Departamento de Ingeniería de Sistemas Telemáticos
Escuela Técnica Superior de Ingenieros de Telecomunicación
Universidad Politécnica de Madrid

TESIS DOCTORAL

ESPECIFICACIÓN DE MODELOS DE INFORMACIÓN DE GESTIÓN DE RED INTEGRADA MEDIANTE EL USO DE ONTOLOGÍAS Y TÉCNICAS DE REPRESENTACIÓN DEL CONOCIMIENTO

Autor

Jorge Enrique López de Vergara Méndez

Ingeniero de Telecomunicación

Director

Víctor Abraham Villagrá González

Doctor Ingeniero de Telecomunicación

Madrid, 2003

Título: Especificación de Modelos de Información de Gestión de Red Integrada Mediante el Uso de Ontologías y Técnicas de Representación del Conocimiento.

Autor: Jorge Enrique López de Vergara Méndez

Director: Víctor A. Villagrá González

TRIBUNAL CALIFICADOR

Presidente:

- ❑

Vocales:

- ❑
- ❑
- ❑

Secretario:

- ❑

Realizado el acto de defensa y lectura de Tesis en Madrid, el día de de .

CALIFICACIÓN:

El presidente Los Vocales

El secretario

Agradecimientos

En primer lugar debo dar las gracias muy especialmente a mi director de Tesis, Víctor Villagrá. Su trabajo, apoyo y valiosa orientación han hecho posible la finalización de este trabajo.

También quisiera expresar mi más sincera gratitud a Julio Berrocal por la confianza y el apoyo demostrados, así como al resto de los miembros del Departamento de Ingeniería de Sistemas Telemáticos, en particular a los integrantes del grupo de Redes y Servicios Telemáticos, por crear un excelente ambiente de trabajo.

Gracias a Juan Ignacio Asensio por dedicar algo de su tiempo cuando venía de visita desde Valladolid para discutir algunas ideas que le planteaba acerca de la Tesis que me ayudaron a comprender mejor diversos aspectos, así como por su excelente disposición en todo momento para ayudar en cualquier cuestión.

Igualmente quiero agradecer a todas las personas con las que trabajé durante mi estancia en los Laboratorios de Hewlett-Packard en Bristol, especialmente a Julio Guijarro. Gracias a ellos amplié mis conocimientos acerca de los modelos de información de gestión, lo que sin duda ha ayudado a mejorar contenido de la Tesis.

Mi agradecimiento también a Ana Belén García, mi compañera de cubículo durante todo este tiempo, así como a los demás compañeros y doctorandos, tanto del B-203 como del resto del departamento, por los buenos momentos pasados, por la ayuda y el apoyo que me han dado, y por haber contribuido en gran medida a un ambiente de trabajo agradable. En especial nuevamente a Ana por las aclaraciones y ayuda prestadas al respecto de los trámites de la Tesis.

Quiero asimismo destacar y agradecer a Alma, a quién dedico este trabajo, y el resto de mi familia por el apoyo, la comprensión y el cariño que me han demostrado durante el prolongado y en ocasiones difícil periodo de tiempo que he estado dedicado a la realización de esta Tesis.

Resumen

La multiplicidad de modelos de Gestión de Red (SNMP, CMIP, DMI, CORBA, WBEM...) ha supuesto en la última década el establecimiento de mecanismos que posibiliten la interoperabilidad entre los distintos dominios de gestión. Uno de los pilares básicos de dicha interoperabilidad es la creación de correspondencias entre los modelos de información de gestión que especifica cada dominio, y que en general se ha llevado a cabo mediante traducciones sintácticas que no tienen en cuenta los aspectos semánticos de la información definida. Llevar estas traducciones a un nivel semántico pasa por la utilización de ontologías, ampliamente utilizadas en el campo de la Inteligencia Artificial, que justamente se enfocan en el significado de los conceptos que componen un modelo de información.

Esta tesis tiene como objetivo perfeccionar las técnicas actuales de interoperabilidad entre dominios de gestión de red mediante el estudio y aplicación de ontologías formales que permitan especificar la información de gestión de una forma óptima, consiguiendo integrar en un mismo modelo todos los conceptos que actualmente pertenecen a distintos dominios de gestión. Una de las grandes ventajas adicionales que tiene esta aproximación es poder aprovechar la capacidad que tienen las ontologías para especificar comportamientos para incluirlo como parte de las definiciones de recursos gestionados, de forma que gestores de red inteligentes puedan interpretar estas reglas para realizar tareas de gestión.

De esta manera, se puede simplificar el manejo de la información de gestión, evitando el problema actual de los gestores, que para cada dominio deben utilizar una información independiente del resto, y sin posibilidad de establecer relaciones directas entre ellas.

Palabras clave: Ontología, Gestión de Red, Lenguaje de Información de Gestión, Integración de Modelos de Información, Información de Comportamiento.

Abstract

The multiplicity of Network Management models (SNMP, CMIP, DMI, CORBA, WBEM...) has posed in the last decade the set up of mechanisms to allow the interoperability among all involved management domains. One of the basic pillars of such interoperability is the mapping between the information models that each domain specifies. Usually, these mappings have been carried out with syntactical translations that do not bear in mind the semantic aspects of the defined information. These translations could reach the semantic level by using ontologies. These ontologies, widely used in Artificial Intelligence, exactly focus on the meaning of those concepts composing an information model.

The goal of this thesis is the improvement of current network management interoperability techniques through the study and application of formal ontologies. They would eventually allow the specification of management information in an optimal way, obtaining the integration of all concepts that currently belongs to different management domains in the same model. One of the additional advantages of this approach is the ability of using the capacity of the ontologies to specify behavior for including it as part of the managed resources definitions, so that intelligent network managers can interpret these rules to perform management tasks.

In this way, the management information handling can be simplified, avoiding managers' current problems, which have to deal with independent information models for each domain, and without the possibility of setting up direct relationships between them.

Keywords: Ontology, Network Management, Management Information Language, Information Models Integration, Behavior Information.

A Alma, por su apoyo constante durante estos cuatro años y medio

Al bebé que está por venir

Índice general

Índice de figuras

Índice de tablas

Capítulo 1 Introducción y objetivos

1.1 Introducción y motivación

La gestión de redes y servicios ha sido un campo en el que tradicionalmente 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. Ante este escenario, a finales de los años 80 y principios de los 90 del siglo XX surgieron los denominados modelos de gestión de red integrada, que definían de manera normalizada un protocolo y un modelo de información de gestión que rompían el acceso de gestión remota propietario, permitiendo teóricamente la interoperabilidad entre gestores y elementos gestionados de múltiples fabricantes.

Debido a diversas razones, se plantearon varios modelos de gestión de red integrada. Inicialmente aparecieron SNMP (*Simple Network Management Protocol*, Protocolo de Gestión de Red Simple) [Harrington02], propuesto en Internet, y OSI-SM (*Open Systems Interconnection - Systems Management*, Interconexión de Sistemas Abiertos, Gestión de Sistemas) [ITUT97b], estandarizado por ISO (*International Organization for Standardization*, Organización Internacional para la Estandarización.). Posteriormente, otros modelos se han ido sumando a éstos, como DMI (*Desktop Management Interface*, Interfaz de Gestión de Equipos de Sobremesa) [DMTF03a], especificado por el DMTF (antiguamente *Desktop Management Task Force*, Grupo de Trabajo de Gestión de Equipos de Sobremesa, en la actualidad *Distributed Management Task Force*, Grupo de Trabajo de Gestión Distribuida), y CORBA (*Common Object Request Broker Architecture*, Arquitectura Común de un Intermediario de Peticiones de Objetos) [OMG02c], definida por OMG (*Object Management Group*, Grupo de Gestión de Objetos). Todos estos modelos han posibilitado el acceso de manera uniforme a los recursos gestionados de una red de comunicaciones al normalizar el protocolo de comunicaciones, la estructura de la información de gestión y conjuntos de definiciones de dicha información de gestión.

Las distintas soluciones que han permitido en este tiempo una gestión integrada de recursos heterogéneos también han supuesto la paradoja de que no pueda existir realmente una gestión totalmente integrada, dado que son modelos incompatibles entre sí. Al cabo de los años cada modelo se ha utilizado en un ámbito de aplicación distinto: mientras que la gestión OSI se utiliza sobre todo para la gestión de equipos y servicios en redes de

telecomunicación y se ha aplicado en redes de operadoras de telecomunicación a través de TMN (*Telecommunication Management Network*, Red de Gestión de las Telecomunicaciones) [ITUT00a], SNMP se utiliza ampliamente en Internet y está más orientada a la gestión de equipos y servicios sobre redes de datos, DMI se usa para la gestión de equipos de sobremesa, tales como ordenadores personales (PCs, *Personal Computers*), y la aplicación de una tecnología de procesamiento distribuido como CORBA permite acceder directamente a los parámetros gestionables de aplicaciones que proporcionan un servicio.

El problema de esta falta de interoperabilidad reside en que cada modelo de gestión utiliza su propio protocolo, su propia estructura de la información y sus propias definiciones de información:

- En el caso del protocolo, en OSI se utiliza CMIS (*Common Management Information Service*, Servicio de Información de Gestión Común) [ITUT97e] y CMIP (*Common Management Information Protocol*, Protocolo de Información de Gestión Común) [ITUT97f], un protocolo potente que balancea la carga de la gestión entre gestor y agentes, mientras que en Internet se emplea SNMP [Case02], un protocolo simple que basa su funcionamiento en un gestor centralizado. DMI permite el acceso desde RPCs (*Remote Procedure Calls*, Llamadas a Procedimientos Remotos) y agentes SNMP [DMTF03a], y CORBA permite la comunicación con los recursos a través de un nuevo protocolo de acceso (IIOP, *Internet Inter-ORB Protocol*, Protocolo Inter ORB para Internet) [OMG02e].
- En lo que se refiere a la estructura de la información, OSI definió GDMO (*Guidelines for the Definition of Managed Objects*, Guía para la Definición de Objetos Gestionados) [ITUT92a], un modelo orientado a objetos con gran capacidad expresiva, mientras que en Internet se ha definido SMI (*Structure of Management Information*, Estructura de la Información de Gestión) [McCloghrie99a], que especifica estructuras de datos muy simples (variables y tablas), existiendo de nuevo grandes diferencias entre estos modelos. Actualmente se está estudiando una nueva versión de SMI llamada SMIng (*SMI Next Generation*, la Siguiente Generación de SMI) [Schönwälder99] que permita definir más claramente las relaciones existentes entre conceptos definidos en SMI [Schönwälder01]. En cualquier caso, SMIng no llega a ser un modelo con tanta capacidad expresiva como GDMO. DMI utiliza su propio formato para definir la información (MIF, *Managed Information Format*, Formato de la Información Gestionada) [DMTF03a], que es tan simple como SMI, pero con algunas características propias. Las interfaces de CORBA usan un lenguaje orientado a objetos (IDL, *Interface Definition Language*, Lenguaje de Definición de Interfaces) [OMG02d], pero también tiene estructuras diferentes a las que posee GDMO.

- La definición de la información en todos los modelos, partiendo de que se han definido mediante dos lenguajes distintos (GDMO, SMI, MIF e IDL) con distinta capacidad expresiva, también es diferente. Además, cada modelo ha definido su información de manera totalmente independiente al otro. Por ejemplo, TMN definió en la recomendación M.3100 [ITUT95a] un modelo genérico de información de red. En Internet, por su parte, se ha definido un conjunto de información contenidas en distintas RFCs (*Request for Comments*, Solicitud de Comentarios) entre las que cabe destacar la MIB-II (*Management Information Base II*, Base de Información de Gestión II) [McCloghrie91]. En DMI se ha especificado la Master MIF [DMTF02b], que posee componentes referidos a los equipos de sobremesa. Finalmente, también existirán distintas definiciones de información descritas como interfaces IDL.

Ante la heterogeneidad de modelos de Gestión de Red integrada se hace necesario establecer mecanismos que posibiliten la interoperabilidad entre los distintos dominios implicados, dado que existen diversos entornos en los que varios de estos modelos deben coexistir con la problemática que esto conlleva. Con esto, se proporcionaría una vista unificada de los sistemas gestionados, independientemente del dominio al que pertenezcan. Los estudios existentes dividen este trabajo básicamente en dos cuestiones, que son el protocolo de comunicaciones y el modelo de información: si es posible definir un conjunto de reglas que traduzcan ambos, la interoperabilidad es posible. El ejemplo más notable que trata de establecer una arquitectura de interoperabilidad de modelos de gestión es la iniciativa WBEM (*Web Based Enterprise Management*, Gestión de Empresa Basada en Web) y su modelo de información asociado CIM (*Common Information Model*, Modelo de Información Común) [DMTF99], que permite el acceso a diversos dominios de gestión con una interfaz neutra a todos ellos.

Sin embargo, un aspecto que todavía no tiene fácil solución se presenta cuando en dos dominios distintos de gestión se representa un mismo concepto de distinta manera: una traducción meramente sintáctica de un modelo origen no proporcionará el concepto existente en el modelo destino. Por otro lado, también es conveniente que un gestor sea capaz de relacionar la información de dos dominios distintos, de forma que si se produce un fallo en un dominio se pueda asociar a la existencia de otro problema en el otro dominio. Se hace por tanto necesaria una traducción semántica que haga corresponder directamente los conceptos de ambos dominios, para poder alcanzar una gestión totalmente integrada de todos los recursos, independientemente de la forma en que estén especificados. Las aproximaciones actuales, incluyendo CIM, no son capaces de aportar soluciones a esta cuestión, pues hasta la fecha sólo han aplicado traducciones sintácticas entre los distintos lenguajes de definición de información de gestión.

Mientras tanto, en el campo de la Inteligencia Artificial, a principios de los años 90 se definen las ontologías como medio de compartir y reutilizar el conocimiento [Neches91], habiéndose utilizado con éxito para solventar problemas similares en otros ámbitos de

aplicación como la Web Semántica [BernersLee01], donde estas técnicas basadas en el conocimiento permiten actualmente añadir a las páginas y servicios Web la semántica de la que carecían hasta el momento.

La aplicación de estas técnicas al campo de la gestión de redes y servicios permitiría afrontar una gestión realmente integrada de los diversos recursos que forman parte de un sistema en red, que normalmente pertenecen a distintos dominios de gestión: redes de telecomunicación gestionadas según TMN, conmutadores y encaminadores gestionados con SNMP, ordenadores administrados con DMI o servicios electrónicos accesibles con una interfaz CORBA se gestionarían de manera unificada desde un gestor con un único modelo de información.

Además, la capacidad declarativa que añaden las ontologías permite incluir definiciones de reglas que modelen el comportamiento del gestor, de forma añadida al modelo de información de gestión que se haya especificado. Con esto, se podría especificar de manera conjunta toda la información relativa a la gestión: la información definida para gestionar los distintos recursos junto con la información relativa al comportamiento de un gestor que quiera administrar dichos recursos.

Como consecuencia de la situación anteriormente expuesta, esta tesis se enmarca en el proceso de búsqueda de soluciones que faciliten la interoperabilidad entre modelos de información, analizando los requisitos necesarios para llevar a cabo una traducción semántica entre los mismos. Para ello se utiliza la técnica de representación del conocimiento conocida como ontología, dado que proporciona las construcciones necesarias para añadir semántica a la información representada. La Figura 1.1 ilustra el ámbito de la tesis doctoral, sus objetivos y las contribuciones con las que pretende alcanzarlos.

La tesis doctoral parte de un estudio general de los modelos de gestión integrada, prestando especial atención a la forma en que se especifican los distintos modelos de información. Igualmente, se muestran las aproximaciones existentes actualmente que proporcionan mecanismos de interoperabilidad, comprobando las limitaciones de dichas propuestas. Así, se presentan las ontologías como un método de representación del conocimiento que tiene en cuenta los aspectos semánticos de la información. De este estudio se puede concluir que los actuales métodos de integración de información no son suficientes para llevar a cabo una gestión con un punto de vista común para todos los dominios de gestión y que las técnicas utilizadas en las ontologías permiten obtener un modelo común que se genere a partir de los existentes, teniendo en cuenta el significado de la información que contiene cada uno de ellos.

Para tratar de solucionar el problema de la heterogeneidad de modelos de gestión integrada, esta tesis doctoral realiza dos propuestas que mejoran la interoperabilidad de los modelos de información de gestión, analizando los requisitos necesarios para llevar a cabo la traducción semántica:

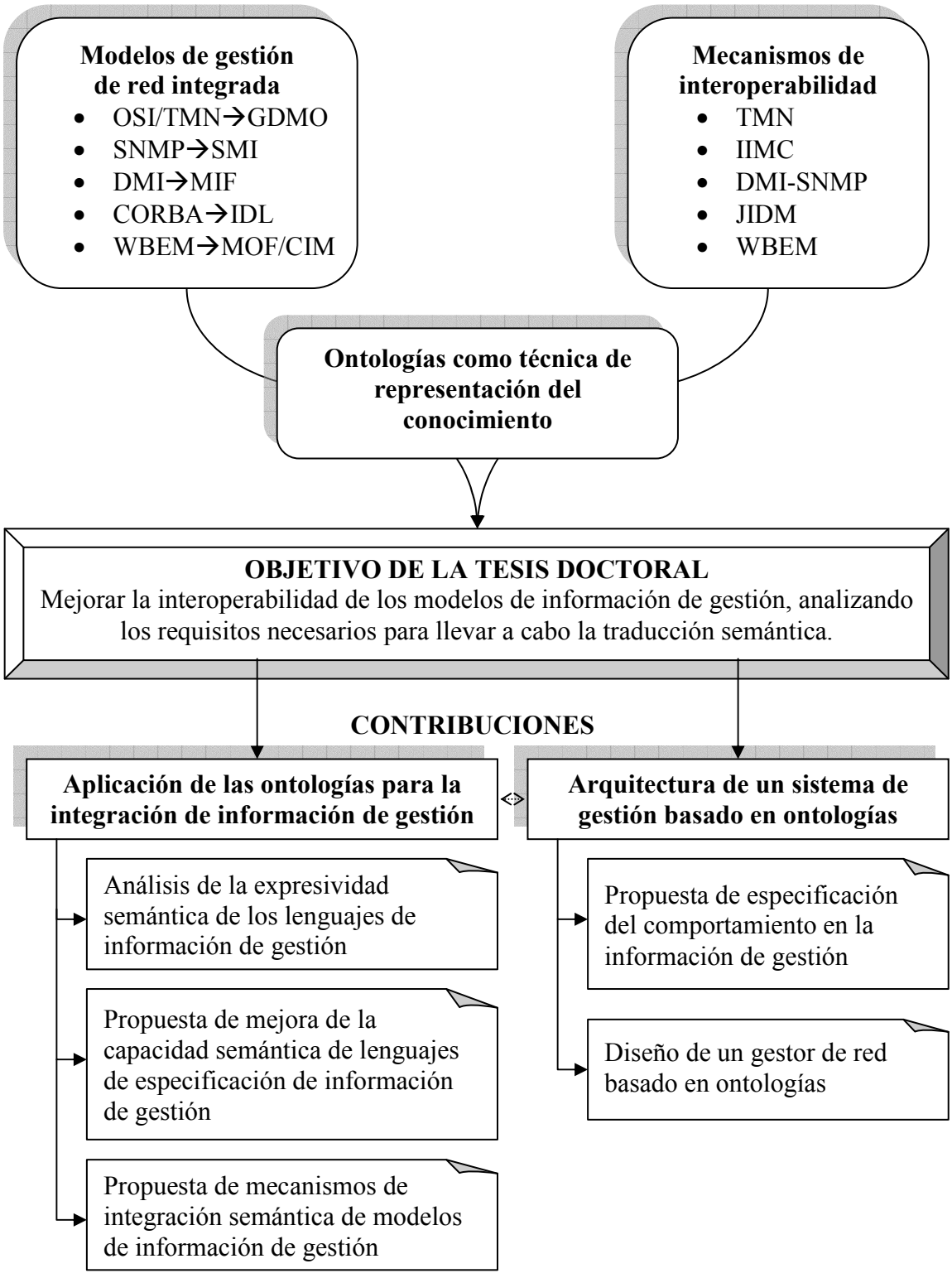

Figura 1.1. Contexto de la tesis doctoral.

1. La aplicación de las ontologías a la integración de información de gestión, aprovechando sus construcciones para comparar la expresividad semántica de los lenguajes de información de gestión, utilizando sus lenguajes de definición para especificar información de gestión y aplicando sus mecanismos de integración a los modelos de gestión existentes.

2. La definición de una arquitectura de un sistema de gestión que utilice el modelo integrado generado a partir de la propuesta anterior al que se puede añadir un conjunto de reglas que especifiquen el comportamiento.

A continuación se presenta en mayor profundidad los objetivos que se han planteado para esta tesis doctoral.

1.2 Objetivos de la Tesis Doctoral

Tradicionalmente se han establecido mecanismos que posibiliten la interoperabilidad entre los distintos dominios implicados en la gestión de recursos, debido a la heterogeneidad de modelos de Gestión de Red Integrada (SNMP, OSI, DMI, WBEM). La interoperabilidad es posible si se consigue definir un conjunto de reglas que traduzcan el protocolo de comunicaciones y modelo de información de gestión. Sin embargo, las reglas existentes en la actualidad no pasan de realizar traducciones sintácticas de los modelos de información. Esto quiere decir que cuando un mismo recurso se describe con dos lenguajes de definición de información de gestión, es posible aplicar una traducción directa entre las estructuras que contienen la descripción, pero no entre sus significados. Se hace por tanto necesaria una traducción semántica que haga corresponder directamente los conceptos de ambos dominios.

En este contexto, se plantea el objetivo general: **proponer soluciones que permitan mejorar la interoperabilidad en lo que se refiere a los modelos de información de gestión, analizando los requisitos necesarios para llevar a cabo la traducción semántica**. Para ello, la técnica de representación del conocimiento conocida como ontología es una de las respuestas más adecuadas a este problema, dado que proporciona las construcciones necesarias para añadir semántica a la información representada, pudiendo ser la clave que permita realizar una gestión realmente integrada e inteligente de los diversos recursos que posee una empresa.

Con objeto de alcanzar el objetivo general, este trabajo de tesis doctoral incluye los siguientes objetivos concretos, ilustrados en la Figura 1.2, agrupados en dos propuestas:

- **Propuesta de aplicación de las ontologías para la integración de información de gestión**. Ésta es la propuesta principal de la tesis, que trata de integrar semánticamente los modelos de información de gestión existentes. Para ello, se sirve de tres pasos:

    1. **Análisis y comparación de la expresividad semántica de los modelos actuales de información de gestión**. El primer objetivo a alcanzar debe ser el estudio de la expresividad semántica que proporcionan los distintos lenguajes de información de gestión. Para ello, se utilizarán los términos habitualmente empleados en las ontologías. Esto permitirá establecer la capacidad semántica

que posee cada uno de ellos, y así facilitar los mecanismos de integración que se deban realizar.

2. **Propuesta de mejora de la capacidad semántica de lenguajes de especificación de información de gestión**. Una vez realizado el análisis previo, será necesario añadir las características de un lenguaje que incluya la capacidad semántica para la especificación de información de gestión. Para ello se puede optar por utilizar algún lenguaje de ontologías, o bien un lenguaje de gestión de red, adaptándolo para que la expresividad semántica que posea sea similar a la de dichos lenguajes de ontologías.

3. **Diseño y propuesta de mecanismos de integración semántica de modelos de información**. Asimismo, y conectando con los otros objetivos, se deberán establecer los mecanismos que posibiliten la integración semántica entre los distintos modelos de información de gestión existentes, permitiendo la fusión de modelos pertenecientes a distintos dominios de gestión, y estableciendo de manera unívoca las correspondencias necesarias para que un sistema de gestión pueda obtener la información de dichos dominios a partir del modelo fusionado. El resultado de esta propuesta consistiría en la obtención, por un lado, de una ontología de gestión fruto de la fusión de la información de gestión existente, y por otro lado, de una ontología de correspondencia para realizar las traducciones semánticas entre la ontología global y la información de cada dominio de gestión.

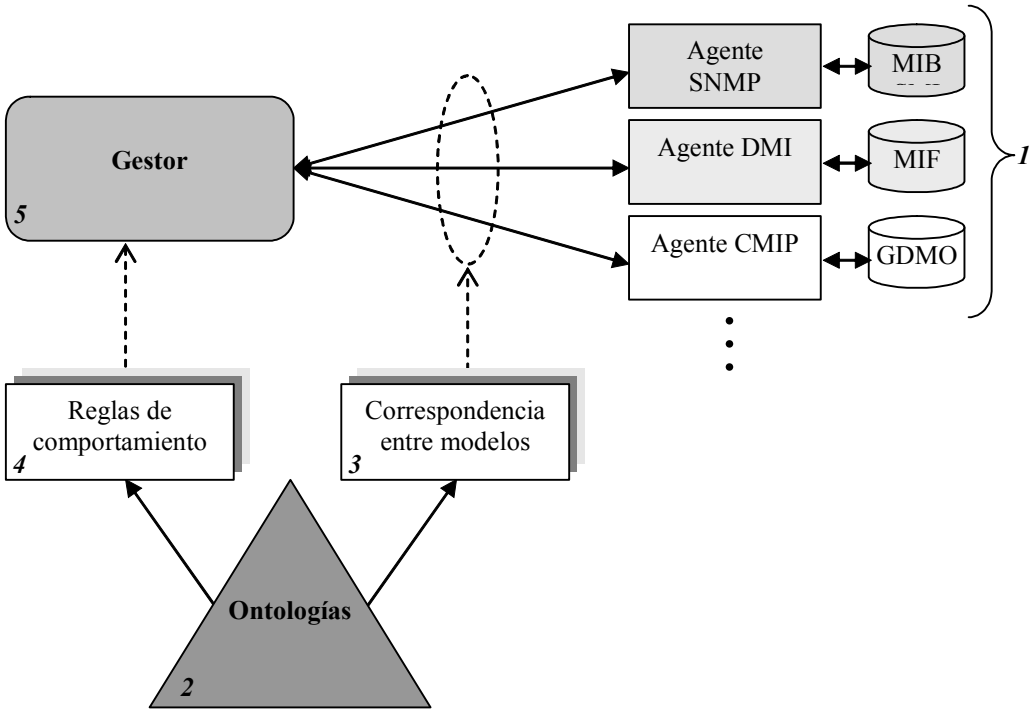

Figura 1.2. Objetivos de la Tesis Doctoral.

