1.8.1 Definición de Clase de Objeto Gestionado.

```
puertoMultiplexorXXX MANAGED OBJECT CLASS
    DERIVED FROM multiplexorXXX;
    CHARACTERIZED BY puertoMultiplexorXXXPackage;
        -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 8}
```

C.1.8.2 Definición de Enlace de Nombre.

```
puertoMultiplexorNB NAME BINDING
        SUBORDINATE OBJECT CLASS puertoMultiplexor
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS multiplexorXXX
                AND SUBCLASSES;
        WHIT ATTRIBUTE puertoMultiplexorId;
REGISTERED AS {nm-nb 8}
```

C.1.8.3 Definición de Paquete.

```
puertoMultiplexorXXXPackage PACKAGE
    ATTRIBUTES
        numeroPuerto            GET,
        nombrePuerto            GET,
        estadoPuerto            GET;
     NOTIFICATIONS
        perdidaSeñal;
REGISTERED AS {nm-package 4}
```

C.1.9 Multiplexor.

Realizan la multiplexión de 30 canales de voz /datos de 64 Kbit/s, en un canal de 2 Mbit/s.

C.1.9.1 Definición de Clase de Objeto Gestionado.

```
multiplexor MANAGED OBJECT CLASS
   DERIVED FROM  sistema;
   CHARACTERIZED BY  multiplexorPackage;
    -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 9}
```

C.1.9.2 Definición de Enlace de Nombre.

```
multiplexorNB NAME BINDING
        SUBORDINATE OBJECT CLASS multiplexor
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS sistema
                AND SUBCLASSES;
        WHIT ATTRIBUTE multiplexorId;
REGISTERED AS {nm-nb 9}
```

C.1.9.3 Definición de Paquete.

```
multiplexorPackage PACKAGE
        ATTRIBUTES
                modeloMultiplexor           GET,
                nombreMultiplexor           GET,
                orientacionMultiplexor      GET,
                velociadadTransferencia     GET;
REGISTERED AS {nm-package 5}
```

C.1.10 MultiplexorPrimario.

C.1.10.1 Definición de Clase de Objeto Gestionado.

```
multiplexorPrimario MANAGED OBJECT CLASS
  DERIVED FROM  multiplexor;
  ATTRIBUTES
        numeroCanalesAnalogolicos       GET,
        numeroCanalesDigitales          GET;
REGISTERED AS {nm-MobjectClass 10}
```

C.1.10.2 Definición de Enlace de Nombre.

```
multiplexorPrimarioNB NAME BINDING
        SUBORDINATE OBJECT CLASS multiplexorPrimario
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS multiplexor
                AND SUBCLASSES;
        WHIT ATTRIBUTE multiplexorPrimarioId;
REGISTERED AS {nm-nb 10}
```

C.1.11 PrimerOrdenMP31.

C.1.11.1 Definición de Clase de Objeto Gestionado.

```
primerOrdenMP31 MANAGED OBJECT CLASS
        DERIVED FROM  multiplexorPrimario;
        CHARACTERIZED BY primerOrdenMP31Package;
         -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 11}
```

C.1.11.2 Definición de Enlace de Nombre.

```
primerOrdenMP31NB NAME BINDING
        SUBORDINATE OBJECT CLASS primerOrdenMP31
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS multiplexorPrimario
                AND SUBCLASSES;
        WHIT ATTRIBUTE primerOrdenMP31Id;
REGISTERED AS {nm-nb 11}
```

C.1.11.3 Definición de Paquete.

```
primerOrdenMP31 Package PACKAGE
        ATTRIBUTES
                alarmaExternaFonia          GET,
                alarmaIndirectaFonia        GET,
                tasaErroresFonia            GET,
                señal64K                    GET,
                falloSincronizacion         GET,
                señalizacionExternaInterna  GET,
                señalizacionIndirecta       GET,
                estadoAlimentacion          GET;
        NOTIFICATIONS
                MP31_EXT_FONIA ,            -- alarma externa fonia
                MP31_IND_FONIA,             -- alarma indirecta fonia
                MP31_INT_FONIA,             -- alarma interna fonia
                MP31_EXC_T_E_FON,           -- alarma de tasa excesiva de errores fonia
                MP31_F_SEN_IT16,            -- alarma de perdida de señal en IT16
                MP31_F_SINCRO,              -- alarma fallo sincronizacion
                MP31_INT+EXT_SEN,           -- alarma externa+interna de señalizacion
```

```
                MP31_IND_SEN,           -- alarma indirecta de señalizacion
                MP31_OR_ALIM;           -- fallo en alimentacion
        RULES   SSR28, SSR30,
                SSR31, SSR32, SSR33,
                SSR34, SSR35, SSR36, SSR37;
REGISTERED AS {nm-package 6}
```

C.1.11.4 Definición de Reglas Expertas de Gestión.

```
SSR28 RULE
    BEHAVIOR SSR28Behavior;
    PRIORITY 0;
    IF
        (?fecha ?h1 ?remota MP31_EXT_FONIA ?destino ALARMA)
        (NOT(?fecha ?h2 ?remota MP31_F_SINCRO ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
    THEN
        ("Grado de Severidad: 3") ,
        ("Averia Asociada: FALLO EN SEÑAL DE ENTRADA A MP31, ESTACION " ?remota),
        ("Recomendacion: REVISAR CANAL DE ENTRADA AL MULTIPLEXOR.");
REGISTERED AS {nm-rule 28};

SSR30 RULE
    BEHAVIOR SSR30Behavior;
    PRIORITY 0;
    IF
        (? ? ?remota MP31_IND_FONIA ?destino ALARMA)
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada:
            AVERIA EN TERMINAL LEJANO O DETECCION DE SEÑAL DE AIS, ESTACION " ?remota),
        ("Recomendacion: REVISAR LISTA DE ALARMAS PARA VERIFICAR CUAL DE LOS
                EQUIPOS CONECTADOS PROVOCA LA ALARMA.");
REGISTERED AS {nm-rule 30};

SSR31 RULE
    BEHAVIOR SSR31Behavior;
    PRIORITY 0;
    IF
        (OR (? ? ?remota MP31_INT_FONIA ?destino ALARMA)
            (?fecha ? ?remota MP31_INT+EXT_SEN ?destino ALARMA))
        (NOT (?fecha ? ?remota MP31_IND_SEN ?destino ALARMA))
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: AVERIA EN MULTIPLEXOR MP31, ESTACION " ?remota),
        ("Recomendacion: REVISAR MULTIPLEXOR.");
REGISTERED AS {nm-rule 31};

SSR32 RULE
    BEHAVIOR SSR32Behavior;
    PRIORITY 0;
    IF
        (?fecha ?h1 ?remota MP31_EXC_T_E_FON ?destino ALARMA)
        (?fecha ?h2 ?remota MP31_EXT_FONIA ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: FALLO EN EQUIPO CONECTADO AL MUX MP31, ESTACION " ?remota),
        ("Recomendacion: REVISAR EQUIPOS CONECTADOS AL MUX.");
REGISTERED AS {nm-rule 32};

SSR33 RULE
    BEHAVIOR SSR33Behavior;
    PRIORITY 0;
    IF
        (? ? ?remota MP31_F_SEN_IT16 ?destino ALARMA)
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: FALTA DE SEÑAL EN UN CANAL DIGITAL, ESTACION " ?remota),
        ("Recomendacion: REVISAR EQUIPOS ASOCIADOS A LOS CANALES DIGITALES.");
REGISTERED AS {nm-rule 33};
```

```
SSR34 RULE
      BEHAVIOR SSR34Behavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota MP31_F_SINCRO ?destino ALARMA)
            (?fecha ?h2 ?remota MP31_EXT_FONIA ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
      THEN
             ("Grado de Severidad: 3"),
             ("Averia Asociada: FALLO EN SEÑAL DE ENTRADA AL MUX MP31 POR PERDIDA DEL
                                SINCRONISMO,  ESTACION " ?remota),
             ("Recomendacion: REVISAR EQUIPOS CONECTADOS A ENTRADA DE MULTIPLEXOR");
REGISTERED AS {nm-rule 34};

SSR35 RULE
      BEHAVIOR SSR35Behavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota MP31_INT+EXT_SEN ?destino ALARMA)
            (?fecha ?h2 ?remota MP31_IND_SEN ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
      THEN
             ("Grado de Severidad: 3"),
             ("Averia Asociada:
               ALARMA EXTERNA, FALLA UN EQUIPO CONECTADO AL MP31,  ESTACION " ?remota),
             ("Recomendacion: REVISAR LA LISTA DE ALARMAS PRESENTES PARA DETERMINAR
                        DE DONDE  PROVIENE EL FALLO.");
REGISTERED AS {nm-rule 35};

SSR36 RULE
      BEHAVIOR SSR36Behavior;
      PRIORITY 0;
      IF
            (? ? ?remota MP31_OR_ALIM ?destino ALARMA)
      THEN
             ("Grado de Severidad: 3"),
             ("Averia Asociada: FALLO EN EL ALIMENTADOR O EN LA ALIMENTACION DEL
                        ALIMENTADOR DEL MP31, ESTACION " ?remota),
             ("Recomendacion: REVISAR ALIMENTADOR Y/O SU ALIMENTACION.");
REGISTERED AS {nm-rule 36};

SSR37 RULE
      BEHAVIOR SSR37Behavior;
      PRIORITY 0;
      IF
            (? ? ?remota MP31_IND_SEN ?destino ALARMA)
      THEN
             ("Grado de Severidad: 3"),
             ("Averia Asociada:
                AVERIA EN TERMINAL LEJANO O  DETECCION  DE SEÑAL DE AIS, ESTACION " ?remota),
             ("Recomendacion: REVISAR LISTA DE ALARMAS PARA VERIFICAR CUAL DE LOS
                        EQUIPOS CONECTADOS PROVOCA LA ALARMA.");
REGISTERED AS {nm-rule 37};
```

C.1.12 PrimarioZAK30/5.

C.1.12.1 Definición de Clase de Objeto Gestionado.

```
primarioZAK30/5 MANAGED OBJECT CLASS
            DERIVED FROM  multiplexorPrimario;
            CHARACTERIZED BY primarioZAK30/5Package;
             -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 12}
```

C.1.12.2 Definición de Enlace de Nombre.

```
primarioZAK30/5NB NAME BINDING
            SUBORDINATE OBJECT CLASS primarioZAK30/5
                  AND SUBCLASSES;
            NAMED BY SUPERIOR OBJECT CLASS multiplexorPrimario
                  AND SUBCLASSES;
            WHIT ATTRIBUTE primarioZAK30/5Id;
REGISTERED AS {nm-nb 12}
```

C.1.12.3 Definición de Paquete.

```
primarioZAK30/5Package PACKAGE
        ATTRIBUTES
                perdidaSeñalTributariaEnt       GET,
                altaTasaErroresBFL3             GET,
                falloInterno                    GET,
                falloAlineamientoTrama          GET,
                AISIND                          GET,
                falloAlineamientoMultitrama     GET,
                FalloAlinmentacion              GET;
        NOTIFICATIONS
                ZAK?30-5_FALLO_TRIB,    -- perdida de señal tributaria entrante
                ZAK?30-5 BER,           -- tasa de errores BER
                ZAK?30-5 BFL,           -- tasa de errores BFL
                ZAK?30-5 BFL3,          -- tasa de errores BFL3
                ZAK?30-5_FALLO_INT,     -- fallo interno
                ZAK?30-5_F.AL.TR,       -- alineamiento de trama
                ZAK?30-5_AIS+IND,       -- recepcion de señal AIS+IND
                ZAK?30-5_FALLO_ALIM;    -- fallo alimentacion
        RULES   SSR62, SSR64, SSR65, SSR67, SSRB69;
REGISTERED AS {nm-package 7}
```

C.1.12.4 Definición de Reglas Expertas de Gestión.

```
SSR62 RULE
    BEHAVIOR SSR62Behavior;
    PRIORITY 0;
    IF
        (? ? ?remota ZAK_30?5_FALLO_TRIB ?destino ALARMA)
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada:
            AVERIA EN EQUIPO ASOCIADO AL MULTIPLEXOR ZAK 30?5, ESTACION " ?remota),
        ("Recomendacion: REVISAR EQUIPOS ASOCIADOS.");
REGISTERED AS {nm-rule 62};

SSR64 RULE
    BEHAVIOR SSR64Behavior;
    PRIORITY 0;
    IF
      (OR
        (? ? ?remota ZAK_30?5_BER ?destino ALARMA)
        (? ? ?remota ZAK_30?5_BFL3 ?destino ALARMA)
        (? ? ?remota ZAK_30?5_BFL ?destino ALARMA)
        (? ? ?remota ZAK_30?5_F.AL.TR. ?destino ALARMA))
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: AVERIA EN EQUIPO ASOCIADO AL MULTIPLEXOR ZAK 50?5,
                ESTACION " ?remota "  O PROBLEMAS EN EL ENLACE ENTRE " ?remota " Y " ?destino),
        ("Recomendacion: REVISAR EQUIPOS ASOCIADOS AL MULTIPLEXOR ZAK 30?5.");
REGISTERED AS {nm-rule 64};

SSR65 RULE
    BEHAVIOR SSR65Behavior;
    PRIORITY 0;
    IF
        (? ? ?remota ZAK_30?5_FALLO_INT ?destino ALARMA)
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: AVERIA EN EL MULTIPLEXOR ZAK 50?5, ESTACION " ?remota),
        ("Recomendacion: REVISAR EL MULTIPLEXOR.");
REGISTERED AS {nm-rule 65};

SSR67 RULE
    BEHAVIOR SSR67Behavior;
    PRIORITY 0;
    IF
        (? ? ?remota ZAK_30?5_AIS+IND ?destino ALARMA)
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: AVERIA EN TERMINAL LEJANO O FALLO EN SEÑAL DE ENTRADA
                DETECTADO POR MUX ZAK 30?5, ESTACION " ?remota),
        ("Recomendacion: REVISAR EQUIPOS ASOCIADOS Y/O TERMINALES LEJANOS.");
REGISTERED AS {nm-rule 67};
```

```
SSR69 RULE
      BEHAVIOR SSR69Behavior;
      PRIORITY 0;
      IF
            (? ? ?remota ZAK_30?5_FALLO_ALIM ?destino ALARMA)
      THEN
             ("Grado de Severidad: 3"),
             ("Averia Asociada: FALLO EN EL ALIMENTADOR O EN LA ALIMENTACION DEL
                    ALIMENTADOR DEL ZAK50 ?5, ESTACION " ?remota),
             ("Recomendacion: REVISAR ALIMENTADOR Y/O ALIMENTACION DEL ALIMENTADOR");
REGISTERED AS {nm-rule 69};
```

C.1.13 PrimarioZAKD/I.

C.1.13.1 Definición de Clase de Objeto Gestionado.

```
primarioZAKD/I  MANAGED OBJECT CLASS
            DERIVED FROM  multiplexorPrimario;
            CHARACTERIZED BY primarioZAKD/IPackage;
             -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 13}
```

C.1.13.2 Definición de Enlace de Nombre.

```
primarioZAKD/INB NAME BINDING
            SUBORDINATE OBJECT CLASS primarioZAKD/I
                  AND SUBCLASSES;
            NAMED BY SUPERIOR OBJECT CLASS multiplexorPrimario
                  AND SUBCLASSES;
            WHIT ATTRIBUTE primarioZAKD/I Id;
REGISTERED AS {nm-nb 13}
```

C.1.13.3 Definición de Paquete.

```
PrimarioZAKD/I5Package PACKAGE
            ATTRIBUTES
                  perdidaSeñalTributariaEnt          GET,
                  altaTasaErroresBFL3                GET,
                  falloInterno                       GET,
                  falloAlineamientoTrama             GET,
                  AISIND                             GET,
                  falloAlineamientoMultitrama        GET,
                  FalloAlinmentacion                 GET;
            NOTIFICATIONS
                  ZAK_D/I_FALLO_TRIB,      -- perdida de señal tributaria entrante
                  ZAK_D/I_FALLO_INT,       -- fallo interno
                  ZAK_D/I_F.AL.TR,         -- alineamiento de trama
                  ZAK_D/I_AIS+IND,         -- recepcion de señal AIS+IND
                  ZAK_D/I BFL3,            -- alineamiento de multitrama
                  ZAK_D/I_FALLO_ALIM;      -- fallo alimentacion
            RULES    SSR63, SSR66, SSR68, SSR70, SSR71;
REGISTERED AS {nm-package 8}
```

C.1.13.4 Definición de Reglas Expertas de Gestión.

```
SSR63 RULE
      BEHAVIOR SSR63Behavior;
      PRIORITY 0;
      IF
            (? ? ?remota ZAK_D/I_FALLO_TRIB ?destino ALARMA)
      THEN
             ("Grado de Severidad: 3"),
             ("Averia Asociada:
                  AVERIA EN EQUIPO ASOCIADO AL MULTIPLEXOR ZAK D/I, ESTACION " ?remota),
             ("Recomendacion: REVISAR EQUIPOS ASOCIADOS.");
REGISTERED AS {nm-rule 63};

SSR66 RULE
      BEHAVIOR SSR66Behavior;
      PRIORITY 0;
      IF
            (? ? ?remota ZAK_D/I_FALLO_INT ?destino ALARMA)
      THEN
             ("Grado de Severidad: 3"),
             ("Averia Asociada: AVERIA EN EL MULTIPLEXOR ZAK D/I, ESTACION " ?remota),
             ("Recomendacion: REVISAR EL MULTIPLEXOR.");
REGISTERED AS {nm-rule 63};
```

```
SSR68 RULE
      BEHAVIOR SSR68Behavior;
      PRIORITY 0;
      IF
            (? ? ?remota ZAK_D/I_AIS+IND ?destino ALARMA)
      THEN
            ("Grado de Severidad: 3"),
            ("Averia Asociada: AVERIA EN TERMINAL LEJANO O FALLO EN SEÑAL EN LA
                  ENTRADA DETECTADO  POR MUX ZAK D/I, ESTACION " ?remota),
            ("Recomendacion: REVISAR EQUIPOS ASOCIADOS Y/O TERMINALES LEJANOS.");
REGISTERED AS {nm-rule 68};

SSR70 RULE
      BEHAVIOR SSR70Behavior;
      PRIORITY 0;
      IF
            (? ? ?remota ZAK_D/I_FALLO_ALIM ?destino ALARMA)
      THEN
            ("Grado de Severidad: 3"),
            ("Averia Asociada: FALLO EN EL ALIMENTADOR O EN LA ALIMENTACION DEL
                  ALIMENTADOR DEL ZAK D/I, ESTACION " ?remota),
            ("Recomendacion: REVISAR ALIMENTADOR Y/O ALIMENTACION DEL ALIMENTADOR");
REGISTERED AS {nm-rule 70};

SSR71 RULE
      BEHAVIOR SSR71Behavior;
      PRIORITY 0;
      IF
        (OR
            (? ? ?remota ZAK_D/I_BFL3 ?destino ALARMA)
            (? ? ?remota ZAK_D/I_F.AL.TR. ?destino ALARMA))
      THEN
            ("Grado de Severidad: 3"),
            ("Averia Asociada: AVERIA EN EQUIPO ASOCIADO AL MULTIPLEXOR ZAK D/I,
                  ESTACION " ?remota "  O PROBLEMAS EN EL ENLACE ENTRE " ?remota " Y " ?destino),
            ("Recomendacion: REVISAR EQUIPOS ASOCIADOS AL MULTIPLEXOR ZAK 30?5.");
REGISTERED AS {nm-rule 71};
```

C.1.14 OrdenSuperiorMP31/X.

C.1.14.1 Definición de Clase de Objeto Gestionado.

```
ordenSuperiorMP31/X  MANAGED OBJECT CLASS
            DERIVED FROM  multiplexor;
REGISTERED AS {nm-MobjectClass 14}
```

C.1.14.2 Definición de Enlace de Nombre.

```
ordenSuperiorMP31/XNB NAME BINDING
            SUBORDINATE OBJECT CLASS ordenSuperiorMP31/X
                  AND SUBCLASSES;
            NAMED BY SUPERIOR OBJECT CLASS multiplexor
                  AND SUBCLASSES;
            WHIT ATTRIBUTE ordenSuperiorMP31/X Id;
REGISTERED AS {nm-nb 14}
```

C.1.15 OrdenTerceroMP31/X3.