- **Propuesta de arquitectura para un sistema de gestión basado en ontologías**. Partiendo de los resultados obtenidos en la propuesta anterior, ésta la complementa aprovechando las posibilidades que aportan las ontologías para un sistema de gestión:

    4. **Análisis de lenguajes de especificación de comportamiento desde el punto de vista de su inclusión en las definiciones ontológicas**. El modelo de información de gestión basado en ontologías se puede completar incluyendo reglas de restricción que definan su comportamiento. Para ello se estudiarán las capacidades que en este sentido poseen los lenguajes utilizados para especificar la información de gestión en términos de ontologías.

    5. **Diseño y propuesta de la arquitectura de un gestor de red basado en ontologías**. Por último, y siguiendo con un modelo semántico basado en ontologías, es necesario definir la arquitectura que debe poseer un gestor que maneje las ontologías de gestión y sea capaz de actuar según el comportamiento definido en las mismas. Esta arquitectura debe ser capaz de trabajar con la ontología producto de la fusión de la información de gestión existente, que además contendrá información adicional relacionada con el comportamiento, y a la vez, ser capaz de utilizar la ontología de correspondencia para obtener la información necesaria de los distintos dominios de gestión implicados.

1.3 Estructura de la memoria

La memoria de esta tesis sigue el esquema presentado en la Figura 1.1:

- El Capítulo 2 presenta el estado del arte de la gestión de red integrada, dando un repaso a los distintos modelos de gestión integrada existentes, haciendo especial hincapié en las diferencias existentes en sus modelos de información.
- El Capítulo 3 introduce los mecanismos de interoperabilidad que se han propuesto, teniendo en cuenta tanto los que se han definido por organismos de estandarización como otros trabajos realizados sobre esta cuestión por diversos grupos de investigación.
- El Capítulo 4 estudia el estado del arte de las ontologías dentro del contexto de los modelos de información de gestión, definiendo su significado y enumerando los lenguajes existentes y los mecanismos propuestos para la integración de ontologías.
- El Capítulo 5 detalla la primera propuesta de la tesis, realizando un análisis de la expresividad semántica de los lenguajes de definición de información de gestión, aplicando los lenguajes de definición de ontologías a los modelos de gestión, y proponiendo qué mecanismos son válidos para integrar dichos modelos desde un punto de vista semántico.

- El Capítulo 6 completa la primera propuesta abordando la forma en que se debería implantar en un sistema de gestión, aprovechando al mismo tiempo la capacidad de definición de reglas que aportan las ontologías.
- El Capítulo 7 describe las conclusiones, aportaciones realizadas y líneas de investigación abiertas tras la realización de la presente tesis doctoral.
- El Apéndice A formaliza el metamodelo de CIM, especificando las reglas de restricción que debe cumplir.
- El Apéndice B lista el código que define una ontología de correspondencia.
- El Apéndice C propone un caso de estudio, en el que se aplica la propuesta del Capítulo 5.

Capítulo 2 Gestión de Red Integrada

2.1 Introducción

Una vez se ha presentado el ámbito y objetivos de la tesis doctoral, este capítulo presenta una panorámica del estado del arte de la Gestión de Red Integrada desde el punto de vista de la problemática expuesta, explicando los distintos modelos existentes. Así, se verá que existen múltiples aproximaciones, que poseen:

- Distintos mecanismos de intercambio de información, aunque parcialmente similares.
- Distintos lenguajes de definición de información, con distinta complejidad y capacidad semántica.
- Distinta información especificada utilizando dichos lenguajes.

Todas estas diferencias suponen problemas de compatibilidad, por lo que no es posible realizar una gestión realmente integrada si para acceder a los recursos gestionados es necesario manejar varios de estos modelos.

A continuación se define el concepto de gestión de red integrada, pasando seguidamente a explicar en detalle los modelos de gestión integrada existentes en TMN y OSI, Internet, equipos de sobremesa, plataformas de procesamiento distribuido y la gestión basada en Web. Para cada uno de ellos se estudian los mecanismos de intercambio, los lenguajes que existen y la información que se ha definido con dichos lenguajes.

2.2 Definición de Gestión de Red Integrada

Para definir qué es la Gestión de Red Integrada, primeramente es necesario explicar qué es la Gestión de Red. Para ello se presentan varias definiciones 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 y garantizar la disponibilidad y un 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 que 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 y las aplicaciones y servicios que proporcionan dichos sistemas. Este mismo autor propone distintas dimensiones de la gestión, mostradas en la Figura 2.1.

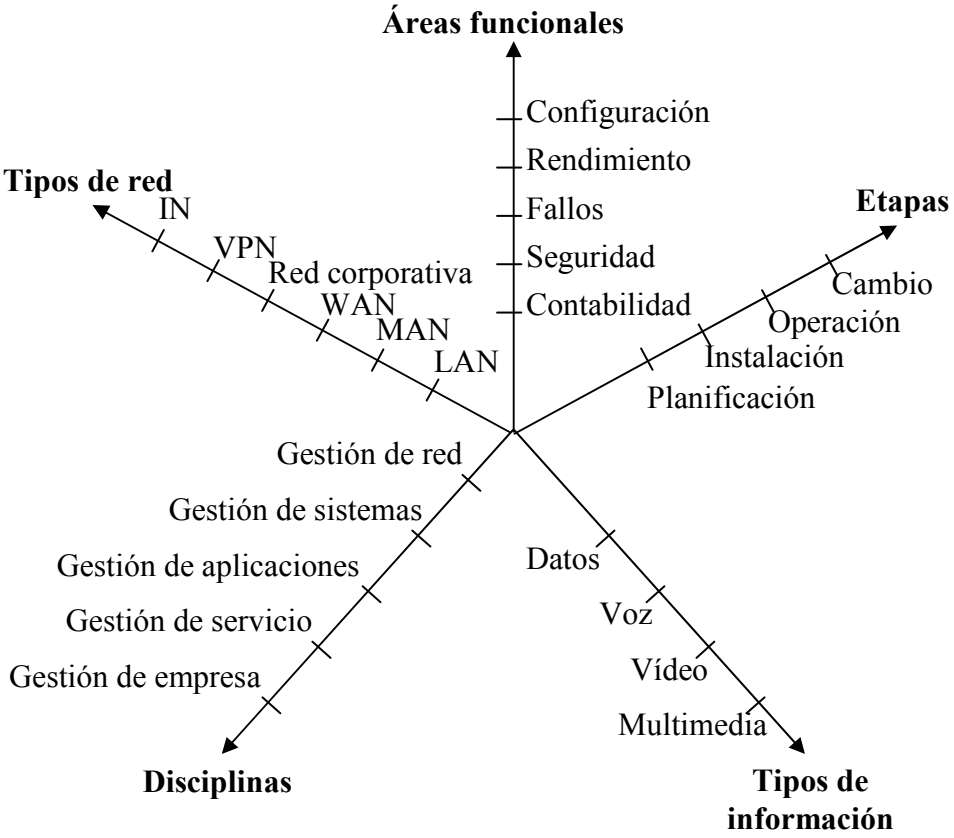

Figura 2.1. Dimensiones de la gestión según [Hegering99].

- 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].

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.

Una vez definido lo que es la gestión de red, conviene introducir el paradigma gestor-agente, un concepto fundamental que es compartido por la mayoría de los sistemas de gestión que se pueden encontrar en el mercado. Este paradigma se basa en que los componentes de una red se pueden clasificar en dos grandes grupos (ver Figura 2.2):

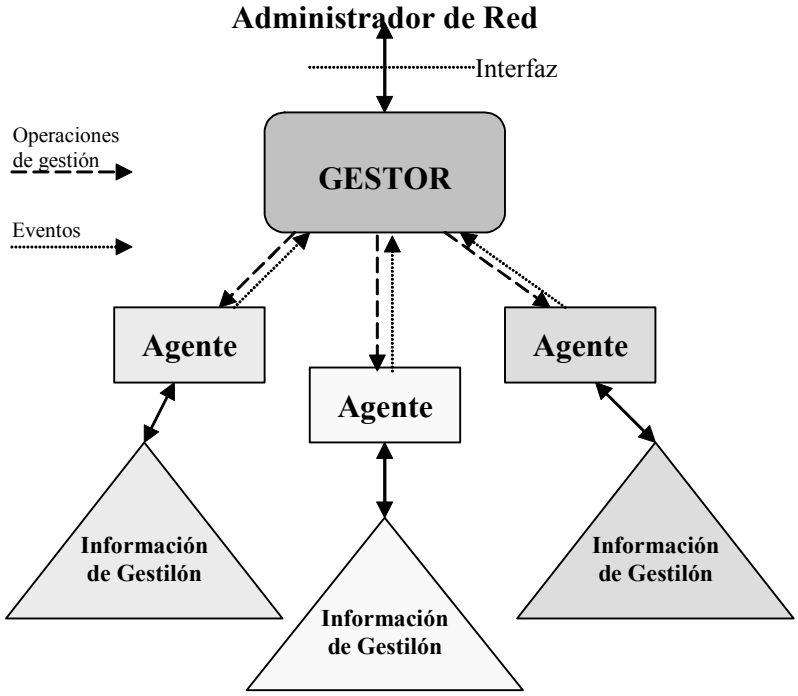

Figura 2.2. El paradigma gestor-agente.

- 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.

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 esos 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.

Con las definiciones anteriores se puede explicar cuándo la Gestión de Red es Integrada. Ésta surge como consecuencia de la necesidad de gestionar equipos de telecomunicación heterogéneos, y consiste en la posibilidad de gestionar diferentes tipos de recursos de red (recursos de transmisión y conmutación) procedentes de diversos fabricantes utilizando un mismo conjunto de herramientas de gestión. Para ello, la gestión integrada usa conceptos estandarizados de bases de datos de gestión globales; permite una aproximación integral a distintos aspectos, incluyendo los organizacionales, soporte de sistemas heterogéneos; y ofrece programación abierta e interfaces de usuario [Hegering99].

Por tanto, estos modelos de gestión integrada, permiten, teóricamente, la interconexión de una manera abierta, entre las aplicaciones de gestión de red o gestores y los agentes situados en los recursos gestionados. Para llegar a este tipo de integración hay que tener en cuenta los dos puntos siguientes:

1. Normalizar las comunicaciones para intercambiar información entre los diferentes componentes del sistema de gestión. Esto permitirá definir el tipo de operaciones que los gestores pueden hacer en los agentes. En general serán solicitudes de información (supervisar), o solicitudes de cambio de información (controlar).
2. Normalizar la información. El objetivo es conseguir una definición sintácticamente uniforme de todos los elementos gestionados, con independencia del fabricante. Para ello, este objetivo se suele dividir en dos subobjetivos.
    a. Normalizar el lenguaje que se emplea para definir la información. Se trata de usar un mismo modelo para definir toda la información, de forma que su sintaxis sea fácilmente interpretada independientemente de lo que defina.
    b. Definiciones de la información gestionada. A partir de la sintaxis anterior se pueden definir Bases de Información de Gestión o MIBs (*Management Information Bases*) que contienen la especificación de la información gestionada. Así, se puede reutilizar para todos los recursos que sean de un mismo tipo, independientemente de quién los haya fabricado.

Por motivos históricos, han sido varios los modelos de gestión de red integrada definidos por los distintos organismos de estandarización. Las siguientes secciones presentan los modelos de gestión en Internet, OSI, equipos de sobremesa, entornos distribuidos y gestión basada en Web. Cada uno de ellos propone su propio mecanismo de intercambio, su lenguaje de definición y su conjunto de definiciones de información. Por esta razón, cada uno es incompatible con el resto, y su heterogeneidad impide alcanzar la meta de la gestión integrada, que es conseguir gestionar los distintos recursos desde una misma herramienta de gestión. La presente tesis doctoral precisamente se centra en cómo integrar la información ante la existencia de múltiples modelos de gestión integrada.

2.3 Gestión de red en OSI e ITU

El objetivo pretendido por la ITU[1] (*International Telecommunication Union*, Unión Internacional de las Telecomunicaciones) con recomendación de la arquitectura TMN (*Telecommunication Management Network*, Red de Gestión de las Telecomunicaciones) [ITUT00a] es proporcionar 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 normalizados.

TMN define una arquitectura física (estructura y entidades de la red), un modelo organizativo (niveles de gestión, mostrados en la Figura 2.3), un modelo funcional (servicios, componentes y funciones de gestión) y un modelo de información (definición de recursos gestionados).

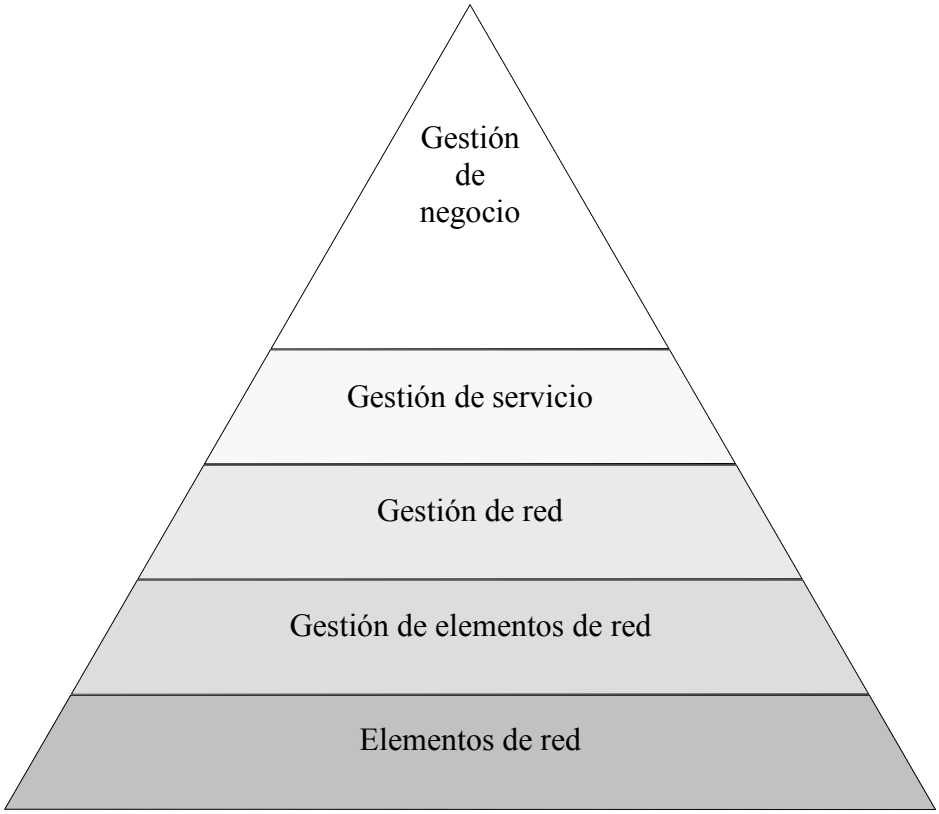

Figura 2.3. Niveles de gestión según TMN según [ITUT00b].

TMN reutiliza muchos conceptos del modelo de gestión propuesto por ISO[2] (*International Organization for Standardization*, Organización Internacional para la Estandarización) en OSI-SM (*Open Systems Interconnection – Systems Management*, Interconexión de Sistemas Abiertos – Gestión de Sistemas):

[1] http://www.itu.int/

[2] http://www.iso.ch/

- CMIS (*Common Management Information Service*, Servicio Común de Información de Gestión) y CMIP (*Common Management Information Protocol*, Protocolo Común de Información de Gestión) componen el mecanismo de intercambio que se emplea en las interfaces Q_3 [ITUT97a].
- GDMO (*Guidelines for the Definition of Managed Objects*, Guías para la Definición de Objetos Gestionados) es el lenguaje de información con el que se ha especificado el Modelo de Información Genérico contenido en la recomendación M.3100.

2.3.1 CMIS/CMIP

CMIS [ITUT97e] y CMIP [ITUT97f] conforman el mecanismo definido por OSI para el intercambio de información de gestión. Para CMIS se definen dos grandes tipos de servicio:

- Servicios de operación. Son servicios usados por el gestor para invocar operaciones de gestión a los agentes y permitir a éstos devolver los resultados de dichas operaciones a los gestores. Dichos servicios son M-GET (pedir información de los objetos gestionados), M-SET (modificar o añadir información en los objetos), M-ACTION (llevar a cabo una acción por parte de un objeto), M-CREATE (crear un nuevo objeto gestionado), M-DELETE (eliminar un objeto gestionado) y M-CANCEL-GET (cancelar una petición anterior). Asimismo, se han definido una serie de métodos para poder invocar dichos servicios sobre un conjunto de objetos distintos: alcance (especifica el conjunto de objetos sobre los que aplicar la operación), filtrado (restringe la operación según el valor de una lista de atributos) y sincronización (especifica que la operación sea de tipo atómico o bien *best effort*).
- Servicios de notificación. Se usa 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 implementa un mecanismo de filtrado de las notificaciones que recibe y posee un atributo que contiene la dirección del agente destino. También 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. Dado que la gestión OSI distribuye la carga de la gestión entre gestor y agentes permitiendo a los últimos enviar notificaciones al primero, este servicio es uno de los que más caracteriza a este modelo.

La arquitectura de CMIP se representa en la Figura 2.4. Este protocolo es orientado a conexión, aportando una mayor fiabilidad pero introduciendo a la vez una sobrecarga en

las comunicaciones, aunque esto no es un problema en las redes TMN, que adoptan un modelo *sidestream* en el que se despliega una red de gestión para acceder a los elementos gestionados. La codificación de este protocolo es la propia del nivel de presentación de OSI, es decir, ASN.1.

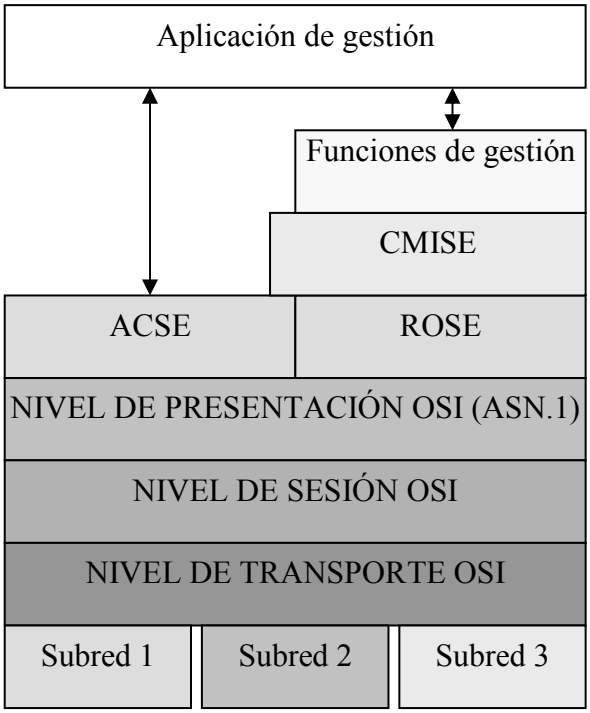

Figura 2.4. Arquitectura de protocolos de CMIS/CMIP.

2.3.2 GDMO

GDMO [ITUT92a] define un conjunto de plantillas para especificar la información de gestión, como parte de la norma de ISO referida a la estructura de la información de gestión. GDMO usa el paradigma de orientación a objetos, en el que existen clases de objetos gestionados y ejemplares. Las clases se pueden especializar siguiendo un árbol de herencia, en el que una clase puede poseer múltiples clases madres de las que hereda sus propiedades (por ejemplo, una tarjeta Ethernet es una clase hija de una tarjeta de red genérica, y posee las características generales de cualquier tarjeta de red). Al mismo tiempo, también existe un árbol de agregación que representa las relaciones de composición entre clases, en el que un ejemplar de una clase puede estar compuesto de otro ejemplar de otra clase (por ejemplo, un ordenador se compone de un procesador, un módulo de memoria, un disco y una tarjeta de red). Este árbol de agregación se aprovecha para nombrar unívocamente cada uno de los ejemplares gestionados, junto con un atributo que permite distinguir los ejemplares de las clases en que está contenido (por ejemplo, la tarjeta de red número X que se encuentra en el ordenador número Y). Para definir otro tipo de relaciones entre las clases se puede utilizar GRM (*General Relationship Model*, Modelo General de Relaciones) [ITUT95b]. Por otro lado también existe un árbol de registro, que identifica con un OID (*Object Identifier*, Identificador de Objeto) la definición de cada

construcción (clase, atributos, notificaciones…). GDMO también añade otras características para permitir una mejor reutilización de la información de gestión definida, como por ejemplo la definición de paquetes. Este conjunto de construcciones hace de GDMO un lenguaje complejo de manejar.

La Figura 2.5 muestra un diagrama de clases de UML (*Unified Modeling Language*, Lenguaje de Modelado Unificado) de las plantillas definidas en GDMO. Este diagrama es meramente ilustrativo, pues se ha creado a partir de la definición de las plantillas sin definir un conjunto de reglas para ello. También se han dibujado diagramas semejantes para el resto de lenguajes de definición de información que se explican en este capítulo.

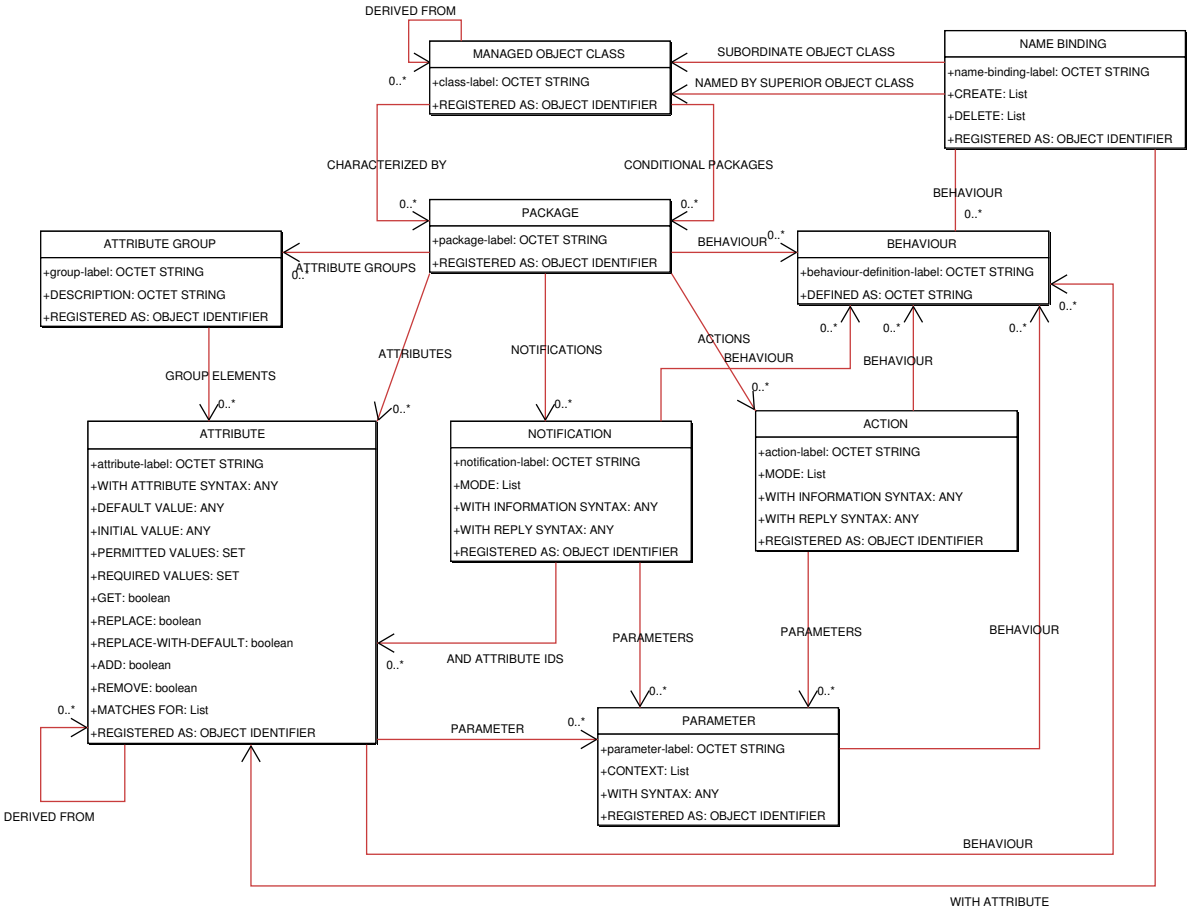

Figura 2.5. Representación de las plantillas de GDMO en UML.