C.1.15.1 Definición de Clase de Objeto Gestionado.

```
thirtyMultiplexorMP31/X3 MANAGED OBJECT CLASS
            DERIVED FROM ordenSuperiorMP31/X;
            CHARACTERIZED BY ordenSuperiorMP31/X3Package;
REGISTERED AS {nm-MobjectClass 15}
```

C.1.15.2 Definición de Enlace de Nombre.

```
thirtyMultiplexorMP31/X3NB NAME BINDING
            SUBORDINATE OBJECT CLASS ordenSuperiorMP31/X3
                  AND SUBCLASSES;
            NAMED BY SUPERIOR OBJECT CLASS ordenSuperiorMP31/X
                  AND SUBCLASSES;
            WHIT ATTRIBUTE thirtyMultiplexorMP31/X3Id;
REGISTERED AS {nm-nb 15}
```

C.1.15.3 Definición de Paquete.

```
ordenSuperiorMP31/XPackage PACKAGE
        ATTRIBUTES
                alarmaExterna                   GET,
                alarmaIndirecta                 GET,
                GenEq                           GET,
                estadoTributarios               GET,
                estadoAlimentacion              GET;
        NOTIFICATIONS
                MP31/X.3_EXTERNA,               -- alarma externa
                MP31/X.3_INDIRECTA              -- alarma indirecta
                MP31/X.3_GEN_EQ,                -- alarma GEN EQ
                MP31/X.3_TRIBUTARIOS,           -- fallo en tributarios
                MP31/X.3_COMUN_8MB/S,           -- fallo en interfaz principal
                MP31/X.3_OR_ALIM;               -- fallo alimentacion
        RULES
                SSR21A, SSR23A, SSR24A, SSR25A, SSR26, SSR27A;
REGISTERED AS {nm-package 9}
```

C.1.15.4 Definición de Reglas Expertas de Gestión.

```
SSR21A RULE
    BEHAVIOR SSR21ABehavior;
    PRIORITY 0;
    IF
        (? ? ?remota MP31/X.3_GEN_EQ ?destino ALARMA)
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: AVERIA EN MUX MP31/X.3, ESTACION " ?remota),
        ("Recomendacion: REVISAR MULTIPLEXOR.");
REGISTERED AS {nm-rule 21A};

SSR23A RULE
    BEHAVIOR SSR23ABehavior;
    PRIORITY 0;
    IF
        (?fecha ?h1 ?remota MP31/X.3_EXTERNA ?destino ALARMA)
        (OR(?fecha ?h2 ?remota CMF?38_BER_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
            (?fecha ?h3 ?remota CMF?38_BER_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00))
            (?fecha ?h4 ?remota CMF?38_F_AL_TR_C1 ?destino ALARMA & : (<(ABS(? ?h1 ?h4)) 1.00))
            (?fecha ?h5 ?remota CMF?38_F_AL_TR_C2 ?destino ALARMA & : (<(ABS(? ?h1 ?h5)) 1.00)))
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: FALLO EN EQUIPO CONECTADO AL MUX MP31/X.3, ESTACION " ?remota),
        ("Recomendacion: REVISAR EQUIPOS CONECTADOS AL MUX.");
REGISTERED AS {nm-rule 23A};

SSR24A RULE
    BEHAVIOR SSR24ABehavior;
    PRIORITY 0;
    IF
        (? ? ?remota MP31/X.3_INDIRECTA ? ALARMA)
    THEN
        ("Grado de Severidad: 3”),
        ("Averia Asociada: AVERIA EN TERMINAL LEJANO O CORTE DEL ENLACE,
                        DETECTADO POR ESTACION " ?remota),
        ("Recomendacion: REVISAR LISTA DE ALARMAS PARA LOCALIZAR EL EQUIPO QUE FALLA");
REGISTERED AS {nm-rule 24A};

SSR25A RULE
    BEHAVIOR SSR25ABehavior;
    PRIORITY 0;
    IF
        (? ? ?remota MP31/X.3_TRIBUTARIOS ?destino ALARMA)
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: FALTA DE SEÑAL EN TRIBUTARIOS DEL MP31/X.3, ESTACION " ?remota)
        ("Recomendacion: REVISAR TRIBUTARIOS O EQUIPOS ASOCIADOS.");
REGISTERED AS {nm-rule 25A};

SSR26 RULE
    BEHAVIOR SSR26Behavior;
    PRIORITY 0;
    IF
        (? ? ?remota MP31/X.3_COMUN_8MB/S ?destino ALARMA)
```

```
        THEN
            ("Grado de Severidad: 3"),
            ("Averia Asociada: FALTA DE SINCRONIZACION EN UN TRIBUTARIO DE 8MB/S,
                                ESTACION "  ?remota),
            ("Recomendacion: REVISAR TRIBUTARIOS DE 8MB/S.");
    REGISTERED AS {nm-rule 26};

    SSR27A RULE
        BEHAVIOR SSR27ABehavior;
        PRIORITY 0;
        IF
            (? ? ?remota MP31/X.3_OR_ALIM ?destino ALARMA)
        THEN
            ("Grado de Severidad: 3"),
            ("Averia Asociada: FALLO EN EL ALIMENTADOR O EN LA ALIMENTACION DEL
                ALIMENTADOR DEL MUX MP31/X.3, ESTACION " ?remota),
            ("Recomendacion: REVISAR ALIMENTADOR.");
    REGISTERED AS {nm-rule 27A};
```

C.1.16 OrdenSegundoMP31/X3.

C.1.16.1 Definición de Clase de Objeto Gestionado.

```
ordenSegundoMP31/X3 MANAGED OBJECT CLASS
        DERIVED FROM ordenSuperiorMP31/X;
        CHARACTERIZED BY ordenSegundoMP31/X3Package;
         -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 16}
```

C.1.16.2 Definición de Enlace de Nombre.

```
ordenSegundoMP31/X3 NAME BINDING
        SUBORDINATE OBJECT CLASS  ordenSegundoMP31/X3
            AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS ordenSuperiorMP31/X
            AND SUBCLASSES;
        WHIT ATTRIBUTE ordenSegundoMP31/X3Id;
REGISTERED AS {nm-nb 16}
```

C.1.16.3 Definición de Paquete.

```
ordenSuperiorMP31/X3 Package PACKAGE
        ATTRIBUTES
                alarmaExterna                   GET,
                alarmaIndirecta                 GET,
                GenEq                           GET,
                estadoTributarios               GET,
                estadoAlimentacion              GET;
NOTIFICATIONS
                MP31/2_EXTERNA,         -- alarma externa
                MP31/2_INDIRECTA        -- alarma indirecta
                MP31/2_GEN EQ,          -- alarma GEN EQ
                MP31/2_TRIBUTARIOS,     -- fallo en tributarios
                MP31/2_COMUN_8MB/S,     -- fallo en interfaz principal
                MP31/2_OR_ALIM;         -- fallo alimentacion
        RULES
                SSR21B,  SSR23B,
                SSRB24B, SSR25B, SSR27B;
REGISTERED AS {nm-package 10}
```

C.1.16.4 Definición de Reglas Expertas de Gestión.

```
SSR21B RULE
    BEHAVIOR SSR21BBehavior;
    PRIORITY 0;
    IF
        (? ? ?remota MP31/2_GEN_EQ ?destino ALARMA)
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: AVERIA EN MULTIPLEXOR MP31/2, ESTACION "),
        ("Recomendacion: REVISAR MULTIPLEXOR. ");
REGISTERED AS {nm-rule 21B};
```

```
SSR23B RULE
      BEHAVIOR SSR23BBehavior;
      PRIORITY 0;
      IF
        (?fecha ?h1 ?remota MP31/2_EXTERNA ?destino ALARMA)
        (OR(?fecha ?h2 ?remota CMF?38_BER_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
        (?fecha ?h3 ?remota CMF?38_BER_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00))
        (?fecha ?h4 ?remota CMF?38_F_AL_TR_C1 ?destino ALARMA & : (<(ABS(? ?h1 ?h4)) 1.00))
        (?fecha ?h5 ?remota CMF?38_F_AL_TR_C2 ?destino ALARMA & : (<(ABS(? ?h1 ?h5)) 1.00)))
      THEN
         ("Grado de Severidad: 3."),
         ("Averia Asociada: FALLO EN EQUIPO CONECTADO AL MUX MP31/2, ESTACION " ?remota),
         ("Recomendacion: REVISAR EQUIPOS CONECTADOS AL MUX.");
REGISTERED AS {nm-rule 23B};

SSR24B RULE
      BEHAVIOR SSR24BBehavior;
      PRIORITY 0;
      IF
           (? ? ?remota MP31/2_INDIRECTA ? ALARMA)
      THEN
           ("Grado de Severidad: 3"),
           ("Averia Asociada: AVERIA EN TERMINAL LEJANO O CORTE DEL ENLACE,
           DETECTADO
                    POR ESTACION " ?remota),
           ("Recomendacion: REVISAR LISTA DE ALARMAS PARA LOCALIZAR EL EQUIPO QUE  FALLA");
REGISTERED AS {nm-rule 24B};

SSR25B RULE
      BEHAVIOR SSR25BBehavior;
      PRIORITY 0;
      IF
           (? ? ?remota MP31/2_TRIBUTARIOS ?destino ALARMA)
      THEN
           ("Grado de Severidad: 3"),
           ("Averia Asociada: FALTA DE SEÑAL EN TRIBUTARIOS DEL MP31/2, ESTACION" ?remota),
           ("Recomendacion: REVISAR TRIBUTARIOS O EQUIPOS ASOCIADOS.");
REGISTERED AS {nm-rule 25B};

SSR27B RULE
      BEHAVIOR SSR27BBehavior;
      PRIORITY 0;
      IF
           (? ? ?remota MP31/2_OR_ALIM ?destino ALARMA)
      THEN
           ("Grado de Severidad: 3"),
           ("Averia Asociada: FALLO EN EL ALIMENTADOR O EN LA ALIMENTACION DEL
                ALIMENTADOR DEL MUX MP31/2, ESTACION " ?remota),
           ("Recomendacion: REVISAR ALIMENTADOR.");
REGISTERED AS {nm-rule 27B};
```

C.1.17 OrdenSuperiorMDM.

C.1.17.1 Definición de Clase de Objeto Gestionado.

```
ordenSuperiorMDM  MANAGED OBJECT CLASS
          DERIVED FROM  multiplexor;
          CHARACTERIZED BY ordenSuperiorMDMPackage;
          -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 17}
```

C.1.17.2 Definición de Enlace de Nombre.

```
ordenSuperiorMDMNB NAME BINDING
          SUBORDINATE OBJECT CLASS ordenSuperiorMDM
                AND SUBCLASSES;
          NAMED BY SUPERIOR OBJECT CLASS multiplexor
                AND SUBCLASSES;
          WHIT ATTRIBUTE ordenSuperiorMDMId;
REGISTERED AS {nm-nb 17}
```

C.1.17.3 Definición de Paquete.

```
ordenSuperiorMDMPackage PACKAGE
        ATTRIBUTES
                falloAlineamientoTrama          GET,
                tasaAltaErroresBFL3             GET,
                señalAIS                        GET,
                falloBIIR                       GET,
                perdidaSeñalTributaria          GET,
                desbordamientoMemoria           GET,
                falloAlimentacion               GET;
        NOTIFICATIONS
                MDM_8/2_F.AL.TR,        -- fallo alineamiento trama
                MDM_8/2_BFL,            -- tasa alta de errores BFL
                MDM_8/2_BFL3,           -- tasa alta de errores BFL3
                MDM_8/2_AIS,            -- recepción de señal AIS
                MDM_8/2_B11R,           -- alarma BIIR
                MDM_8/2_FALLO_TRIB,     -- perdida señal tributaria
                MDM_8/2_FALLO_ALIM;     -- fallo alimentación
        RULES
                SSR75, SSR76, SSR77,
                SSR78, SSR79;
REGISTERED AS {nm-package 11}
```

C.1.17.4 Definición de Reglas Expertas de Gestión.

```
SSR75 RULE
    BEHAVIOR SSR75Behavior;
    PRIORITY 0;
    IF
      (OR
        (? ? ?remota MDM_8/2_F.AL.TR. ?destino ALARMA)
        (? ? ?remota MDM_8/2_BFL3 ?destino ALARMA)
        (? ? ?remota MDM_8/2_BFL ?destino ALARMA))
    THEN
        ( "Grado de Severidad: 3"),
        ( "Averia Asociada: AVERIA EN EQUIPO ASOCIADO AL MUX MDM 8/2, ESTACION "
                ?remota " O PROBLEMAS EN EL ENLACE ENTRE " ?remota " Y " ?destino),
        ("Recomendacion: REVISAR ENLACE Y/O EQUIPOS ASOCIADOS.");
REGISTERED AS {nm-rule 75};

SSR76 RULE
    BEHAVIOR SSR76Behavior;
    PRIORITY 0;
    IF
        (? ? ?remota MDM_8/2_AIS ?destino ALARMA)
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: FALLO EN SEÑAL DE ENTRADA AL MDM 8/2, ESTACION " ?remota),
        ("Recomendacion: REVISAR ENLACE Y/O EQUIPOS ASOCIADOS.");
REGISTERED AS {nm-rule 76};

SSR77 RULE
    BEHAVIOR SSR77Behavior;
    PRIORITY 0;
    IF
        (? ? ?remota MDM_8/2_B11R ?destino ALARMA)
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: AVERIA EN TERMINAL LEJANO DETECTADA POR MDM 8/2,
                        ESTACION " ?remota )
        ("Recomendacion: REVISAR TERMINALES LEJANOS.");
REGISTERED AS {nm-rule 77};

SSR78 RULE
    BEHAVIOR SSR78Behavior;
    PRIORITY 0;
    IF
        (? ? ?remota MDM_8/2_FALLO_TRIB ?destino ALARMA)
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: AVERIA EN EQUIPO ASOCIADO AL MDM 8/2 DETECTADA POR FALTA
                DE SEÑAL EN UN CANAL DEL 2Mb/s, ESTACION " ?remota )
        ("Recomendacion: REVISAR EQUIPOS ASOCIADOS.");
REGISTERED AS {nm-rule 78};

SSR79 RULE
    BEHAVIOR SSR79Behavior;
```

```
        PRIORITY 0;
        IF
              (? ? ?remota MDM_8/2_FALLO_ALIM ?destino ALARMA)
        THEN
              ("Grado de Severidad: 3"),
              ("Averia Asociada: FALLO EN EL ALIMENTADOR DEL MDM 8/2 O EN LA ALIMENTACION
                    DE ALIMENTADOR, ESTACION " ?remota )
              ("Recomendacion: REVISAR ALIMENTADOR Y/O ALIMENTACION DEL ALIMENTADOR.");
REGISTERED AS {nm-rule 79};
```

C.1.18 OrdenSegundoMDM8/2.

C.1.18.1 Definición de Clase de Objeto Gestionado.

```
ordenSegundoMDM8/2 MANAGED OBJECT CLASS
        DERIVED FROM  ordenSuperiorMDM;
REGISTERED AS {nm-MobjectClass 18}
```

C.1.18.2 Definición de Enlace de Nombre.

```
ordenSegundoMDM8/2NB NAME BINDING
        SUBORDINATE OBJECT CLASS ordenSegundoMDM8/2
              AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS ordenSuperiorMDM
              AND SUBCLASSES;
        WHIT ATTRIBUTE ordenSegundoMDM8/2Id;
REGISTERED AS {nm-nb 18}
```

C.1.19 OrdenTerceroMDM34/8.

C.1.19.1 Definición de Clase de Objeto Gestionado.

```
ordenTerceroMDM34/8 MANAGED OBJECT CLASS
        DERIVED FROM ordenSuperiorMDM;
REGISTERED AS {nm-MobjectClass 19}
```

C.1.19.2 Definición de Enlace de Nombre.

```
ordenTerceroMDM34/8NB NAME BINDING
        SUBORDINATE OBJECT CLASS ordenTerceroMDM34/8
              AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS ordenTerceroMDM
              AND SUBCLASSES;
        WHIT ATTRIBUTE ordenTerceroMDM34/8Id;
REGISTERED AS {nm-nb 19}
```

C.1.20 AnalogicoSBB/CCTF2.

C.1.20.1 Definición de Clase de Objeto Gestionado.

```
analogicoSBB/CCTF2 MANAGED OBJECT CLASS
        DERIVED FROM  multiplexor;
        ATTRIBUTES
              perdidaPortadora                GET;
        NOTIFICATIONS
              CCTF/2 AL PORTADORA;      --perdida de señal portaora
        RULES
              SSR38;
REGISTERED AS {nm-MobjectClass 20}
```

C.1.20.2 Definición de Enlace de Nombre.

```
analogicoSBB/CCTF2NB NAME BINDING
        SUBORDINATE OBJECT CLASS analogicoSBB/CCTF2
              AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS multiplexor
              AND SUBCLASSES;
        WHIT ATTRIBUTE analogicoSBB/CCTF2Id;
REGISTERED AS {nm-nb 20}
```

C.1.20.3 Definición de Reglas Expertas de Gestión.

```
SSR38 RULE
      BEHAVIOR SSR38Behavior;
      PRIORITY 0;
      IF
            (? ? ?remota CCTF/2_AL_PORTADORA ALARMA)
      THEN
            ("Grado de Severidad: 3")
            ("Averia Asociada: FALLO EN CANAL DE SERVICIO DEL TRANSCEPTOR CTR190/7,
              ESTACION" ?remota),
```

```
            ("Recomendacion: REVISAR EL MULTIPLEXOR CCTF/2.")
REGISTERED AS {nm-rule 38};
```

C.1.21 EquipoTerminalLineaOptica.

C.1.21.1 Definición de Clase de Objeto Gestionado.

```
equipoTerminalLineaOptica  MANAGED OBJECT CLASS
        DERIVED FROM  sistema;
          ATTRIBUTES
                ModeloTerminal          GET,
                nombreTerminal          GET,
                orientacionTerminal     GET;
REGISTERED AS {nm-MobjectClass 21}
```

C.1.21.2 Definición de Enlace de Nombre.

```
equipoTerminalLineaOpticaNB NAME BINDING
        SUBORDINATE OBJECT CLASS equipoTerminalLineaOptica
            AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS sistema
            AND SUBCLASSES;
        WHIT ATTRIBUTE equipoTerminalLineaOptica Id;
REGISTERED AS {nm-nb 21}
```

C.1.22 TerminalETLO-34.

C.1.22.1 Definición de Clase de Objeto Gestionado.

```
terminalETLO-34 MANAGED OBJECT CLASS
        DERIVED FROM  equipoTerminalLineaOptica;
        CHARACTERIZED BY terminalETLO-34Package;
         -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 22}
```

C.1.22.2 Definición de Enlace de Nombre.

```
terminalETLO-34NB NAME BINDING
        SUBORDINATE OBJECT CLASS terminalETLO-34
            AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS equipoTerminalLineaOptica
            AND SUBCLASSES;
        WHIT ATTRIBUTE terminalETLO-34Id;
REGISTERED AS {nm-nb 22}
```

C.1.22.3 Definición de Paquete.

```
terminalETLO-34Package PACKAGE
        ATTRIBUTES
                falloAlineamientoRemota         GET,
                falloDegradacionLaser           GET,
                perdidaSeñalRX                  GET,
                perdidaSeñalTX                  GET,
                tasaAltaBFL3                    GET,
                tasaAltaBFL6                    GET,
                falloAlimentacion               GET;
NOTIFICATIONS
                FALLO_DE_ALIMENTACIÓN_REMOTA,
                FALLO_DE_DEGRADACIÓN_DEL_LÁSER,
                PÉRDIDA_DE_SEÑAL_ENTRANTE_RX,
                PÉRDIDA_DE_SEÑAL_SALIENTE TX,
                TASA_ALTA_DE_ERRORES_BFL3,
                TASA_DE_ERRORES_BFL6,
                FALLO_DE_ALIMENTACIÓN_LOCAL;
REGISTERED AS {nm-package 12}
```

C.1.23 Minilink.

Equipo empleado para realizar enlaces inalámbricos, utilizando altas frecuencias de radio para reemplazar conexiones por cable físico, como es el caso de *Wireless LAN's* , redes LAN inalámbricas. Esto agiliza enormemente la instalación de redes.

C.1.23.1 Definición de Clase de Objeto Gestionado.

```
minilink MANAGED OBJECT CLASS
        DERIVED FROM  sistema;
        CHARACTERIZED BY minilinkPackage;
         -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 23}
```

C.1.23.2 Definición de Enlace de Nombre.

```
minilinkNB NAME BINDING
        SUBORDINATE OBJECT CLASS minilink
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS sistema
                AND SUBCLASSES;
        WHIT ATTRIBUTE minilink Id;
REGISTERED AS {nm-nb 23}
```

C.1.23.3 Definición de Paquete.

```
minilinkPackage PACKAGE
        ATTRIBUTES
                PerdidaSeñalTX          GET,
                perdidaSeñalRX          GET,
                falloAlimentacion       GET;
NOTIFICATIONS
                MINILINK_TX,            -- perdida señal saliente TX
                MINILINK_RX,            -- perdida señal entrante RX
                MINILINK_RA,            -- fallo alimentacion estacion remota
                MINILINK_FALLO_ALIM;    -- fallo alimentacion
        RULES
                SSR80, SSR81, SSR82;
REGISTERED AS {nm-package 13}
```

C.1.23.4 Definición de Reglas Expertas de Gestión.

```
SSR80 RULE
    BEHAVIOR SSR80Behavior;
    PRIORITY 0;
    IF
        (? ? ?remota MINILINK_TX ?destino ALARMA)
    THEN
         ("Grado de Severidad: 3")
         ("Averia Asociada: AVERIA EN TRANSMISOR MINILINK, ESTACION " ?remota )
         ("Recomendacion: REVISAR TRANSMISOR.")
REGISTERED AS {nm-rule 80};

SSR81 RULE
    BEHAVIOR SSR81Behavior;
    PRIORITY 0;
    IF
        (? ? ?remota MINILINK_RX ?destino ALARMA)
    THEN
         ("Grado de Severidad: 3"),
         ("Averia Asociada: AVERIA EN RECEPTOR MINILINK, ESTACION " ?remota )
         ("Recomendacion: REVISAR RECEPTOR.");
REGISTERED AS {nm-rule 81};

SSR82 RULE
    BEHAVIOR SSR82Behavior;
    PRIORITY 0;
    IF
      (OR
        (? ? ?remota MINILINK_RA ?destino ALARMA)
        (? ? ?remota MINILINK_FALLO_ALIM ?destino ALARMA))
    THEN
         ("Grado de Severidad: 3"),
         ("Averia Asociada: FALLO EN LA ALIMENTACION DEL TRANSCEPTOR MINILINK,
                        ESTACION" ?remota )
         ("Recomendacion: REVISAR ALIMENTACION DEL TRANSCEPTOR.");
REGISTERED AS {nm-rule 82};
```

C.1.24 RadioEquipo.