Las plantillas definidas en GDMO incluyen:

- Clase de objeto gestionado (*MANAGED OBJECT CLASS*). Todos los paquetes incluyen una etiqueta con un nombre simbólico y un identificador de objeto (OID, *Object Identifier*) asociado. En este caso, además se indica la clase o clases de las que hereda propiedades (es posible la herencia múltiple), y los paquetes obligatorios y condicionales que posee.

- Paquete (*PACKAGE*). En cada paquete se define su etiqueta y OID, así como su comportamiento. También los atributos, grupo de atributos, acciones y notificaciones de que está compuesto.
- Atributo (*ATTRIBUTE*). Para cada atributo se especifica su etiqueta y OID asociado, así como su comportamiento, las operaciones de filtrado posibles, comportamiento y parámetros de error. Al igual que las clases, los atributos se pueden derivar de otros ya existentes. Su sintaxis es cualquier tipo ASN.1 sin ninguna restricción, con toda la potencialidad que esto supone. Otras propiedades o facetas del atributo son el valor por defecto, el valor inicial, los valores permitidos, los valores requeridos y el acceso.
- Grupo de atributos (*ATTRIBUTE GROUP*). Permite identificar un conjunto de atributos sobre los que se pueden realizar operaciones de manera conjunta. Para ello, además de su etiqueta y OID, incluye una lista de atributos y una descripción.
- Acción (*ACTION*). Al nombre, comportamiento y OID se añaden en este caso el modo de confirmación de la operación, así como los parámetros necesarios.
- Notificación (*NOTIFICATION*). Tiene una sintaxis parecida a las acciones, pues se definen con su etiqueta, comportamiento, OID, modo de confirmación y parámetros. A esto hay que añadir los atributos que se notifican.
- Parámetro (*PARAMETER*). Especifican su nombre, comportamiento, OID, contexto (si es un parámetro de consulta o de respuesta) y sintaxis.
- Ligadura de Nombres (*NAME BINDING*). Permite especificar la relación de contención entre las diferentes clases, indicando la clase subordinada, la clase superior y su atributo asociado, así como la etiqueta, OID y comportamiento. El modo de creación y borrado de las clases también se incluye en esta plantilla.
- Comportamiento (*BEHAVIOUR*). Para especificar el comportamiento únicamente se incluye un nombre y una cadena de caracteres, por lo que esta definición se reduce a una mera descripción en lenguaje natural.

2.3.3 Modelos de información en OSI e ITU

Para el caso de OSI e ITU dos son los modelos de información más importantes:

- La recomendación X.721 [ITUT92b] define en GDMO un conjunto de clases con sus atributos, acciones y notificaciones para describir objetos vinculados al proceso de gestión como parte de la definición de OSI-SM. Entre éstas se encuentran la clase `top`, de la que heredan el resto de clases; la clase `system`, que proporciona información del sistema gestionado; la clase `discriminator`, que permite filtrar los eventos según su importancia y la clase `log`, que actúa de registro de eventos, según se contó al hablar de CMIS. A su vez, esta última clase puede contener

entradas al registro, con clases `logRecord` que suelen estar especializadas según el tipo de evento (alarma, cambio de valor de un atributo, creación y borrado de un objeto, cambio de estado…).

- La recomendación M.3100 [ITUT95a] describe en GDMO un "Modelo de Información de Red Genérico" con clases y propiedades de objetos gestionados que son útiles para describir la información que se intercambia entre todos los interfaces de TMN descritos en [ITUT00b]. Las clases se pueden utilizar con distintos grados de generalidad, pudiendo ser comunes a las redes de telecomunicación gestionadas, o de un tipo genérico que se puede usar para gestionar a un nivel independiente de la tecnología, o superclases de objetos con tecnología específica, u objetos de soporte que son necesarios para la gestión de la red de telecomunicación. A través de las clases definidas se proporcionan mecanismos homogéneos para realizar gestión de fallos, configuración, rendimiento, seguridad y contabilidad.

En cualquier caso, estos modelos, con un lenguaje de definición que emplea un paradigma de modelado basado en la herencia de propiedades y la composición de clases, son claramente independientes de la información definida en el otro gran modelo de gestión integrada, utilizado en Internet.

2.4 Gestión de red en Internet

El modelo de gestión de red integrada definido por el IETF[3] (*Internet Engineering Task Force*, Grupo de Trabajo de Ingeniería para Internet) para Internet es el conocido por el nombre de su protocolo, SNMP (*Simple Network Management Protocol*, Protocolo Simple de Gestión de Red). Este modelo también define el lenguaje de definición de información SMI (*Structure of Management Information*, Estructura de la Información de Gestión). Además, existe un gran conjunto de MIBs estándar que se han ido definiendo a lo largo de los años de existencia de SNMP.

Antes de exponer la arquitectura de gestión de red utilizada en Internet, es conveniente citar el axioma fundamental de diseño en el que se basa: "El impacto de añadir gestión de red a los nodos debe ser mínimo, reflejando su mínimo común denominador". Este axioma está basado en las mismas causas que han llevado al protocolo IP a tener tan amplia aceptación: los mínimos requisitos que impone a su capa de interfaz inferior. Las consecuencias de este axioma suponen la imposibilidad de adoptar bajo este esquema el modelo de gestión de red OSI, que exige gran capacidad a los agentes para realizar operaciones tales como ámbito y filtrado sobre los objetos gestionados, aparte de tener que implementar toda la torre de protocolos de OSI en los agentes. Frente a ello, se ha

[3] http://www.ietf.org/

adoptado un modelo cuya característica principal es su simplicidad, que pueda implantarse en todo tipo de recursos gestionados.

2.4.1 SNMP

El protocolo SNMP es el heredero de SGMP (*Simple Gateway Monitoring Protocol*, Protocolo Simple de Supervisión de Pasarelas), definido a finales de los ochenta como propuesta para realizar tareas de gestión en Internet. A lo largo del tiempo se han propuesto varias versiones, que trataban de mejorar la funcionalidad y el modelo administrativo, pero no siempre han tenido la implantación deseada, coexistiendo en la actualidad la primera versión, la 2C, que incluía mejoras en la definición de la información y el rendimiento del protocolo pero que seguía basando su modelo administrativo en comunidades, y la 3, que añade cuestiones relativas a la seguridad y la arquitectura. Todas las versiones, no obstante, han mantenido la filosofía de simplicidad.

Las operaciones que permite efectuar están restringidas por simplicidad a la obtención, modificación y notificación de información, no teniendo la potencia que proporciona CMIS/CMIP. Un gestor SNMP puede enviar a un agente tres tipos de peticiones: GetRequest, GetNextRequest y SetRequest. A partir de la versión 2C se incluye además GetBulkRequest, que mejora el rendimiento al solicitar información. Todas son confirmadas por el agente mediante un mensaje del tipo Response. Además, los agentes pueden enviar en situaciones críticas un mensaje no solicitado o Trap que no será confirmado. También pueden enviarse mensajes Inform entre gestores, similares a las Traps pero que sí son confirmados. Hay que destacar que las notificaciones SNMP no tienen la misma funcionalidad que los eventos de la gestión OSI. En SNMP su propósito es el de informar de situaciones muy específicas y no están pensados para llevar a cabo una gestión orientada a eventos.

Utilizando inteligentemente el modelo de información se pueden añadir otras operaciones. Por ejemplo, a través de la definición de una columna de tipo `RowStatus` se pueden crear y borrar filas en las tablas.

En lo que se refiere a la arquitectura de protocolos, ésta se ilustra en la Figura 2.6. El protocolo SNMP se basó inicialmente en UDP, reservándose los puertos 161 y 162 para acceder a los agentes y gestores respectivamente. Debido a su éxito se ha especificado su transporte en protocolos de redes OSI, Apple y Novell. Para la codificación con sintaxis neutra se utiliza un subconjunto de ASN.1, y a partir de la versión 3, se emplea MD.5 (*Message Digest 5*, Resumen de Mensaje 5) para autenticar y DES (*Data Encryption Standard*, Estándar de Cifrado de Datos) en su modo CBC (*Cipher Block Chain*, Encadenado de Bloques Cifrados) para cifrar los mensajes.

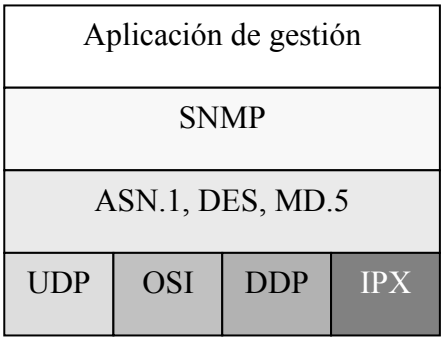

Figura 2.6. Arquitectura de protocolos de SNMP.

2.4.2 SMI

SMI se especificó para definir la información de gestión que se debía mantener acerca de los recursos en Internet. SMIv2 [McCloghrie99a,McCloghrie99b,McCloghrie99c] es una versión mejorada de SMI introducida en SNMPv2 aunque sigue manteniendo la filosofía de simplicidad de SNMP y es actualmente el lenguaje estándar de definición de información de gestión del IETF.

Una de las razones de esta simplicidad es que no usa un modelo como el de la programación orientada a objetos. Las definiciones contienen objetos, especificados con macros ASN.1, pero sus ejemplares sólo pueden ser grupos de variables escalares y celdas de tablas. A pesar de no ser un modelo orientado a objetos común, las tablas se pueden ver como clases, donde los atributos son las columnas de la tabla, y cada fila contiene a un ejemplar de la clase. Al igual que en GDMO, cada objeto lleva asociado un OID, que en este caso servirá directamente para solicitar la información a los recursos gestionados. Los OIDs se definen con estructura de árbol en el que se registra la información, pero no hay árboles de herencia ni agregación. Tampoco hay ninguna propiedad para indicar el comportamiento, que tendrá que incluirse en la descripción de cada macro si se considera necesario.

La Figura 2.7 presenta una representación en UML de las macros definidas en [McCloghrie99a]. Estas macros son las siguientes:

- Módulo (*MODULE-IDENTITY*). Define un módulo de información de gestión o MIB. Dicho módulo poseerá propiedades tales como su nombre y OID, las fechas de revisión y última actualización, la organización y datos de contacto de quién lo definió y una descripción textual acerca de la información incluida en dicho módulo.
- Tipo de objeto (*OBJECT-TYPE*). Forman lo que serían atributos de una clase, e incluyen nombre, OID, acceso (ninguno, notificación, lectura, escritura, creación), estado (actual, obsoleto, caducado) y descripción textual. Los tipos de datos que definen la sintaxis del objeto son los tipos ASN.1 `INTEGER` (y sus derivados `Counter32`, `Gauge32`, `Unsigned32`, `Counter64`, `TimeTicks`), `OBJECT IDENTIFIER`,

`OCTET STRING` (y sus derivados, como `IpAddress` u `Opaque`) y `BITS`. Se puede incluir en algunos casos la longitud o los valores máximo y mínimo, así como cierta semántica (si el número entero es un contador o un medidor, o si es una enumeración de valores). Además, los tipos de datos se pueden redefinir con la macro TEXTUAL-CONVENTION, no incluida en el diagrama. Opcionalmente, también se pueden incluir unidades de medida, valor por defecto, referencia, e indicador de índice si el objeto define la entrada de una tabla.

- Identidad de objeto (*OBJECT-IDENTITY*). Representan OIDs con un significado específico. Para ello, se les dota de un nombre, un estado y una descripción, y opcionalmente, una referencia.
- Notificación (*NOTIFICATION-TYPE*). Las notificaciones se describen con un nombre, un OID, un estado, una descripción, y opcionalmente una referencia. También pueden tener una lista de los tipos de objetos que se enviarán en la notificación.
- Otras macros no incluidas en el diagrama son las declaraciones de conformidad que indican las opciones de implementación para distintos grupos de objetos y notificaciones.

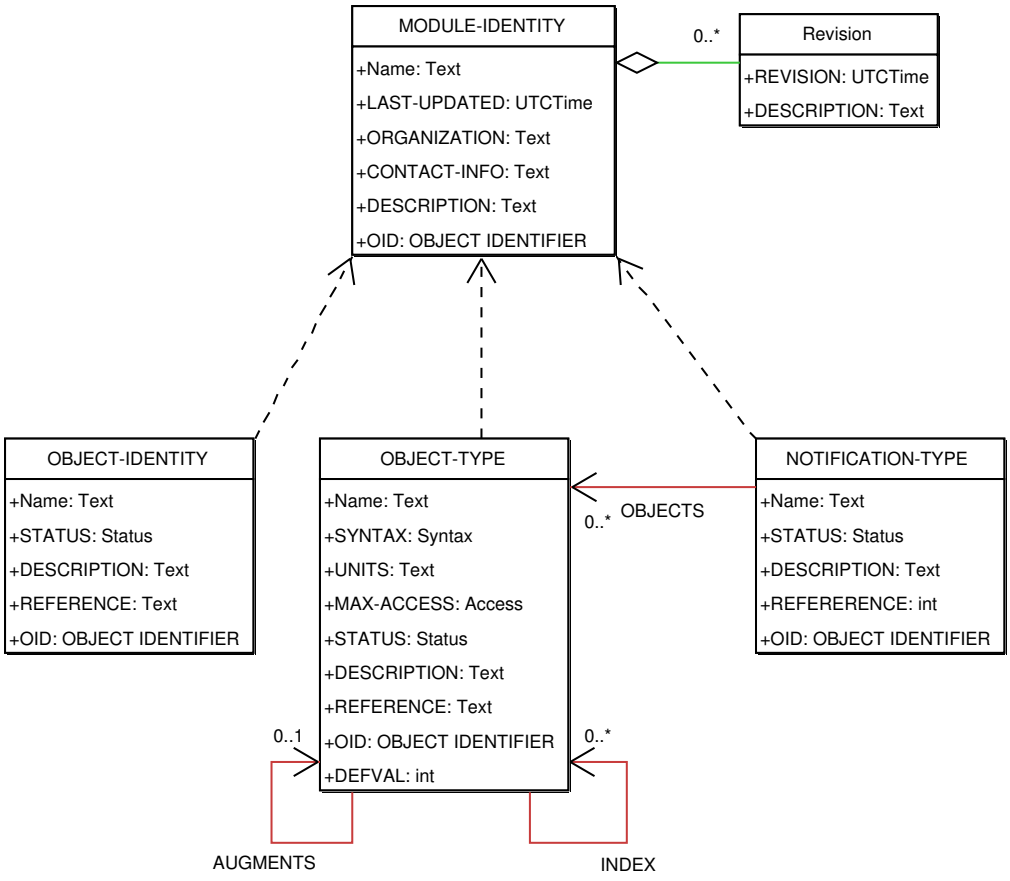

Figura 2.7. Representación de las macros de SMIv2 en UML.

SMIng [Schönwälder99, Elliot01] es la siguiente generación (ng, *next generation*) de SMI propuesta por el Grupo de Investigación de Gestión de Red del IRTF (*Internet Research Task Force*, Grupo de Trabajo de Investigación de Internet) para definir información de gestión e integrar SMI y SPPI (*Structure of Policy Provisioning Information*, Estructura de la Información de Provisión de Políticas), utilizando una sintaxis similar a C/C++, evitando el uso de ASN.1. Al no utilizarse esta sintaxis, tampoco se asocian OIDs a los nombres.

Aunque no es todavía un estándar y está todavía en proceso de definición, la expresividad de SMIng es mucho mayor que la de SMI, dado que en este caso sí se trata de un lenguaje orientado a objetos en sentido general, que permite la definición de clases y relaciones de herencia simple. Además de clases con atributos y eventos, también es posible la definición de extensiones, que especifican nuevas estructuras al proporcionar la sintaxis que deben cumplir. La Figura 2.8 representa un diagrama de clases UML de las distintas construcciones de SMIng. Éstas son:

- Módulo (*module*). Un modulo posee un identificador, datos de la organización y contacto del que ha definido el módulo, datos de revisión, una descripción textual y una referencia, al estilo de la macro de SMIv2. Dentro de un módulo se definen clases, identidades, tipos de datos y extensiones.
- Clase (*class*). Para cada clase se define un identificador, una posible clase de la que hereda, un estado de implementación, una descripción textual de la clase y una posible referencia a otras definiciones. También puede existir una cláusula de unicidad para indicar el conjunto de atributos que identifican unívocamente un ejemplar de la clase. Las clases contienen atributos y eventos.
- Atributo (*attribute*). Para cada atributo se especifica un identificador, un acceso (de lectura, escritura y notificación), el valor por defecto, el formato del tipo de datos, las unidades de medida del atributo, el grado de implementación del atributo, una descripción y una referencia. El tipo de datos del atributo puede ser uno de los siguientes, basados en los que existían en SMIv2: OctetString, Pointer, ObjectIdentifier, Integer (32 y 64 bits, con y sin signo (Unsigned)), Float (32, 64 y 128 bits), Enumeration, Bits. Estos tipos de datos también se pueden redefinir y restringir en rangos de valores o longitudes con la construcción `typedef`.
- Evento (*event*). Los eventos de SMIng son muy restringidos, especificándose únicamente un identificador, un estado de implementación, una descripción y una referencia.
- Definición de tipos (*typedef*). Se pueden definir tipos de datos nuevos a partir de los ya existentes. Esta definición incluirá el nombre del nuevo tipo, así como en el que se basa. También incluirán un posible valor por defecto, el formato, las unidades, el estado de implementación, la descripción del nuevo tipo y una referencia.

- Identidad (*identity*). Ante la ausencia de OIDs, permiten definir nombres con identidad propia. Para ello se les proporciona un identificador, un estado, una descripción y una referencia.
- Extensión (*extension*). Define una extensión del lenguaje, proporcionando la sintaxis ABNF (*Augmented Backus-Naur Form*, Forma Backus-Naur Aumentada) que debe cumplir. Además, se especifica un identificador, un estado, una descripción y una referencia.

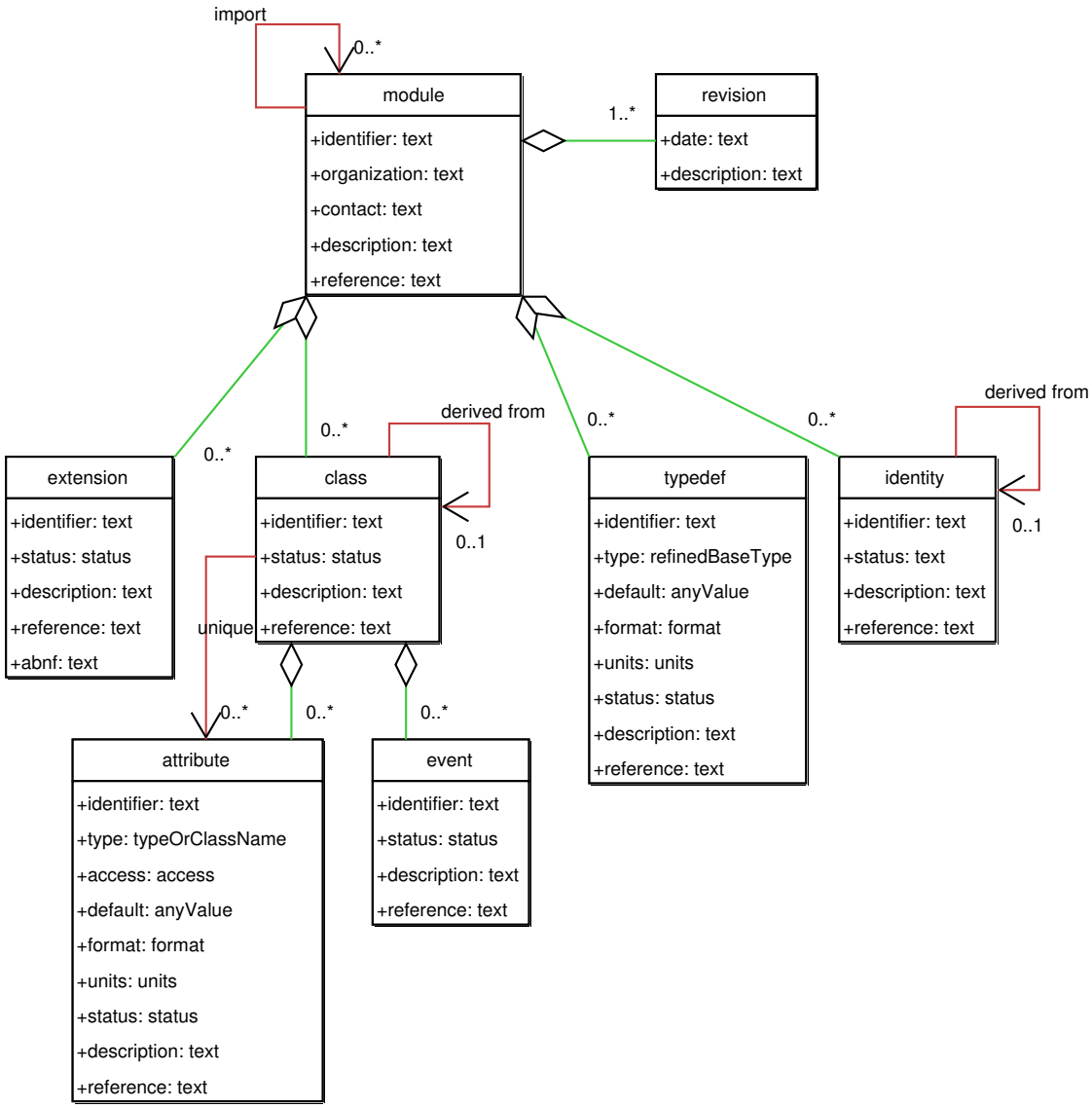

Figura 2.8. Representación de las cláusulas de SMIng en UML.

2.4.3 MIBs

Las MIBs definidas en SMI y propuestas en RFCs (*Request For Comments*, Solicitud de Comentarios) del IETF suman más de doscientas. Además, hay que contar las MIBs

privadas propuestas por cada fabricante. El servidor SimpleWeb[4] contiene todas las MIBs incluidas en RFCs, y punteros a gran parte de las privadas.

Las MIBs de SNMP se podrían clasificar según la información que definen en los siguientes grupos:

- Gestión de protocolos de comunicaciones. Las MIBs que definen información relativa a protocolos de comunicaciones suman el mayor número, siendo la más importante la MIB-II [McCloghrie91] y sus actualizaciones. En general miden datos estadísticos de los protocolos para hacer gestión de rendimiento.
- Gestión de sistemas. Estas MIBs permiten gestionar cuestiones relativas a los sistemas y no a la comunicación. MIBs de este tipo son la HOST-RESOURCES-MIB [Waldbusser00a] y la ENTITY-MIB [McCloghrie99d]. Muchas MIBs privadas también definen este tipo de recursos, ocurriendo que a veces la misma información puede encontrarse en varias MIBs diferentes, especificadas de forma distinta. Por ejemplo, la MIB privada HP-UNIX, definida por Hewlett-Packard[5] para sus sistemas, describe información parecida a la HOST-RESOURCES-MIB.
- Gestión distribuida. Este grupo de MIBs permite delegar a un sistema remoto para que éste realice tareas de gestión. Incluye a RMON [Waldbusser00b] y sus extensiones, o las de DISMAN[6] (*Distributed Management Workgroup*, Grupo de Gestión Distribuida del IETF).
- Gestión administrativa. Son las MIBs que se han definido en los distintos modelos administrativos de SNMP que permiten configurar a las entidades gestionadas. Como ejemplo de este caso están las MIBs de SNMPv3 [Harrington02, Case02, Levi02, Blumenthal02, Wijnen02, Presuhn02].

Como puede verse, el tipo de información definida es bastante distinta a la del caso de TMN. Mientras que anteriormente se modelaban los elementos de red, aquí sobre todo se especifican los protocolos que manejan dichos elementos. Esto se debe a que sus ámbitos de aplicación son claramente distintos.

2.5 Gestión en equipos de sobremesa

A finales de los 80 empiezan a proliferar los ordenadores personales o PCs (*Personal Computers*) conectados en redes de área local. La gestión de dichos ordenadores heterogéneos, a diferencia de SNMP que se utiliza para dispositivos de red, era propietaria

[4] http://www.simpleweb.org/

[5] http://www.hp.com/

[6] http://www.ietf.org/html.charters/disman-charter.html

y dependiente de cada vendedor. El DMTF[7] (entonces *Desktop Management Task Force*, Grupo de Trabajo de Gestión de Equipos de Sobremesa, actual *Distributed Management Task Force*, Grupo de Trabajo de Gestión Distribuida) trata de salvar este problema, realizando una gestión integrada, independiente del sistema operativo y protocolos de red. Para ello define DMI (*Desktop Management Interface*, Interfaz de Gestión de Equipos de Sobremesa), pensado como un complemento a SNMP. DMI utiliza MIF (*Managed Information Format*, Formato de la Información Gestionada) como lenguaje de definición de información, y especifica el conjunto de información útil para la gestión de PCs en la denominada Master MIF.

Posteriormente, y debido a que gran parte de la información se maneja a nivel de la BIOS (*Basic Input Output System*, Sistema Básico de Entrada y Salida), trata también de normalizar este acceso con SMBIOS (*System Management BIOS*, BIOS de Gestión de Sistemas).

2.5.1 DMI y SMBIOS

DMI [DMTF03a] define un marco normalizado para gestionar y rastrear componentes en un ordenador de sobremesa, portátil o servidor, habiendo sido DMI el primer estándar referido específicamente a este fin. Para ello, especifica la arquitectura mostrada en la Figura 2.9. En ella, la capa de servicio de gestión que se encarga de comunicar la interfaz de gestión a la que acceden las aplicaciones gestoras con la interfaz de los componentes a la que se conectan los recursos gestionados. Las operaciones posibles que incluye se refieren a la obtención, modificación, registro y notificación de información, así como la creación y borrado de ejemplares de la MIF. A partir de la segunda versión de DMI se hace uso de RPCs (*Remote Procedure Calls*, Llamadas a Procedimientos Remotos) confirmadas, permitiendo también la gestión remota de dispositivos y un marco para el desarrollo de software independiente de la plataforma de gestión. También se ha definido un mecanismo para interoperar con SNMP, que se describirá en el Capítulo 3.

SMBIOS [DMTF02c] aborda cómo los vendedores de equipos y placas pueden presentar la información de gestión sobre sus productos con un formato estándar extendiendo la interfaz de la BIOS en sistemas con arquitectura de Intel. La especificación está pensada para permitir la entrega de esta información con una instrumentación genérica a aplicaciones de gestión que usen DMI o tengan acceso directo, eliminando la posibilidad de operaciones que produzcan fallos, como el sondeo de sistemas para detectar la presencia de hardware.

[7] http://www.dmtf.org/

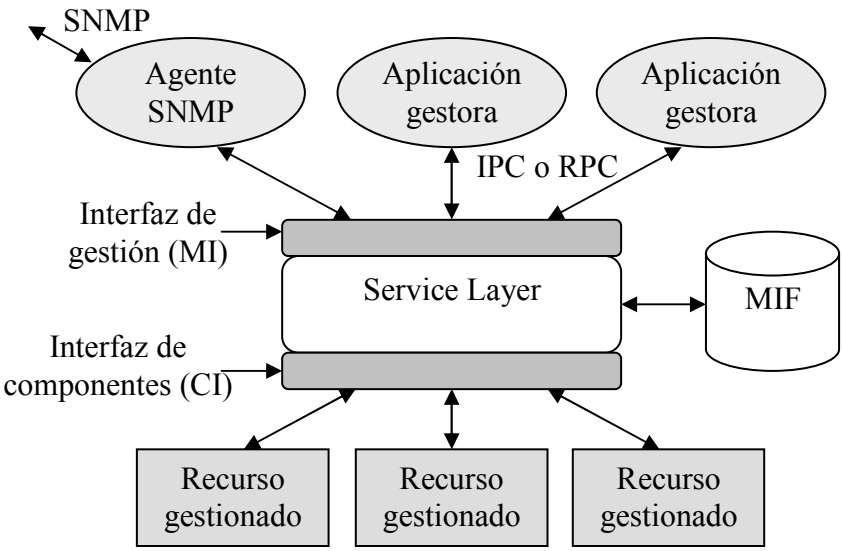

Figura 2.9. Arquitectura de DMI.

2.5.2 MIF

El lenguaje empleado en DMI [DMTF03a] se ha usado para definir información relacionada con ordenadores de escritorio. Su paradigma es en cierta manera similar al de SMI: sólo se pueden definir componentes, que poseen grupos de variables simples (llamadas en este caso atributos) y tablas. Sin embargo, MIF es un lenguaje incluso más simple que SMI puesto que las claves de cada tabla son siempre internas a dicha tabla, y por lo tanto, no se pueden definir asociaciones a partir de dichas claves. Para identificar los atributos en este caso no existe un árbol de identificadores de objeto, sino que la identificación se basa en el par {clase de grupo, identificador de atributo}, donde la clase de grupo es una cadena que identifica unívocamente a un grupo de atributos, y el identificador es un número entero. Sin embargo, cada grupo también puede tener asignado un OID debajo de la rama `enterprises.dmtf` para permitir el acceso desde SNMP.

Las cláusulas especificadas en MIF que se ilustran con el diagrama de clases de la Figura 2.10 son:

- Componente (*component*). Un componente reúne distintos grupos de atributos, caminos, enumeraciones y tablas. Además, posee un nombre, una descripción y una cláusula pragma, que puede indicar el OID SNMP con que se corresponde, dependencias con otros grupos, implementación, y registro de Windows.
- Camino (*path*). Un camino posee un nombre e indica la localización de los archivos que se usan para la gestión del componente en distintos sistemas operativos (DOS, MacOS, OS/2, Unix y distintas versiones de Windows).
- Grupo (*group*). Un grupo de atributos se define con un nombre, una clase que indica el organismo que define la información, el nombre de lo definido y la versión; una descripción; un identificador numérico del grupo dentro del

componente; una clave, que indica qué atributos forman la clave; y una cláusula pragma.

- Atributo (*attribute*). Un atributo incluye un nombre, una descripción, un identificador numérico del atributo dentro del grupo, el tipo de acceso, el tipo de almacenamiento, el valor que posee y una cláusula pragma. El tipo de datos puede ser un número entero (integer (o int), integer64 (o int64), gauge, counter, counter64), string (n) o displaystring(n), octetstring(n) y date. También puede ser una enumeración.
- Enumeración (*enum*). Una enumeración indica la relación entre números enteros y cadenas de caracteres.
- Tabla (*table*). Define ejemplares de tabla dentro de un grupo de atributos. Para ello indica el nombre, identificador y clase, así como los valores de las distintas filas y columnas de la tabla.

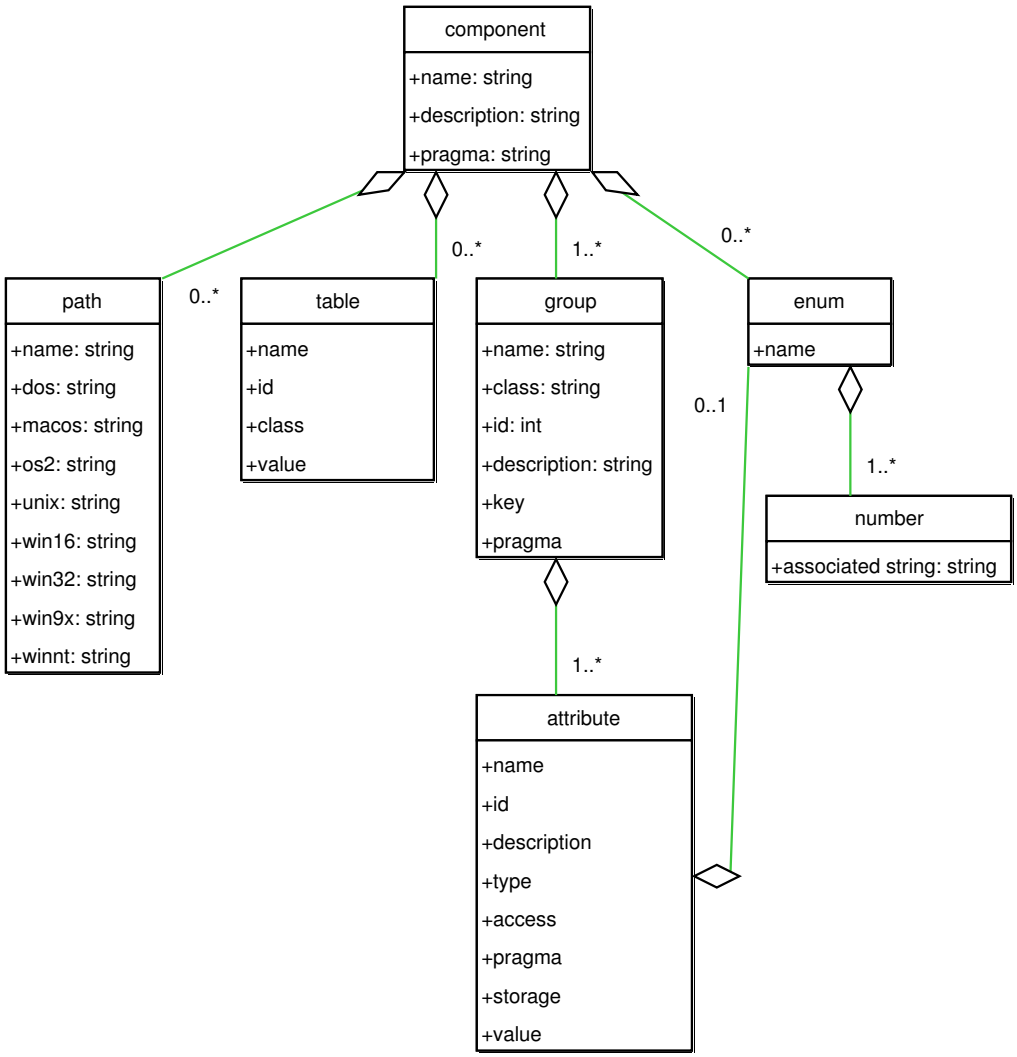

Figura 2.10. Representación de las cláusulas de MIF en UML.

2.5.3 Master MIF

La Master MIF [DMTF02b] contiene todos los grupos de información estandarizados por el DMTF. Dichos grupos se refieren sobre todo a datos de configuración, permitiendo hacer gestión de inventario respecto a los recursos físicos (discos, memoria, tarjetas de red, procesador…) y lógicos (programas instalados) que posee. Una gran parte de esta información se podría corresponder con la HOST-RESOURCES-MIB, si bien se define de manera totalmente independiente. La información que se refiere a los recursos físicos normalmente se obtiene a partir de SMBIOS, ocurriendo que la mayor parte de las BIOS que vienen incorporadas en los PCs actuales implementan esta función.

2.6 Gestión en plataformas de procesamiento distribuido

La tecnología de las plataformas de procesamiento distribuido orientado a objetos posee cualidades que la hacen una buena candidata para ser aplicada a la gestión integrada. El acceso a la información de gestión mantenida por los recursos gestionados se puede realizar utilizando los mecanismos de comunicación propios de la plataforma de procesamiento distribuido [Pavlou97]. De esa manera se dispondría de un protocolo de intercambio de información de gestión con una interfaz de acceso a sus servicios estándar que se puede aplicar para acceder directamente a los parámetros gestionables de recursos tales como aplicaciones distribuidas que proporcionan un servicio [Asensio00, Valera01, Villagrá02].

Así, la arquitectura CORBA (*Common Object Request Broker Architecture*, Arquitectura Común de un Intermediario de Peticiones a Objetos), definida por OMG[8], ha sido propuesta a través de su Grupo de Trabajo de Telecomunicaciones (TTF, *Telecom Task Force*) [OMG01a], conjuntamente con otros organismos de normalización como T1M1 (Comité T1[9] de Telecomunicaciones de ANSI, Subcomité M1 de Operaciones de Interconexión, Administración, Mantenimiento y Provisión) o el ETSI[10] (*European Telecommunications Standards Institute*, Instituto Europeo de Estándares de Telecomunicación), en varias normas de la ITU para su uso en TMN. Éstas se refieren tanto al intercambio de la información como a la definición de la misma utilizando IDL (*Interface Description Language*, Lenguaje de Descripción de Interfaces).

[8] http://www.omg.org/

[9] http://www.t1.org/

[10] http://www.etsi.org/

2.6.1 CORBA

CORBA [OMG02c] es una implementación parcialmente conforme a ODP (*Open Distributed Processing,* Procesamiento Distribuido y Abierto) [ITUT97d] que propone una solución para el modelo de ingeniería y notaciones y mecanismos para construir modelos computacionales. La Figura 2.11 muestra la arquitectura que se define para ello. En ella existe un ORB (*Object Request Broker*, Intermediario de Peticiones a Objetos) que se encarga de poner en comunicación un cliente y un conjunto de objetos. De esta manera, si un cliente desea comunicarse con un objeto concreto, puede acceder al repositorio de interfaces para conocer las operaciones y atributos que implementa. A partir de dichos interfaces podrá realizar una invocación dinámica, o bien hacer uso de cabos o *stubs* que se encarguen de serializar la petición. Por su parte, los objetos recibirán las peticiones a través de esqueletos generados a partir de las interfaces, o bien con esqueletos dinámicos.

CORBA define IIOP (*Internet Inter-ORB Protocol*, Protocolo Inter-ORB de Internet) [OMG02e] como protocolo para las comunicaciones entre objetos distribuidos, pudiéndose acceder y modificar valores de atributos, así como solicitar la ejecución de métodos que devuelven resultados o lanzan excepciones.

Al mismo tiempo CORBA define un conjunto de servicios conocidos como CORBAservices (Servicios CORBA) que incluyen entre otros un servicio de nombres [OMG02g] para la localización de ejemplares y un servicio de notificaciones [OMG02f]. El TTF ha definido servicios adicionales como el de registro [OMG02h]. Para el caso de TMN, la recomendación Q.816 [ITUT01a] utiliza un conjunto de servicios que incluye al de nombres y al de notificaciones y el de registro, así como otros servicios nuevos que permiten operar de manera similar a CMIP. Éstos son, por ejemplo, el localizador de fábricas (para crear ejemplares), de terminación (para borrar ejemplares), operación de múltiples objetos (que permite realizar operaciones dentro de un ámbito o con filtrado) o localizador de canal (para encontrar canales de eventos).

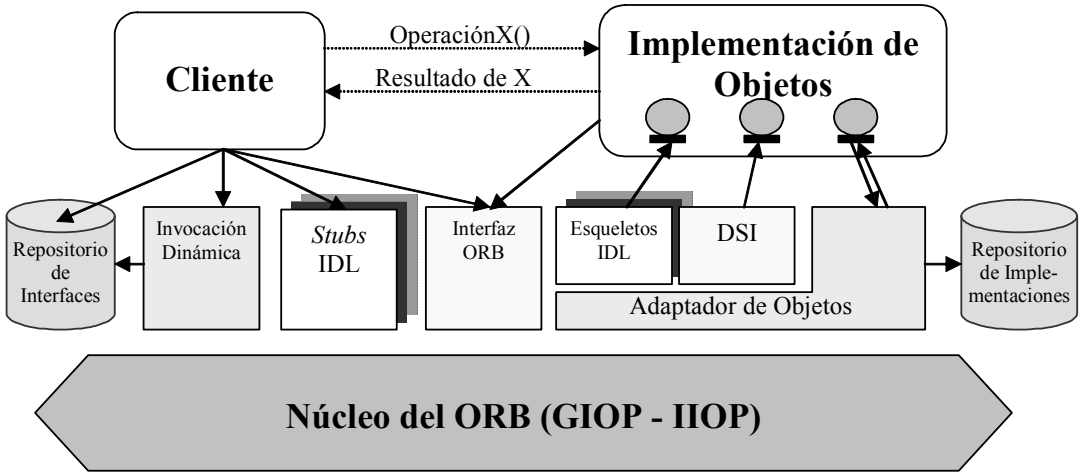

Figura 2.11. Arquitectura CORBA.

2.6.2 IDL

IDL [OMG02d] es el lenguaje utilizado en CORBA para definir interfaces. Es un modelo orientado a objetos en el que las clases son interfaces con atributos, operaciones y relaciones de herencia. A su vez, se pueden definir asociaciones mediante el uso de referencias a otras interfaces. Las últimas versiones también incluyen características para definir componentes.

La recomendación M.3020 [ITUT00c] define una metodología de definición de interfaces denominada UTRAD (*Unified TMN Requirements, Analysis and Design*, Requisitos, Análisis y Diseño de TMN Unificados) que indica que a partir de su diseño en UML (*Unified Modeling Language*, Lenguaje de Modelado Unificado) [OMG01b] se deben poder generar definiciones tanto en GDMO como en IDL.

IDL no es un lenguaje concebido para definir información de gestión sino para describir las interfaces de acceso que permiten la comunicación entre aplicaciones distribuidas, que pueden estar implementadas en cualquier lenguaje de programación con el que tenga ligaduras. Esto hace que aunque se puedan definir los tipos de datos de los atributos o si éstos son de lectura, no incluya la adición de descripciones ni los identificadores de objetos. Sí existen, sin embargo, el llamado IOR (*Interoperable Object Reference*, Referencia Interoperable de Objeto) que se puede obtener a través del servicio de nombres y permite acceder a las interfaces.

Las construcciones de IDL, sin tener en cuenta aquéllas referidas a componentes, se ilustran en la Figura 2.12. Esta representación está basada en la definición de la gramática de IDL, y no en el perfil UML de CORBA, especificado en [OMG02a]. Éstas incluyen:

- Módulo (*module*). Los módulos poseen un identificador y poseen declaraciones. Dichas declaraciones pueden ser de definición de tipos, constantes, interfaces, tipos de valores o excepciones.
- Interfaz (*interface*). Las interfaces se definen con un identificador, las interfaces de las que hereda y los modificadores de abstracción (la interfaz no puede tener ejemplares directamente) y local (la interfaz no es remota). Contienen exportaciones, que pueden ser de definición de tipos, constantes, excepciones, atributos y operaciones.
- Atributo (*attribute*). Los atributos se describen con un identificador y el acceso (lectura y escritura). También pueden lanzar excepciones cuando se obtenga o modifique su valor. Además, poseen un tipo de datos, que puede ser de distintas clases. Los tipos básicos pueden ser números de coma flotante (float, double, long double), enteros con y sin signo (octet, short, long, long long), booleanos, caracteres (char, wchar), cadenas de caracteres y cualquiera de éstos (tipo any). Otros tipos incluyen las estructuras (struct y union), enumeraciones, secuencias y *arrays*.

- Operación (*operation*). Las operaciones se especifican con un identificador, el tipo de datos que devuelve, el modo de respuesta (normal o de sólo ida) y un contexto, que indica qué elementos del contexto del cliente pueden afectar al rendimiento de la solicitud. Puede contener un conjunto de parámetros, y emitir una excepción si se produce algún problema durante su ejecución.
- Parámetro (*parameter*). Los parámetros de una operación poseen un identificador, un tipo de datos y los modificadores que indican si es un parámetro de entrada o de salida.
- Excepción (*exception*). Indican que se ha producido una condición excepcional durante la realización de una solicitud. Pueden contener un conjunto de miembros, con identificador y tipo de datos.
- Constante (*const*). Definen valores constantes, esto es, que no varían, mediante un identificador, un tipo de datos y una expresión que especifica el valor.
- Definición de tipos (*typedef*). Los tipos se pueden redefinir, añadiendo un identificador al tipo definido.
- Tipos de valores (*valuetype*). Para permitir el paso de objetos por valor y no por referencia, se pueden definir tipos de valores. Estos valores soportan interfaces, pero también poseen valores de estado, que son atributos con modificadores públicos, y factorías, que representan métodos constructores del valor.

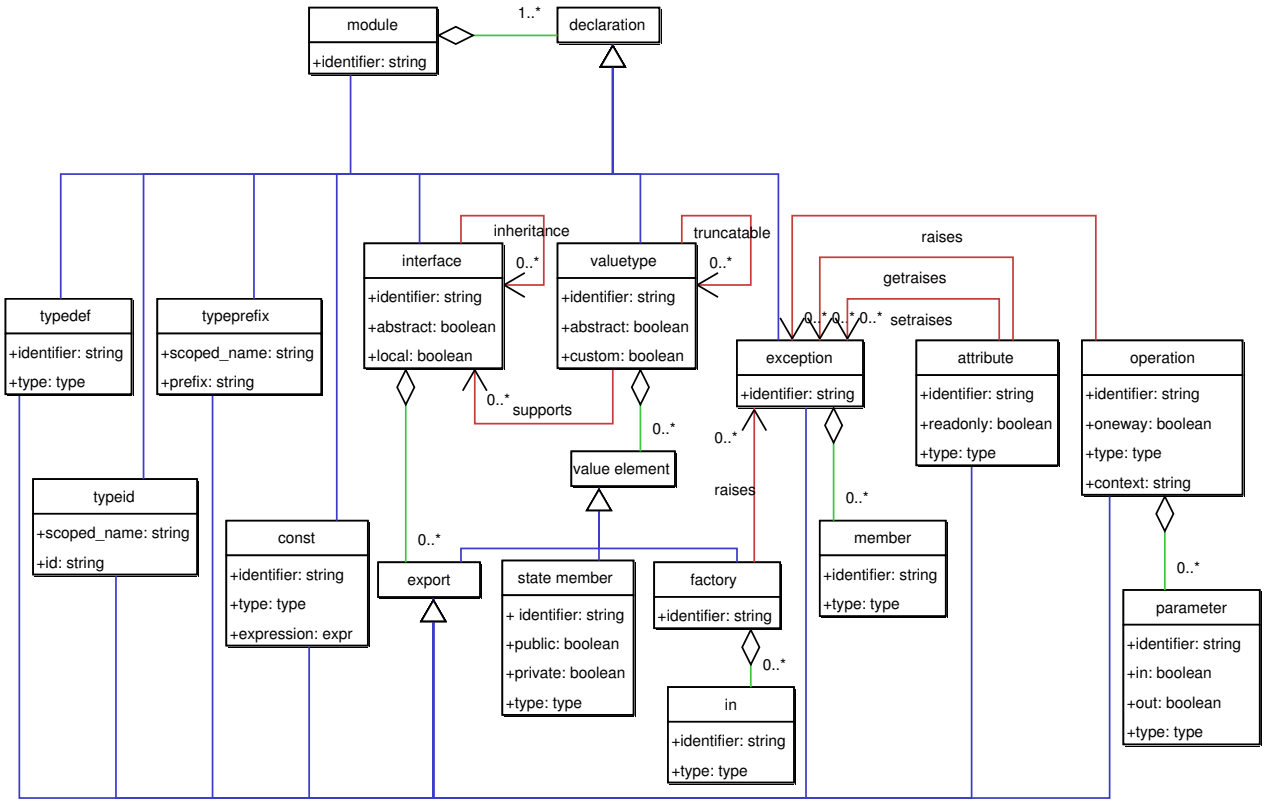

Figura 2.12. Representación parcial de la gramática de IDL en UML.

2.6.3 Modelos de información en CORBA

En el caso de CORBA, y sin tener en cuenta los módulos que definen los servicios anteriormente mencionados, se han definido en IDL dos módulos para describir la información de gestión relativa a TMN:

- La recomendación X.780 [ITUT01b] especifica un conjunto de guías para definir interfaces CORBA de objetos que representen recursos gestionados en TMN. Cubre guías de modelado de información, reglas para traducir modelos de GDMO, y convenciones de estilo en el código IDL. También proporciona un módulo IDL que define tipos de datos, superclases y notificaciones para usar en la especificación de modelos como el que se comenta a continuación.
- La recomendación M.3120 [ITUT01c] define un modelo de información de red genérico para ser usado para la gestión basada en CORBA de redes de telecomunicación. Define en IDL un conjunto de interfaces y constantes, siendo éste un modelo similar al definido en las recomendaciones X.721 y M.3100. Las interfaces genéricas se pueden extender para gestionar tecnologías de red específicas. La traducción desde la recomendación M.3100 se ha hecho a mano teniendo en cuenta las guías de la recomendación X.780.

2.7 Gestión basada en Web

El modelo de gestión basado en Web también conocido por WBEM (*Web Based Enterprise Management*, Gestión de Empresa Basada en Web) se ha definido a partir de una iniciativa que pretende la interoperabilidad de los modelos anteriores, aunque también se puede usar directamente como un modelo de gestión integrada. Para ello el DMTF (presentado anteriormente al hablar de DMI) ha propuesto un protocolo, una sintaxis y un modelo de información, que son neutros e independientes de los ya vistos. Al mismo tiempo, trata de reutilizar las tecnologías que con tan buenos resultados han sido aplicadas a la Web: HTTP (*Hyper-Text Transfer Protocol*, Protocolo de Transferencia de Hipertexto) y XML (*eXtensible Markup Language*, Lenguaje de Marcas Extensible) se emplean de manera conjunta para el intercambio de información. Sin embargo, la mayor aportación de esta iniciativa es la de CIM (*Common Information Model*, Modelo de Información Común), que se ha definido con el objetivo de poder modelar todos los recursos a gestionar de una empresa. Con dicho modelo se han especificado los esquemas CIM, que definen la información de gestión relacionada con dichos recursos, utilizando el lenguaje conocido como MOF (*Managed Object Format*, Formato de Objetos Gestionados).

Las ventajas que conlleva utilizar este modelo como medio de integración se verán en el capítulo siguiente, dedicando esta sección únicamente a presentar las características relacionadas con el modelo de gestión integrada.

2.7.1 HTTP y XML

Como su nombre indica, WBEM hace uso de las tecnologías de la Web. El uso de estas tecnologías, habituales para los servicios basados en Web, permiten reutilizar herramientas ya existentes. Para adaptar HTTP [Fielding99] a las necesidades propias de comunicaciones de gestión, se aprovechan las extensiones de HTTP definidas en [Nielsen00] que permiten incluir nuevas cabeceras específicas para una aplicación concreta. En [DMTF03b] se describen dichas cabeceras, que permiten indicar al servidor que la solicitud que se está realizando es relativa a operaciones de gestión, pudiendo también el servidor devolver respuestas a dichas operaciones. Tras las cabeceras es necesario especificar el tipo de operación que se realiza, y sobre qué información se realiza. Para ello, servidor y cliente intercambian mensajes expresados en XML [Bray00], definiéndose también en [DMTF03b] la forma en que se codifican dichos mensajes, a partir de un DTD (*Document Type Definition*, Definición de Tipos de Documento) conocido como CIM-XML. Éstos pueden ser <`SIMPLEREQ`> y `<SIMPLERSP>` para peticiones y respuestas simples, y `<MULTIREQ>` y `<MULTIRSP>` para peticiones y respuestas múltiples, conteniendo a su vez `<METHODCALL>`, `<METHODRESPONSE>` o `<ERROR>`. Además, existen `<PARAMVALUE>` y `<RETURNVALUE>`.