C.1.24.1 Definición de Clase de Objeto Gestionado.

```
radioEquipo MANAGED OBJECT CLASS
        DERIVED FROM  sistema;
          ATTRIBUTES
                modeloEquipo            GET,
                nombreEquipo            GET,
                orientacionEquipo       GET,
                velocidadtransEquipo    GET,
                configuracionEquipo     GET,
                frecuenciaTXCanal1      GET,
                frecuenciaRXCanal1      GET,
```

```
                frecuenciaTXCanal2          GET,
                frecuenciaRXCanal2          GET;
REGISTERED AS {nm-MobjectClass 24}
```

C.1.24.2 Definición de Enlace de Nombre.

```
radioEquipoNB NAME BINDING
        SUBORDINATE OBJECT CLASS radioEquipo
            AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS sistema
            AND SUBCLASSES;
        WHIT ATTRIBUTE radioEquipoId;
REGISTERED AS {nm-nb 24}
```

C.1.25 RadioTransceptor.

C.1.25.1 Definición de Clase de Objeto Gestionado.

```
radioTransceptorRT21 MANAGED OBJECT CLASS
        DERIVED FROM  radioEquipo;
        CHARACTERIZED BY radioTransnceptorRT21Package;
         -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 25}
```

C.1.25.2 Definición de Enlace de Nombre.

```
radioTransceptorRT21NB NAME BINDING
        SUBORDINATE OBJECT CLASS radioTransceptorRT21
            AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS radioEquipo
            AND SUBCLASSES;
        WHIT ATTRIBUTE radioTransceptorRT21Id;
REGISTERED AS {nm-nb 25}
```

C.1.25.3 Definición de Paquete.

```
radioTransnceptorRT21Package PACKAGE
        ATTRIBUTES
                potenciaSalidaTX1           GET,
                potenciaEntradaTX2          GET,
                potenciaRecepcionRX1        GET,
                potenciaRecepcionRX2        GET,
                estadoDSTX1                 GET,
                estadoDSTX2                 GET,
                estadoTrasnmisorTX1         GET,
                estadoTrasnmisorTX2         GET,
                estadoReceptorRX1           GET,
                estadoReceptorRX2           GET,
                estadoFuente1               GET,
                estadoFuente2               GET,
                SIATXtramaBB-I              GET,
                SIATXtramaBB-II             GET,
                SIARXtramaBB-I              GET,
                SIARXtramaBB-II             GET,
                estadoMultiplexorTX1        GET,
                estadoMultiplexorTX2        GET,
                estadoMultiplexorRX1        GET,
                estadoMultiplexorRX2        GET,
                tasaErroresBER1             GET,
                tasaErroresBER2             GET,
                indicadorPrealarmaBER1      GET,
                indicadorPrealarmaBER2      GET,
                automaticoManualTX          GET,
                automaticoManualRX          GET,
                estadoAleatorizatorTX1      GET,
                estadoAleatorizatorTX2      GET,
                estadoDesaleatorizatorRX1   GET,
                estadoDesaleatorizatorRX2   GET;
        ATTRIBUTES GROUP
                 medidas,
                 powerState,
                 sPU1,
                 SPU2;
        NOTIFICATIONS
                SPU_1_AIS_TX,               -- insercion SIA TX trama BB-I
                SPU_2_AIS_TX,               -- insercion SIA TX trama BB-II
                SPU_1_AIS_RX,               -- recepcion SIA RX trama BB-I
                SPU_2_AIS_RX,               -- recepcion SIA RX trama BB-II
                SPU_1_F_SC_MX_TX1,          -- fallo TX1 o generador sincronismo
                SPU_1_F_SC_MX_TX2,          -- fallo TX2 o generador sincronismo
```

```
                SPU_1_F_SC_DMX_RX1,      -- fallo demultiplexor RX1
                SPU_1_F_SC_DMX_RX2,      -- fallo demultiplexor RX2
                SPU_1_BER_1,             -- tasa alta errores BER1
                SPU_1_BER_2,             -- tasa alta errores BER2
                SPU_1_PREBER_1,          -- prealarma BER1
                SPU_1_PREBER_2,          -- prealarma BER2
                SPU_1_C_A/M_TX,          -- conmutacion manual TX
                SPU_1_C_A/M_RX,          -- conmutacion manual RX
                SPU_2_ALEATOR_TX1,       -- fallo aleatorizador TX1
                SPU_2_ALEATOR_TX2,       -- fallo aleatorizador TX2
                SPU_2_DESALEA_RX1,       -- fallo desaleatorizador RX1
                SPU_2_DESALEA_RX2,       -- fallo desaleatorizador RX2
                RT21_PTX1,               -- bajada de señal de salida TX1
                RT21_PTX2;               -- bajada de señal de salida TX2
        RULES
                SSR44,  SSR45A,  SSR45B,
                SSR46,  SSR47A,  SSR47B,
                SSR48A, SSRB48B, SSR49A,
                SSR49B, SSR49C,  SSR49D,
                SSR50A, SSR50B,  SSR51A,
                SSRB51B,SSRB51C, SSRB51D,
                SSR90A, SSR90B;
REGISTERED AS {nm-package 8}
```

C.1.25.4 Definición de Reglas Expertas de Gestión.

```
SSR44 RULE
    BEHAVIOR SSR44Behavior;
    PRIORITY 0;
    IF
        (OR
            (? ? ?remota SPU_1_AIS_TX ?destino ALARMA)
            (? ? ?remota SPU_2_AIS_TX ?destino ALARMA))
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: FALLO EN SEÑAL DE ENTRADA AL EQUIPO DE TRANSMISION RT21/3,
                        ESTACION " ?remota),
        ("Recomendacion: REVISAR LISTA DE ALARMAS PRESENTES PARA DETECTAR EL
            EQUIPO QUE  PRODUCE EL FALLO.”),
REGISTERED AS {nm-rule 44};

SSR45A RULE
    BEHAVIOR SSR45ABehavior;
    PRIORITY 0;
    IF
        (?fecha ?h1 ?remota SPU_1_F_SC_MX_TX1 ?destino ALARMA)
        (NOT(?fecha ?h2 ?remota RT21/3_F_ALIM_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
    THEN
        ("Grado de Severidad: 2"),
        ("Averia Asociada: AVERIA EN CANAL 1 EN SPU 1, ESTACION " ?remota),
        ("Recomendacion: REVISAR SPU 1.”);
REGISTERED AS {nm-rule 45A};

SSR45B RULE
    BEHAVIOR SSR45BBehavior;
    PRIORITY 0;
    IF
        (?fecha ?h1 ?remota SPU_1_F_SC_MX_TX2 ?destino ALARMA)
        (NOT(?fecha ?h2 ?remota RT21/3_F_ALIM_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
    THEN
        ("Grado de Severidad: 2"),
        ("Averia Asociada: AVERIA EN CANAL 2 EN SPU 1, ESTACION " ?remota),
        ("Recomendacion: REVISAR SPU 1.”);
REGISTERED AS {nm-rule 45B};

SSR46 RULE
    BEHAVIOR SSR46Behavior;
    PRIORITY 0;
    IF
        (OR
            (? ? ?remota SPU_1_AIS_RX ?destino ALARMA)
            (? ? ?remota SPU_2_AIS_RX ?destino ALARMA))
    THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: FALLO EN RECEPCION EN RT21, ESTACION " ?remota),
        ("Recomendacion: REVISAR EQUIPO SPU, REVISAR RT21 Y REVISAR EQUIPOS COLATERALES.”);
REGISTERED AS {nm-rule 46};
```

```
SSR47A RULE
      BEHAVIOR SSR47ABehavior;
      PRIORITY 0;
      IF
        (?fecha ?h1 ?remota SPU_1_PREBER_1 ?destino ALARMA)
        (NOT(?fecha ?h2 ?remota SPU_1_PREBER_1 ?destino DESAPARECE_ALARMA
           & : (<(ABS(? ?h1 ?h2)) 1.00)))
           (NOT(?fecha ?h3 ?remota SPU_1_BER_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
           (NOT(CONTROL_DEL_FLUJO))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: DEGRADACION EN EL CANAL 1 DEL ENLACE RT21 ENTRE "
                              ?remota  " Y " ?destino),
            ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 47A};

SSR47B RULE
      BEHAVIOR SSR47Behavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota SPU_1_PREBER_2 ?destino ALARMA)
           (NOT(?fecha ?h2 ?remota SPU_1_PREBER_2 ?destino DESAPARECE_ALARMA
           & : (<(ABS(? ?h1 ?h2)) 1.00)))
           (NOT(?fecha ?h3 ?remota SPU_1_BER_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
           (NOT(CONTROL_DEL_FLUJO))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: DEGRADACION EN EL CANAL 2 DEL ENLACE RT21 ENTRE "
                              ?remota " Y " ?destino),
            ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 47B};

SSR48A RULE
      BEHAVIOR SSR48ABehavior;
      PRIORITY -1;
      IF
            ?A<?(?fecha ?h1 ?remota SPU_1_PREBER_1 ?destino ALARMA)
            ?B<?(?fecha ?h2 ?remota SPU_1_PREBER_1 ?destino DESAPARECE_ALARMA
               & : (<(ABS(? ?h1 ?h2)) 0.20))
            ?C<?(?fecha ?h3 ?remota SPU_1_PREBER_1 ?destino ALARMA)
            ?D<?(?fecha ?h4 ?remota SPU_1_PREBER_1 ?destino DESAPARECE_ALARMA
                & : (<(ABS(? ?h3 ?h4)) 0.20))
           (TEST(NOT(equal ?A ?C)))
           (TEST(NOT(equal ?B ?D)))
           (NOT(?fecha ?h5 ?remota SPU_1_BER_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h5)) 3.00)))
           (NOT(CONTROL_DEL_FLUJO))
      THEN
           (retract ?A),
           (retract ?B),
           (retract ?C),
           (retract ?D),
           ("Grado de Severidad 2"),
           ("Averia Asociada: POSIBLE PERTURBACION EN EL ENLACE RT21 ENTRE " ?remota " Y " ?destino),
           ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 48A};

SSR48B RULE
      BEHAVIOR SSR48BBehavior;
      PRIORITY -1;
      IF
       ?A<?(?fecha ?h1 ?remota SPU_1_PREBER_2 ?destino ALARMA)
       ?B<?(?fecha ?h2 ?remota SPU_1_PREBER_2 ?destino DESAPARECE_ALARMA & : (<(ABS(? ?h1 ?h2)) 0.20))
       ?C<?(?fecha ?h3 ?remota SPU_1_PREBER_2 ?destino ALARMA)
       ?D<?(?fecha ?h4 ?remota SPU_1_PREBER_2 ?destino DESAPARECE_ALARMA &: (<(ABS(? ?h3 ?h4)) 0.20))
           (TEST(NOT(equal ?A ?C)))
           (TEST(NOT(equal ?B ?D)))
           (NOT(?fecha ?h5 ?remota SPU_1_BER_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h5)) 3.00)))
           (NOT(CONTROL_DEL_FLUJO))
      THEN
           (retract ?A),
           (retract ?B),
           (retract ?C),
           (retract ?D),
           ("Grado de Severidad: 2"),
           ("Averia Asociada: POSIBLE PERTURBACION EN EL ENLACE RT21 ENTRE " ?remota " Y " ?destino),
           ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 48B};
```

```
SSR49A RULE
      BEHAVIOR SSR49ABehavior;
      PRIORITY 0;
      IF
        (?fecha ?h1 ?remota SPU_1_BER_1 ?destino ALARMA)
        (NOT(?fecha ?h2 ?remota SPU_1_BER_1 ?destino DESAPARECE_ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
        (NOT(?fecha ?h3 ?remota SPU_1_F_SC_DMX_RX1 ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
      THEN
            ("Grado de Severidad: 3"),
            ("Averia Asociada: ALTA TASA DE ERRORES EN EL CANAL 1 DEL ENLACE RT21/3  ENTRE "
                              ?remota " Y " ?destino),
            ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 49A};

SSR49B RULE
      BEHAVIOR SSR49BBehavior;
      PRIORITY 0;
      IF
        (?fecha ?h1 ?remota SPU_1_BER_2 ?destino ALARMA)
        (NOT(?fecha ?h2 ?remota SPU_1_BER_2 ?destino DESAPARECE_ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
        (NOT(?fecha ?h3 ?remota SPU_1_F_SC_DMX_RX2 ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
      THEN
            ("Grado de Severidad: 3"),
            ("Averia Asociada: ALTA TASA DE ERRORES EN EL CANAL 2 DEL ENLACE RT21/3  ENTRE "
                              ?remota " Y " ?destino),
            ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 49B};

SSR49C RULE
      BEHAVIOR SSR49CBehavior;
      PRIORITY -2;
      IF
         ?A<?(?fecha ?h1 ?remota SPU_1_BER_1 ?destino ALARMA)
         ?B<?(?fecha ?h2 ?remota SPU_1_BER_1 ?destino DESAPARECE_ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
         ?C<?(?fecha ?h3 ?remota SPU_1_BER_1 ?destino ALARMA)
         ?D<?(?fecha ?h4 ?remota SPU_1_BER_1 ?destino DESAPARECE_ALARMA & : (<(ABS(? ?h3 ?h4)) 1.00))
           (TEST(NOT(equal ?A ?C)))
           (TEST(NOT(equal ?B ?D)))
      THEN
            (retract ?A),
            (retract ?B),
            (retract ?C),
            (retract ?D),
            (assert (CONTROL_DEL_FLUJO)),
            ("Grado de Severidad: -2"),
            ("Averia Asociada: PROBLEMAS EN EL MEDIO DE TRANSMISION,
                        CORRESPONDIENTE  AL ENLACE RT21 ENTRE " ?remota " Y " ?destino),
             ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 49C};

SSR49D RULE
      BEHAVIOR SSR49DBehavior;
      PRIORITY -2;
      IF
         ?A<?(?fecha ?h1 ?remota SPU_2_BER_2 ?destino ALARMA)
         ?B<?(?fecha ?h2 ?remota SPU_2_BER_2 ?destino DESAPARECE_ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
          ?C<?(?fecha ?h3 ?remota SPU_2_BER_2 ?destino ALARMA)
          ?D<?(?fecha ?h4 ?remota SPU_2_BER_2 ?destino DESAPARECE_ALARMA & : (<(ABS(? ?h3 ?h4)) 1.00))
           (TEST(NOT(equal ?A ?C)))
           (TEST(NOT(equal ?B ?D)))
      THEN
            (retract ?A),
            (retract ?B),
            (retract ?C),
            (retract ?D),
            (assert (CONTROL_DEL_FLUJO)),
            ("Grado de Severidad: -2"),
            ("Averia Asociada: PROBLEMAS EN EL MEDIO DE TRANSMISION,
                        CORRESPONDIENTE AL ENLACE RT21 ENTRE " ?remota " Y " ?destino),
             ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 49D};
```

```
SSR50A RULE
      BEHAVIOR SSR50ABehavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota SPU_1_F_SC_DMX_RX1 ?destino ALARMA)
            (NOT(?fecha ?h2 ?remota SPU_2_DESALEA_RX1 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
            (NOT(?fecha ?h3 ?destino SPU_1_F_SC_MX_TX1 ?remota ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
            (NOT(?fecha ?h4 ?destino RT21/3_F_DAT_TX1 ?remota ALARMA & : (<(ABS(? ?h1 ?h4)) 1.00)))
            (NOT(?fecha ?h5 ?destino RT21/2_F_DAT_TX1 ?remota ALARMA & : (<(ABS(? ?h1 ?h5)) 1.00)))
            (NOT(?fecha ?h6 ?destino RT21/3_TX1_PWR/VCO ?remota ALARMA & : (<(ABS(? ?h1 ?h6)) 1.00)))
            (NOT(?fecha ?h7 ?destino RT21/2_TX1_PWR/VCO ?remota ALARMA & : (<(ABS(? ?h1 ?h7)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: AVERIA EN CANAL 1 EN SPU 1, ESTACION " ?remota),
            ("Recomendacion: REVISAR EL ENLACE, INCLUYENDO EL RECEPTOR RT21.");
REGISTERED AS {nm-rule 50A};

SSR50B RULE
      BEHAVIOR SSR50BBehavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota SPU_1_F_SC_DMX_RX2 ?destino ALARMA)
            (NOT(?fecha ?h2 ?remota SPU_2_DESALEA_RX2 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
            (NOT(?fecha ?h3 ?destino SPU_1_F_SC_MX_TX2 ?remota ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
            (NOT(?fecha ?h4 ?destino RT21/3_F_DAT_TX2 ?remota ALARMA & : (<(ABS(? ?h1 ?h4)) 1.00)))
            (NOT(?fecha ?h5 ?destino RT21/2_F_DAT_TX2 ?remota ALARMA & : (<(ABS(? ?h1 ?h5)) 1.00)))
            (NOT(?fecha ?h6 ?destino RT21/3_TX2_PWR/VCO ?remota ALARMA & : (<(ABS(? ?h1 ?h6)) 1.00)))
            (NOT(?fecha ?h7 ?destino RT21/2_TX2_PWR/VCO ?remota ALARMA & : (<(ABS(? ?h1 ?h7)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: AVERIA EN CANAL 2 EN SPU 1, ESTACION " ?remota),
            ("Recomendacion: REVISAR EL ENLACE, INCLUYENDO EL RECEPTOR RT21.");
REGISTERED AS {nm-rule 50B};

SSR51A RULE
      BEHAVIOR SSR51ABehavior;
      PRIORITY 0;
      IF    (?fecha ?h1 ?remota SPU_2_ALEATOR_TX1 ?destino ALARMA)
            (NOT(?fecha ?h2 ?remota RT21/3_F_ALIM_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN CANAL 1 DE SPU 2, ESTACION " ?remota),
            ("Recomendacion: REVISAR SPU 2.");
REGISTERED AS {nm-rule 51A};

SSR51B RULE
      BEHAVIOR SSR51BBehavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota SPU_2_ALEATOR_TX2 ?destino ALARMA)
            (NOT(?fecha ?h2 ?remota RT21/3_F_ALIM_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN CANAL 2 DE SPU 2, ESTACION " ?remota),
            ("Recomendacion: REVISAR SPU 2.");
REGISTERED AS {nm-rule 51B};

SSR51C RULE
      BEHAVIOR SSR51CBehavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota SPU_2_DESALEA_RX1 ?destino ALARMA)
            (NOT(?fecha ?h2 ?destino SPU_2_ALEATOR_TX1 ?remota ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
            (NOT(?fecha ?h3 ?remota SPU_1_F_SC_DMX_RX1 ?destino ALARMA  & : (<(ABS(? ?h1 ?h3)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: AVERIA EN CANAL 1 DE SPU 2, ESTACION " ?remota),
            ("Recomendacion: REVISAR SPU 2.");
REGISTERED AS {nm-rule 51C};

SSR51D RULE
      BEHAVIOR SSR51DBehavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota SPU_2_DESALEA_RX2 ?destino ALARMA)
            (NOT(?fecha ?h2 ?destino SPU_2_ALEATOR_TX2 ?remota ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
            (NOT(?fecha ?h3 ?remota SPU_1_F_SC_DMX_RX2 ?destino ALARMA  & : (<(ABS(? ?h1 ?h3)) 1.00)))
```

```
        THEN
                ("Grado de Severidad: 2"),
                ("Averia Asociada: AVERIA EN CANAL 2 DE SPU 2, ESTACION " ?remota),
                ("Recomendacion: REVISAR SPU 2.");
REGISTERED AS {nm-rule 51D};

SSR90A RULE
        BEHAVIOR SSR90ABehavior;
        PRIORITY 0;
        IF
                (?fecha ?h1 ?remota SPU_1_F_SC_DMX_RX1 ?destino ALARMA)
                (?fecha ?h2 ?remota SPU_2_DESALEA_RX1 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
        THEN
                ("Grado de Severidad: 2"),
                ("Averia Asociada: FALLO EN EL CANAL 1 DEL ENLACE RT21 ENTRE " ?remota  " Y " ?destino),
                ("Recomendacion: REVISAR EL ENLACE");
REGISTERED AS {nm-rule 90A};

SSR90B RULE
        BEHAVIOR SSR90BBehavior;
        PRIORITY 0;
        IF
                (?fecha ?h1 ?remota SPU_1_F_SC_DMX_RX2 ?destino ALARMA)
                (?fecha ?h2 ?remota SPU_2_DESALEA_RX2 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
        THEN
                ("Grado de Severidad: 2"),
                ("Averia Asociada: FALLO EN EL CANAL 2 DEL ENLACE RT21 ENTRE " ?remota  " Y " ?destino),
                ("Recomendacion: REVISAR EL ENLACE");
REGISTERED AS {nm-rule 90B};
```

C.1.26 RadioTransceptorRT21/3.