Los tipos de operaciones, o nombres de estos métodos, pueden ser sobre datos (ejemplares de clase), metadatos (definición de clases) o consultas (obtener ejemplares que cumplan una condición), pudiendo ser:

- Sobre datos. Para este caso existen las operaciones `GetInstance`, `DeleteInstance`, `CreateInstance`, `ModifyInstance`, `EnumerateInstances`, `EnumerateInstanceNames`, `GetProperty`, `SetProperty`.
- Sobre metadatos. Éstas incluyen `GetClass`, `DeleteClass`, `CreateClass`, `ModifyClass`, `EnumerateClasses`, `EnumerateClassNames`, `GetQualifier`, `SetQualifier`, `DeleteQualifier`, `EnumerateQualifiers`.
- Consultas. Las operaciones relacionadas con consultas son `ExecQuery`, `Associators`, `References`, `Enumerators`.

Por otro lado, además de las operaciones es necesario definir la sintaxis en que se transmitirán los datos y metadatos contenidos en los mensajes correspondientes. Dicha codificación se define en [DMTF02a], representándose en XML la información CIM a partir del mismo DTD que se comentaba anteriormente.

Con todo, la arquitectura de protocolos que se utiliza podría representarse como en la Figura 2.13.

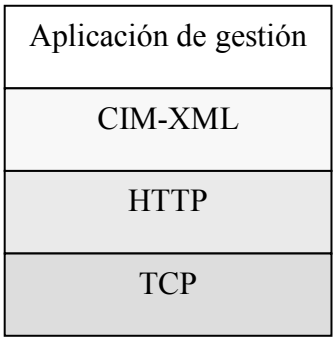

Figura 2.13. Arquitectura de protocolos de HTTP-XML.

Actualmente se trabaja en la adopción de SOAP (*Simple Object Adaptor Protocol*, Protocolo Simple de Adaptación de Objetos) [Mitra02], recomendación del *World Wide Web Consortium*[11] (W3C, Consorcio de la World Wide Web) para el intercambio de mensajes sobre HTTP, con lo que se aprovecharían todavía más las herramientas que existen actualmente.

2.7.2 MOF/CIM

La especificación de CIM [DMTF99] describe un modelo orientado a objetos a partir de su metamodelo. Este metamodelo se describe mediante un diagrama de clases de UML, y define las construcciones básicas de CIM incluyendo Clases, Propiedades, Métodos, Calificadores, y Esquemas, siendo lo suficientemente expresivo para permitir traducciones sintácticas entre CIM y los otros modelos de información. A continuación se detallan los distintos elementos que lo componen, ilustrados en la Figura 2.14:

- Elemento nombrado (*Named Element*). Cualquiera de los elementos del metaesquema.
- Clase (*Class*). Colección de ejemplares que soportan el mismo tipo: es decir, las mismas propiedades y métodos. Las clases se pueden organizar en una jerarquía de generalización que representa una relación de subtipos entre clases. En este caso no se soporta herencia múltiple. Las clases tienen propiedades y métodos, que se describen a continuación. Una clase puede participar en asociaciones siendo el destino de una de las referencias que posee la asociación. Las clases también tienen ejemplares.
- Propiedad (*Property*). Valor usado para caracterizar ejemplares de una clase. Se puede pensar como un par de funciones de obtención y modificación que cuando se aplican a un objeto, devuelven o cambian su estado respectivamente. Las propiedades poseen un tipo de datos.

[11] http://www.w3.org

- Método (*Method*). Representan el comportamiento relevante de una clase. Se declaran con los parámetros de la operación, y el tipo de datos que devuelve. En el caso de una clase no abstracta puede implicar una implementación.

  Se incluyen asociaciones reflexivas para las propiedades y métodos que permiten su redefinición. Un método puede redefinir un método heredado, lo que implica que cualquier acceso al método heredado resultará en la invocación de la implementación del método redefinido. Una interpretación similar se aplica a la redefinición de propiedades.

- Referencia (*Reference*). Especialización de una propiedad que define el papel que juega cada objeto en una asociación. Las referencias indican los ejemplares de las clases que participan en una asociación concreta.

- Asociación (*Association*). Especialización de una clase que contiene dos o más referencias. Representa una relación entre dos o más objetos. Debido a la forma en que se definen las asociaciones, se puede establecer una relación entre clases sin afectar a ninguna de las clases implicadas. Es decir, añadir una asociación no afecta a la interfaz de dichas clases. Sólo las asociaciones pueden tener referencias. Una asociación no puede ser una subclase de una clase que no sea una asociación y cualquier subclase de una asociación es una asociación. Las asociaciones soportan la provisión de múltiples ejemplares de relaciones para un objeto dado. Por ejemplo, un sistema puede estar relacionado con un gran número de componentes de dicho sistema.

- Disparador (*Trigger*). Indica el reconocimiento de un cambio de estado de un ejemplar de una clase (como la creación, borrado, actualización, o acceso a un objeto) o la actualización o acceso a una propiedad.

- Indicación (*Indication*). Objeto que se crea como resultado de un disparador. Debido a que las indicaciones son subtipos de clases, pueden tener propiedades y métodos, y organizarse jerárquicamente.

- Calificador (*Qualifier*). Se usan para proporcionar características adicionales de Elementos Nombrados (una clase, una propiedad o un método, por ejemplo). Los calificadores proporcionan un mecanismo que hace posible la extensión del metaesquema de una forma limitada y controlada. Es posible añadir nuevos tipos de calificadores definiendo sus nombres, proporcionando así nuevos tipos de meta-datos para procesar la gestión y manipulación de clases, propiedades y otros elementos del metaesquema. El uso estos calificadores que permiten ampliar el metamodelo fácilmente también reduce la formalización del mismo, al ser todos los calificadores opcionales y tener una definición únicamente textual.

- Esquema (*Schema*). Grupo de clases definidas por una misma entidad. Los esquemas se usan para administración y nombrado de clases. Los nombres de las

clases deben ser únicos dentro de sus esquemas. Un esquema contiene ejemplares del conjunto de elementos del metamodelo.

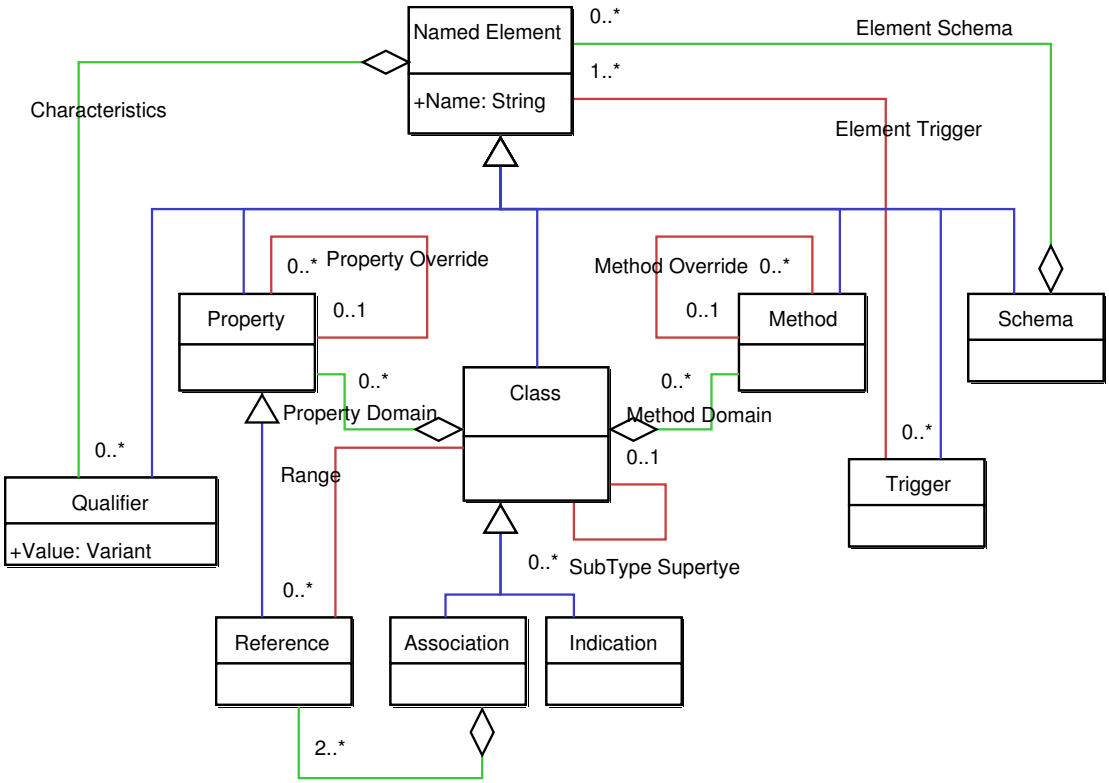

Figura 2.14. Metamodelo en UML de CIM v2.2 según [DMTF99].

Este conjunto de construcciones se puede representar de múltiples maneras:

- MOF (*Managed Object Format*, Formato de Objetos Gestionados) es el método de representación habitual en los gestores que soportan CIM. Aunque emplee el mismo acrónimo, MOF no tiene nada que ver con la Facilidad de Meta Objetos (*Meta Object Facility*) [OMG02b] de OMG. Para diferenciarlo de ésta, en el resto de la tesis se hará referencia a MOF/CIM. Este lenguaje permite describir textualmente las construcciones de CIM, con una sintaxis de lenguaje orientado a objetos parecida a IDL. Es bastante expresivo, gracias al uso de calificadores, siendo su complejidad bastante menor que la que presenta GDMO.
- Asimismo, se pueden emplear diagramas de clases en UML para modelar la información de forma gráfica, y XML para la transmisión de la información, como se comentaba anteriormente.

2.7.3 Esquemas CIM

El nombre de CIM se emplea para definir tanto el lenguaje como los modelos de información, llamados esquemas CIM. Estos esquemas tratan de unificar los modelos de información ya existentes en un modelo común (de ahí el nombre de CIM) y unificado.

Los esquemas se estructuran de forma jerárquica, con un núcleo central que se va extendiendo mediante el uso de herencia de las clases definidas en dicho esquema. Esta jerarquía va de lo más general a lo particular, existiendo tres niveles (ver Figura 2.15):

1. Modelo nuclear (*core model*). Contiene clases y asociaciones aplicables a todos los dominios de gestión, definiendo los conceptos básicos comunes a cualquier dominio: Elemento Gestionado, Sistema, Servicio, Dispositivo, Elemento Físico...
2. Esquemas de dominio (*extension schemas*). Partiendo del modelo nuclear se definen los esquemas para cada dominio particular. Para abordar dominios específicos, los modelos comunes especializan clases del modelo nuclear. Los distintos dominios pueden ser: gestión de sistemas (*system*), nivel físico (*physical*), dispositivos (*device*), aplicaciones (*application*), redes (*network*), integración con directorios (*directory*), etc..
3. Esquemas de extensión (*extension schema*). Finalmente, se puede crear un esquema específico para un caso concreto. Los esquemas de extensión suelen ser definidos por los fabricantes particularizando un modelo común para una arquitectura específica (por ejemplo, los esquemas de Windows o de Solaris).

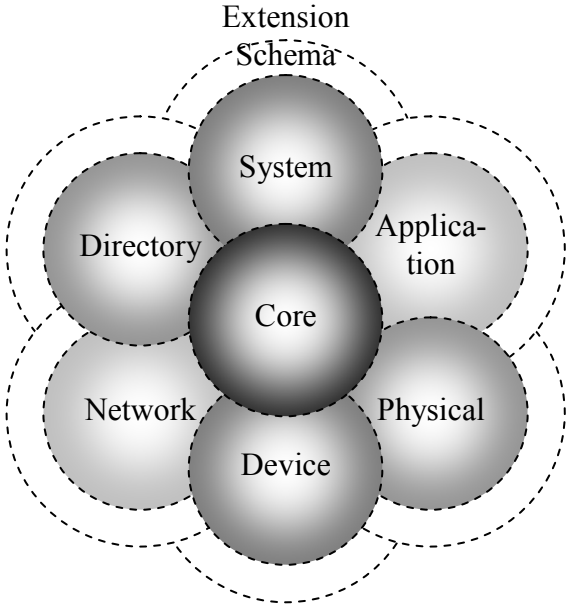

Figura 2.15. Esquemas CIM.

2.8 Conclusiones

El presente capítulo ha presentado el concepto de Gestión de Red Integrada y los distintos modelos de gestión de red integrada existentes. Éstos se resumen en la siguiente tabla:

Tabla 2.1. Modelos de Gestión de Red Integrada.

Modelo	Protocolo de intercambio	Lenguaje de definición de información	Modelos de información
TMN/OSI-SM	CMIP	GDMO	X.721, M.3100
Internet	SNMP	SMI	MIB-II, otras
Equipos de sobremesa	DMI	MIF	Master MIF
Plataformas de procesamiento distribuido	CORBA/IIOP	IDL	X.780, M.3120
Gestión basada en Web	HTTP-XML	MOF/CIM	Esquemas CIM

Por tanto, existen múltiples modelos incompatibles que a su vez poseen lenguajes de definición con distinto grado de expresividad. Además, estos lenguajes son poco formales y con poca o nula capacidad para definir el comportamiento del gestor, lo que los hace difícil de implantar en gestores inteligentes. Este hecho puede implicar graves problemas de interoperabilidad que dificulten el objetivo planteado al principio del capítulo de gestionar recursos heterogéneos de una manera homogénea.

En el siguiente capítulo se presentan los estudios más relevantes que se han realizado hasta la fecha para tratar de solventar estas cuestiones, analizando los distintos mecanismos que se han propuesto para integrar los distintos modelos de gestión aquí expuestos.

Capítulo 3 Mecanismos de integración de información de gestión

3.1 Introducción

Las distintas soluciones de gestión integrada de recursos heterogéneos presentadas en el capítulo anterior también han supuesto la paradoja de que no pueda existir realmente una gestión totalmente integrada. Cada modelo de gestión utiliza su propio protocolo (CMIS/CMIP, SNMP, DMI, IIOP, HTTP), su propio lenguaje de definición (GDMO, SMI, MIF, IDL, CIM) y sus propias definiciones de información (M.3100, MIB-II, Master MIF, M.3120, esquemas CIM), incompatibles entre sí. La multiplicidad de modelos de gestión de red puede suponer el uso de varios de ellos en algunos escenarios para poder acceder a la totalidad de recursos gestionados.

Por ello se ha planteado la necesidad de establecer mecanismos de interoperabilidad, que permitan coordinar la gestión de una forma unificada. Diversos estudios que se presentarán a lo largo del presente capítulo dividen este problema en dos puntos que se refieren al protocolo de comunicaciones (integrar el método para intercambiar los datos) y al modelo de información (integrar las especificaciones que definen los distintos recursos gestionados). Si se puede establecer una regla que traduzca ambas cuestiones, la interoperabilidad es posible.

También se verá que en lo que se refiere a la interoperabilidad de modelos de información, los estudios actuales únicamente consiguen realizar una traducción sintáctica entre los distintos lenguajes de definición existentes. Sin embargo, esto no permite que la semántica que contenía cada concepto definido en un lenguaje se pueda integrar con los que existen en el lenguaje de destino. Por ello se hace necesario algún mecanismo que permita llevar a cabo una traducción semántica de los modelos de información que haga corresponder los conceptos de cada dominio según su significado.

El capítulo se ha estructurado de la siguiente manera: una primera sección explicará los mecanismos propuestos por organismos de estandarización para proporcionar interoperabilidad entre los modelos de gestión integrada existentes, analizando la solución que se consigue. A continuación se descubrirá la arquitectura de gestión basada en Web como integradora de los modelos anteriores, estudiando las mejoras que se obtienen con respecto a las soluciones previas. Tras esto, se incluirán otros estudios realizados por equipos de investigación independientes, presentando los trabajos que han tenido

relevancia en este sentido. Finaliza el capítulo el apartado de conclusiones, que analiza de manera general los resultados obtenidos en el conjunto de propuestas presentadas.

3.2 Mecanismos propuestos por organismos de estandarización

A continuación se exponen las ideas propuestas en este sentido por organismos de estandarización. En general todas ellas han tenido en cuenta cómo integrar distintos protocolos y distintos modelos de información. En primer lugar merece la pena considerar los conceptos que en este sentido se proponen en las recomendaciones de la ITU sobre TMN. Tras esto se verán otras propuestas más pragmáticas, que tratan de integrar los modelos de gestión OSI, SNMP, DMI y CORBA. El estudio de WBEM se deja para la sección siguiente, al tener entidad propia como propuesta integradora de modelos de gestión.

3.2.1 ITU

Con la recomendación de TMN la ITU ha formalizado en cierta manera los problemas de interoperabilidad. La recomendación M.3010 [ITUT00b] incluye varias referencias a esta cuestión, relacionadas con la interacción entre las interfaces Q y otros modelos de gestión no integrados en TMN.

Así, su apartado 11.1.2 presenta los bloques físicos de transformación. La transformación proporciona la conversión entre distintos protocolos y formatos de datos para el intercambio de información entre bloques físicos. Hay dos tipos de transformación: adaptación y mediación, que actúan a modo de pasarelas.

- Un dispositivo de adaptación o adaptador proporciona transformación entre una entidad física no-TMN a un Elemento de Red o un Sistema de Operación de TMN.
- Un dispositivo de mediación proporciona la transformación entre bloques físicos TMN que incorporan mecanismos de comunicación incompatibles.

Además, su apartado 11.4 presenta el concepto de la interfaz de interoperabilidad, que es útil para simplificar los problemas de comunicaciones que aparecen en una red de proveedores y capacidades múltiples. Este tipo de interfaz está constituido por el conjunto formalmente definido de protocolos, procedimientos, formatos de mensaje y semántica utilizados para las comunicaciones de gestión. Asimismo, también se destaca la existencia de un conocimiento de gestión compartido que incluye el entendimiento del modelo de información de la red gestionada (clases de objetos soportadas, funciones soportadas, etc.), de los objetos de soporte de gestión, de las opciones, del contexto de aplicación soportado, etc. El conocimiento de gestión compartido asegura que cada extremo de la interfaz (gestores y agentes) entienda el significado exacto de un mensaje enviado por el otro extremo.

Finalmente, su apartado 12.3 amplía la información en lo referente al conocimiento de gestión compartido, indicándose que para que los sistemas de gestión de las comunicaciones interoperen, deben compartir un enfoque o entendimiento común de al menos la siguiente información: capacidades de protocolo soportadas, funciones de gestión soportadas, clases de objetos gestionados soportadas, ejemplares de objetos gestionados disponibles, capacidades autorizadas y relaciones entre objetos.

En resumen, según esta propuesta la interoperabilidad se puede conseguir teóricamente de varias maneras que consistirán básicamente en la forma en que se defina tanto el intercambio como la definición de la información. Sin embargo, dado su enfoque teórico que no indica salvo en líneas muy generales la manera en se debe implementar tales soluciones, este análisis no es útil para ser llevado a la práctica. Por tanto, se hace necesario estudiar otras aproximaciones que incluyan detalles sobre la forma de realizar los dispositivos de adaptación y mediación y los algoritmos de transformación de la llamada interfaz de interoperabilidad.

3.2.2 IIMC

Tras la visión general que ha proporcionado el apartado anterior, el primer trabajo que trató estos problemas de una manera concreta fue llevado a cabo por el grupo de trabajo conocido como IIMC (*ISO/CCITT to Internet Management Coexistence*, Coexistencia de la Gestión ISO/CCITT e Internet), formado por algunas de las principales organizaciones del mundo de la gestión (el IETF y lo que actualmente son el Open Group[1] y Tele Management Forum[2], entonces X/Open y Network Management Forum, respectivamente).

Dicho grupo de trabajo estudió las cuestiones relativas a la interoperabilidad entre la gestión según OSI y según Internet, generando varios documentos sobre el tema:

- IIMCIMIBTRANS (*IIMC Internet MIBs TRANSlation*, Traducción de MIBs de Internet según IIMC) [NMF93a]. Detalla las reglas que permiten traducir la sintaxis de una MIB de Internet en una MIB escrita en GDMO. Para ello, trata las tablas y grupos de variables como clases de objetos gestionados.
- IIMCOMIBTRANS (*IIMC OSI MIBs TRANSlation*, Traducción de MIBs de OSI según IIMC) [NMF93e]. De manera semejante al caso anterior, propone las reglas que permiten traducir en lo posible (no pudiéndose para algunos casos) una MIB escrita en GDMO a otra en SMI, tratando las clases de objetos gestionados como tablas.

[1] http://www.opengroup.org/

[2] http://www.tmforum.org/

- IIMCMIB-II (*IIMC internet MIB-II translation*, Traducción de la MIB-II de Internet según IIMC) [NMF93d]. Especifica la MIB-II de Internet con plantillas de GDMO empleando las reglas de traducción antes comentadas.
- IIMCPROXY (*IIMC management PROXY*, Delegado de Gestión según IIMC) [NMF93c]. Define las características de un sistema intermedio que actúe de pasarela y traduzca peticiones CMIP a un agente SNMP, al estilo del dispositivo de mediación antes explicado. Cabe decir que el inverso (una pasarela de SNMP a CMIP) no se ha definido, puesto que la complejidad inherente a una entidad CMIP no es abordable desde SNMP.
- IIMCSEC (*IIMC management SECurity*, Seguridad de la Gestión según IIMC) [NMF93b]. Explica cómo se deben abordar las cuestiones referentes a la seguridad en ambos modelos de gestión. Este documento está obsoleto, puesto que trata el modelo de seguridad de la primera versión de SNMP, basado en comunidades, y el modelo de seguridad de gestión en Internet se ha reformado y mejorado con la aprobación de SNMPv3 como estándar del IETF.

La propuesta de IIMC es mucho más concreta que el caso anterior permitiendo, en definitiva, llevar a cabo la gestión de dominios SNMP y CMIP desde un mismo gestor CMIP, usando una pasarela en el caso de acceder al dominio SNMP. Sin embargo, los agentes finales seguirían utilizando su propio modelo de información de gestión, con lo que el gestor debería manejar ambos modelos de manera separada, incluso en el caso en el que los conceptos que se trataran fueran similares, con el consiguiente aumento de complejidad. Además, se obvia la existencia de otros modelos de gestión integrada que también pueden aparecer en un escenario más amplio, con lo que la solución no es válida para un ámbito general.

3.2.3 DMI y SNMP

Al especificar DMI se partió de la existencia de otros dominios de gestión, definiéndose desde su principio su interoperabilidad con SNMP. Para ello, la solución definida en [DMTF97] posee los siguientes elementos:

- Un procedimiento administrativo para asignar OIDs de SNMP a grupos de clases MIF de forma que los grupos relacionados se puedan agrupar en MIBs. Para ello se define un método para incluir OIDs en el archivo MIF a través de cláusulas `pragma` de forma que el par de cadena de caracteres de clase y el OID estén disponibles en tiempo de ejecución en la Interfaz de Gestión.
- Un conjunto de algoritmos para convertir los atributos MIF en objetos SMI, de forma que se pueda implementar un agente de correspondencia general para realizar el algoritmo inverso en tiempo de ejecución utilizando sólo información disponible a través de la Interfaz de Gestión. Con estos algoritmos se podría crear

un programa *MIF-to-MIB* (de MIF a MIB) que tome como entrada un archivo MIF y devuelva uno o más archivos MIB de acuerdo con el esquema de traducción definido.

- Una MIB de DMI estándar del DMTF que proporcione acceso a metadatos DMI tales como nombres, tipos, enumeraciones, descripciones, etc. para todas las MIF instaladas en un sistema. Una aplicación SNMP podría cargar la información desde esta MIB y generar las mismas MIBs que con el programa *MIF-to-MIB*.
- Un agente general de correspondencia entre DMI y SNMP que resida en el nodo gestionado. El agente de correspondencia asignará un subárbol de OIDs por cada componente DMI registrado. Con esto podrá satisfacer operaciones SNMP entrantes sobre dichos OIDs haciendo llamadas apropiadas a la interfaz de gestión. El agente de correspondencia también implementará completamente la MIB DMI referida anteriormente.

La novedad de esta propuesta es que además de las reglas para traducir sintácticamente la información y definir un sistema de mediación, también se define información para traducir metadatos, para poder obtener a través del sistema mediador no sólo los ejemplares de la información, sino también la propia información. Sin embargo, esta solución adolece de los mismos problemas que el caso anterior. Se define la forma de acceder a un dominio desde el otro dominio, pero no la inversa. Además, tampoco contempla la posibilidad de que existan otros modelos de gestión integrada. Finalmente, aunque permita el tratamiento de metadatos, el gestor deberá manejar distintos modelos de información de gestión dado que los agentes finales seguirían utilizando su propio modelo de información de gestión, a pesar de que la MasterMIF se solapa en muchos conceptos con las distintas MIBs de SNMP.