C.1.26.1 Definición de Clase de Objeto Gestionado.

```
radioTransnceptorRT21/3 MANAGED OBJECT CLASS
        DERIVED FROM  radioEquipo;
        CHARACTERIZED BY radioTransnceptorRT21/3Package;
         -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 26}
```

C.1.26.2 Definición de Enlace de Nombre.

```
radioTransnceptorRT21/3NB NAME BINDING
        SUBORDINATE OBJECT CLASS radioTransnceptorRT21/3
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS radioEquipo
                AND SUBCLASSES;
        WHIT ATTRIBUTE radioTransnceptorRT21/3Id;
REGISTERED AS {nm-nb 26}
```

C.1.26.3 Definición de Paquete.

```
radioTransnceptorRT21/3Package PACKAGE
                NOTIFICATIONS
                RT21/3_F_DAT_TX1,          -- perdida señal digital TX1
                RT21/3_F_DAT_TX2,          -- perdida señal digital TX2
                RT21/3_PTX1,               -- fallo en transmision TX1
                RT21/3_PTX2,               -- fallo en transmision TX2
                RT21/3_PRX1,               -- fallo en RX1
                RT21/3_PRX2,               -- fallo en RX2
                RT21/3_TX1_PWR/VCO,        -- fallo en la unidad de potencia 1
                RT21/3_TX2_PWR/VCO,        -- fallo en la unidad de potencia 2
                RT21/3_RX1_VCO/DEM,        -- fallo en lel oscilador local 1
                RT21/3_RX2_VCO/DEM,        -- fallo en lel oscilador local 2
                RT21/3_F_ALIM_1,           -- fallo alimentacion 1
                RT21/3_F_ALIM_2,           -- fallo alimentacion 2
                RT21_PRX1,                 -- bajada de señal de recepcion TX1
                RT21_PRX2;                 -- bajada de señal de recepcion TX2
        RULES
                SSR52A, SSR52B, SSR53A,
                SSR53B, SSR53C, SSR53D,
                SSR54A, SSR54B, SSR54C,
                SSR54D, SSR55A, SSR55B,
                SSR56A, SSR57A, SRB57B;
REGISTERED AS {nm-package 15}
```

C.1.26.4 Definición de Reglas Expertas de Gestión.

```
SSR52A RULE
      BEHAVIOR 52ABehavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota RT21/3_F_DAT_TX1 ?destino ALARMA)
            (NOT(?fecha ?h2 ?remota RT21/3_F_ALIM_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN SPU O FALLO EN CANAL 1 DEL RT21/3, ESTACION " ?remota),
            ("Recomendacion: REVISAR EQUIPO SPU Y/O EQUIPO DE TRANSMISION RT21/3.");
REGISTERED AS {nm-rule 52A};

SSR52B RULE
      BEHAVIOR SSR52BBehavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota RT21/3_F_DAT_TX2 ?destino ALARMA)
            (NOT(?fecha ?h2 ?remota RT21/3_F_ALIM_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN SPU O FALLO EN CANAL 2 DEL RT21/3, ESTACION " ?remota),
            ("Recomendacion: REVISAR EQUIPO SPU Y/O EQUIPO DE TRANSMISION RT21/3.");
REGISTERED AS {nm-rule 52B};

SSR53ARULE
      BEHAVIOR SSR53ABehavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota RT21/3_TX1_PWR/VCO ?destino ALARMA)
            (NOT(?fecha ?h2 ?remota RT21/3_F_ALIM_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN LA UNIDAD DE POTENCIA O FALLO EN LA UNIDAD DE
                  OSCILACIÓN CORRESPONDIENTE AL CANAL 1, ESTACION " ?remota),
            ("Recomendacion: REVISAR UNIDAD DE POTENCIA Y/O UNIDAD DE OSCILACION DEL RT21/3.");
REGISTERED AS {nm-rule 53A};

SSR53B RULE
      BEHAVIOR SSR53BBehavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota RT21/3_TX2_PWR/VCO ?destino ALARMA)
            (NOT(?fecha ?h2 ?remota RT21/3_F_ALIM_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN LA UNIDAD DE POTENCIA O FALLO EN LA UNIDAD DE
                  OSCILACION CORRESPONDIENTE AL CANAL 2, ESTACION " ?remota),
            ("Recomendacion: REVISAR UNIDAD DE POTENCIA Y/O UNIDAD DE OSCILACION DEL RT21/3.");
REGISTERED AS {nm-rule 53B};

SSR53C RULE
      BEHAVIOR SSR53CBehavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota RT21/2_TX1_PWR/VCO ?destino ALARMA)
            (NOT(?fecha ?h2 ?remota RT21/2_F_ALIM_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN LA UNIDAD DE POTENCIA O FALLO EN LA UNIDAD DE
                  OSCILACION CORRESPONDIENTE AL CANAL 1, ESTACION " ?remota),
            ("Recomendacion: REVISAR UNIDAD DE POTENCIA Y/O UNIDAD DE OSCILACION DEL RT21/2.");
REGISTERED AS {nm-rule 53C};

SSR53D RULE
      BEHAVIOR SSR53DBehavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota RT21/2_TX2_PWR/VCO ?destino ALARMA)
            (NOT(?fecha ?h2 ?remota RT21/2_F_ALIM_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN LA UNIDAD DE POTENCIA O FALLO EN LA UNIDAD DE
                  OSCILACION CORRESPONDIENTE AL CANAL 2, ESTACION " ?remota),
            ("Recomendacion: REVISAR UNIDAD DE POTENCIA Y/O UNIDAD DE OSCILACION DEL RT21/2.");
REGISTERED AS {nm-rule 53D};
```

```
SSR54A RULE
      BEHAVIOR SSR54ABehavior;
      PRIORITY 0;
      IF
        (?fecha ?h1 ?remota RT21/3_RX1_VCO/DEM ?destino ALARMA)
        (OR(?fecha ?h2 ?remota RT21/3_PRX1 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR
            ?num & : (<(ABS(? ?h1 ?h2)) 1.00))
        (?fecha ?h2 ?remota RT21_PRX1 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num
                & : (<(ABS(? ?h1 ?h2)) 1.00)))
            (NOT(?fecha ?h3 ?remota RT21/3_F_ALIM_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN EL CANAL 1 DEL OSCILADOR LOCAL DEL RT21/3 O FALLO
                EN LA SEÑAL DE ENTRADA, ESTACION " ?remota),
            ("Recomendacion: REVISAR OSCILADOR DE RT21/3 Y/O COMPROBAR LISTA DE ALARMAS
                PRESENTES PARA DETECTAR EL EQUIPO QUE PROVOCA EL FALLO.");
REGISTERED AS {nm-rule 54A};

SSR54B RULE
      BEHAVIOR SSR54BBehavior;
      PRIORITY 0;
      IF
        (?fecha ?h1 ?remota RT21/3_RX2_VCO/DEM ?destino ALARMA)
        (OR(?fecha ?h2 ?remota RT21/3_PRX2 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR
                ?num  & : (<(ABS(? ?h1 ?h2)) 1.00))
        (?fecha ?h2 ?remota RT21_PRX2 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num
                    & : (<(ABS(? ?h1 ?h2)) 1.00)))
        (NOT(?fecha ?h3 ?remota RT21/3_F_ALIM_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN EL CANAL 2 DEL OSCILADOR LOCAL DEL RT21/3 O FALLO
                EN LA SEÑAL DE ENTRADA, ESTACION " ?remota),
            ("Recomendacion: REVISAR OSCILADOR DE RT21/3 Y/O COMPROBAR LISTA DE ALARMAS
                PRESENTES  PARA DETECTAR EL EQUIPO QUE PROVOCA EL FALLO.");
REGISTERED AS {nm-rule 54B};

SSR54C RULE
      BEHAVIOR SSR54CBehavior;
      PRIORITY 0;
      IF    (?fecha ?h1 ?remota RT21/2_RX1_VCO/DEM ?destino ALARMA)
            (?fecha ?h2 ?remota RT21/2_PRX1 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR
            ?num & : (<(ABS(? ?h1 ?h2)) 1.00))
            (NOT(?fecha ?h3 ?remota RT21/2_F_ALIM_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN EL CANAL 1 DEL OSCILADOR LOCAL DEL RT21/2 O FALLO
                EN LA SEÑAL DE ENTRADA, ESTACION " ?remota),
            ("Recomendacion: REVISAR OSCILADOR DE RT21/2 Y/O COMPROBAR LISTA DE ALARMAS
                PRESENTES PARA DETECTAR EL EQUIPO QUE PROVOCA EL FALLO.");
REGISTERED AS {nm-rule 54C};

SSR54D RULE
      BEHAVIOR SSR54DBehavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota RT21/2_RX2_VCO/DEM ?destino ALARMA)
            (?fecha ?h2 ?remota RT21/2_PRX2 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR
                ?num  & : (<(ABS(? ?h1 ?h2)) 1.00))
            (NOT(?fecha ?h3 ?remota RT21/2_F_ALIM_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN EL CANAL 2 DEL OSCILADOR LOCAL DEL RT21/2 O FALLO
                EN LA SEÑAL DE ENTRADA, ESTACION " ?remota),
            ("Recomendacion: REVISAR OSCILADOR DE RT21/2 Y/O COMPROBAR LISTA DE ALARMAS
                PRESENTES PARA DETECTAR EL EQUIPO QUE PROVOCA EL FALLO.");
REGISTERED AS {nm-rule 54D};

SSR55A RULE
      BEHAVIOR SSR55ABehavior;
      PRIORITY 0;
      IF  (? ? ?remota RT21/3_F_ALIM_1 ?destino ALARMA)
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN FUENTE DE ALIMENTACION 1 DEL RT21/3 O FALLO EN LA
                ALIMENTACION DE LA FUENTE, ESTACION " ?remota),
            ("Recomendacion: REVISAR FUENTE DE ALIMENTACION Y/O ALIMENTACION DE LA FUENTE.");
REGISTERED AS {nm-rule 55A};
```

```
SSR55B RULE
        BEHAVIOR SSR55BBehavior;
        PRIORITY 0;
        IF
                (? ? ?remota RT21/3_F_ALIM_2 ?destino ALARMA)
        THEN
                ("Grado de Severidad: 2"),
                ("Averia Asociada: FALLO EN FUENTE DE ALIMENTACION 2 DEL RT21/3 O FALLO EN LA
                        ALIMENTACION DE LA FUENTE, ESTACION " ?remota),
                ("Recomendacion: REVISAR FUENTE DE ALIMENTACION Y/O ALIMENTACION DE LA FUENTE.");
REGISTERED AS {nm-rule 55B};

SSR56A RULE
        BEHAVIOR SSR56ABehavior;
        PRIORITY 0;
        IF
           (OR
              (?fecha ?h1 ?remota RT21/3_PTX1 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?)
              (?fecha ?h1 ?remota RT21_PTX1 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?))
              (NOT(?fecha ?h2 ?remota RT21/3_F_ALIM_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
        THEN
                ("Grado de Severidad: 2"),
                ("Averia Asociada: FALLO EN EL CANAL 1 DEL TRANSMISOR RT21, ESTACION " ?remota),
                ("Recomendacion: REVISAR EL TRANSMISOR RT21.");
REGISTERED AS {nm-rule 56A};

SSR57A RULE
  BEHAVIOR SSR57ABehavior;
   PRIORITY 0;
   IF
     (OR(?fecha ?h1 ?remota RT21/3_PRX1 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num)
     (?fecha ?h1 ?remota RT21_PRX1 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num))
      (NOT(?fecha ?h2 ?remota RT21/3_RX1_VCO/DEM ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
       (NOT(?fecha ?h3 ?destino RT21/3_TX1_PWR/VCO ?remota ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
        (NOT(?fecha ?h4 ?destino RT21_PTX1 ?remota MEDIDA_PASA_A_ALARMA _POR_LIMITE_INFERIOR ?
                & : (<(ABS(? ?h1 ?h4)) 1.00)))
        THEN
                ("Grado de Severidad: 2"),
                ("Averia Asociada:   BAJADA DE NIVEL EN LA SEÑAL CORRESPONDIENTE AL CANAL 1
                        DEL RECEPTOR RT21, ESTACION " ?remota),
                ("Recomendacion: REVISAR EL ENLACE ENTRE " ?remota " Y " ?destino);
REGISTERED AS {nm-rule 57A};

SSR57B RULE
   BEHAVIOR SSR57Behavior;
    PRIORITY 0;
    IF
    (OR(?fecha ?h1 ?remota RT21/3_PRX2 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num)
     (?fecha ?h1 ?remota RT21_PRX2 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num))
            (NOT(?fecha ?h2 ?remota RT21/3_RX2_VCO/DEM ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
            (NOT(?fecha ?h3 ?destino RT21/3_TX2_PWR/VCO ?remota ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
            (NOT(?fecha ?h4 ?destino RT21_PTX2 ?remota MEDIDA_PASA_A_ALARMA _POR_LIMITE_INFERIOR
                ? & : (<(ABS(? ?h1 ?h4)) 1.00)))
        THEN
                ("Grado de Severidad: 3"),
                ("Averia Asociada: BAJADA DE NIVEL EN LA SEÑAL CORRESPONDIENTE AL CANAL 2 DEL
                        RECEPTOR RT21, ESTACION " ?remota),
                ("Recomendacion: REVISAR EL ENLACE ENTRE " ?remota " Y " ?destino);
REGISTERED AS {nm-rule 57B};
```

C.1.27 RadioTransceptorRT21/2.

C.1.27.1 Definición de Clase de Objeto Gestionado.

```
radioTransnceptorRT21/2 MANAGED OBJECT CLASS
        DERIVED FROM  radioEquipo;
        CHARACTERIZED BY radioTransnceptorRT21/2Package;
          -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 27}
```

C.1.27.2 Definición de Enlace de Nombre.

```
radioTransnceptorRT21/2NB NAME BINDING
        SUBORDINATE OBJECT CLASS radioTransnceptorRT21/2
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS radioEquipo
                AND SUBCLASSES;
        WHIT ATTRIBUTE radioTransnceptorRT21/2Id;
```

```
REGISTERED AS {nm-nb 27}
```

C.1.27.3 Definición de Paquete.

```
radioTransnceptorRT21/2Package PACKAGE
        NOTIFICATIONS
                T21/2_F_DAT_TX1,        -- perdida señal digital TX1
                T21/2_F_DAT_TX2,        -- perdida señal digital TX2
                RT21/2_PTX1,            -- fallo en transmisión TX1
                RT21/2_PTX2,            -- fallo en transmisión TX2
                RT21/2_PRX1,            -- fallo en RX1
                RT21/2_PRX2,            -- fallo en RX2
                RT21/2_F_ALIM_1,        -- fallo alimentación 1
                RT21/2_F_ALIM_2;        -- fallo alimentación 2
        RULES
                SSR52C, SSR52D, SSR53C,
                SSR53D, SSR54C, SSR54D,
                SSR55C, SSR55D, SSR56C,
                SSR56D, SSR57C, SSR57D;
REGISTERED AS {nm-package 16}
```

C.1.27.4 Definición de Reglas Expertas de Gestión.

```
SSR52C RULE
    BEHAVIOR SSR52CBehavior;
    PRIORITY 0;
    IF   (?fecha ?h1 ?remota RT21/2_F_DAT_TX1 ?destino ALARMA)
         (NOT(?fecha ?h2 ?remota RT21/2_F_ALIM_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
    THEN
          ("Grado de Severidad: 2"),
          ("Averia Asociada: FALLO EN SPU O FALLO EN CANAL 1 DEL RT21/2, ESTACION " ?remota),
          ("Recomendacion: REVISAR EQUIPO SPU Y/O EQUIPO DE TRANSMISION RT21/3.");
REGISTERED AS {nm-rule 52C};

SSR52D RULE
    BEHAVIOR SSR52DBehavior;
    PRIORITY 0;
    IF
        (?fecha ?h1 ?remota RT21/2_F_DAT_TX2 ?destino ALARMA)
        (NOT(?fecha ?h2 ?remota RT21/2_F_ALIM_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
    THEN
          ("Grado de Severidad: 2"),
          ("Averia Asociada: FALLO EN SPU O FALLO EN CANAL 2 DEL RT21/2, ESTACION " ?remota),
          ("Recomendacion: REVISAR EQUIPO SPU Y/O EQUIPO DE TRANSMISION RT21/3.");
REGISTERED AS {nm-rule 52D};

SSR53C RULE
    BEHAVIOR SSR53CBehavior;
    PRIORITY 0;
    IF
        (?fecha ?h1 ?remota RT21/2_TX1_PWR/VCO ?destino ALARMA)
        (NOT(?fecha ?h2 ?remota RT21/2_F_ALIM_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
    THEN
          ("Grado de Severidad: 2"),
          ("Averia Asociada: FALLO EN LA UNIDAD DE POTENCIA O FALLO EN LA UNIDAD DE
                  OSCILACION CORRESPONDIENTE AL CANAL 1, ESTACION " ?remota),
          ("Recomendacion: REVISAR UNIDAD DE POTENCIA Y/O UNIDAD DE OSCILACION DEL RT21/2.");
REGISTERED AS {nm-rule 53C};

SSR53D RULE
    BEHAVIOR SSR53DBehavior;
    PRIORITY 0;
    IF
        (?fecha ?h1 ?remota RT21/2_TX2_PWR/VCO ?destino ALARMA)
        (NOT(?fecha ?h2 ?remota RT21/2_F_ALIM_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
    THEN
          ("Grado de Severidad: 2"),
          ("Averia Asociada: FALLO EN LA UNIDAD DE POTENCIA O FALLO EN LA UNIDAD DE
                   OSCILACION CORRESPONDIENTE AL CANAL 2, ESTACION " ?remota),
          ("Recomendacion: REVISAR UNIDAD DE POTENCIA Y/O UNIDAD DE OSCILACION DEL T21/2.");
REGISTERED AS {nm-rule 53D};
```

```
SSR54C RULE
      BEHAVIOR SSR54CBehavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota RT21/2_RX1_VCO/DEM ?destino ALARMA)
            (?fecha ?h2 ?remota RT21/2_PRX1 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR
                  ?num & : (<(ABS(? ?h1 ?h2)) 1.00))
            (NOT(?fecha ?h3 ?remota RT21/2_F_ALIM_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN EL CANAL 1 DEL OSCILADOR LOCAL DEL RT21/2 O FALLO EN
                  LA SEÑAL  DE ENTRADA, ESTACION " ?remota),
            ("Recomendacion: REVISAR OSCILADOR DE RT21/2 Y/O COMPROBAR LISTA DE ALARMAS
                  PRESENTES PARA DETECTAR EL EQUIPO QUE PROVOCA EL FALLO.");
REGISTERED AS {nm-rule 54C};

SSR54D RULE
      BEHAVIOR SSR54DBehavior;
      PRIORITY 0;
      IF
       (?fecha ?h1 ?remota RT21/2_RX2_VCO/DEM ?destino ALARMA)
       (?fecha ?h2 ?remota RT21/2_PRX2 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num
                  & : (<(ABS(? ?h1 ?h2)) 1.00))
            (NOT(?fecha ?h3 ?remota RT21/2_F_ALIM_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN EL CANAL 2 DEL OSCILADOR LOCAL DEL RT21/2 O FALLO EN
                  LA SEÑAL DE ENTRADA, ESTACION " ?remota),
            ("Recomendacion: REVISAR OSCILADOR DE RT21/2 Y/O COMPROBAR LISTA DE ALARMAS
                  PRESENTES PARA DETECTAR EL EQUIPO QUE PROVOCA EL FALLO.");
REGISTERED AS {nm-rule 54D};

SSR56C RULE
      BEHAVIOR SSR56CBehavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota RT21/2_PTX1 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?)
            (NOT(?fecha ?h2 ?remota RT21/2_F_ALIM_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN EL CANAL 1 DEL TRANSMISOR RT21/2, ESTACION " ?remota),
            ("Recomendacion: REVISAR EL TRANSMISOR RT21/2.");
REGISTERED AS {nm-rule 56C};

SSR56D RULE
      BEHAVIOR SSR56DBehavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota RT21/2_PTX2 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?)
            (NOT(?fecha ?h2 ?remota RT21/2_F_ALIM_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN EL CANAL 2 DEL TRANSMISOR RT21/2, ESTACION " ?remota),
            ("Recomendacion: REVISAR EL TRANSMISOR RT21/2.");
REGISTERED AS {nm-rule 56D};

SSR57C RULE
      BEHAVIOR SSR57CBehavior;
      PRIORITY 0;
      IF
        (?fecha ?h1 ?remota RT21/2_PRX1 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num)
            (NOT(?fecha ?h2 ?remota RT21/2_RX1_VCO/DEM ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
            (NOT(?fecha ?h3 ?destino RT21/2_TX1_PWR/VCO ?remota ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
            (NOT(?fecha ?h4 ?destino RT21_PTX1 ?remota MEDIDA_PASA_A_ALARMA _POR_LIMITE_INFERIOR
            ? & : (<(ABS(? ?h1 ?h4)) 1.00)))
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: BAJADA DE NIVEL EN LA SEÑAL CORRESPONDIENTE AL CANAL 1 DEL
                  RECEPTOR RT21/2, ESTACION " ?remota),
            ("Recomendaciones: REVISAR EL ENLACE ENTRE " ?remota " Y " ?destino);
REGISTERED AS {nm-rule 57C};
```

```
SSR57D RULE
 BEHAVIOR SSR57DBehavior;
  PRIORITY 0;
 IF
 (?fecha ?h1 ?remota RT21/2_PRX2 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num)
          (NOT(?fecha ?h2 ?remota RT21/2_RX2_VCO/DEM ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
          (NOT(?fecha ?h3 ?destino RT21/2_TX2_PWR/VCO ?remota ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
          (NOT(?fecha ?h4 ?destino RT21_PTX2 ?remota MEDIDA_PASA_A_ALARMA _POR_LIMITE_INFERIOR
          ? & : (<(ABS(? ?h1 ?h4)) 1.00)))
      THEN
           ("Grado de Severidad: 2"),
           ("Averia Asociada: BAJADA DE NIVEL EN LA SEÑAL CORRESPONDIENTE AL CANAL 2 DEL
                 RECEPTOR RT21/2, ESTACION " ?remota),
           ("Recomendaciones: REVISAR EL ENLACE ENTRE " ?remota " Y " ?destino);
REGISTERED AS {nm-rule 57D};
```

C.1.28 RadioTransceptorCTR190.