3.2.4 JIDM

En lo que se refiere a CORBA, OMG junto con el Open Group crearon el grupo de trabajo JIDM (*Joint Inter-Domain Management*, Gestión Inter-Dominio Unificada) [Open00]. Este grupo partió de los trabajos de IIMC y empleó una metodología similar para abordar el problema de interoperabilidad entre CORBA, SNMP y CMIP. Para ello definieron dos tipos de traducciones:

- JIDM-ST (*Specification Translation*, Traducción de Especificaciones). Ésta detalla la traducción entre la estructura de la información de gestión y la definición de interfaces de los objetos CORBA. Esto se hace a dos niveles. El primer nivel se refiere a la traducción entre tipos de datos ASN.1 e IDL, dando las reglas comunes de traducción usadas tanto en los algoritmos de traducción de GDMO como SNMP, al usar ambos estos tipos de datos para sus definiciones. El segundo nivel, no menos importante, trata la traducción en ambos sentidos de especificaciones GDMO e IDL y la traducción de SMI a IDL, incluyendo también un archivo de

relación entre OIDs y nombres. No se especifica la traducción de IDL a SMI puesto que la complejidad de IDL no sería abordable por SMI en la mayoría de los casos.

- JIDM-IT (*Interaction Translation*, Traducción de Interacciones). Aquí se aborda la traducción de operaciones entre CORBA y los dominios de gestión, cubriendo el proceso por el cual las interacciones de un dominio se traducen en una o más interacciones en el otro dominio. La entidad responsable de traducir las interacciones, por ejemplo, puede recibir una PDU (*Protocol Data Unit*, Unidad de Datos del Protocolo) de CMIP o SNMP, traduciéndola a una o más peticiones a interfaces IDL. Los resultados deben ser posteriormente recogidos y formateados en una o más PDUs CMIP o SNMP de respuesta. En suma, la Traducción de Interacciones debe cubrir la inicialización, identificando cómo se inicializa y utiliza la entidad que realice las traducciones, cómo identifica la población de objetos existentes y qué otros ejemplares de servicios puede necesitar. Para ello se define un conjunto de servicios que permitan conocer la referencia a la información entre los distintos dominios y poder de esta manera acceder a ella. A su vez, estos nuevos servicios podrán utilizar otros servicios estándar del dominio CORBA, como por ejemplo, el servicio de nombres para resolver nombres unívocos de ejemplares, el servicio de ciclo de vida para crear nuevos ejemplares de objetos y canales de eventos para la distribución de los mismos.

Con este conjunto de definiciones es posible desarrollar pasarelas que traduzcan entre los distintos dominios. Como idea interesante aparece el uso de servicios que se encarguen de realizar el conjunto de tareas para llevar a cabo el intercambio de información. Además, esta solución tiene en cuenta IIMC, con lo que se amplia el número de dominios de gestión que puede aparecer en un mismo escenario, aunque no es posible acceder a recursos gestionados con DMI. Por otro lado, al trabajar con tres dominios de gestión diferentes aumenta el problema de los modelos de información. En este caso, un gestor CORBA debería manejar tres modelos distintos, uno para cada dominio de gestión, aunque los conceptos que se que contengan dichos modelos sean similares (como las recomendaciones M.3100 y M.3120), dado que las traducciones se hacen sobre las estructuras de información y no sobre el contenido de dichas estructuras.

3.2.5 Análisis general de las soluciones

A partir de los mecanismos presentados en los apartados anteriores se pueden implementar soluciones estandarizadas, normalmente basadas en pasarelas. La Figura 3.1 muestra los elementos mediadores, o pasarelas, que podrían existir según dichas especificaciones [LópezDeVergara02]:

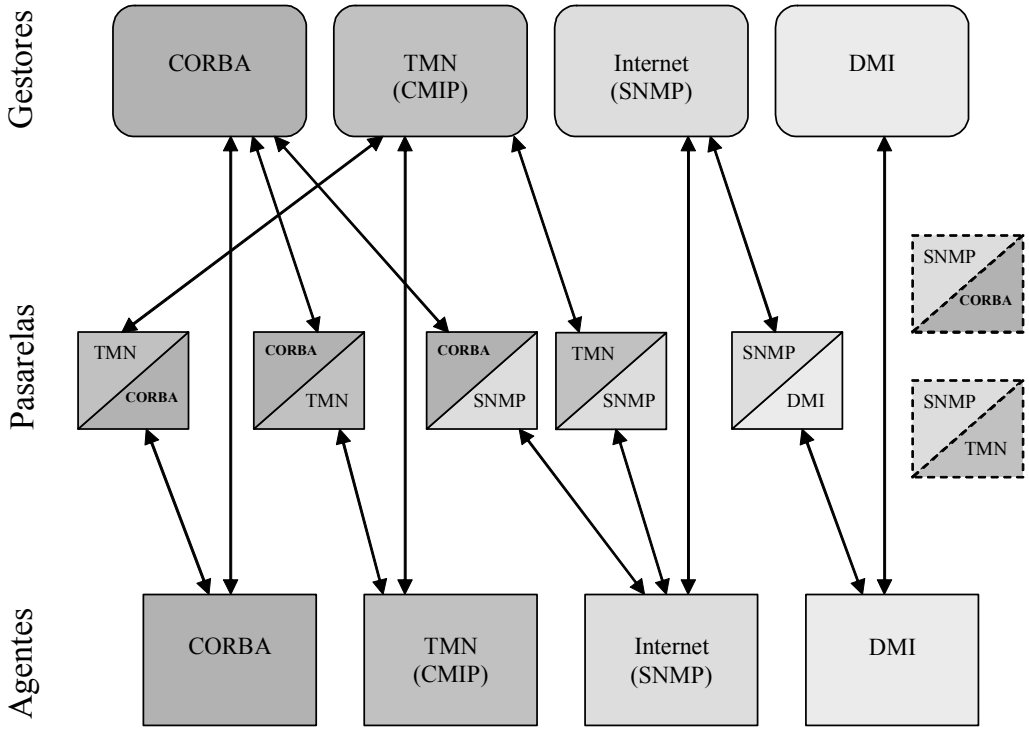

Figura 3.1. Pasarelas definidas para los distintos dominios de gestión.

- Así, desde un gestor CORBA se puede acceder a objetos CORBA, con información definida en IDL y a agentes TMN y SNMP con información definida en GDMO y SMI respectivamente.
- El caso de un gestor TMN es bastante similar, pudiéndose gestionar agentes TMN, con información definida en GDMO, y objetos CORBA y agentes SNMP con información definida en IDL y SMI respectivamente.
- Para el caso de gestores SNMP, el acceso a otros dominios distintos a SNMP está restringido. El acceso a objetos CORBA sólo es posible si la información se ha definido en SMI y traducido posteriormente a IDL [Asensio99]. Otro tanto ocurre en el caso de TMN, donde la traducción de GDMO a SMI no es siempre posible.
- Finalmente, en DMI, el acceso es posible únicamente desde aplicaciones DMI y gestores SNMP, sin otras alternativas. Además, el mecanismo de traducción no sólo hace traducciones de especificaciones sino que también existe una MIB en SMI que deberá poseer el agente para acceder a metadatos de MIF.

Con todo, queda claro que la integración que se consigue en estos casos es bastante parcial, no pudiéndose desde un mismo dominio de gestión acceder a la totalidad de ellos. Esto quiere decir que en aquel escenario en que existan recursos que se gestionan usando todos estos dominios, no será posible realizar esa gestión desde una interfaz única. Además, la integración se restringe a poder realizar un conjunto de operaciones sobre un modelo de información que se ha traducido sintácticamente al lenguaje adecuado, sin existir una integración real con los modelos de información existentes en dicho dominio. La siguiente

sección expone WBEM como un modelo de gestión integrada definido desde la perspectiva de la integración de dominios que trata de solventar estas cuestiones.

3.3 Gestión de red basada en *Web*

Las distintas soluciones existentes para solventar los problemas de interoperabilidad han permitido llevar a cabo una gestión parcialmente integrada de distintos recursos, pero sin conseguir una integración total de los recursos a gestionar en un sistema en red. Quedan varios problemas por resolver, puesto que tanto el método de acceso como la información manejada por el gestor siguen siendo dependientes del dominio en que se encuentre el recurso gestionado. Sería deseable llegar a una solución similar a la de la Figura 3.2, en que las aplicaciones de gestión, mediante un modelo de información único y a través de una única interfaz, se pudieran comunicar con los distintos recursos de una empresa, que habitualmente se gestionarán según su naturaleza: los elementos y servicios de telecomunicación, mediante CMIP; los dispositivos de redes de datos, usando SNMP; los ordenadores de sobremesa, con DMI; y otros recursos, como aplicaciones y servicios distribuidos, empleando CORBA. Un escenario parecido es el que propuso el DMTF al definir la arquitectura WBEM (*Web-Based Enterprise Management*, Gestión Basada en Web de los recursos de una Empresa) [Westerinen01].

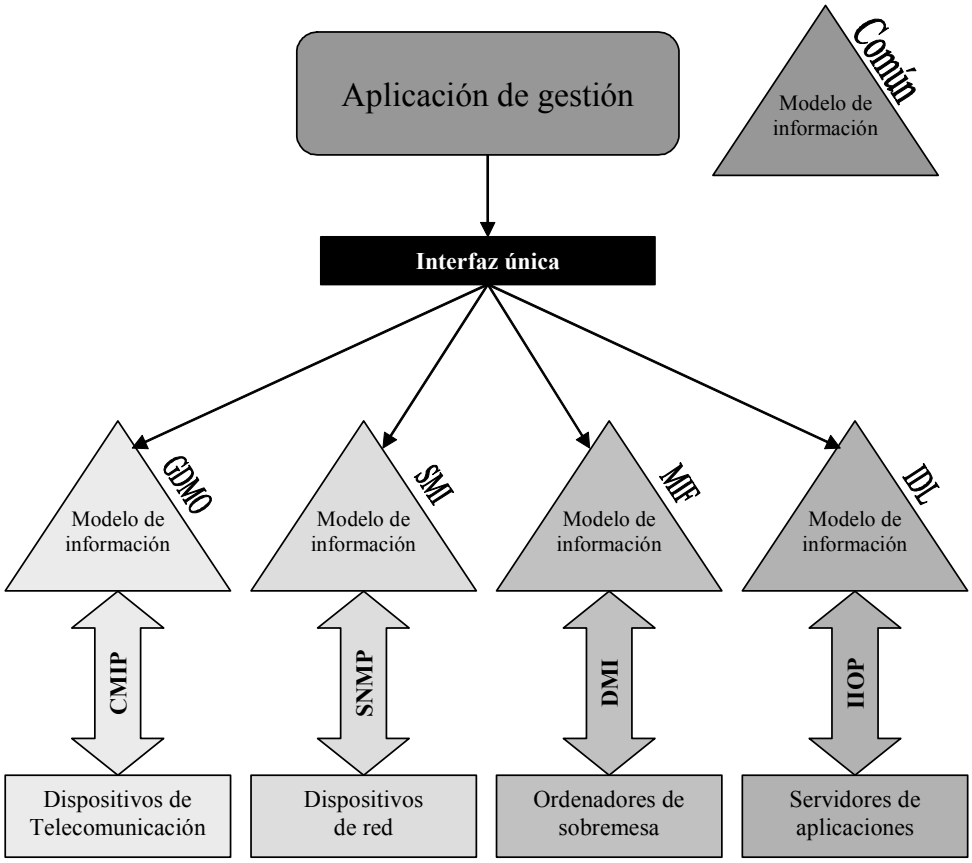

Figura 3.2. Gestión integrada de distintos dominios.

La llegada de WBEM al escenario de la gestión se ha presentado como un nuevo modelo de gestión integradora de los modelos existentes para permitir la ansiada gestión integrada de todos los recursos de la red de una corporación empresarial. La arquitectura presentada, se basa en tres pilares, ya presentados en el capítulo anterior:

- Protocolo de intercambio de la información. Se utiliza HTTP (*Hyper-Text Transfer Protocol*, Protocolo de Transferencia de Hipertexto), adaptado con ciertas cabeceras para permitir el intercambio de la información de gestión.
- Definición de la información. Se emplea CIM (*Common Information Model*, Modelo de Información Común), que trata de modelar la información referida a cualquier dominio. A su vez, CIM posee una sintaxis conocida como MOF (*Managed Object Format*, Formato de Objetos Gestionados).
- Modelo de información. Se ha definido un conjunto de esquemas CIM que engloban varios dominios de recursos diferentes.

WBEM usa un protocolo, una sintaxis y un modelo de información que son neutros e independientes de los ya vistos. La principal ventaja de esta neutralidad es que no hace depender esta solución de otras previas que no fueron capaces de resolver el problema. El inconveniente es añadir más complejidad de protocolos, estructuras y modelos de información.

3.3.1 Arquitectura WBEM

La arquitectura de WBEM [Hegering99] se ilustra en la Figura 3.3. Dicha arquitectura es un modelo de varios niveles que siguen el patrón arquitectónico de Gestión Web presentado en [LópezDeVergara01], y está pensada para integrar el acceso a los recursos gestionados en distintos dominios.

Éstos son de arriba a abajo los niveles según la figura:

1. Un cliente WBEM. Dicho cliente será la interfaz de usuario del operador del sistema gestionado. Se comunica con el siguiente nivel con una adaptación de HTTP y XML que permite enviar mensajes con operaciones de gestión.
2. Un CIMOM (*CIM Object Manager*, Gestor de Objetos CIM), elemento nuclear de la arquitectura. Este gestor de objetos maneja el modelo de información independiente de los dominios de gestión llamado CIM, orientado a objetos y representable en XML.
3. Un conjunto de pasarelas entre el CIMOM y los agentes de los distintos dominios de gestión, denominadas proveedores. Éstos deberán ser capaces de interactuar con el CIMOM, utilizando HTTP y XML, y con cada agente, usando el protocolo (CMIP, SNMP, DMI, IIOP...) para acceder a cada dominio concreto.

4. Un conjunto de agentes que se encuentren en los recursos gestionados. Interactuarán con los proveedores empleando el protocolo asociado a su dominio de gestión.

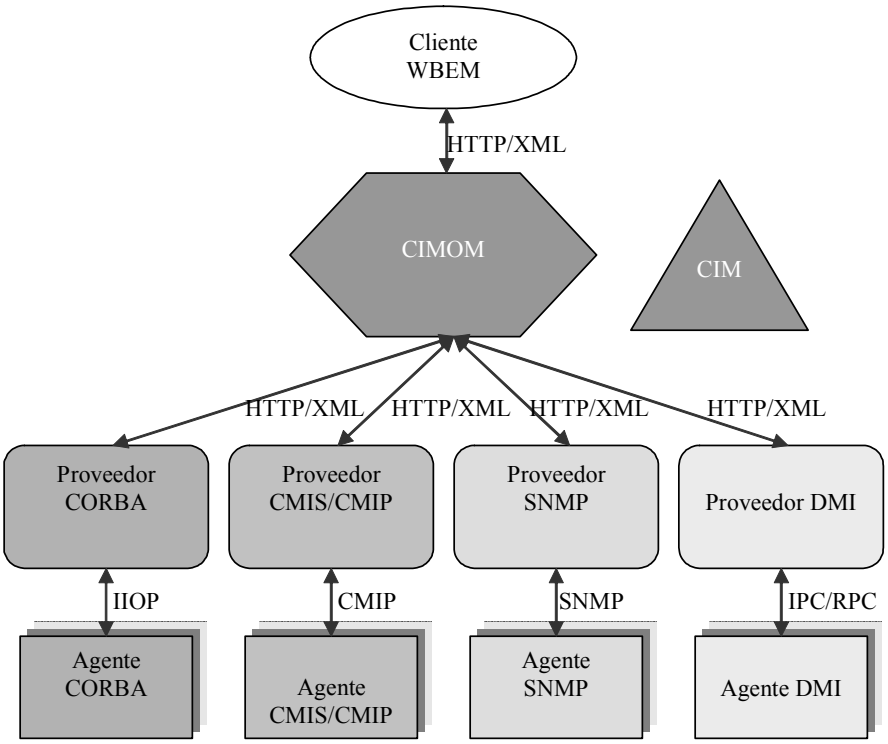

Figura 3.3. Arquitectura de WBEM.

3.3.2 Traducciones de modelos de información

La especificación de CIM tiene en cuenta la necesidad de traducir aquella información definida en otros dominios para poder trabajar con un modelo de información común y para ello define tres posibles métodos:

- Traducción técnica. Las metaconstrucciones de un dominio origen se describen usando las construcciones del dominio destino. Es decir, se define un modelo en el dominio destino en el que se pueda ejemplarizar el modelo de información del dominio origen concepto a concepto. Esto sería similar al caso de la MIB de DMI, que define en SMI las metaconstrucciones de MIF.
- Traducción mediante reescritura (*recast*). Se define una correspondencia entre las metaconstrucciones de ambos dominios, traduciendo el modelo de información del dominio origen al destino utilizando dicha correspondencia. A esta traducción se le podría denominar traducción sintáctica, pues traduce de un lenguaje de especificación a otro siguiendo las normas sintácticas de cada uno. Esta solución es la que se ha aplicado normalmente en los estudios presentados en la sección anterior.

- Traducción de dominio. Realiza una correspondencia entre el contenido de los modelos origen y destino. Esta traducción es posible cuando ambos modelos representan entidades idénticas en distintos dominios. Frente a la reescritura, aquí se realiza la traducción en el nivel semántico, interpretándose los conceptos según su significado y no en función de cómo están escritos.

Además, esta especificación indica que para poder deshacer estas traducciones es necesario incluir anotaciones (*scratch pad*) en forma de calificadores. Por este motivo está definido el calificador `MappingStrings` con el que se puede indicar el identificador que poseía una propiedad en cada dominio. Esta solución es similar a la empleada en DMI, comentada anteriormente, donde se definía la cláusula `pragma oid` para hacer corresponder un atributo con un OID.

3.3.3 Análisis de la solución

Con la aparición de WBEM y CIM, surge la cuestión de si este nuevo marco es capaz de cumplir el objetivo de una integración real de la gestión. Si esto fuera así, debería ser posible integrar el método para intercambiar los datos o protocolo, la estructura de los datos y el modelo de información, tal y como se ha visto en los casos anteriores.

WBEM realiza la integración de protocolos a través de proveedores. Cada proveedor se comporta como una pasarela entre un protocolo independiente, HTTP/XML, y los protocolos de gestión. El conjunto de operaciones posibles de WBEM, sobre metadatos, datos y consultas es lo suficientemente completo como para acceder a cualquier dominio de gestión, entendiéndose que mediante esta arquitectura se puede conseguir una integración de protocolos.

En lo que se refiere a integración de estructura de datos, hay que resolver primero la correspondencia entre tipos de datos. Se puede establecer una correspondencia con los tipos de datos existentes en MOF/CIM y los existentes en ASN.1 e IDL sin muchos problemas. Tras esto, es necesario realizar una integración del modelo de información. Para este caso hay que pensar que el gestor siempre utilizará el modelo de CIM, con lo que la información que se defina a partir de ahora para cualquier dominio se debe definir previamente en CIM. Sin embargo, en general se partirá de agentes con información ya definida que se puede traducir. El DMTF no especifica cómo se debe llevar a cabo esta traducción, pero habitualmente ésta se realiza mediante reescritura, como se verá en algunos trabajos que se exponen en la sección siguiente. En definitiva, CIM y WBEM sólo consiguen una integración sintáctica, al igual que ocurría con IIMC y JIDM.

El problema de esta integración sintáctica es que no incluye las características semánticas de la información: distintos dominios de gestión pueden poseer en su información a gestionar conceptos que se solapen. Lo ideal sería que estos conceptos estuvieran reflejados en conceptos de CIM, como ocurre en aquellos casos en que están anotados

mediante un calificador que indica dicha relación, si bien dicho calificador muchas veces resultará insuficiente.

3.4 Otras aproximaciones propuestas

Durante los últimos años, se han realizado otros trabajos que trataban de solventar en mayor o menor medida los problemas de integración de modelos de gestión integrada. Dichos estudios, realizados en paralelo a la definición de mecanismos normalizados ya comentados, proponen soluciones para mejorar la interoperabilidad entre estos modelos de gestión que han tenido cierta aceptación en la comunidad científica. A continuación se examinan los que se han considerado más relevantes, siguiendo un orden cronológico de aparición.

3.4.1 Trabajos de Kalyanasundaram y Sethi

Uno de los primeros estudios relativo a la integración de modelos de gestión de red es [Kalyanasundaram94]. Aunque trata de ser lo más genérico posible, el enfoque que posee va orientado hacia la interoperabilidad entre OSI-SM y SNMP, puesto que éste era el problema existente cuando se escribió dicho artículo. Así, las soluciones que propone tienen una experimentación previa en este sentido [Kalyanasundaram93]. El estudio enumera distintas cuestiones que afectan a la interoperabilidad (modelo de información, protocolo, transparencia en el paso de un dominio al otro), proponiéndose en él distintos esquemas de interoperabilidad, estudiando además su relación con las cuestiones enumeradas anteriormente. La Figura 3.4 muestra los esquemas de interoperabilidad, que pueden consistir en:

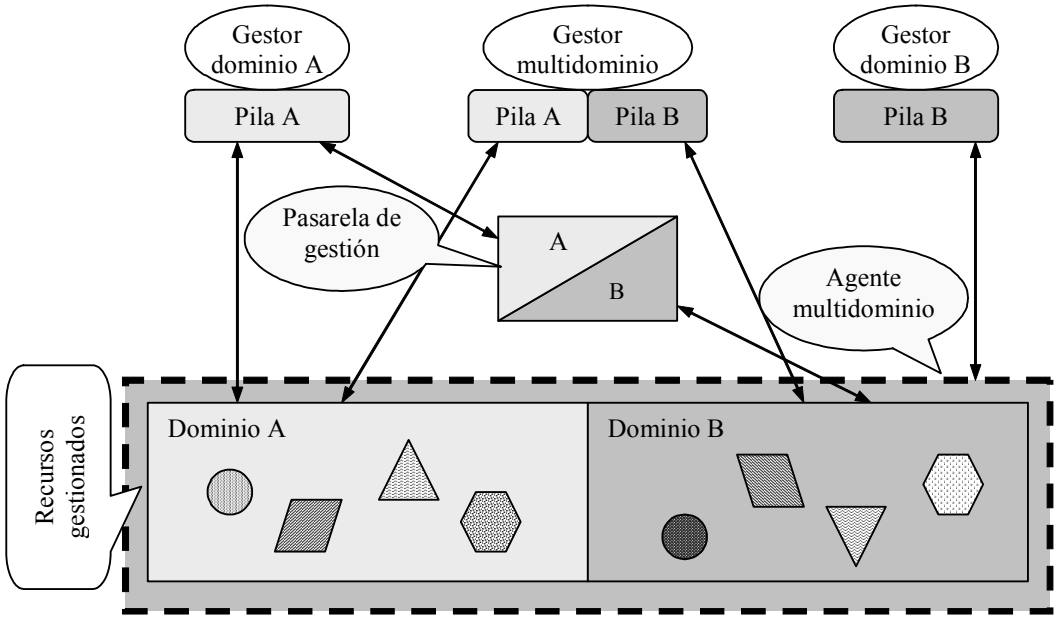

Figura 3.4. Esquemas de interoperabilidad.

- Gestores multidominio. Son gestores con varias pilas de protocolos, una por cada dominio de gestión, incrementando la complejidad y tamaño de los gestores.
- Agentes multidominio. Son agentes con múltiples interfaces para un mismo recurso, una por cada dominio de gestión, o un mismo agente con varias pilas de protocolos. Al igual que antes, se incrementa la complejidad y tamaño de los agentes que hay en la red, con el problema añadido de que el número de agentes es muy superior al de gestores, complicando su despliegue en los recursos.
- Entidades de mediación. Median entre gestores y agentes de distintos dominios. Estas entidades pueden ser de dos tipos: *Proxies* o Delegados, y Pasarelas de Aplicación. Los primeros simplemente repiten las peticiones de un lado al otro con las traducciones pertinentes. Las segundas son *proxies* especializados que proporcionan servicios y correspondencias, además de la traducción de protocolos. Estas entidades mediadoras tienen como inconveniente que deben realizar tareas complejas para interoperar entre distintos dominios, pero tienen la ventaja de que la introducción de un nuevo dominio de gestión no afecta a la infraestructura ya desplegada.

Este trabajo, aunque algo antiguo, contiene los conceptos básicos y define los problemas habituales en este campo. Aporta soluciones en lo que se refiere a interoperabilidad del protocolo, pero no respecto a la semántica de la información, por lo que no resuelve el problema de manera completa.

3.4.2 Análisis del IRIT

El grupo de Redes y Servicios del IRIT[3] (*Institut de Recherche en Informatique de Toulouse*, Instituto de Investigación en Informática de Toulouse) ha trabajado activamente en esta cuestión. En [Rivière96b] se aprovecha el estudio presentado en el apartado anterior e identifica el problema de la semántica, presentando un diagrama similar al de la Figura 3.5. Según dicho artículo, los conceptos que se tienen en cuenta en un sistema de gestión de red dependen de las entidades gestionadas, el nivel de abstracción de modelado y el tipo de conocimiento en el que se enfoca este sistema. Como resultado, la traducción de información tiene que ocuparse del problema de la equivalencia de conceptos y la consecuente pérdida de información. Esta visión es compartida por [Poirier95], que expone mediante símiles con la arquitectura de edificios, que para que distintos dominios de gestión interoperen es necesario trabajar en el nivel de la semántica de la información que hay definida.