C.1.28.1 Definición de Clase de Objeto Gestionado.

```
radioTransnceptorCTR190  MANAGED OBJECT CLASS
          DERIVED FROM  radioEquipo;
          CHARACTERIZED BY RadioTransnceptorCTR190Package;
           -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 28}
```

C.1.28.2 Definición de Enlace de Nombre.

```
sistemaNB NAME BINDING
          SUBORDINATE OBJECT CLASS radioTransnceptorCTR190
                AND SUBCLASSES;
          NAMED BY SUPERIOR OBJECT CLASS radioEquipo
                AND SUBCLASSES;
          WHIT ATTRIBUTE radioTransnceptorCTR190Id;
REGISTERED AS {nm-nb 28}
```

C.1.28.3 Definición de Paquete.

```
RadioTransnceptorCTR190Package PACKAGE
ATRIBUTTES
          potenciaRecepcionRX2          GET,
                sentido                 GET,
                velocidadTransmision    GET,
                estadoTransmisorC1      GET,
                estadoTransmisorC2      GET,
                estadoReceptorC1        GET,
                estadoReceptorC2        GET,
                estadoAlientacion1      GET,
                estadoAlientacion2      GET,
                insBitOr                GET,
                extBitOr                GET,
                tasaErroresBER1         GET,
                tasaErroresBER2         GET,
                indicadorPrealarmaBER1  GET,
                indicadorPrealarmaBER2  GET,
                falloAlineacionTramaC1  GET,
                falloAlineacionTramaC2  GET;
          NOTIFICATIONS
                CTR190/7_TX_C1,         -- fallo en transmision TX C1
                CTR190/7_TX_C2,         -- fallo en transmision TX C2
                CTR190/7_RX_C1,         -- fallo en recepcion RX C1
                CTR190/7_RX_C2,         -- fallo en recepcion RX C2
                CTR190/7_F_ALIM 1,      -- fallo alimentacion 1
                CTR190/7_F_ALIM 2,      -- fallo alimentacion 2
                CMF?38_INS_BIT OR,      -- alarma de insercion de BIT OR
                CMF?38_EXT_BIT OR,      -- alarma de extracción de BIT OR
                CMF?38_BER_1,           -- alta tasa de errores BER1,
                CMF?38_PREBER_1,        -- prealarma errores BER1,
                CMF?38_BER_2,           -- alta tasa de errores BER2,
                CMF?38_PREBER_2,        -- prealarma errores BER2,
                CMF?38_F_AL_TR_C1,      -- perdida de alineacion trama C1
                CMF?38_F_AL_TR_C2;      -- perdida de alineacion trama C2
          RULES
                SSR1A,  SSR1B,  SSR2,
                SSR3A,    SSR5,  SSR6,   SSR8,
                SSR10A,  SSR11A,
                SSR11B, SSR12C, SSR12D;
                SSR15,  SSRB16, SSR17;
REGISTERED AS {nm-package 17}
```

C.1.28.4 Definición de Reglas Expertas de Gestión.

```
SSR1A RULE
      BEHAVIOR SSR1ABehavior;
      PRIORITY 0;
      IF
        (?fecha ?h1 ?remota CTR190/7_TX_C1 ?destino ALARMA)
      THEN
        ("Compruebe el valor de la medida CTR190/7_PRX1 " ?destino"."),
        ("Si dicho valor es menor de ?60, entonces:"),
        ("Grado de Severidad: 3"),
        ("Averia Asociada: FALLO EN MODULO DE TRANSMISION CTR190/7, CANAL 1, ESTACION " ?remota),
        ("Recomendacion: REVISION DEL TRANSCEPTOR.");
REGISTERED AS {nm-rule 1A};

SSR1B RULE
      BEHAVIOR SSR1BBehavior;
      PRIORITY 0;
      IF
        (?fecha ?h1 ?remota CTR190/7_TX_C2 ?destino ALARMA)
      THEN
        ("Compruebe el valor de la medida CTR190/7_PRX2 "?remota".),
        ("Si dicho valor es menor que ?60, entonces:"t),
        ("Grado de Severidad: 3" t),
        ("Averia Asociada: FALLO EN EL MODULO DE TRANSMISION DE CTR190/7, CANAL 2,
           ESTACION " ?remota),
        ("Recomendacion: REVISOR DEL TRANSCEPTOR CTR190/7.");
REGISTERED AS {nm-rule 1B};

SSR2 RULE
      BEHAVIOR SSR2Behavior;
      PRIORITY 0;
      IF
       (OR
        (AND(?fecha ?h1 ?remota CTR190/7_TX_C1 ?destino ALARMA)
        (?fecha ?h2 ?remota CTR190/7_PRX1 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num
                    & : (< (ABS(? ?h1 ?h2))1.0)))
        (AND(?fecha ?h1 ?remota CTR190/7_TX_C2 ?destino ALARMA)
          (?fecha ?h2 ?remota CTR190/7_PRX2 ?destino MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR
                    ?num & : (< (ABS(? ?h1 ?h2)) 1.0))))
     THEN
           ("Grado de Severidad: 3 “),
           ("Averia Asociada: FALLO EN EL OSCILADOR LOCAL, ESTACION " ?remota t " O FALLO EN
                    MODULO DE TRANSMISION, ESTACION " ?destino),
           ("Recomendacion: REVISION DEL TRANSCEPTOR CTR190/7.");
REGISTERED AS {nm-rule 2};

SSR3A RULE
      BEHAVIOR SSR3ABehavior;
      PRIORITY 0;
      IF
        (?fecha ?h1 ?remota CTR190/7_TX_C1 ?destino ALARMA)
        (?fecha ?h2 ?remota CTR190/7_F_ALIM_1 ?destino ALARMA & : (< (ABS(? ?h1 ?h2)) 1.00))
      THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: AVERIA EN FUENTE DE ALIMENTACION 1 DE CTR190/7, ESTACION " ?remota),
        ("Recomendacion: REVISAR LA FUENTE Y EN SU CASO CAMBIARLA.");
REGISTERED AS {nm-rule 3A};

SSR5 RULE
      BEHAVIOR SSR5Behavior;
      PRIORITY 0;
      IF
        (?fecha ?h1 ?remota CTR190/7_F_ALIM_1 ?destino ALARMA)
        (?fecha ?h2 ?remota CTR190/7_F_ALIM_2 ?destino ALARMA & : (< (ABS(? ?h1 ?h2)) 1.0))
      THEN
        ("Grado de Severidad: 5"),
        ("Averia Asociada: AVERIA EN LAS DOS FUENTES DE ALIMENTACION DEL TRANSCEPTOR
           CTR190/7 DE ESTACION " ?remota " O AVERIA EN LA ALIMENTACION DE AMBAS FUENTES"),
        ("Recomendacion: REVISAR LAS FUENTES DE ALIMENTACION DEL TRANSCEPTOR CTR190/7
           Y/O SU ALIMENTACION.");
REGISTERED AS {nm-rule 5};
```

```
SSR6 RULE
        BEHAVIOR SSR6Behavior;
        PRIORITY 0;
        IF
          (? ? ?remota CMF?38_INS_BIT_OR ? ALARMA)
        THEN
          ("Grado de Severidad: 3" t),
          ("Averia Asociada: FALLO EN EL MODULO DE INSERCION DE BITS, ESTACION " ?remota t),
          ("Recomendacion: REVISAR MODULO DE INSERCION DE BITS." t )
REGISTERED AS {nm-rule 6};
SSR8 RULE
        BEHAVIOR SSR8Behavior;
        PRIORITY 0;
        IF
          (?fecha ?h1 ?remota CMF?38_EXT_BIT_OR ?destino ALARMA)
          (NOT(?fecha ?h2 ?destino CMF?38_INS_BIT_OR ?remota ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
        THEN
           ("Grado de Severidad: 3")
           ("Averia Asociada: FALLO EN EL EQUIPO DE EXTRACCION DE BIT, ESTACION " ?remota),
           ("Recomendacion: REVISAR EQUIPO DE EXTRACCION DE BIT.");
REGISTERED AS {nm-rule 8};

SSR10A RULE
        BEHAVIOR SSR10ABehavior;
        PRIORITY 0;
        IF
          (?fecha ?h1 ?remota CMF?38_PREBER_1 ?destino ALARMA)
          (NOT(?fecha ?h2 ?remota CMF?38_PREBER_1 ?destino DESAPARECE_ALARMA
                              & : (<(ABS(? ?h1 ?h2)) 1.00)))
          (NOT(?fecha ?h3 ?remota CCA?34_AIS_DE_BB ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
          (NOT(?fecha ?h4 ?remota CMF?38_BER_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h4)) 1.00)))
          (NOT(CONTROL_DEL_FLUJO))
        THEN
           ("Grado de Severidad: 3"),
           ("Averia Asociada: DEGRADACION EN EL CANAL 1 DEL ENLACE CTR190/7 ENTRE " ?remota
             " Y " ?destino),
           ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 10A};

SSR11A RULE
  BEHAVIOR SSR11ABehavior;
   PRIORITY -1;
   IF
    ?A<?(?fecha ?h1 ?remota CMF?38_PREBER_1 ?destino ALARMA)
    ?B<?(?fecha ?h2 ?remota CMF?38_PREBER_1 ?destino DESAPARECE_ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
    ?C<?(?fecha ?h3 ?remota CMF?38_PREBER_1 ?destino ALARMA)
    ?D<?(?fecha ?h4 ?remota CMF?38_PREBER_1 ?destino DESAPARECE_ALARMA & : (<(ABS(? ?h3 ?h4)) 1.00))
        (TEST(NOT(equal ?A ?C)))
        (TEST(NOT(equal ?B ?D)))
        (NOT(?fecha ?h5 ?remota CMF?38_BER_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h5)) 3.00)))
        (NOT(CONTROL_DEL_FLUJO))
        THEN
         (retract ?A),
         (retract ?B),
         (retract ?C),
         (retract ?D),
          ("Grado de Severidad: 2"),
          ("Averia Asociada: POSIBLE PERTURBACION EN EL ENLACE CTR190/7 ENTRE " ?remota " Y " ?destino),
          ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 11A};

SSR11B RULE
        BEHAVIOR SSR11BBehavior;
        PRIORITY -1;
        IF
          ?A<?(?fecha ?h1 ?remota CMF?38_PREBER_2 ?destino ALARMA)
          ?B<?(?fecha ?h2 ?remota CMF?38_PREBER_2 ?destino DESAPARECE_ALARMA
                          & : (<(ABS(? ?h1 ?h2)) 1.00))
          ?C<?(?fecha ?h3 ?remota CMF?38_PREBER_2 ?destino ALARMA)
          ?D<?(?fecha ?h4 ?remota CMF?38_PREBER_2 ?destino DESAPARECE_ALARMA
                          & : (<(ABS(? ?h3 ?h4)) 1.00))
          (TEST(NOT(equal ?A ?C)))
          (TEST(NOT(equal ?B ?D)))
          (NOT(?fecha ?h5 ?remota CMF?38_BER_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h5)) 3.00)))
          (NOT(CONTROL_DEL_FLUJO))
```

```
        THEN
         (retract ?A),
         (retract ?B),
         (retract ?C),
         (retract ?D),
          ("Grado de Severidad: 2"),
          ("Averia Asociada: POSIBLE PERTURBACION EN EL ENLACE CTR190/7 ENTRE " ?remota " Y " ?destino),
          ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 11B};

SSR12C RULE
        BEHAVIOR SSR12CBehavior;
        PRIORITY -2;
        IF
         ?A<?(?fecha ?h1 ?remota CMF?38_BER_1 ?destino ALARMA)
         ?B<?(?fecha ?h2 ?remota CMF?38_BER_1 ?destino DESAPARECE_ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
         ?C<?(?fecha ?h3 ?remota CMF?38_BER_1 ?destino ALARMA)
         ?D<?(?fecha ?h4 ?remota CMF?38_BER_1 ?destino DESAPARECE_ALARMA & : (<(ABS(? ?h3 ?h4)) 1.00))
         (TEST(NOT(equal ?A ?C)))
         (TEST(NOT(equal ?B ?D)))
        THEN
         (retract ?A),
         (retract ?B),
         (retract ?C),
         (retract ?D),
         (assert (CONTROL_DEL_FLUJO)),
          ("Grado de Severidad: 3"),
           ("Averia Asociada: PROBLEMAS EN EL MEDIO DE TRANSMISION, CORRESPONDIENTE AL
               ENLACE CTR190/7 ENTRE " ?remota " Y " ?destino),
           ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 12C};

SSR12D RULE
        BEHAVIOR SSR12DBehavior;
        PRIORITY -2;
        IF
         ?A<?(?fecha ?h1 ?remota CMF?38_BER_2 ?destino ALARMA)
         ?B<?(?fecha ?h2 ?remota CMF?38_BER_2 ?destino DESAPARECE_ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
         ?C<?(?fecha ?h3 ?remota CMF?38_BER_2 ?destino ALARMA)
         ?D<?(?fecha ?h4 ?remota CMF?38_BER_2 ?destino DESAPARECE_ALARMA & : (<(ABS(? ?h3 ?h4)) 1.00))
         (TEST(NOT(equal ?A ?C)))
         (TEST(NOT(equal ?B ?D)))
        THEN
         (retract ?A),
         (retract ?B),
         (retract ?C),
         (retract ?D),
         (assert (CONTROL_DEL_FLUJO)),
          ("Grado de Severidad: 3"),
          ("Averia Asociada: PROBLEMAS EN EL MEDIO DE TRANSMISION,  CORRESPONDIENTE AL
               ENLACE CTR190/7 ENTRE " ?remota " Y " ?destino),
          ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 12D};

SSR15 RULE
        BEHAVIOR SSR15Behavior;
        PRIORITY 0;
        IF
         (OR
           ((?fecha ?h1 ?remota CTR190/7_RX_C1 ?destino ALARMA)
               (NOT(?fecha ?h2 ?remota CTR190/7_RX_C2 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))))
           ((?fecha ?h1 ?remota CTR190/7_RX_C2 ?destino ALARMA)
               (NOT(?fecha ?h2 ?remota CTR190/7_RX_C1 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))))
        THEN
           ("Grado de Severidad: 3"),
           ("Averia Asociada: FALLO EN RECEPCION, ESTACION " ?remota),
           ("Recomendacion: REVISAR EL RECEPTOR CTR190/7.");
REGISTERED AS {nm-rule 15};

errorTransmissionCTR190/7 RULE
        BEHAVIOR errorTransmissionCTR190/7Behavior;
        PRIORITY 0;
        IF
              (?fecha ?h1 ?remota CTR190/7_TX_C1 ?destino ALARMA)
              (?fecha ?h2 ?destino CTR190/7_RX_C1 ?remota ALARMA  & : (<(ABS(? ?h1 ?h2)) 1.00))
```

```
        THEN
                ("Grado de Severidad: 3"),
                ("Averia Asociada: FALLO EN CANAL 1 DEL MODULO DE TRANSMISION, ESTACION " ?remota),
                ("Recomendacion: REVISAR EL TRANSMISOR CTR190/7.");
REGISTERED AS {nm-rule 16A};

SSR16B RULE
        BEHAVIOR SSR16BBehavior;
        PRIORITY 0;
        IF
                (?fecha ?h1 ?remota CTR190/7_TX_C2 ?destino ALARMA)
                (?fecha ?h2 ?destino CTR190/7_RX_C2 ?remota ALARMA  & : (<(ABS(? ?h1 ?h2)) 1.00))
        THEN
                ("Grado de Severidad: 3"),
                ("Averia Asociada: FALLO EN CANAL2 DEL MODULO DE TRANSMISION, ESTACION " ?remota),
                ("Recomendacion: REVISAR EL TRANSMISOR CTR190/7.");
REGISTERED AS {nm-rule 16B};

SSR17 RULE
        BEHAVIOR SSR17Behavior;
        PRIORITY 0;
        IF
                (?fecha ?h1 ?remota CTR190/7_RX_C1 ?destino ALARMA)
                (?fecha ?h2 ?remota CTR190/7_RX_C2 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
        THEN
                ("Grado de Severidad: 3"),
                ("Averia Asociada: FALLO EN ENLACE CTR190/7 ENTRE " ?remota " Y " ?destino),
                ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 17};
```

C.1.29 EquipoCCA34.

C.1.29.1 Definición de Clase de Objeto Gestionado.

```
equipoCCA34 MANAGED OBJECT CLASS
        DERIVED FROM  sistema;
        CHARACTERIZED BY equipoCCAA34Package;
          -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 29}
```

C.1.29.2 Definición de Enlace de Nombre.

```
equipoCCA34NB NAME BINDING
        SUBORDINATE OBJECT CLASS equipoCCA34
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS sistema
                AND SUBCLASSES;
        WHIT ATTRIBUTE equipoCCA34Id;
REGISTERED AS {nm-nb 29}
```

C.1.29.3 Definición de Paquete.

```
equipoCCAA34Package PACKAGE
        ATTRIBUTES
                estadoAutomaticoManual          GET,
                estadoConmutacion               GET,
                indicadorRecepcionAIS           GET,
                estadoEquipo1                   GET,
                estadoEquipo1                   GET;
        ACTIONS
                conmutacionAutomaticoManual,
                conmutacionC1/C2;
        NOTIFICATIONS
                CCA-34_C_A/M,           -- cambio automatico/manual
                CCA-34_C_C1/C2,         -- cambio C1/C2
                CCA-34_AIS_DE_BB,       -- recepcion de alarma AIS de BB
                CCA-34_EQ_1_ON,         -- equipo 1 en linea
                CCA-34_EQ_2_ON;         -- equipo 2 en linea
        PARAMETERS
                codigoConmutacionAutoamtico, codigoComuntacionC1/C2;
        RULES
                SSR14;
REGISTERED AS {nm-package 18}
```

C.1.29.4 Definición de Reglas Expertas de Gestión.

```
SSR14 RULE
      BEHAVIOR SSR14Behavior;
      PRIORITY 0;
      IF
       (?fecha ? ?remota CCA?34_AIS_DE_BB ?destino ALARMA)
      THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: PROBLEMAS EN EL ENLACE CTR190/7 ENTRE " ?remota " Y " ?destino),
        ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 14};
```

C.1.30 EquipoCCA33.

C.1.30.1 Definición de Clase de Objeto Gestionado.

```
equipoCCA33 MANAGED OBJECT CLASS
            DERIVED FROM  sistema;
            CHARACTERIZED BY equipoCCAA33Package;
              -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 30}
```

C.1.30.2 Definición de Enlace de Nombre.

```
equipoCCA33NB NAME BINDING
            SUBORDINATE OBJECT CLASS equipoCCA33
                  AND SUBCLASSES;
            NAMED BY SUPERIOR OBJECT CLASS sistema
                  AND SUBCLASSES;
            WHIT ATTRIBUTE equipoCCA33Id;
REGISTERED AS {nm-nb 30}
```

C.1.30.3 Definición de Paquete.

```
equipoCCCA33Package PACKAGE
            ATTRIBUTES
                  estadoAutomaticoManual          GET,
                  estadoC1/C2                     GET;
                  indicadorRecepcionAIS           GET,
                  estadoEquipo1                   GET,
                  estadoEquipo1                   GET,
                  estadoAlimentacion1             GET,
                  estadoAlimentacion2             GET,
                  ausenciaDatosC1                 GET,
                  ausenciaDatosC2                 GET;
            ACTIONS
                  conmutarC1/C2;
            NOTIFICATIONS
                  CCA-33_C_A/M,           -- cambio automatico/manual
                  CCA-33_C_C1/C2,         -- cambio C1/C2
                  CCA-33_AIS_DE_BB,       -- recepcion de alarma AIS de BB
                  CCA-33_EQ_1_ON,         -- equipo 1 en linea
                  CCA-33_EQ_2_ON,         -- equipo 2 en linea
                  CCA-33_F_ALIM_1,        -- fallo en alimentacion 1
                  CCA-33_F_ALIM_2,        -- fallo en alimentacion 2
                  CCA-33_F_DAT_C1,        -- perdida de datos en entrada C1
                  CCA-33_F_DAT_C2,        -- perdida de datos en entrada C2
                  CCA-33_F_DAT_HDB3;      -- fallo entrada al sistema de conmutacion
            PARAMETERS
                  codigoConmutacionAutoamtico, codigoComuntacionC1/C2;
            RULES
                  SSR14, SSR72,
                  SSR73A, SSR73B, SSR74;
REGISTERED AS {nm-package 19}
```

C.1.30.4 Definición de Reglas Expertas de Gestión.

```
SSR72 RULE
      BEHAVIOR SSR72Behavior;
      PRIORITY 0;
      IF  (OR
            (? ? ?remota CCA?33_F_ALIM_1 ?destino ALARMA)
            (? ? ?remota CCA?33_F_ALIM_2 ?destino ALARMA))
      THEN
            ("Grado de Severidad: 3"),
             ("Averia Asociada: FALLO EN EL ALIMENTADOR DEL CCA?33 O FALLO EN LA ALIMENTACION
                  DEL ALIMENTADOR, ESTACION " ?remota),
             ("Recomendacion: REVISAR ALIMENTADOR Y/O ALIMENTACION DEL ALIMENTADOR.");
REGISTERED AS {nm-rule 72};
```

```
SSR73A RULE
      BEHAVIOR SSR73ABehavior;
      PRIORITY 0;
      IF    (? ? ?remota CCA?33_F_DAT_C1 ?destino ALARMA)
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: FALLO EN RECEPCION DE SEÑAL EN EQUIPO RADIO CCA?33,
                  ESTACION " ?remota),
            ("Recomendacion: REVISAR CANAL 1.");
REGISTERED AS {nm-rule 73A};

SSR73B RULE
      BEHAVIOR SSR73BBehavior;
      PRIORITY 0;
      IF      (? ? ?remota CCA?33_F_DAT_C2 ?destino ALARMA)
      THEN
            ("Grado de Severidad: 3"),
            ("Averia Asociada: FALLO EN RECEPCION DE SENAL EN EQUIPO RADIO CCA?33,
                  ESTACION " ?remota),
            ("Recomendacion: REVISAR CANAL 2.");
REGISTERED AS {nm-rule 73B};

SSR74 RULE
      BEHAVIOR SSR74Behavior;
      PRIORITY 0;
      IF
            (? ? ?remota CCA?33_F_DAT_HDB3 ?destino ALARMA)
      THEN
            ("Grado de Severidad: 3"),
            ("Averia Asociada: FALLO EN SENALES DE ENTRADA AL CCA?33, ESTACION " ?remota),
            ("Recomendacion: REVISAR ENLACE CCA?33 ENTRE " ?remota " Y " ?destino);
REGISTERED AS {nm-rule 74};
```

C.1.31 GrupoElectrogeno.

C.1.31.1 Definición de Clase de Objeto Gestionado.

```
grupoElectrogeno MANAGED OBJECT CLASS
            DERIVED FROM  sistema;
            CHARACTERIZED BY grupoElectrogenoPackage;
             -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 31}
```

C.1.31.2 Definición de Enlace de Nombre.

```
grupoElectrogeno NB NAME BINDING
            SUBORDINATE OBJECT CLASS grupoElectrogeno
                  AND SUBCLASSES;
            NAMED BY SUPERIOR OBJECT CLASS sistema
                  AND SUBCLASSES;
            WHIT ATTRIBUTE grupoElectrogenoId;
REGISTERED AS {nm-nb 31}
```

C.1.31.3 Definición de Paquete.

```
grupoElectrogenoPackage PACKAGE
            ATTRIBUTES
                  mediaTensionUtilizacion          GET,
                  falloArranque                    GET;
                  bajaPrecionAceite                GET,
                  temperaturaMotorAlta             GET,
                  sobreintencidadAlternador        GET,
                  tensionInsuficiente              GET,
                  bajoNivelAceite                  GET,
                  bajaTensionBateria               GET,
                  falloVentilador                  GET,
                  disparoAutoamtico                GET,
                  bajoNivelD-50                    GET,
                  cargadorMantenimiento            GET,
                  cargaBateriaMotor                GET,
                  sobrevelocidadMotor              GET,
                  bajoNivelD-100                   GET;
            ACTIONS
                  ActuaSobreGrupoDisponible, ActuaSobreGrupoServicio ;
            NOTIFICATIONS
                  FALLO_DE_ARRANQUE,
                  BAJA_PRESIÓN_ACEITE,
                  TEMPERATURA_MOTOR_ALTA,
```

```
                    SOBREINTENSIDAD_ALTERNADOR,
                    INSUFICIENTE_TENSIÓN_UTILIZACIÓN,
                    BAJO_NIVEL_ACEITE,
                    BAJA_TENSIÓN_BATERIA,
                    FALLO_VENTILADOR_DE_SALA,
                    DISPARO_AUTOMÁTICO,
                    BAJO_NIVEL_D-50,
                    CARGADOR_DE_MANTENIMIENTO,
                    AVISO_CARGA_DE_BATERÍA_DEL_MOTOR,
                    SOBREVELOCIDAD_DEL_MOTOR,
                    BAJO_NIVEL_D-1000;
        PARAMETERS
                    codigoActuacionGrupoDisponible, codigoActuacionGrupoServicio;
REGISTERED AS {nm-package 20}
```

C.1.32 Caseta.

C.1.32.1 Definición de Clase de Objeto Gestionado.

```
caseta MANAGED OBJECT CLASS
        DERIVED FROM  base;
        CHARACTERIZED BY casetaPackage;
         -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 32}
```

C.1.32.2 Definición de Enlace de Nombre.

```
casetaNB NAME BINDING
        SUBORDINATE OBJECT CLASS caseta
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS base
                AND SUBCLASSES;
        WHIT ATTRIBUTE casetaId;
REGISTERED AS {nm-nb 32}
```

C.1.32.3 Definición de Paquete.

```
casetaPackage PACKAGE
        ATTRIBUTES
                medidaTensionCaseta             GET,
                medidaTensionCuadroGeneral      GET,
                tensionEntradaCaseta            GET,
                tensionEntradaCuadroGeneral     GET,
                descargaAtmosferica             GET,
                temperaturaAlta                 GET,
                puertaAbierta                   GET,
                deteccionHumos                  GET,
                anomaliaAireAcondicionado       GET;
        NOTIFICATIONS
                TENSION_CUADRO_DIS_GEN,         – fallo tesión entr. cuadro distribucion
                TENSION_ENTRADA_CASETA,         -- fallo tesion entrada caseta
                TENSION_DE_UTILIZACION,         – fallo alimentacion
        RULES
                SSR58, SSR59, SSR60, SSR61;
REGISTERED AS {nm-package 21}
```

C.1.32.4 Definición de Reglas Expertas de Gestión.

```
SSR58 RULE
    BEHAVIOR SSR58Behavior;
    PRIORITY 0;
    IF
     (?fecha ?h1 ?remota TENSION_ENTRADA_CASETA
          MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num)
    (?fecha ?h2 ?remota TENSION_CUADRO_DIS_GEN MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR
          ?num & : (<(ABS(? ?h1 ?h2)) 1.00))
    THEN
          ("Grado de Severidad: 3"),
          ("Averia Asociada: FALLO EN LA ALIMENTACION DE LA CASETA, ESTACION " ?remota),
          ("Recomendacion: REVISAR ALIMENTACION DE LA CASETA.");
REGISTERED AS {nm-rule 58};

SSR59 RULE
   BEHAVIOR SSR59Behavior;
   PRIORITY -1;
   IF
    ?A<?(?fecha ?h1 ?remota TENSION_CUADRO_DIS_GEN
          MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num)
   ?B<?(?fecha ?h2 ?remota TENSION_CUADRO_DIS_GEN
     MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num  & : (>(ABS(? ?h1 ?h2)) 30))
```

```
      (NOT (?fecha ?h3 ?remota TENSION_CUADRO_DIS_GEN
            MEDIDA_EN_ALARMA_LIM.INF_PASA_A_NORMAL & : (<(ABS(? ?h1 ?h3)) 30)))
       (NOT(?fecha ?h4 ?remota TENSION_CUADRO_DIS_GEN DESAPARECE_ALARMA
            & : (<(ABS(? ?h2 ?h4)) 30)))
      THEN
         (retract ?A),
         (retract ?B),
          ("Grado de Severidad: 3"),
          ("Averia Asociada: FALLO EN EL GRUPO ELECTROGENO O EN SISTEMA ASOCIADO,
                    ESTACION" ?remota),
          ("Recomendacion: REVISAR EL GRUPO ELECTROGENO.");
REGISTERED AS {nm-rule 59};

SSR60 RULE
   BEHAVIOR SSR60Behavior;
   PRIORITY 0;
   IF
     (?fecha ?h1 ?remota
          TENSION_DE_UTILIZACION MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num)
     (?fecha ?h2 ?remota TENSION_CUADRO_DIS_GEN MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR
          ?num & : (<(ABS(? ?h1 ?h2)) 1.00))
     THEN
          ("Grado de Severidad:  0"),
          ("Averia Asociada: FALLO EN LA DISTRIBUCION DE ALTERNA, ESTACION " ?remota),
          ("Recomendacion: REVISAR ALIMENTACION DE ALTERNA DEL RECTIFICADOR.");
REGISTERED AS {nm-rule 60};

SSR61 RULE
  BEHAVIOR SSR61Behavior;
   PRIORITY 0;
   IF
     (?fecha ?h1 ?remota
          TENSION_DE_UTILIZACION MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num)
     (NOT(?fecha ?h2 ?remota TENSION_CUADRO_DIS_GEN
          MEDIDA_PASA_A_ALARMA_POR_LIMITE_INFERIOR ?num & : (<(ABS(? ?h1 ?h2)) 1.00)))
     THEN
          ("Grado de Severidad: 3"),
          ("Averia Asociada: FALLO EN EL RECTIFICADOR O EN LA BATERIA, ESTACION " ?remota),
          ("Recomendacion: REVISAR BATERIA Y RECTIFICADOR.");
REGISTERED AS {nm-rule 61};
```

C.1.33 Presualizador.

C.1.33.1 Definición de Clase de Objeto Gestionado.

```
presuarizador MANAGED OBJECT CLASS
          DERIVED FROM  base;
          CHARACTERIZED BY presuarizadorPackage;
           -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 33}
```

C.1.33.2 Definición de Enlace de Nombre.

```
presuarizadorNB NAME BINDING
          SUBORDINATE OBJECT CLASS presuarizador
                AND SUBCLASSES;
          NAMED BY SUPERIOR OBJECT CLASS base
                AND SUBCLASSES;
          WHIT ATTRIBUTE presuarizadorId;
REGISTERED AS {nm-nb 33}
```

C.1.33.3 Definición de Paquete.

```
presuarizadorPackage PACKAGE
          ATTRIBUTES
                flujo_alto                GET,
                bajaPresion               GET,
                humedadAlta               GET;
          NOTIFICATIONS
                PRESURIZ_ALTO_FLUJO,          -- fisura o rotura en guia de ondas
                PRESURIZ_BAJA_PRESION,        -- fallo en equipo
                PRESURIZ_ALTA_HUMEDAD;        -- entrada de humedad en guia de ondas
          RULES
                SSR39, SSR40, SSR41, SSR42;
REGISTERED AS {nm-package 22}
```

C.1.33.4 Definición de Reglas Expertas de Gestión.

```
SSR39 RULE
      BEHAVIOR SSR39Behavior;
      PRIORITY 0;
      IF
            (? ? ?remota PRESURIZ_ALTO_FLUJO ALARMA)
      THEN
            ("Grado de Severidad: 3"),
            ("Averia Asociada: FISURA O ROTURA EN LA GUIA DE ONDAS; ESTACION " ?remota),
            ("Recomendacion: REVISAR LA GUIA DE ONDAS.");
REGISTERED AS {nm-rule 39};

SSR40 RULE
      BEHAVIOR SSR40Behavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota PRESURIZ_BAJA_PRESION ALARMA)
            (?fecha ?h2 ?remota PRESURIZ_ALTO_FLUJO ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
      THEN
            ("Grado de Severidad: 1"),
            ("Averia Asociada: ROTURA DE LA GUIA DE ONDAS, ESTACION " ?remota),
            ("Recomendacion: REVISAR LA GUIA DE ONDAS.");
REGISTERED AS {nm-rule 40};

SSR41 RULE
      BEHAVIOR SSR41Behavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota PRESURIZ_BAJA_PRESION ALARMA)
            (NOT(?fecha ?h2 ?remota PRESURIZ_ALTO_FLUJO ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
      THEN
            ("Grado de Severidad: 1"),
            ("Averia Asociada: FALLO EN EQUIPO DE PRESURIZACION, ESTACION " ?remota),
            ("Recomendacion: REVISAR EQUIPO DE PRESURIZACION.");
REGISTERED AS {nm-rule 41};

SSR42 RULE
      BEHAVIOR SSR42Behavior;
      PRIORITY 0;
      IF
            (? ? ?remota PRESURIZ_ALTA_HUMEDAD ALARMA)
      THEN
            ("Grado de Severidad: 2"),
            ("Averia Asociada: ENTRADA DE HUMEDAD A LA GUIA DE ONDAS A TRAVES DEL
                  PRESURIZADOR, ESTACION " ?remota),
            ("Recomendacion: REVISAR SALES DESECADORAS DEL PRESURIZADOR.");
REGISTERED AS {nm-rule 42};
```

C.1.34 TorreMeteorologica.

C.1.34.1 Definición de Clase de Objeto Gestionado.

```
torreMeteorologica MANAGED OBJECT CLASS
            DERIVED FROM  base;
            ATTRIBUTES
               velocidad Viento, direccionViento, tempertaturaExterior;
REGISTERED AS {nm-MobjectClass 34}
```

C.1.34.2 Definición de Enlace de Nombre.

```
torreMeteorologicaNB NAME BINDING
            SUBORDINATE OBJECT CLASS torreMeteorologica
                  AND SUBCLASSES;
            NAMED BY SUPERIOR OBJECT CLASS base
                  AND SUBCLASSES;
            WHIT ATTRIBUTE torreMeteorologicaId;
REGISTERED AS {nm-nb 34}
```

C.1.35 Rectificador.

C.1.35.1 Definición de Clase de Objeto Gestionado.

```
rectificador MANAGED OBJECT CLASS
            DERIVED FROM  base;
            CHARACTERIZED BY rectificadorPackage;
              -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 35}
```

C.1.35.2 Definición de Enlace de Nombre.

```
sistemaNB NAME BINDING
        SUBORDINATE OBJECT CLASS rectificador
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS base
                AND SUBCLASSES;
        WHIT ATTRIBUTE baseId;
REGISTERED AS {nm-nb 35}
```

C.1.35.3 Definición de Paquete.

```
rectificadorPackage PACKAGE
        ATTRIBUTES
                tensionEntrada              GET,
                maximaTensionRectificador1  GET,
                maximaTensionRectificador2  GET,
                minimaTensionBateria        GET,
                minimaTensionFlotRect1      GET,
                minimaTensionFlotRect2      GET,
                cargaRapidaBateria          GET,
                cargaFondo                  GET,
                disparoSalidaR1             GET;
        NOTIFICATIONS
                BAJA_TENSIÓN_ENTRADA_AL_RECTIFICADOR,
                MÁXIMA_TENSIÓN_RECTIFICADOR_1,
                MÁXIMA_TENSIÓN_RECTIFICADOR_2,
                MÍNIMA_DESCARGA_BATERÍA,
                MÍNIMA_TENSIÓN_FLOT_RECTIFICADOR_1,
                MÍNIMA_TENSIÓN_FLOT_EN_RECTIFICADOR_2,
                BATERÍA_EN_CARGA_RÁPIDA,
                DISPARO_SALIDA_DEL_R1,
                CARGA_A_FONDO;
REGISTERED AS {nm-package 23}
```

C.1.36 RectificadorI.

C.1.36.1 Definición de Clase de Objeto Gestionado.

```
rectificadorI MANAGED OBJECT CLASS
        DERIVED FROM  base;
        CHARACTERIZED BY rectificadorIPackage;
         -- no condicional packages are defined for this class
REGISTERED AS {nm-MobjectClass 36}
```

C.1.36.2 Definición de Enlace de Nombre.

```
rectificadorI NB NAME BINDING
        SUBORDINATE OBJECT CLASS rectificadorI
                AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS base
                AND SUBCLASSES;
        WHIT ATTRIBUTE rectificadorIId;
REGISTERED AS {nm-nb 36}
```

C.1.36.3 Definición de Paquete.

```
rectificadorPackageI PACKAGE
        ATTRIBUTES
                FalloFaseR1                 GET,
                FalloFaseR2                 GET,
                maximaTensionUtilizacion    GET,
                minimaTensionUtilizacion    GET,
                averiaR1                    GET,
                averiaR2                    GET,
                maximaTensionBateria        GET,
                cargaManualR1               GET,
                cargaIntR1                  GET,
                cargaFlotR1                 GET,
                cargaExcepR1                GET,
                cargaAutomatica             GET,
                cargaManual                 GET,
                cargaIntR2                  GET,
                cargaFlotR2                 GET,
                cargaExcepR2                GET;
        NOTIFICATIONS
                FALLO_FASE_R-1,  FALLO_FASE_R-2,
                MÁXIMA_TENSIÓN_DE_UTILIZACIÓN,
                MÍNIMA_TENSIÓN_DE_UTILIZACIÓN,
                AVERÍA_R-1, AVERÍA_R-2,
```

```
                    MÁXIMA_TENSIÓN_BATERÍA,
                    CARGA_MANUAL_R-1,
                    CARGA_INT_R-1, CARGA_FLOT_R-1,
                    CARGA_EXCEP_R-1,
                    CARGA_AUTOMÁTICA_R-1/R-2,
                    CARGA_MANUAL_R-2,
                    CARGA_INT_R-2, CARGA_FLOT_R-2,
                    CARGA_EXCEP_R-2;
REGISTERED AS {nm-package 24}
```

C.1.37 Clases Virtuales.

C.1.37.1 Definición de Clase de Objeto Gestionado Virtual .

```
CTR190_CCAA33/34  MANAGED OBJECT CLASS
        DERIVED FROM  radioTransnceptorCTR190, equipoCCAA33;
        -- no condicional packages are defined for this class
        RULES
           damageFrecuentCTR190/7,
           SSR4B, SSR10B, SSR12A, SSR12B,SSR13;
REGISTERED AS {nm-MobjectClass 37}

damageFrecuentCTR190/7 RULE
    BEHAVIOR damageFrecuentCTR190/7Behavior;
    PRIORITY 0;
    IF
      (?fecha ? ?remota CTR190/7_F_ALIM_1 ?destino ALARMA)
      (NOT(?fecha ? ?remota CCA?34_AIS_DE_BB ?destino ALARMA))
    THEN
      ("Grado de Severidad: 4"),
      ("Averia Asociada: AVERIA EN FUENTE DE ALIMENTACION 1 DE CTR190/7, ESTACION " ?remota "
                  O AVERIA EN ALIMENTACION DE LA FUENTE 1"),
       ("Recomendacion: REVISAR LA FUENTE Y/O SU ALIMENTACION.");
REGISTERED AS {nm-rule 4A};

SSR4B RULE
    BEHAVIOR SSR4BBehavior;
    PRIORITY 0;
    IF
      (?fecha ? ?remota CTR190/7_F_ALIM_2 ?destino ALARMA)
      (NOT(?fecha ? ?remota CCA?34_AIS_DE_BB ?destino ALARMA))
    THEN
       ("Grado de Severidad: 4"),
       ("Averia Asociada: AVERIA EN FUENTE DE ALIMENTACION 2 DE CRT190/7, ESTACION " ?remota "
                  O AVERIA EN LA ALIMENTACION DE LA FUENTE 2"),
       ("Recomendacion: REVISAR LA FUENTE Y/O SU ALIMENTACION.");
REGISTERED AS {nm-rule 4B};

SSR10B RULE
    BEHAVIOR SSR10BBehavior;
    PRIORITY 0;
    IF
      (?fecha ?h1 ?remota CMF?38_PREBER_2 ?destino ALARMA)
      (NOT(?fecha ?h2 ?remota CMF?38_PREBER_2 ?destino DESAPARECE_ALARMA
                  & : (<(ABS(? ?h1 ?h2)) 1.00)))
      (NOT(?fecha ?h3 ?remota CCA?34_AIS_DE_BB ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
      (NOT(?fecha ?h4 ?remota CMF?38_BER_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h4)) 1.00)))
      (NOT(CONTROL_DEL_FLUJO))
    THEN
       ("Grado de Severidad: 2"),
       ("Averia Asociada: DEGRADACION EN EL CANAL 2 DEL ENLACE CTR190/7 ENTRE "
          ?remota " Y " ?destino),
       ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 10B};
```

```
SSR12A RULE
      BEHAVIOR SSR12ABehavior;
      PRIORITY 0;
      IF
        (?fecha ?h1 ?remota CMF?38_BER_1 ?destino ALARMA)
        (NOT(?fecha ?h2 ?remota CMF?38_BER_1 ?destino DESAPARECE_ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
        (NOT(?fecha ?h3 ?remota CCA?34_AIS_DE_BB ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
        (NOT(?fecha ?h4 ?remota CMF?38_F_AL_TR_C1 ?destino ALARMA & : (<(ABS(? ?h1 ?h4)) 1.00)))
      THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: ALTA TASA DE ERRORES EN EL CANAL 1 DEL ENLACE CTR190/7 ENTRE "
            ?remota " Y " ?destino),
         ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 12A};

SSR12B RULE
      BEHAVIOR SSR12BBehavior;
      PRIORITY 0;
      IF
        (?fecha ?h1 ?remota CMF?38_BER_2 ?destino ALARMA)
        (NOT(?fecha ?h2 ?remota CMF?38_BER_2 ?destino DESAPARECE_ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00)))
        (NOT(?fecha ?h3 ?remota CCA?34_AIS_DE_BB ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
        (NOT(?fecha ?h4 ?remota CMF?38_F_AL_TR_C2 ?destino ALARMA & : (<(ABS(? ?h1 ?h4)) 1.00)))
      THEN
         ("Grado de Severidad: 3"),
         ("Averia Asociada: ALTA TASA DE ERRORES EN EL CANAL 2 DEL ENLACE CTR190/7 ENTRE "
            ?remota " Y " ?destino),
         ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 12B};


SSR13 RULE
      BEHAVIOR SSR13Behavior;
      PRIORITY 0;
      IF
       (OR
        (?fecha ?h1 ?remota CMF?38_F_AL_TR_C1 ?destino ALARMA)
        (?fecha ?h1 ?remota CMF?38_F_AL_TR_C2 ?destino ALARMA))
       (NOT (?fecha ?h2 ?remota CCA?34_AIS_DE_BB ?destino ALARMA & : (< (ABS(? ?h1 ?h2)) 1.00)))
      THEN
        ("Grado de Severidad: 4"),
        ("Averia Asociada: PROBLEMAS EN EL ENLACE CTR190/7 ENTRE " ?remota " Y " ?destino),
        ("Recomendacion: REVISAR EL ENLACE.");
REGISTERED AS {nm-rule 13};
```