[3] http://www.irit.fr/

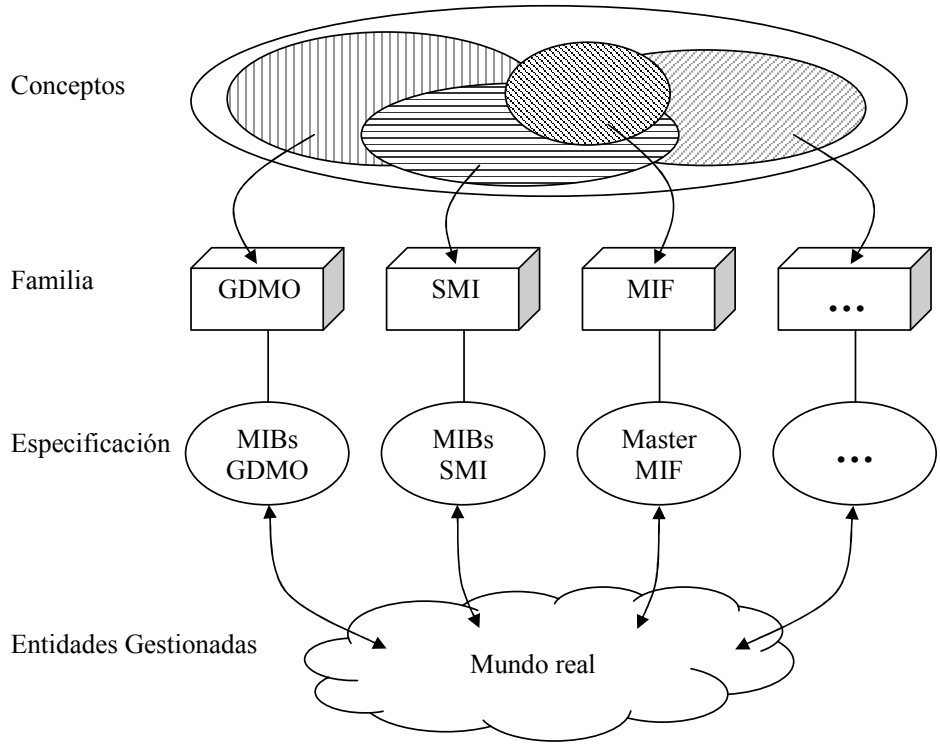

Figura 3.5. Heterogeneidad de modelos de información de gestión según [Rivière96b].

En [Rivière98] se completa el estudio anterior profundizando en las posibles integraciones de modelos de información, separándolas en cuatro aspectos que se derivan del análisis del problema según puntos de vista diferentes:

- Filosofía. Según el nivel de abstracción al que se realice la traducción se llevará a cabo una integración distinta. La Figura 3.6 ilustra este aspecto, siendo semejante al que define CIM, ya explicado en el apartado 3.3.2.

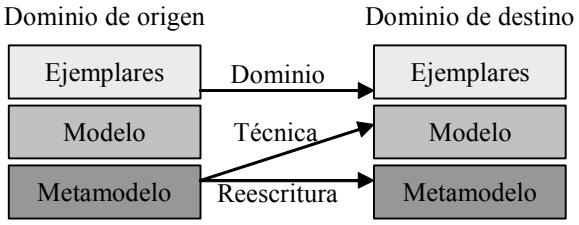

Figura 3.6. Punto de vista desde la filosofía según [Rivière98].

Así, una traducción puede ser:

  - Técnica. Si la correspondencia se hace respectivamente entre modelo y metamodelo, se denomina traducción técnica. Éste podría ser el caso de la MIB de DMI.
  - Reescritura. Si se establece una correspondencia entre metamodelos. Éstas son las habituales traducciones sintácticas que se establecen entre los distintos lenguajes de definición.

    - Dominio. Con correspondencia de ejemplares de modelos, que conservan la semántica de la información definida. A diferencia de las anteriores, éstas son las más extrañas.

- Principio. Según el grado de heterogeneidad entre los modelos de información, la traducción puede ser:
    - Directa. Si se puede realizar mediante un conjunto de reglas de correspondencia entre los modelos origen y destino.
    - Abstracta. Si para realizar la traducción hay que añadir o quitar información de manera manual.
- Modo. Según el tipo de información empleado para posibilitar la interoperabilidad:
    - Estático. para conseguir interoperabilidad sólo se utiliza la información de gestión definida.
    - Dinámico. además de la información se tienen en cuenta cuestiones tales como la identificación de los ejemplares de la información gestionada.
- Organización. Dependiendo de si el mecanismo que se emplea para realizar la traducción es:
    - Independiente. El mecanismo de integración es independiente de los sistemas origen o destino. Esta solución podría ser la de un *proxy* o pasarela, indicada anteriormente.
    - Dependiente. Los propios sistemas origen o destino poseen el mecanismo de traducción. Serían los gestores o agentes multiarquitecturales.

Según esta clasificación se podría decir que las soluciones típicas normalmente se basan en la filosofía de reescritura, el principio de traducción directa, modo dinámico y organización independiente, aunque otras son también posibles.

Si bien este trabajo propone esquemas de interoperabilidad a partir del modelo de información e intuye que parte del problema está en el significado de los conceptos que se describen con dicha información, no llega a proponer ningún método para llevar a cabo una traducción semántica: las experimentaciones en que se basa [Rivière95, Rivière96a, Rivière96b] se quedan en la sintaxis.

3.4.3 Aproximaciones de Ban y Deri

En [Ban97] se propone una aproximación para llevar a cabo la interoperabilidad entre los modelos de gestión integrada de TMN, Internet y CORBA, teniendo en cuenta los metamodelos de cada lenguaje de definición (GDMO, SMI e IDL respectivamente). Para ello, define GOM (*Generic Object Model*, Modelo de Objeto Genérico), un modelo en el que existe un objeto gestionado genérico sobre el que se pueden obtener o modificar

valores. Este objeto es una interfaz IDL de acceso a un adaptador que será el que realice las operaciones. Para traducir la información referida a cada objeto, se sigue el formato (*layout*) de GDMO, SMI e IDL, descomponiendo las definiciones de información en unidades simples que llama metaobjetos, que se incluyen en dicho objeto genérico, enlazados de forma que se pueda volver a obtener la información de cada lenguaje.

Este trabajo es anterior a la especificación final de JIDM. Según su autor, la ventaja de usar GOM frente a la versión entonces existente de JIDM es que no se pierde información en la traducción a IDL. Sin embargo, dado que las traducciones se basan en la descripción de los metamodelos con un modelo de objeto genérico, la solución que propone según la clasificación anterior es de tipo técnico, por lo que no incluye aspectos de semántica en la misma. Es más, dado que la estructura de metaobjetos que se genera para cada lenguaje es diferente, el gestor no sólo tendrá que tener en cuenta los distintos modelos de información, sino que además deberá conocer dicha estructura, con lo que en este sentido, esta solución es incluso menos transparente que la de JIDM.

Por la misma época, en [Deri96] se propone *Liaison*, otra aproximación para integrar los modelos de gestión de Internet y TMN que utiliza en este caso un *proxy* con una interfaz HTTP que maneja objetos Java siguiendo, como en el caso anterior, una traducción técnica. Estos objetos poseen interfaces de acceso a operaciones comunes para obtener o modificar el valor de atributos, así como a otras propias de los metamodelos de SMI y GDMO, que permiten obtener el OID o el tipo de datos tiene un atributo, o invocar una acción a un ejemplar de objeto gestionado.

En [Deri97] ambos autores comparan sus propuestas, que resultan ser bastante similares salvo por las interacciones con HTTP y Java propuestas por Deri. Por ello, las carencias de esta aproximación son similares a las de la anterior. La interfaz para acceder a objetos SMI y GDMO es distinta, por lo que la transparencia de acceso a la información desde el gestor se limita a que el protocolo de transporte es HTTP, y que no es necesario hacer una descomposición de las especificaciones de gestión. Al igual que antes, no sólo no trata el problema de la semántica sino que ni siquiera trata de hacer una correspondencia sintáctica entre modelos. Además, este trabajo se limita al acceso a SNMP y GDMO, por lo que abarca menos dominios de gestión que el anterior.

3.4.4 Propuestas de Neumair y Keller

Casi al mismo tiempo, y partiendo de los trabajos de Kalyanasundaram y Sethi, en [Neumair98] y [Keller99] se introduce el concepto de gestión paraguas (*umbrella management*), que propone el uso de un único modelo de información y una interfaz de acceso única a través de la cual se pueda comunicar con cualquier recurso de la empresa, perteneciente a un dominio concreto como Internet o TMN (ver Figura 3.7). Esta aproximación se debe a que para cada recurso hay que utilizar el modelo de gestión que mejor se adapte. No es posible poner un agente CMIP en un concentrador o *hub*, y

tampoco tiene sentido utilizar DMI, pues los datos de gestión útiles a este recurso normalmente se obtienen a través de SNMP. La arquitectura de WBEM se podría considerar una implementación de este concepto, si bien ambos artículos utilizan CORBA como la plataforma que actúa de paraguas.

En [Neumair98], esta gestión paraguas se destina a la gestión de aplicaciones, teniendo en cuenta las perspectivas del modelo de referencia de ODP (*Open Distributed Processing,* Procesamiento Distribuido y Abierto) [ITUT97d]. Las definiciones de objetos gestionados no se corresponden con un modelo de gestión concreto, sino que se especifican basándose en los conceptos de ODP que sean relevantes, siendo las perspectivas computacional y de ingeniería las más importantes para la gestión. En [Keller99] se aplica la gestión paraguas para gestionar los propios sistemas de gestión. Esto es necesario en un escenario con múltiples proveedores de servicios que necesiten interoperar. Para definir la información a gestionar también se utiliza la aproximación de emplear las perspectivas de ODP.

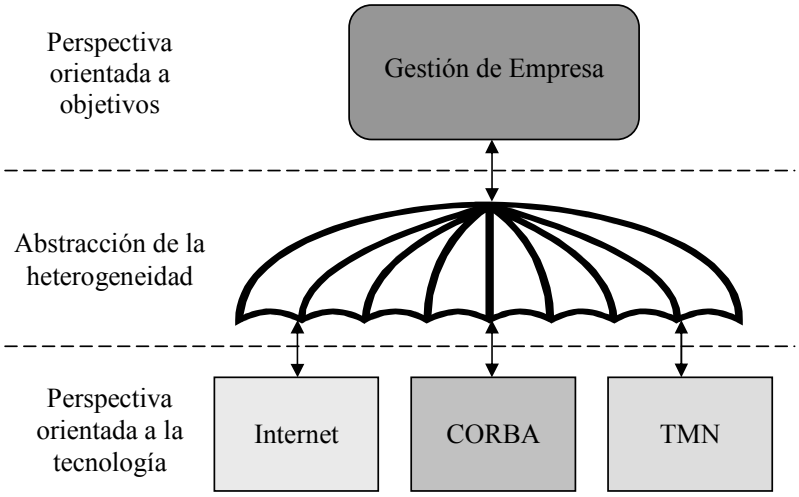

Figura 3.7. Gestión paraguas según [Keller99].

El punto más positivo de estos trabajos es que se trata la gestión desde múltiples niveles y perspectivas, y de manera independiente a los modelos de gestión existentes. Sin embargo, estas soluciones son aplicaciones de casos concretos, con lo que no se resuelve el problema de la interoperabilidad para un caso general, ni tampoco se explica cómo llevar a cabo la gestión paraguas. Sólo proponen la forma de definir la información de gestión para casos que se deban integrar en un escenario en el que la información esté en una perspectiva distinta a la de la forma de obtenerla.

3.4.5 Trabajos relacionados con WBEM

En lo que se refiere a definir los mecanismos de interoperabilidad entre WBEM y los distintos dominios de gestión, existen varios trabajos. En primer lugar, se encuentran aquéllos relacionados con la traducción entre CIM y otros lenguajes de definición de información de gestión. Por ejemplo, en [Festor99] se han propuesto varias reglas de correspondencia entre CIM y GDMO, basándose en las ideas de IIMC y JIDM. También

en [Pablos01] se hace uso de las ideas de JIDM para traducir a IDL información definida en CIM y poder acceder a objetos CORBA desde un gestor WBEM. Otra aproximación similar es la de [Bénech00], si bien para ello define estructuras genéricas en IDL que representen clases CIM, pudiéndose entender como una traducción técnica. En lo que se refiere a Internet, existen distintos sistemas operativos que proporcionan herramientas para acceder a SNMP mediante un proveedor [Microsoft02, Sun02], pero que son mutuamente incompatibles. Si en el caso de Microsoft se define un espacio de nombres distinto para cada agente SNMP que incluye las MIBs que mantiene, Sun utiliza un mismo espacio de nombres en el que hay tantos objetos de cierta clase como agentes existan, y otro objeto que relaciona al primero con las distintas MIBs. Además, la traducción de las MIBs en SMI a MOF/CIM es también diferente en ambos casos.

De cualquier manera, en todos estos estudios sobre la implementación de proveedores para gestionar distintos dominios desde WBEM, la traducción que se realiza es como mucho mediante reescritura por lo que en definitiva sólo consiguen una integración sintáctica.

Tras esto se llega a los últimos trabajos realizados sobre este tema, que ya tratan de atacar el problema de la semántica de la información entre dominios de gestión. En [MartinFlatin01] se identifica un conjunto de problemas, proponiendo un modelo de dos capas según se muestra en la Figura 3.8: una universal y otra dependiente de cada dominio, aunque no explica cómo llevar a cabo esta correspondencia.

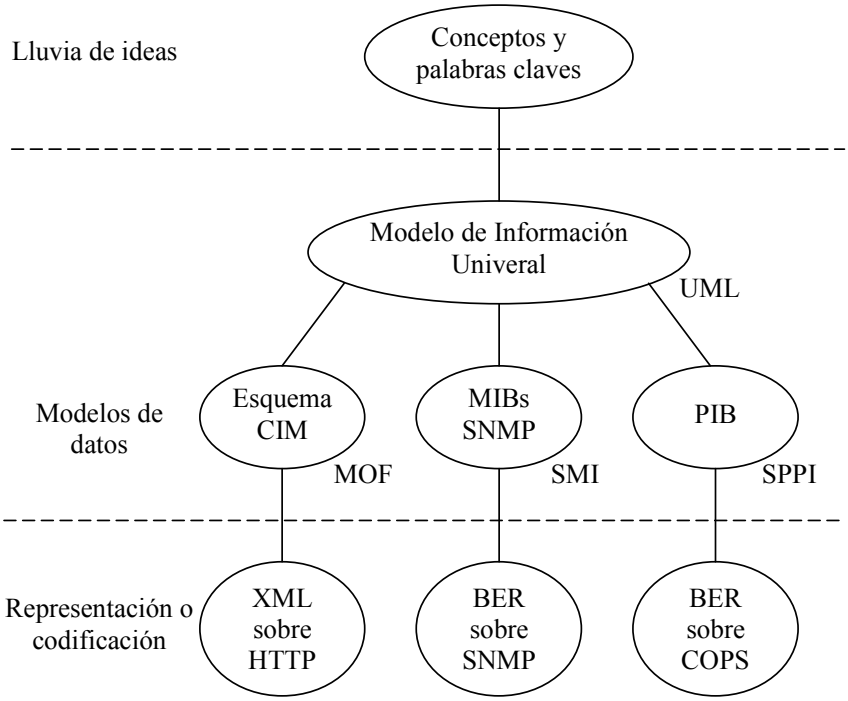

Figura 3.8. Modelo de información en dos capas según [MartinFlatin01].

Más adelante, y como parte del mismo trabajo [Schott02] indica cómo conseguir una correspondencia de términos de MIBs escritas en SMI con otros términos incluidos en el modelo de CIM. Otra desventaja de estos trabajos es que únicamente inciden en la

definición de correspondencias directas entre distintos modelos y no proponen ningún método que permita realizar el alineamiento de la información definida de manera rápida.

3.5 Conclusiones

A lo largo de diversas secciones este capítulo ha presentado las distintas propuestas y aproximaciones que existen para establecer mecanismos de interoperabilidad entre los distintos modelos de información de gestión.

Para tratar con la heterogeneidad de dichos modelos, los organismos de estandarización han propuesto distintos métodos en IIMC, DMI-SNMP y JIDM. En todos ellos se definen mecanismos para traducir tanto las interacciones de los protocolos de intercambio como las especificaciones de información definidas en los distintos lenguajes existentes. Sin embargo, la integración que se consigue es parcial, puesto que dichos mecanismos no están definidos para conseguir que todos los dominios de gestión existentes interactúen y además, las traducciones de información se realizan únicamente de manera sintáctica.

La arquitectura conocida como WBEM trata de solventar estos problemas definiendo un nuevo modelo de gestión integrada pensado para poder gestionar todos los recursos de una empresa independientemente de la tecnología con que se acceda a ellos. Así, adapta HTTP como protocolo de acceso a un sistema que actúe de intermediario, utilizando XML para la codificación, y define CIM como un modelo de información común en el que tenga cabida toda la información especificada con anterioridad. WBEM soluciona alguno de los problemas existentes en las aproximaciones anteriores. Sin embargo, siguen existiendo algunos problemas relativos a la forma en que se maneja la información. Aunque se introduce la posibilidad de las traducciones de dominio o semánticas, no se indica cómo llevarlas a cabo.

Y por último pero no menos importante, varios grupos de investigación también han realizado distintas propuestas acerca de mecanismos de interoperabilidad. Kalyanasundaram y Sethi proponen diversas posibilidades para las interacciones entre gestores y agentes de distinto dominio. El grupo de Redes y Servicios del IRIT define varias perspectivas de integración. Ban y Deri proponen dos alternativas para la interoperabilidad entre la gestión OSI e Internet, en el primer caso, con CORBA, y en el segundo, con tecnologías Web. Neumair y Keller, desde la perspectiva de la gestión paraguas, abstraen el modelo de información a utilizar a través de las perspectivas de ODP. Finalmente, hay distintos trabajos que han tratado de formalizar la forma en que se integran los distintos dominios con WBEM. Sin embargo, ninguna de estas aproximaciones consigue atacar la interoperabilidad semántica de una manera efectiva.

La Tabla 3.1 presenta un resumen de dichos trabajos, indicando a qué dominios de gestión se aplican, y los problemas que estudian. Cuando las soluciones que se obtienen son parcialmente válidas se ha incluido la palabra “parcial”, y “sí” si son totalmente válidas. La palabra “no” se incluye cuando no se ha estudiado el problema.

A pesar de la existencia de todos los mecanismos aquí presentados, que han tratado de alcanzar una vista unificada de todos los dominios de gestión, hasta la fecha sólo se han aplicado como mucho traducciones sintácticas entre los diversos lenguajes de definición de información. Esto implica que si un mismo concepto se representa en dos o más dominios de gestión, es posible una traducción directa entre las estructuras que lo definen, pero no según su significado. Éste es un problema si se quiere realizar una gestión unificada entre distintos dominios. Para resolver esta cuestión es necesario alcanzar una interoperabilidad semántica.

Tabla 3.1. Relación de aproximaciones propuestas.

Solución propuesta	**Dominios de gestión implicados**	**Problemas que estudia**		
		Protocolo	*Lenguaje*	*Modelo*
TMN	Independiente	Parcial	Parcial	Parcial
IIMC	OSI-SM, SNMP	Sí	Sí	No
DMI	DMI, SNMP	Sí	Sí	No
JIDM	OSI-SM, CORBA, SNMP	Sí	Sí	No
WBEM	Todos	Sí	Sí	Parcial
Kalyanasundaram	OSI-SM, SNMP	Sí	Sí	No
IRIT	Independiente	Parcial	Sí	Parcial
Ban	OSI-SM, SNMP, CORBA	Sí	Sí	No
Deri	OSI-SM, SNMP, Web	Sí	Sí	No
Neumair y Keller	Gestión paraguas	No	No	Parcial
Interoperabilidad con WBEM	OSI-SM, SNMP, CORBA, WBEM	Sí	Sí	No
Martin Flatin	WBEM	No	Sí	Parcial

En el siguiente capítulo, y para tratar de solventar estas cuestiones, se presenta la técnica de representación del conocimiento conocida como ontología. Dicha técnica actualmente proporciona mecanismos que se pueden aprovechar para facilitar la interoperabilidad de la

información, mediante su tratamiento a partir de su semántica. Las ontologías se han aplicado de manera exitosa para solventar problemas semánticos similares en otros dominios como la Web Semántica [BernersLee01], donde estas técnicas basadas en el conocimiento proporcionan a las páginas y servicios Web la semántica de la que normalmente carecen.

Capítulo 4 Las Ontologías como Técnicas de Representación del Conocimiento

4.1 Introducción

El capítulo anterior presentaba el problema de interoperabilidad debido a la heterogeneidad de modelos de gestión integrada existentes. Cada modelo define un mecanismo de intercambio de información, un lenguaje de definición y un conjunto de modelos de información que son incompatibles entre sí. Además, las aproximaciones actuales que definen mecanismos de interoperabilidad son muy rudimentarias. La arquitectura propuesta por WBEM puede proporcionar mecanismos de integración válidos, pero presenta problemas para atacar la interoperabilidad de la información teniendo en cuenta la semántica de la misma.

Esta cuestión se está resolviendo en el mundo de la gestión del conocimiento e inteligencia artificial mediante el uso de ontologías [Fernández01], que últimamente han ganado relevancia con la aparición de la Web Semántica [BernersLee01, Cherry02]. La mayoría de los contenidos de la Web están actualmente diseñados para ser leídos por humanos, y no por programas que traten de extraer su significado. Los ordenadores pueden analizar las páginas Web para mostrar su diseño, pero no tienen una manera fiable de procesar su semántica. Con la Web Semántica se pretende que un agente inteligente pueda obtener la información que se le haya solicitado recorriendo distintas páginas, de las que extraerá su significado a partir de las ontologías que maneje. Al respecto de esta cuestión, se están realizando actividades de normalización en el seno de varios proyectos, tales como el proyecto OntoWeb[1], del programa europeo IST (*Information Society Technologies*, Tecnologías de la Sociedad de la Información); el proyecto DAML[2] (*DARPA Agent Markup Language*, Lenguaje de Marcas de Agentes de DARPA), de la agencia de investigación de defensa de Estados Unidos (DARPA, *Defense Advanced Research Projects Agency*); y la Web SemanticWeb.org[3], iniciativa conjunta de varias universidades europeas y norteamericanas.

---

[1] http://www.ontoweb.org, que redirige a http://ontoweb.aifb.uni-karlsruhe.de/

[2] http://www.daml.org/

[3] http://www.semanticweb.org/

En la Web Semántica se produce un problema similar al que se ha presentado con los modelos de gestión de red integrada al tratar de componer servicios Web. En [Castillo02] se denomina el problema ontológico: es la situación que se presenta cuando dos entidades software, que deben intercambiar información, estructuran de forma diferente la información que poseen. En el campo de los agentes inteligentes, es la situación que se presenta cuando estas entidades estructuran el conocimiento que deben intercambiar de forma diferente. Por ello, las soluciones que se aplican a esta cuestión pueden valer también para el caso de la integración de información de gestión, aunque no se conocen propuestas que hasta la fecha hayan realizado ningún estudio en este sentido.

El presente capítulo presenta la técnica de representación del conocimiento conocida como ontología. Para ello, en la siguiente sección se definirá qué es una ontología y cuál es su relación con los modelos de información de gestión existentes. Tras esto, se presentarán distintos tipos de lenguajes de representación del conocimiento. A continuación se expondrán distintos trabajos realizados para la integración de ontologías, centrándose más tarde en las ontologías de correspondencia y las metodologías que existen para llevar a cabo esta integración.

4.2 Las ontologías y su relación con los modelos de información de gestión

Para comprender cómo se pueden aplicar las ontologías al campo de estudio de la integración de información de distintos dominios de gestión es necesario explicar previamente en qué consisten las ontologías. Este término, tomado de la Filosofía, es la *parte de la metafísica que trata del ser en general y de sus propiedades trascendentales*. Su definición en el campo de la Inteligencia Artificial es algo difusa, puesto que los expertos en el tema no han acordado una definición conjunta y existen múltiples referencias con definiciones complementarias. Quizás la que las describe de manera más completa es la contenida en [Studer98], que explica una ontología como *una especificación explícita y formal de una conceptualización compartida*. Esta definición se puede entender de la siguiente manera:

- Es explícita porque define los conceptos, propiedades, relaciones, funciones, axiomas y restricciones que la componen.
- Es formal porque es interpretable por máquinas.
- Es una conceptualización porque es un modelo abstracto y vista simplificada de fenómenos del dominio que se quiere representar.
- Finalmente, es compartida porque la información ha sido consensuada previamente entre distintos grupos de expertos.

De forma breve, se puede decir que una ontología es la definición de un conjunto de conceptos, su taxonomía, interrelación y las reglas que gobiernan dichos conceptos. Una base de conocimiento toma estos conceptos para representar aserciones sobre un mundo real o hipotético [Swartout99].

Las ontologías se pueden clasificar en ligeras y pesadas. Las primeras son aquéllas que son capaces de modelar la información referida a un dominio, pero que no incluyen axiomas ni restricciones, por lo que razonar con ellas será una cuestión complicada. Las ontologías pesadas incluyen todos los elementos que permiten utilizarlas para realizar inferencias sobre el conocimiento que contienen. De esta forma, los modelos de información de gestión existentes se podrían entender como ontologías ligeras: modelos como los esquemas de CIM, la MIB-II de SNMP o la recomendación M.3100 de TMN definen la información del dominio de gestión con una sintaxis normalizada y han sido consensuadas en grupos de trabajo. Sin embargo, su semántica está limitada y es mejorable. No se pueden considerar ontologías pesadas al no incorporar las construcciones que permiten inferir conocimiento basándose en el ya existente [LópezDeVergara02].