C.1.37.2 Definición de Clase de Objeto Gestionado Virtual .

```
CCAA33/34_Presuarizador  MANAGED OBJECT CLASS
            DERIVED FROM  equipoCCAA33, Presuarizador;
             -- no condicional packages are defined for this class
            RULES
               SSR19;
REGISTERED AS {nm-MobjectClass 38}

SSR19 RULE
      BEHAVIOR SSR19Behavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota CCA?34_AIS_DE_BB ?destino ALARMA)
              (OR(?fecha ?h2 ?remota PRESURIZ_ALTO_FLUJO ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
                 (?fecha ?h3 ?remota PRESURIZ_BAJA_PRESION ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00)))
      THEN
            ("Grado de Severidad: 5"),
            ("Averia Asociada: ROTURA DE LA GUIA DE ONDAS, ESTACION " ?remota),
            ("Recomendacion: REVISAR LA GUIA DE ONDAS.");
REGISTERED AS {nm-rule 19};
```

C.1.37.3 Definición de Clase de Objeto Gestionado Virtual.

```
CTR190_ MP31  MANAGED OBJECT CLASS
            DERIVED FROM  radioTransnceptorCTR190, primerOrdenMP31;
            DERIVED FROM  equipoCCAA33, Presuarizador;
             -- no condicional packages are defined for this class
            RULES
               SSR22A, SSR22B, SSR29;
REGISTERED AS {nm-MobjectClass 39}
```

```
SSR22A RULE
      BEHAVIOR SSR22A Behavior;
      PRIORITY 0;
      IF
            (? ? ?remota MP31/X.3_EXTERNA ?destino ALARMA)
      THEN
             ("Grado de Severidad: 3"),
             ("Averia Asociada: FALLO EN SENAL DE ENTRADA DE 34Mb/s AL MULTIPLEXOR
                        MP31/X.3, ESTACION " ?remota),
             ("Recomendacion: REVISAR SENAL DE ENTRADA AL MUX.");
REGISTERED AS {nm-rule 22A};

SSR22B RULE
      BEHAVIOR SSR22Behavior;
      PRIORITY 0;
      IF
            (? ? ?remota MP31/2_EXTERNA ?destino ALARMA)
      THEN
        ("Grado de Severidad: 3"),
        ("Averia Asociada: FALLO EN SENAL DE ENTRADA DE 8Mb/s AL MULTIPLEXOR MP31/2,
            ESTACION " ?remota),
        ("Recomendacion: REVISAR CANAL DE ENTRADA AL MUX.");
REGISTERED AS {nm-rule 22B};

SSR29 RULE
      BEHAVIOR SSR29Behavior;
      PRIORITY 0;
      IF
            (?fecha ?h1 ?remota MP31_EXT_FONIA ?destino ALARMA)
            (OR(?fecha ?h2 ?remota CMF?38_BER_1 ?destino ALARMA & : (<(ABS(? ?h1 ?h2)) 1.00))
            (?fecha ?h3 ?remota CMF?38_BER_2 ?destino ALARMA & : (<(ABS(? ?h1 ?h3)) 1.00))
            (?fecha ?h4 ?remota CMF?38_F_AL_TR_C1 ?destino ALARMA & : (<(ABS(? ?h1 ?h4)) 1.00))
            (?fecha ?h5 ?remota CMF?38_F_AL_TR_C2 ?destino ALARMA & : (<(ABS(? ?h1 ?h5)) 1.00)))
      THEN
             ("Grado de Severidad: 3"),
             ("Averia Asociada: FALLO EN EQUIPO CONECTADO AL MP31, ESTACION " ?remota),
             ("Recomendacion: REVISAR EQUIPOS CONECTADOS AL MULTIPLEXOR.");
REGISTERED AS {nm-rule 29};
```

Anexo D.

Art*Enterprise.

Históricamente, los primeros Sistemas Basados en Conocimiento (SBC) fueron desarrollados utilizando lenguajes de programación como LISP y PROLOG. A medida que el desarrollo de los SBC iba aumentado en cantidad y complejidad, la comunidad científica comenzó a buscar formas de desarrollar los sistemas en menor tiempo y esfuerzo.

Esto dio lugar a la aparición, en primer lugar a sistemas vacíos como el EMYCIN, a los que se denominaron *shells*, ya que ofrecían toda la arquitectura de un SBC a la que había que incorporar la base de conocimientos.

Posteriormente surgieron en el mercado otras herramientas que incorporaron, además de opciones de representación del conocimiento, esquemas de inferencia y control. Estas herramientas pasaron a tomar el nombre de *Entornos de Desarrollo de SBC*, entre las que cabe citar el Art*Enterprise.

ART es una herramienta desarrollada a mediados de los 80, por Inference Corporation. Es un entorno de desarrollo para la construcción de Sistemas Expertos, dirigido a grandes aplicaciones empresariales. ART*Enterprise es la última versión de esta familia de entornos de desarrollo. Su precio puede rondar los US$ 60.000. y US$ 85.000.

D.1 Herramienta Avanzada de Programación.

ART*Enterprise posee un lenguaje de programación propio, denominado ARTScript. Éste paradigma de programación del ART*Enterprise es utilizado para facilitar y agilizar todas las tareas de programación desarrolladas. ARTScript permite realizar todas las operaciones relacionadas con la construcción de Sistemas Expertos: definir funciones, crear objetos, escribir reglas, indexar casos, acceder a bases de datos, etc.

El ARTScript parece poco familiar al principio, utiliza una notación a base de funciones delimitadas por paréntesis anidados, que puede parecer complicada. Realmente ARTScript es un lenguaje de programación fácil de entender (leer y escribir), especialmente a la hora de trabajar en el desarrollo de Sistemas Expertos para dominios específicos. Dispone de elementos que como el Browser, permiten la edición de Aplicaciones y posibilitan la rápida detección y corrección de fallos en las funciones programadas. Además aporta las herramientas necesarias para comprobación y posterior ejecución inmediata de los procedimientos, evitando tediosos procesos de compilado y linkado de los programas y sus funciones.

El depurador de ARTScript, dispone además de una pila que contiene las llamadas a las distintas funciones ARTScript, así como los argumentos que se pasan entre ellas. Permite el desplazamiento por la pila de arriba abajo, para examinar el flujo de datos. Esta

herramienta es muy eficaz a la hora de hacer un seguimiento y detectar los errores de ejecución.

Todas estas características, hace de ARTScript una poderosa herramienta para la construcción de prototipos de sistemas expertos.

D.2 Características.

Entre sus características principales, podemos enumerar las siguientes:

- **Razonamiento basado en casos, CBR (case-based reasoning).**

El funcionamiento del RBC parte de casos almacenados previamente en la base de casos. ART puede incorporar reglas expertas, en el orden de miles. Un nuevo problema se compara con estos y se recuperan uno o varios casos. Posteriormente se utiliza y evalúa una solución, sugerida por los casos que han sido seleccionados con anterioridad, para ver si se aplica al problema actual.

- Programación orientada a objeto, OOP (Object Oriented Programming).

Utiliza características de lenguajes de programación orientas a objetos, actualmente presentes en C++, junto con una gran colección de las clases del objeto definidas.

- Sistema de manejo B.D., DBMS (DataBase Management System).

Permite el acceso a bases de datos mediante SQL-based y ODBC-based.

- Interfaz gráfica de usuario, GUI Graphical User Interface.

Dispone de un avanzado entorno operativo, basado en menús gráficos compuestos por ventanas, iconos y menú de opciones desplegables, que posibilita su programación de forma cómoda y fácil, incluso a usuarios inexpertos.

- Motor de encadenamiento hacia delante.

Proporciona también posibilidad de ejecución de encadenamiento hacia atrás, aunque el utilizado por defecto es el encaminamiento hacia delante.

- Plataformas soportadas.

Amplio abanico de posibilidades, corre sobre estaciones de trabajo Windows (3.1, 95, NT), OS/2, UNIX (AIX, HP-UX, Solaris) y series DEC VAX

- Desarrollo multi-usuario.

De utilidad en entornos de grupos de trabajo en red.

En los siguientes apartados se estudian aspectos importantes de ART*Enterprise a la hora de activar y disparar las distintas reglas de gestión del sistema, como son el establecimiento de prioridades y las estrategias de resolución de conflictos.

D.3 Evaluación de la Prioridad.

La prioridad (en ingles, salience) de una regla no es más que un valor numérico. Por defecto es 0, y está comprendido entre -10000 y +10000. Sin embargo, el usuario puede asignar directamente dicho valor mediante una constante o una expresión numérica. La asignación se realiza poniendo la siguiente expresión entre el nombre de la regla y el primer elemento condicional de la regla:

(salience <expresión-numérica>)

Los valores de prioridad pueden evaluarse en alguno de los siguientes momentos:

- Cuando se define la regla: La prioridad se establece al realizar la definición de la regla experta de gestión, en nuestro caso cuando se define la regla con la plantilla RULE de GDMO+.

- Cuando se activa la regla: La prioridad se establece al realizar la definición de la plantilla RULE y posteriormente cuando la regla se activa.

- En cada ciclo de ejecución: La prioridad se establece al realizar la definición de la plantilla RULE, posteriormente cuando la regla se activa y después de cada disparo (ver apartado 5.8.6).

Por defecto la prioridad se evalúa cuando se define la plantilla RULE. Para modificar el comportamiento sobre la evaluación de prioridades, habrá que realizarlo en el SHELL utilizado, en nuestro caso se utiliza **ART Enterprise 3.0.**

D.3.1 Utilización de la Prioridad, Consideraciones.

El uso de prioridades es una herramienta muy potente para controlar la ejecución y puede ser usada para establecer el orden de disparo de las reglas. De hecho, nos atreveríamos a decir que para la programación basada en reglas, el establecimiento de prioridades es un aspecto muy importante. Debemos evitar el abuso de esta potencia y ser conscientes de las limitaciones en el control explícito del sistema, pueden aparecer los siguientes inconvenientes:

1.- Está convirtiendo esta técnica de programación en una programación imperativa o precedural, ya que se está estableciendo la secuencia de ejecución de las reglas. Con lo que se elimina en parte las cualidades y ventajas que nos aporta un lenguaje de Sistema Experto.

2.- Se estará desarrollando y código muy pobre. La principal ventaja de la programación basada en reglas es que el programador no debe preocuparse de la ejecución del programa. Tan sólo se limita a codificar el conocimiento que se tiene del problema para que, posteriormente, atendiendo a ciertos *criterios naturales* de ejecución, se disparen las reglas activas de una forma óptima.

Se ha de tener siempre presente que un experto no establece, en general, una jerarquía de prioridades *ad hoc* en las reglas. Aún en el caso de que fuese necesario establecer valores, se estima que no más de 3 ó 4 valores son necesarios para definir el conocimiento en las reglas de gestión bien codificadas (en nuestro caso las plantillas RULES). Es más, es casi seguro que si se profundiza un poco más en el problema a

resolver, posiblemente se puedan eliminar los valores de prioridad en las que, inicialmente, se consideraba que eran imprescindibles.

D.4 Estrategias de Resolución de Conflictos.

En el apartado anterior se vio la forma de definir los valores de las prioridades de las reglas, a continuación se comentarán algunas de las estrategias más importantes para la resolución de conflictos que surgen cuando tenemos varias reglas con una misma prioridad.

Para la activación de cada estrategia en el **ART Entrerprise 3.0**, es necesario utilizar el comando correspondiente tal y como se detalla en los siguientes apartados, en donde describimos las distintas técnicas de resolución de conflictos soportadas por Art*enterprise.

D.4.1 Estrategia en Profundidad.

Las nuevas reglas, con igual preferencia, se colocarán en la cima de la pila. Es decir, si R1 se activa para el hecho f-1 y R2 se activa para f-2, y f-1 se afirma antes que f-2, entonces R2 se colocará por encima de la regla R1.

Esta estrategia podemos activarla mediante el comando:

```
>set-strategy depth
```

D.4.2 Estrategia en Anchura.

Las nuevas reglas, con la misma preferencia, se colocarán en el fondo de la pila. Es decir, si R1 se activa para el hecho f-1 y R2 se activa para f-2, y f-1 se afirma antes que f-2, entonces R1 se mantendrá por encima de la regla R2.

Este tipo de estrategia se activa a través del siguiente comando:

```
>set-strategy breadth
```

D.4.3 Estrategia Basada en la Simplicidad.

Las nuevas reglas, con la misma preferencia, se colocarán por encima de aquellas que tengan igual o mayor especificidad. La especificidad de una regla se determina, por el número de comparaciones que deben realizarse en el antecedente de una regla. La especificidad es una variable numérica que se incrementa en una unidad en los casos siguientes:

- Cuando se realiza una comparación con una constante o variable de valor asignado.

- Cuando la llamada a una función que involucre a los elementos condicionales ***:***, ***=*** o ***test***.

No se añade especificidad en estos otros casos:

- En las llamadas a las funciones booleanas ***not***, ***and*** y ***or***. Aunque si pueden añadir especificidad sus argumentos.

- En la llamada a una función dentro de una función que no añade especificidad a la regla.

```
example RULE
   BEHAVIOR exampleBehavior;
   IF
     (?remota ?valor1 ?fecha ?destino ?fecha ?valor2)
     (test (AND ?valor1 (> ?valor1 (+ 10 ?valor2)) (< ?valor1 100)))
   THEN
    …
REGISTERED {XXX}
```

tiene valor de especicidad 4. En efecto:

- El término (`?remota ?valor1 ?fecha ?destino ?fecha ?valor2`) tiene especicidad 1, ya que la primera aparición de ?remota ?valor1 ?fecha ?destino ?valor2 no añaden especicidad, pero sí la segunda aparición de ?fecha.

- El término `(test(AND ?valor1 (> ?valor1 (+ 10 ?valor2))(< ?valor1 100)))` tiene especicidad 3, ya que incrementan las siguientes funciones: TEST, > y <. Notar que no añaden especicidad las funciones AND y +.

Este tipo de estrategia se activa a través del siguiente comando:

```
>set-strategy  simplicity
```

D.4.4 Estrategia basada en la Complejidad.

Las nuevas reglas, con la misma preferencia, se colocarán por encima de aquellas que tengan igual o menor especificidad. La estrategia se activa mediante el comando:

```
>set-strategy complecity
```

D.4.5 Estrategia Lex y Mea.

De entre las reglas con una misma preferencia, se establece una ordenación temporal. Estas estrategias se activan respectivamente, mediante los comandos:

```
>set-strategy lex
>set-strategy mea
```

D.4.6 Estrategia Aleatoria.

En cada activación de una regla determinada se le asigna un número aleatorio. Este número será utilizado para determinar el lugar que ocuparán reglas activas con igual preferencia.

Cuando se cambia a otro tipo de estrategia, los números aleatorios asignados a cada regla serán recordados, con el objetivo para preservar el mismo ordenamiento de las reglas cuando volvamos a elegir de nuevo esta estrategia. Para esto, sólo se tendrán en cuenta las reglas activas, que también lo estaban en el momento del cambio de estrategia.

La estrategia se activa mediante el comando:

```
>set-strategy random
```

Anexo E

Referencias

A

[Abeck,95] Abeck, Sebastian – *Integrated Network and System Management* –1995

[Anderson02] Anderson, J. R. (John R.) - *Intelligent networks: principles and applications* London : Institution of Electrical Engineers, cop. 2002

[Albus01] Albus, James Sacra. Meystel, Alexander M. - *Engineering of mind : an introduction to the science of intelligent systems* -Publicación New York [etc.] : John Wiley, 2001

B

[Bauer88] Bauer, D.S.; Koblentz, M.E.- *NIDX-an expert system for real - time network intrusion detection* - Computer Networking Symposium, 1988., Proceedings of the, 11-13 Apr 1988, Page(s): 98 –106

[Barba99] Barba Martí, Antoni. *Gestión de Redes*. Edicions Universitat Politécnica de Catalunya – 1999.

[Black95] Black, U.D. *Network Management Standars.* McGraw Hill -1995.

C

[Cisco99] Cisco Systems, Inc. - *Cisco IOS 12.0 Network Security* - Cisco Systems, Inc. Publicación Indianapolis. IN Cisco, 1999.

[Callan03] Autor Callan, Robert - *Título Artificial intelligence* - Publicación Basingstoke: Palgrave Macmillan, 2003

[Celestino93] Celestino, J., Jr.; L'alanne, S.; Claude, J.-P. - *A computing platform for Telecommunication Management Networks* -Networks, 1993. International Conference on Information Engineering '93. 'Communications and Networks for the Year 2000', Proceedings of IEEE Singapore International Conference on , Volume: 1 , 6-11 Sep 1993, Page(s): 440 -444 vol.1

[Chen00] Chen, Graham - *Integrated Telecommunications Management Solutions* - Piscataway, New Jersey IEEE, 2000

[Cisco99] Cisco Systems, Inc. - *Cisco IOS 12.0 Network Security* - Cisco Systems, Inc. Publicación Indianapolis. IN Cisco, 1999.

D

[Derbort91] Derbort, H.J.; Cobb, J.W.; Denton, R.W. - *NETHELP user friendly interface* - Southeastcon '91., IEEE Proceedings of , 7-10 Apr 1991 Page(s): 746 -750 vol.2

E

[ERIC-91] *Multiplexor de primer orden ZAK 30/5*. Ericsson. Documento 151-ZFK-401 01 Ues. 1991.

[ERIC-85] *Multiplexor de segundo orden MP31/X.* Ericsson. Documento 614-102/83-TM-S. 1985.

F

[Friant00] Friant, Jean - *Juegos lógicos en el mundo de la inteligencia artificial* – Publicación Barcelona : Gedisa, 2000

G

[Garcia01] Garcia, R.C.; Cannady, J. - *Boundary expansion of expert systems: incorporating evolutionary computation with intrusion detection solutions* - SoutheastCon 2001. Proceedings. IEEE , 2001 Page(s): 96 –99

[Giarratano01] Giarratano, Riley – *Sistemas Expertos. Principios y Programación* – International Thomson Editores, 2001.

[Ghosh00] Ghosh, Sumit and Lee, Tony - *Intelligent transportation systems : new principles and architectures* - Publicación Boca Raton, Fla. : CRC Press, cop. 2000

[GTE-81] *Multiplexor de primer orden MP31.* GTE Telecomunicazioni.spa. Documento 67-3102-101/S. 1981.

H

[Hebrawi95] Hebrawi, B. *GDMO*, *Object modelling and definition for network management.* Technology appraisals – 1995.

[Hegering94] Hegering, H.G. and Abeck, S. *Integrated network and System management*. Addison-Wesley –1994.

[Hicks90] Hicks, R.C. - *A composite methodology for low maintenance expert systems development* - System Sciences, 1990., Proceedings of the Twenty-Third Annual Hawaii International Conference on , Volume: iii , 2-5 Jan 1990, Page(s): 293 -302 vol.3

[Hopgood01] Autor Hopgood, Adrian A. - I*ntelligent systems for engineers and scientists* - Boca Raton : CRC Press, 2001

I

[ISO96] CCITT X.660 / ISO 9834-1. *Procedures for the Operation of OSI Registration Authorities, Gereneral Procedures, Amendment 1: Incorporation of Object Identifiers Components* – 1996.

[ISO92A] CCITT X. 701 / ISO 10040. *Information Processing Systems - Open Systems Interconnection - Systems Management Overview* - Ginebra, 1992.

[ISO92B] CCITT X. 720 / ISO 10165-2. *Information Tecnology – Open System Interconection – System Management Overview* - 1992.

[ITUT00] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), *TMN management functions,* Recomendation M.3400, febrero de 2000

[ITUT93] CCITT X. 722 / ISO 10165-4 ISO, *Structure of management information. Part 1: Guidelines for the definition of managed objects* – 1992.

[ITUT91] ITU-T - *Recomemendation X.711 : Common Management Information Service for CCITT applications* – 1991

[ITUT92] ITU-T | ISO/IEC 10040 - *Recommendation X.70*, *Information technology. Open Systems Interconnection, Systems management overview.* – 1992

[ITUT93] ITU-T - *Recomemendation X.734: Information Techology. Open system Interconnection, system management: Event report manamement function* – 1993

K

[Kauffels92] Kauffels, Franz-Joachim - *Network Management: Problems, Standards and Strategies* –fels Wokingham, England Addison-Wesley, 1992

[Klementtinen99] M. Klementtinen - *A Knowledge Discovery Methodology for Telecommunication Network Alarm Databases.* - PhD thesis, University of Helsinki, 1999.

L

[Lai98] Lai, Loi Lei - *Intelligent System Applications in Power Engineering: Evolutionary programming and neural networks* - Publicación Chichester [etc.] John Wiley and Sons, 1998

[León00] León, C., Martín, S., Mejías, M., Romero, M. C., Medina, A.V.: *"Integration of Expert System Rules into Standarized Object Description Model for Telecommunication Network Management"*. ICEIS 2000: 2ndInternational Conference onEnterprise InformationSystems. Stafford (UK). pp. 141-175. 2000. ISBN/D.L.: 972-98050-1-6.

[León99] Carlos León, Manuel Mejías, Joaquín Luque, Fernando Gonzalo: "*Expert System for the Integrated Management of a Power Utility's Communication System*". IEEE Trans on Power Delivery, Vol. 14, No. 4, Octubre, 1999, pp 1208-1212

[Lewis93A] Lewis, L. - *A case-based reasoning approach to the management of faults in communication networks* - INFOCOM '93. Proceedings.Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking: Foundation for the Future. IEEE, 1993 Page(s): 1422 -1429 vol.3

[Lewis93B] Lewis, L. - *A case-based reasoning approach to the management of faults in communication networks* - Artificial Intelligence for Applications, 1993. Proceedings., Ninth Conference on , 1-5 Mar 1993, Page(s): 114 – 120

[Liebowitz98] Liebowitz, Jay – *The Handbook of APPLIED EXPERD SYSTEMS* – Edited by Jay Liebowitzm 1998.

[Lindqvist99] Lindqvist, U.; Porras, P.A. - *Detecting computer and network misuse through the production-based expert system toolset (P-BEST)* - Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on , 1999 Page(s): 146 –161

[Litman02] Litman, Peter f. and Patel-Schneider, F.- *"R++: Adding Path-Bassed Rules to C++"* - IEEE Transactions on Knowledge and Data Engineering, Volume 14, No. 3 - May/June 2002 .

[Louis00] Artificial Neural Networks in Engineering Conference (2000. St. Louis) - *Smart engineering system design: neural networks, fuzzy logic, evolutionary programming, data mining, and complex systems : proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE 2000), held November 5-8, 2000, in St. Louis, Missouri, U.S.A.* - Editors, Cihan H. Dagli... [et al.], New York : ASME Press, 2000

[Lutsky95] Lutsky, P.- *Automating testing by reverse engineering of software documentation* - Reverse Engineering, 1995., Proceedings of 2nd Working Conference on , 14-16 Jul 1995 Page(s): 8 –12

[Loberg88] Loberg, G. – *SMART II: knowledge requirements for expert systems* - Communications, 1988. ICC 88. Digital Technology - Spanning the Universe. Conference Record. IEEE International Conference on , 12-15 Jun 1988, Page(s): 537 -541 vol.1

M

[Malheiros98] Malheiros M. D. and Marcos Silva N. J., - *.Modeling a telecommunication network for fault management applications,.* - Network Operations and Management Symposium, NOMS 98, IEEE, Volume: 3 , Page(s): 723 .732,
1998

[Martin03] A. Martín , C. León, J. M. Elena – *Integractioon of Expert Rules of Management in GDMO standard* – 3rd IASTED International Conference on Artifical Intelligent and Applications (AIA 2003). September, 2003. Benalmadena, Spain

[Martin01] Martín del Brío, Bonifacio - *Redes neuronales y sistemas borrosos* - Madrid : Ra-Ma, 2001

[Melchiors00] Melchiors, C.; Tarouco, L.M.R. - *Troubleshooting network faults using past experience* - Network Operations and Management Symposium, 2000. NOMS 2000. 2000 IEEE/IFIP, 2000 Page(s): 549 –562

[Meystel02] Meystel, Alexander M. and Meystel, James - *Intelligent systems: architecture, design, and control* - New York: John Wiley, 2002

[Merz96] Merz, C.J.; Pazzani, M.J.; Danyluk, A.P. - *Tuning numeric parameters to troubleshoot a telephone-network loop Expert* - IEEE, 1996.

[Moret00] Moret Bonillo, Vicente- *Fundamentos de Inteligencia Artificial* – Servicio de publicaciones, Universidad de la Coruña, 2000.

N

[Nilsson01] Nilsson, Nils J. - *Inteligencia artificial: una nueva síntesis* - Madrid [etc.] : McGraw-Hill, cop. 2001

[Negnevitsky02] Negnevitsky, Michael. - *Artificial intelligence: a guide to intelligent systems* – Publicación New York: Addison Wesley, 2002.

[Nygate95] Nygate, Y, - ECXpert: Exploiting Event Correlaion in Telecommunications - 2nd International Conference on the Practical Applications of Pmlog, I.smdmt Enghmd -1995.

P

[Padgman01] Padgham, Lin and Winikoff , Michael - *Developing intelligent agent systems: a practical guide* - Chichester, England ; Hoboken, NJ: John Wiley, cop. 2004

[Parenll97] Parnell T., *"Guía de Redes de Alta Velocidad"* – Oborne/McGraw-Hill, 1997

[Pino01] Pino Díez, Raúl - *Introducción a la inteligencia artificial: sistemas expertos, redes neuronales artificiales y computación evolutiva* – Publicación Oviedo: Servicio de Publicaciones, Universidad de Oviedo, 2001

[Prerau90] Prerau,D.S.; Gunderson, A.S.; Reinke, R.E.; Adler, M.R. -*Maintainability techniques in developing large expert systems* - Expert, IEEE [see also IEEE Intelligent Systems] , Volume: 5 Issue: 3 , Jun 1990, Page(s): 71 – 80

[Prentzas02] J. Prentzas and I. Hatzilygeroudis, "*Integrating Hybrid Rule-Based with Case-Based Reasoning*", in S. Craw and A. Preece (Eds), Advances in Case-Based Reasoning, Procs 2002 European Conference on Case-Based Reasoning, LNAI 2416, Springer-Verlag, pp. 336-349

R

[Raman99] Raman, Lakshmi G. - *Fundamentals of Telecommunications Network Management* / Lakshmi G. Raman Piscataway, NJ IEEE Press, 1999

[Rich99] Raman, Lakshmi G. - *Fundamentals of Telecommunications Network Management* / Lakshmi G. Raman, Piscataway, NJ IEEE Press, 1999

[Ross88] Ross, M.J.; Covo, A.A.; Hart, C.D., Jr.- *An AI-based network management system* - Computers and Communications, 1988. Conference Proceedings., Seventh Annual International Phoenix Conference on , 16-18 Mar 1988, Page(s): 458 – 461

[Russell04] Russell, Stuart J. and Peter Norvig - *Artificial intelligence : a modern approach* - Publicación Upper Saddle River, NJ: Pearson Education, cop. 2004

S

[Sagrado00] Sagrado Martínez, José del - *Fusión topológica y cuantitativa de redes causales [Archivo de ordenador]* - Almería: Universidad de Almería, Servicio de Publicaciones, c2000

[Sasisekharan96] Sasisekharan, R.; Seshadri, V.; Weiss, S.M. - *Data mining and forecasting in large-scale telecommunication networks* - IEEE Expert, Volume: 11 Issue: 1 , Feb 1996 Page(s): 37 – 43

[Schreiber01] Schreiber, Guus - *Knowledge engineering and management: the commonKADS methodology* - Cambridge, Massachusetts [etc.] : MIT Press, 2001

[shen01] Shen, Weiming, Douglas H. Norrie and Jean-Paul Barthès -*Multi-agent systems for concurrent intelligent design and manufacturing* - Publicación London ; New York : Taylor & Francis, 2001

[Siegel96] Siegel, Jon - *CORBA Fundamentals and Programming / Written and Edited by Jon Siegel* **-** New York [etc.] : John Wiley and Sons, 1996

[Singhal98] Singhal Anoop, Johannes P. Ros and Gary M. Weiss -*ANSWER: Network Monitoring Using Object-Oriented Rules* - AT&T Labs 480 Red Hill Road Middletown, NJ 07748

[Sloman95] Ed. M. Sloman, *Network and Distributed Systems Management, Addison*-Wesley, 1995.

[Stallings00] Stallings, William - *SNMP, SNMPv2, and CMIP: the practical guide to network*-Publicación Reading, Mass. [etc.] Addison-Wesley, 2000.

T

[Tanenbaum91] Tanenbaum, Andrew S.- *Redes de ordenadores* -Ed. Prentice-Hall – 1991

[Tiwana99] Tiwana, Amrit - *The Knowledge Management Toolkit* - Publisher: Prentice Hall PTR, Pub Date: December 06, 1999

[TELE-85] *Sistema radio RT-21.* T.C. Telecomunicación y Control, S.A. Edición 1.1985.

[Townsend95] Townsend, Robert L. - *SNMP Application Developer's Guide* - New York [etc.] John Wiley and Sons, 1995

[Tsing-Hwa91] Tsing-Hwa Chi; Minder Chen; Yihwa Kiang - *A generalized case based reasoning system for personnel performance evaluation* - System Sciences, 1991. Proceedings of the Twenty-Fourth Annual Hawaii International Conference on, Volume: iii , 8-11 Jan 1991 Page(s): 82 -89 vol.3

[T101] Stanadars Commitee T1 Telecommunications, *Telecom Glossary 200*, American National Stanadart T1.523-2001, febrero de 2001.

[Truemper04] Truemper, K., 1942 - *Design of logic-based intelligent systems* - Hoboken, NJ: John Wiley, 2004.

U

[Udupa99] Udupa, Divakara K. *TMN : Telecommunications Management Network* - McGraw-Hill, 1999.

W

[Weiss98A] Gary Weiss, John Eddy, Sholom Weiss - *INTELLIGENT TELECOMMUNICATION TECHNOLOGIES* - Appears in Knowledge-Based Intelligent Techniques in Industry (chapter 8), L. C. Jain, editor, CRC Press, 249-275, 1998.

[Weiss98B] Weiss, Gary and Ros, Johannes p. – *"Implementing Design Patters with Object-Oriented Rules"* - Appears in the Journal of Object-Oriented Programming, 11(7): 25-35, SIGS Publication Inc., Nov/Dec 1998.

[Winston94] Winston, Patrick Henry - *Título Inteligencia artificial* - Publicación Argentina [etc.]: Addison-Wesley Iberoamericana, 1994.

Anexo F

Glosario.

A

- Abducción: Es un método de razonamiento comúnmente utilizado para generar explicaciones. A diferencia de la inducción, la abducción no garantiza que se puedan lograr conclusiones verdaderas, por lo tanto no es un método sólido de inferencia.

- Algoritmo A*: Algoritmo de búsqueda inteligente o informada que busca el camino más corto desde un estado inicial al estado meta a través de un espacio de problema, usando una heurística óptima. Como ignora los pasos más cortos en algunos casos rinde una solución subóptima.

- Adquisición del Conocimiento: Proceso por el cual se introducen informaciones dentro de la Base de Conocimientos de un sistema. Puede ser manual o automática, en este último caso hablamos de aprendizaje.

- Aprendizaje: Subdomino de la Inteligencia Artificial que trata del desarrollo de teorías computacionales del aprendizaje y la construcción de sistemas con capacidad discente. En términos prácticos decimos que ha ocurrido un proceso de aprendizaje cuando el sistema mejora su desempeño al realizar determinada tarea, como resultado de instrucciones dadas por otros sistemas o de la repetición de la misma tarea o tarea similares. Es decir un proceso de acceso a nuevo conocimiento de objetos, fenómenos o procedimientos, así como nuevo conocimiento de las relaciones entre objetos y/o fenómenos.

- Atributo: Cualidad de un objeto. Por ejemplo el color es un atributo del objeto "materia", En relación con "frames" el atributo es sinónimo de la ranura o el "slot".

B

- ***Base de conocimientos:*** Subsistema que representa los conocimientos de un dominio dado, de manera accesible y manipulable por un “Mecanismo de Inferencia”.

- Base de reglas: Un conjunto de fórmulas lógicas que describen las relaciones o las funciones del mundo real, tales como se usan en muchos programas de inteligencia artificial como, por ejemplo, los de sistemas expertos para ejecutar inferencias. Cada regla de la base de reglas es una proposición lógica, aunque puede adoptar la forma exigida por uno cualquiera de los diversos formatos en uso. Se afirma que un sistema experto es un programa que contiene un motor de inferencia generalizado y una base de reglas particularizada. En esa nomenclatura, base de reglas y base de conocimiento son sinónimos.

- Búsqueda primero en amplitud: Se trata de una búsqueda desinformada cuyo algoritmo prefiere expandir al nodo con menos descendientes de una estructura de árbol de búsqueda, o más próximo. Es el del tope de la lista de espera (first in, first out - FIFO). Cuando se expande un nodo, los sucesores van al final de dicha lista.

- Búsqueda primero en profundidad: Se trata de una búsqueda desinformada donde el nodo más profundo no terminal es el que primero se expande. Aquí la lista de espera de nodos por procesar crece por el tope. Los "sucesores", que son los nodos recién expandidos (siempre que no sean terminales, que no se podrían expandir) son anotados en el tope de la lista de espera (last in first out - LIFO).

C

- Cognición: Procesamiento intelectual avanzado de la información, maduración de la información por el gran salto de encontrarle significado. Pensar, Considerar. Procesamiento cerebral de datos.

- Conocimiento, Representación del Conocimiento: Se denomina conocimiento al resultado, por parte del humano, de la maduración semántica de la información y su comprensión experiencial. En inteligencia artificial, una técnica, una manera consistente y útil de organizar la información en la computadora, cuyo objetivo es facilitar su procesamiento. Se lo denomina KR (knowledge representation). Entre los esquemas de KR aparecen reglas de lógica simbólica, frames, redes semánticas y gráficos conceptuales. El tema general se denomina "modelado del conocimiento" (knowledge modeling).

- Correlación de variables: Método matemático para estudiar la distribución de los valores de dos funciones, para encontrar similitudes.

D

- Determinismo: Explicación de la realidad basada en la existencia de productores de resultados predectibles. Los programas determinísticos se ramifican al llegar a un IF ... THEN, sólo en base a condiciones absolutas. El determinismo aplica habitualmente lógica dicotómica para verificar variables de tal manera de mantener bajo control el flujo del programa, llevando a éste hacia la producción de resultados deseados. Cuanto más una técnica de programación sea comandada por los datos, tanto menos predecible es el resultado.

E

- Encadenamiento Hacia Adelante: Llamada Forward Chaining, es un razonamiento dentro de un sistema lógico que avanza desde los datos o premisas hacia las conclusiones cuya validez aún no sabíamos. Aplica en forma directa la regla del Modus Ponens genérico.

F

-Fórmula bien formada (well-formed formula, wff): Una fórmula bien formada puede ser una proposición simple o compuesta que tiene sentido completo y cuyo valor de veracidad, puede ser determinado. La lógica proposicional proporciona un mecanismo para asignar valores de veracidad a la proposición compuesta, basado en los valores de veracidad de las proposiciones simples y en la naturaleza de los conectores lógicos involucrados.

G

- Guión: (Script), secuencia imperativa de acciones que guía a un sistema para actuar. Contiene además de esas acciones, las expectativas acerca de lo que normalmente ocurre en la situación típica en que se desenvuelve el sistema.

H

- Heurística: Conocimiento imperfecto pero útil empleado en el razonamiento o la solución de un problema. También, regla de inferencia aproximada. Usado a veces como sinónimo de “Regla Heurística”.

I

Incertidumbre: En teoría de la información, la certidumbre es el grado cómo la información se puede considerar verdadera, completa y digna de fe. La incertidumbre se origina a partir de elementos de datos falsos o de un equívoco, a partir de datos incompletos o de un contexto ambiguo. La confianza es la respuesta humana a la certidumbre. Por ello ciertos modelos de información usan valores de confianza.

- Inferencia: Proceso mental por el cual se extraen conclusiones a partir de premisas más o menos explícitas, proceso resultante ya sea del "sentido común", ya sea de silogismos informales y formales, o bien por aplicación de las reglas muy detallistas y cálculos de la inferencia estadística

- Inteligencia Artificial: Disciplina dedicada a desarrollar y aplicar enfoques computacionales al comportamiento inteligente. Estudia preferencialmente los comportamientos y fenómenos de percepción, solución de problemas, razonamientos, utilización de lenguajes naturales y planeamiento de actividades.

L

- LISP: Lenguaje utilizado en inteligencia artificial, basado en el manejo de listas por medio de funciones.

- Lógica Difusa: Una forma de razonamiento que incorpora criterios múltiples para tomar decisiones y valores múltiples para evaluar posibilidades.

M

- Marco: (Frame), Estructura de datos con casillas (slots) a las cuales se le puede vincular otras estructuras. Normalmente las casillas poseen valores asumidos por omisión que se pueden reemplazar por valores específicos. Los marcos permiten representar las clasificaciones jerárquicas y la herencia de características entre objetos, además permite trabajar con “orientación a objetos”. Es una estructura de datos muy utilizada en las investigaciones y aplicaciones de inteligencia artificial.

- Metareglas: Reglas sobre las reglas aplicables a un dominio. Indican, por ejemplo el orden de prioridad en que deben utilizarse las reglas del dominio de aplicación.

- Métodos Débiles: Métodos generales de solución de problemas, aplicable en ausencia de conocimientos específicos del dominio del problema. Ejemplo de aplicación en el algoritmo de Análisis de Medios y Metas.

- Modus ponens: Una regla de inferencia usada para probar la corrección o valor de verdad de proposiciones lógicas. Sean dos afirmaciones cualesquiera A y B. La forma de la regla es: A=>B, A. Se sabe que si A es verdad, B también es verdad.

- Modus tollens: Regla de inferencia usada para probar la corrección o el valor de verdad para proposiciones lógicas. Sean A y B dos afirmaciones genéricas cualesquiera. La forma de la regla puede abarcar A, B y otras más: A=>B, ~B. Se sabe que cuando A es verdad, B es verdad. Se sabe también que B no es verdad, se puede inferir que A no es verdad.

N

- No-determinismo: La producción de resultados impredecibles. Las técnicas de programación no-determinísticas se emplean para implementar la lógica difusa o para simular el caos. Los algoritmos no-determinísticos se respaldan a menudo en procesos aleatorios, accidentales o impredectivos. Muchos autores sugieren que la actividad cerebral es no-determinística. Aún los mejores algoritmos de aleatorización resultan finalmente predecibles de suerte que entre los sistemas computacionales no se ha hallado una forma de generar un no-determinismo puro.

P

- Perceptrón: Un modelo electrónico probabilístico del almacenaje y de la organización cerebral. Propuesto en la década de 1950 por Rosenblatt, sus unidades son células electrónicas que exhiben características de procesamientos adaptables o cambiables a través de la exposición repetida al aprendizaje, esto es, a ser expuestas a una entrada o input repetido.

- Peso: En inteligencia artificial la palabra "peso" aparece con motivo del nivel de confianza que en forma abstracta se le confiere a la cantidad de activación que difunde desde una neurona a las que le siguen en una red neuronal natural o desde un nodo a los que lo siguen en una red neural artificial. Cuando se aumenta el peso de las entradas a una neurona o nodo, es más fácil lograr superar el umbral de disparo del mismo y se propaga la señal del potencial de acción.

- Prolog: Lenguaje de programación “lógica”, basado en el cálculo de predicados.

R

- Razonamiento: Es el proceso cognitivo por el cual se realizan inferencias acerca de datos para interpretar una observación o situación basado en restricciones del pasado, del presente o del futuro, restricciones que influyen sobre el resultado.

- Razonamiento no monótono: Según McCarthy, tanto los humanos como los autómatas deben extraer conclusiones que son verdad en los mejores modelos de los hechos que se han tenido en cuenta. Hay diversos conceptos de lo que es mejor en sistemas diferentes. Muchos implican la minimización de algo. A medida que se agregan nuevos hechos, algunas de las anteriores conclusiones no sobreviven. Por este motivo, el razonamiento que permitió arribar a esas conclusiones, se denomina no-monótono.

- Razonamiento probabilístico: Es un caso dentro del razonamiento no monótono. Cuando la probabilidad que la verdad de una oración haya cambiado con respecto a su valor inicial (por ejemplo 1), otras oraciones que anteriormente tenían alta probabilidad pueden decaer hasta tener probabilidad pequeña o nula. Cuando se configuran los modelos probabilísticos, esto es, cuando se define el espacio para una muestra de "eventos" susceptibles de recibir diferentes valores de probabilidad, se está apelando a un razonamiento no-monótono más general, pero la convención es que esto no lo hace la computadora sino una persona que razona informalmente.

- Red Bayesiana: Es un mecanismo para la representación del conocimiento probabilístico. Está más bien relacionado con sistemas expertos. Los algoritmos de inferencia en estas redes usan la estructura de red para generar inferencias en forma eficiente en comparación con las distribuciones de probabilidad conjuntas afectando a todas las variables.

- Red Neuronal: Un modelo artificial de neuronas en el cerebro, consistente usualmente en una computadora provista de nodos o células interconectados que pueden poseer uno de dos valores, sí o no, con enlaces calibrados para tener valores numéricos (llamados pesos) para dejar pasar flujos de potencial. Dichos flujos adquieren patrones dictados por el peso de los enlaces.

- Red Semántica: Es un esquema de representación del conocimiento en que los objetos o los conceptos se almacenan como nodos (sinónimo, vértices) de un gráfico y relacionados entre sí por atributos o arcos etiquetados, tales como 'es_un" o "tiene_un".

- Representación explícita: Un esquema de representación del conocimiento donde el conocimiento se almacena como lógica simbólica o alguna otra forma legible para el humano. Comparar con la representación implícita donde el conocimiento se almacena como un conjunto de datos y pesos que no son legibles para el humano

- Representación implícita: En un esquema de procesamienrto en paralelo tal como lo es una red neural usando técnicas de aprendizaje adaptivas, muchos elementos de procesamiento o de memoria pueden ser entrenados para reconocer inputs sin necesidad de almacenar una copia del input en la memoria

S

- Sistema Basado en Conocimientos (CBS): Sistema computacional que utiliza un "mecanismo de inferencia" y "una Base de Conocimientos" para la solución de problemas.

- Sistema Experto: Un capítulo de la inteligencia artificial que ejecuta, a partir de una base de conocimientos o de reglas, algunas tareas que normalmente requieren cierto nivel de experiencia por parte del humano. Estos programas usualmente logran satisfacer inferencias usando dicho conjunto de reglas para interpretar entradas y generar salidas que suenen a inteligente.

- Sinapsis: Un punto de enlace entre dos neuronas.

- Slot o Ranura: es un componente del frame. Con slots pueden describirse cualidades intrínsecas de un objeto. Sinónimo en sistemas expertos: atributo.

- Soma: El cuerpo celular de una neurona. No es el núcleo de la neurona: abarca tanto el núcleo como otros componentes citoplasmáticos.

V

- Very Large Scale Integration VLSI, (integración a muy alta escala): Técnica para la fabricación de circuitos integrados que contienen más de diez mil elementos en una sola pastilla o chip. Su uso ha facilitado que se diseñen microprocesadores de 32 bits. En inglés se conoce a estos circuitos por las siglas VLSI.