Las ontologías se crearon para compartir y reutilizar el conocimiento [Neches91]. Con ello se trataban de solventar los siguientes impedimentos encontrados a la hora de que interoperaran sistemas inteligentes:

1. Representaciones heterogéneas. Existen múltiples aproximaciones para representar el conocimiento, no pudiéndose representar siempre en un formalismo el conocimiento que está representado con otro formalismo. Una cuestión similar ocurre con los distintos lenguajes de información de gestión, donde no todos son orientados a objetos.
2. Dialectos en familias de lenguajes. Dentro de una misma familia de formalismos de representación del conocimiento, puede ser complicado compartir conocimiento entre dialectos. Por ejemplo, dentro de los lenguajes de información de gestión orientados a objetos, no todos permiten la herencia múltiple.
3. Falta de convenciones de comunicación. No existe un protocolo que especifique cómo distintos sistemas pueden consultarse el conocimiento que poseen. En este punto se ha trabajado mucho en el campo de la gestión de red, habiéndose descrito varios ejemplos en el capítulo anterior.
4. Desemparejamiento de modelos en el nivel de conocimiento. Aunque los problemas del nivel de lenguaje se resuelvan, sigue siendo difícil combinar dos bases de conocimiento. Estas barreras aparecen cuando se usan diferentes términos primitivos para organizarlas. Este problema también se describía para el caso de la gestión de red, donde cada dominio de gestión ha definido su modelo de información de manera totalmente independiente a los otros dominios.

Según [O'Leary98], y en línea con su utilidad de compartir y reutilizar conocimientos, las características deseables para una ontología son, entre otras, que sea descomponible, fácil de entender, extensible, fácil de mantener, modular, traducible y con una buena relación coste/beneficio. Estas características también son deseables en muchos casos para la información de gestión de red, por lo que la mayoría de los modelos tratan de ser modulares y extensibles.

Para facilitar la reutilización de ontologías, normalmente se usan bibliotecas para su definición, siguiendo una estructura piramidal, en la que en la base de la pirámide están las ontologías más generales y por tanto más reutilizables, y en la cúspide, las más específicas y usables y por tanto, menos reutilizables [Gómez99a, Gómez99b]. Así, se pueden encontrar las siguientes bibliotecas de ontologías:

- Ontologías de representación. Capturan las primitivas de representación que se usan para formalizar el conocimiento según distintos paradigmas de representación.
- Ontologías generales comunes. Incluyen vocabulario relativo a cosas, eventos, tiempo, espacio, causalidad, comportamiento, funciones, etc.
- Ontologías de dominio genérico. También llamadas ontologías nucleares, son reutilizables entre varios dominios.
- Ontologías de dominio. Son reutilizables para un dominio dado, proporcionando los conceptos y sus relaciones, sobre las actividades que toman lugar en este dominio, y sobre las teorías y principios elementales que gobiernan el dominio.
- Ontologías de aplicación. Contienen el conocimiento necesario para modelar un dominio particular.

También se podrán encontrar estos tres últimos niveles referidos al vocabulario de términos que se usan para la resolución de problemas (PSM, *Problem Solving Methods*):

- Ontologías de tareas genéricas. Referida a tareas independientes del dominio.
- Ontologías de tareas de dominio. Son reutilizables en un dominio concreto.
- Ontologías de tareas de aplicación. Sólo se pueden usar para la aplicación para la que se definan.

Al igual que antes, también es posible realizar cierta correspondencia entre las bibliotecas de ontologías y los esquemas de CIM [LópezDeVergara02, LópezDeVergara03a], según se muestra en la Figura 4.1. En CIM se ha dispuesto una estructura similar en la que únicamente falta el nivel de una ontología general común:

- El metaesquema de CIM define las construcciones para definir distintos conceptos siguiendo el paradigma de orientación a objetos.
- El modelo nuclear de CIM trata los conceptos generales aplicables a todos los dominios de la gestión.

- Los modelos comunes extienden los conceptos definidos en el modelo nuclear para cada subdominio de gestión (sistema, red, aplicaciones…)
- Los esquemas de extensión toman los conceptos de los modelos comunes y los especializan para los casos concretos de cada fabricante.

Por otro lado, CIM carece de las ontologías de tareas que permitan manejar ese conocimiento para la resolución de problemas.

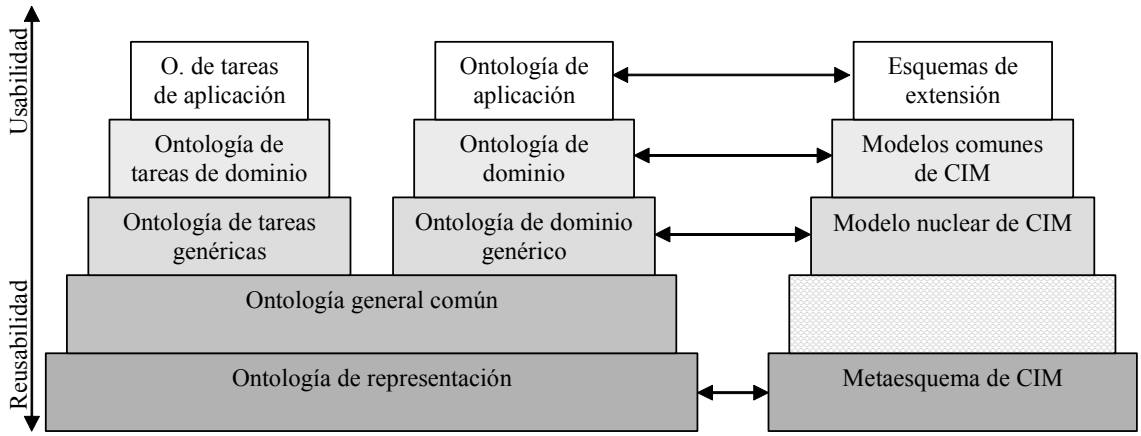

Figura 4.1. Correspondencia entre la arquitectura de CIM y la arquitectura de ontologías.

4.3 Lenguajes de definición de Ontologías

Dependiendo del paradigma empleado para la representación del conocimiento existen múltiples terminologías [Kiryakov01]. Aunque éstas sean distintas, se pueden hacer correspondencias directas entre la mayoría de ellas. Algunos de los paradigmas de representación del conocimiento son:

- Redes semánticas. Éstas tratan normalmente con conceptos, ejemplares, relaciones y propiedades. Los conceptos expresan cualquier clase de fenómeno estático y autónomo. Las entidades que pertenezcan a la interpretación del concepto son ejemplares. Las propiedades son características de las entidades.
- Representación basada en marcos. Usan clases, ejemplares, ranuras (*slots*) y facetas. Comparando con las redes semánticas, las clases se corresponden con los conceptos, y los ejemplares son exactamente lo mismo en ambos casos. Las ranuras son como las propiedades. Finalmente, las facetas son características de las ranuras.
- Lógica descriptiva. Usa conceptos, roles e individuos. En este caso, los roles o papeles se corresponden con propiedades, y los individuos, con ejemplares.
- Orientación a objetos. Finalmente, este paradigma de representación emplea clases, objetos y atributos. En este caso, los objetos son ejemplares de clases, y los atributos, sus propiedades.

Partiendo de estos paradigmas, y según se expone en [OntoWeb01, OntoWeb02a, OntoWeb02b], en la última década se ha definido un gran número de lenguajes de definición de ontologías. Por ejemplo: Ontolingua, Loom, F-Logic, etc. Muchos de éstos ya se utilizaban para representar conocimiento en aplicaciones, otros se adaptaron a partir de lenguajes ya existentes, y otros se crearon específicamente para la representación de ontologías. Este conjunto de lenguajes, llamados tradicionales, usan una sintaxis en texto plano que en muchos casos es similar a LISP (*LISt Processor*, Procesador de Listas), desarrollado para programar aplicaciones de inteligencia artificial.

Posteriormente, en el contexto de la Web Semántica se han desarrollado otros lenguajes. Por ejemplo, RDF (*Resource Description Framework*, Marco de Descripción de Recursos), RDFS (*RDF Schema*, Esquema RDF), OIL (*Ontology Inference Layer*, Capa de Inferencias de Ontologías también *Ontology Interchange Language*, Lenguaje de Intercambio de Ontologías), DAML+OIL (fusión de los lenguajes DAML y OIL) y últimamente OWL (*Web Ontology Language*, Lenguaje de Ontologías para la Web). Estos lenguajes basan su sintaxis en XML, ampliamente adoptado para el intercambio de información en la Web.

Por otro lado, también existen varias propuestas que pretenden aprovechar las características de UML para la especificación de ontologías.

A continuación se presenta un resumen de las principales características de los lenguajes más conocidos.

4.3.1 Lenguajes tradicionales

4.3.1.1 Ontolingua y OCML

Ontolingua [Gruber93] fue desarrollado en el *Knowledge Systems Laboratory* (KSL, Laboratorio de Sistemas de Conocimiento) de la Universidad de Stanford. Este lenguaje se basa en KIF (*Knowledge Interchange Format*, Formato de Intercambio de Conocimiento), sobre el que se ha definido la Ontología de Marcos (*Frame Ontology*), que permite definir ontologías siguiendo el paradigma de marcos. KIF posee una gran expresividad, lo que permite la representación de conceptos, taxonomías, relaciones n-arias, funciones, axiomas, ejemplares y procedimientos. Sin embargo, también es su mayor defecto, al ser un lenguaje muy complejo. La importancia de Ontolingua radica en la biblioteca de ontologías escrita en este lenguaje, disponible en el servidor del KSL[4].

OCML (*Operational Conceptual Modeling Language*, Lenguaje de Modelado Conceptual y Operacional) [Motta99] es un lenguaje desarrollado en la Open University que se ha utilizado en múltiples proyectos. Es muy similar a Ontolingua, aunque añade algunos

[4] http://ontolingua.stanford.edu

componentes adicionales tales como reglas deductivas y de producción y definiciones operacionales para funciones. Por ello se le considera el Ontolingua operacional.

4.3.1.2 OKBC

OKBC (*Open Knowledge Base Connectivity*, Conectividad de Bases de Conocimiento Abiertas) [Chaudhri98] no es propiamente un lenguaje, sino un protocolo para el intercambio de conocimiento, que en principio es independiente del lenguaje de representación que utilizan los sistemas que intercambian ese conocimiento. OKBC proporciona un conjunto de construcciones típicas de los lenguajes basados en marcos: constantes, marcos (*frames*), ranuras (*slots*), facetas, clases, individuos y bases de conocimiento, si bien es lo suficientemente general para que dos sistemas que poseen distintos paradigmas de representación puedan intercambiar conocimiento.

4.3.1.3 F-Logic

F-Logic (*Frame Logic*, Lógica de Marcos) [Kifer95] fue desarrollado por la Universidad de Karlsruhe. Integra el paradigma de marcos y el cálculo de predicados de primer orden. Incluye construcciones de orientación a objetos y marcos, proporcionando identidad de objetos, objetos complejos, herencia, tipos polimórficos, métodos de consulta, etc. Con ello permite representar conceptos, taxonomías, relaciones binarias, funciones, ejemplares, axiomas y reglas deductivas. Asimismo, posee un motor de inferencias que permite comprobar restricciones y deducir conocimiento.

4.3.1.4 Loom

Loom [MacGregor91] es un lenguaje de programación de alto nivel, así como un entorno de desarrollo de sistemas expertos y otras aplicaciones de inteligencia artificial. Al ser un lenguaje descendiente de KL-ONE, está basado en el paradigma de la lógica descriptiva, alcanzando una integración entre los paradigmas basados en reglas y marcos. Hay que notar que la lógica descriptiva difiere de las aproximaciones basadas en marcos, pues trata de clasificar los conceptos definidos según las restricciones que se han impuesto sobre ellos. Loom permite definir conceptos, taxonomías, relaciones n-arias, funciones, axiomas y reglas de producción. Razonando con ontologías definidas en este lenguaje se pueden realizar clasificaciones automáticas, comprobación de restricciones y ejecución de las reglas de producción.

4.3.1.5 Comparación

En sucesivos trabajos del Departamento de Inteligencia Artificial[5] de la Universidad Politécnica de Madrid se han realizado comparaciones de los lenguajes tradicionales de

[5] http://www.dia.fi.upm.es

definición de ontologías. El propósito inicial [Corcho99a, Corcho99b] era permitir la traducción entre F-Logic y Ontolingua, tratando de no perder el significado de la información contenida en las ontologías. Para ello, han analizado las construcciones que poseen dichos lenguajes, tratando de encontrar correspondencias entre las mismas. Posteriormente, [Corcho00] utiliza este marco para comparar el conjunto de lenguajes existentes en ese momento. La comparación tiene en cuenta la capacidad para definir conceptos, atributos, facetas, taxonomías, reglas, funciones, axiomas, ejemplares, reglas de producción y mecanismos de inferencia. Según la comparativa que realiza, Loom resulta ser el lenguaje con mayor capacidad expresiva de los anteriores, pues es el que posee el mayor número de las características estudiadas.

Esta tesis aprovecha el marco propuesto en dicho artículo para comparar en la sección 5.2 los lenguajes de definición de información de gestión en términos similares.

4.3.2 Lenguajes basados en XML

4.3.2.1 Lenguajes independientes de RDF

Antes de que se propusiera aplicar RDF a la definición de ontologías, ha habido otras aproximaciones que utilizaban lenguajes de marcas.

SHOE (*Simple HTML Ontology Extensions*, Extensiones Simples de HTML para Ontologías) [Heflin99] define unas extensiones a HTML (*Hyper-Text Markup Language*, Lenguaje de Marcas de Hipertexto), que es el lenguaje que se emplea para escribir las páginas Web. Por tanto, no emplea XML, aunque su definición en 1996 explica este hecho. Para añadir semántica a las páginas Web, añade marcas que pueden ser de construcción de ontologías y de anotación de documentos. SHOE permite representar conceptos, su taxonomía, relaciones n-arias, ejemplares y reglas deductivas que se pueden usar en un motor de inferencias.

XOL (*XML-based Ontology exchange Language*, Lenguaje de Intercambio de Ontologías Basado en XML) [Karp99] definido por los autores de OKBC, trata de especificar en XML un subconjunto de las primitivas de este protocolo. Es un lenguaje muy restringido, pues sólo permite definir conceptos, taxonomías y relaciones binarias, y no incluye mecanismos de inferencia, al estar pensado únicamente para el intercambio de ontologías.

OML (*Ontology Markup Language*, Lenguaje de Marcas para Ontologías) [Kent00] se basa en lógica descriptiva y grafos conceptuales, y permite representar conceptos organizados en taxonomías, relaciones y axiomas usando lógica de primer orden. En su primer desarrollo traducía a XML los formalismos de SHOE. Posteriormente ha tratado de acercarse a RDFS.

4.3.2.2 RDF y RDFS

RDF[6] es un marco definido por el Consorcio de la World Wide Web (W3C, *World Wide Web Consortium*) que incluye un lenguaje [Lassila99] con sintaxis XML para integrar una variedad de aplicaciones desde catálogos de bibliotecas y directorios de la Web a la agregación de noticias, software, y colecciones personales de música, fotos y eventos. Las especificaciones del marco RDF proporcionan un sistema de ontología ligera para intercambiar el conocimiento en la Web.

RDF no proporciona mecanismos para describir propiedades, ni la relación que existe entre ellas y otros recursos, siendo éste el papel de RDFS [Brickley03], que describe cómo usar RDF para definir vocabularios, creando para esto un conjunto de términos. Con este lenguaje de descripción de vocabularios se pueden definir clases y propiedades, así como jerarquías y restricciones de tipo.

También existe una propuesta [Goad01] para añadir computación a RDF, aunque realmente sólo introduce código entre marcas.

4.3.2.3 OIL, DAML+OIL y OWL

OIL [Fensel00, Fensel01] es una propuesta para representar ontologías en la Web, describiendo un estándar de descripción e intercambio. Para ello combina las primitivas de los lenguajes basados en marcos y la semántica formal y los servicios de razonamiento de la lógica descriptiva. Es compatible con RDFS e incluye una semántica precisa para describir el significado de términos.

OIL define un lenguaje con una aproximación de capas en la que cada capa añade funcionalidad y complejidad a la capa anterior. De esta forma, los agentes que sólo puedan procesar la capa inferior, pueden entender parcialmente las ontologías que se expresan usando capas superiores. La Figura 4.2 muestra esta cuestión, así como su relación con RDFS.

DAML+OIL [McGuinness02, Horrocks02] surge como resultado de los trabajos de OIL, desarrollado en el contexto de un proyecto europeo, y DAML (*DARPA Agent Markup Language*, Lenguaje de Marcas de Agentes de DARPA), fruto de un proyecto estadounidense, para crear un estándar conjunto para la Web Semántica, habiendo sido adoptado por el W3C [Connolly01]. La última versión incluye el tratamiento de tipos de datos, utilizándose los definidos en XSD (*XML Schema Data types*, Tipos de Datos de Esquemas XML) [Biron01]. Asimismo, se ha realizado una definición axiomática de su semántica en KIF [Fikes01].

[6] http://www.w3.org/RDF/

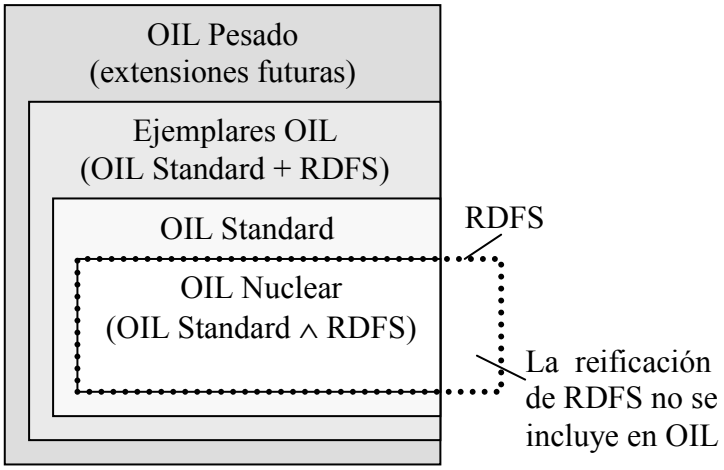

Figura 4.2. Lenguaje en capas de OIL según [Fensel01].

OWL [Dean02] es la última propuesta del W3C como lenguaje de definición de ontologías para la Web Semántica, y actualmente se encuentra en proceso de desarrollo. Este lenguaje tiene pequeñas diferencias con DAML+OIL, siendo la mayor de ellas la existencia de OWL Lite (OWL Ligero), un subconjunto de OWL más fácil de soportar y menos complejo que DAML+OIL.

4.3.2.4 Comparación

En lo que se refiere a lenguajes de definición de ontologías basados en XML, en [Corcho00] también se hace una comparación de SHOE, XOL, RDFS y OIL, en paralelo con los lenguajes tradicionales anteriormente comentados. Este estudio está particularizado para lenguajes de la Web Semántica en [Gómez02], donde se comparan SHOE, XOL, OML, RDFS, OIL y DAML+OIL. Esta comparativa muestra que OIL y DAML+OIL son los que poseen una mayor capacidad expresiva en lo que se refiere a taxonomías, siendo más o menos similares al resto de lenguajes en cuanto a definición de conceptos, relaciones, axiomas y ejemplares.

En [Pan01] se hace una comparación de RDF, RDFS y DAML+OIL desde su metamodelo, explicando los problemas que presentan estos lenguajes debido a la forma en que se entremezclan los distintos niveles de modelado. Para corregir este problema define RDFSFA (*RDFS Fixed metamodeling Architecture*, Arquitectura de Metamodelado Fijo de RDFS), basado en la arquitectura de 4 niveles que se emplea en UML.

4.3.3 UML como lenguaje de definición de ontologías

La aparición del Lenguaje de Modelado Unificado [OMG01b] en el campo de la ingeniería del software ha supuesto muchos beneficios, al haberse normalizado una misma notación y un conjunto de diagramas para los procesos de modelado de programas informáticos. Por este motivo, UML ha tenido una gran acogida, teniendo una amplia comunidad de usuarios. Por otro lado, los lenguajes estudiados en los apartados anteriores son escasamente conocidos fuera del ámbito de la Inteligencia Artificial. De hecho, en

[Wang02] se propone una integración entre el modelado de software y de conocimiento utilizando UML. De esta manera, UML se presenta como una posible alternativa a los lenguajes de definición de ontologías [Cranefield99].

Un posible defecto de UML es que en su momento no se definió inicialmente con un lenguaje formal, puesto que las reglas que definían su metamodelo estaban escritas en inglés. Para solventar esta cuestión, en [Evans99] se presentan los resultados del grupo de trabajo Precise UML[7] (pUML, UML preciso), que tratan de redefinir la semántica del metamodelo de UML utilizando OCL (*Object Constraint Language*, Lenguaje de Restricciones de Objetos) [OMG01c], que es el lenguaje utilizado en UML para definir restricciones sobre los modelos definidos. Al usar OCL para definir las reglas del metamodelo de UML se consigue:

- Clarificar y hacer precisa la semántica de UML.
- Razonar con las propiedades de los modelos UML.
- Verificar la corrección de los diseños UML.
- Construir herramientas que soporten la aplicación rigurosa de UML.

Las ideas propuestas por pUML se han ido incorporado a la especificación del lenguaje, ocurriendo que gran parte de la semántica del metamodelo de UML se define en la actualidad utilizando OCL.

Por tanto, y según esto, UML puede ser un buen lenguaje para modelar ontologías. Permite definir clases, atributos, jerarquías de herencia, ejemplares, y el lenguaje OCL se puede utilizar también para definir reglas de inferencia, como se propone en [Cranefield01]. Además, UML tiene otro conjunto de características que lo hacen una buena elección [Kogut02]:

- UML es el resultado de muchos años de experiencia en el análisis y diseño de software por varias compañías.
- UML está ampliamente adoptado por la industria y enseñado en muchas universidades y cursos de formación.
- UML está soportado por herramientas CASE (*Computer Aided Software Engineering*, Ingeniería del Software Asistida por Ordenador) ya maduras. Por tanto, se pueden aprovechar estas interfaces de usuario y tecnologías de comprobación de consistencia como herramientas de ontologías.
- UML es un estándar abierto mantenido por el Object Management Group (OMG). Hay un proceso sistemático de evolución de UML de acceso público.

[7] http://www.puml.org/, que redirige a http://www.cs.york.ac.uk/puml/

- UML tiene un metamodelo documentado semánticamente. Restricciones explícitas guían el desarrollo de modelos UML bien definidos.
- UML tiene mecanismos de extensión. Se pueden definir subclases de elementos de UML (estereotipos) y se pueden añadir nuevas propiedades a los elementos UML (valores marcados). Se pueden aplicar restricciones definidas por los usuarios a los elementos de UML.
- UML se ha utilizado de manera efectiva para definir DTDs (*Document Type Definitions*, Definiciones de Tipos de Documento) y esquemas XML. Los lenguajes de ontologías basados en XML son una extensión de estos DTDs y esquemas.
- Se ha definido una sintaxis XML neutra para UML conocida como XMI (*XML Metadata Interchange*, Intercambio de Metadatos en XML). Esto permite una integración más fácil entre distintas herramientas.

El uso de diagramas de clases UML para definir los esquemas CIM vuelve a sugerir nuevamente la comparación entre CIM y los lenguajes de ontologías, si bien el metamodelo de CIM no es el mismo que el de UML ni está definido de forma semántica en OCL, por lo que esta aseveración no es del todo cierta, como se verá en el 0.

Con todo, existen actualmente múltiples esfuerzos encaminados al uso de UML para la definición de ontologías, y en concreto su integración con DAML+OIL. De esta forma, el proyecto UBOT[8] (*UML Based Ontology Toolset*, Herramientas de Ontologías Basadas en UML), dentro de la iniciativa DAML ha propuesto una extensión de UML para poder representar ontologías escritas en DAML+OIL, teniendo en cuenta las diferentes cuestiones que se pueden plantear [Backlawski01]. DUET[9] (*DAML UML Enhanced Tool*, Herramienta UML Mejorada para DAML) es otro desarrollo que se ha sumado a la iniciativa de soportar UML para modelar ontologías. Para esto se aportan unos complementos (*plug-ins*) implementados en el proyecto CODIP (*Components for Ontology Driven Information Push*, Componentes para Promover la Información Mediante Ontologías), también dentro de la iniciativa DAML, para poder cargar archivos DAML+OIL en herramientas CASE como Rational Rose[10] y ArgoUML[11]. La herramienta de diseño de ontologías Protégé-2000[12] también posee un complemento[13] para poder cargar y guardar archivos con sintaxis XMI, integrándose de esta manera con UML.

[8] http://ubot.lockheedmartin.com/

[9] http://codip.grci.com/Tools/Tools.html

[10] http://www.rational.com/products/rose/

[11] http://argouml.tigris.org/

[12] http://protege.stanford.edu/

[13] http://protege.stanford.edu/plugins/uml/

Por este conjunto de razones, la representación gráfica que se haga de ontologías en el resto la tesis se hará con diagramas de clases UML. Con esto además se utiliza una misma notación que resulte de fácil comprensión para el lector.

4.4 Integración de ontologías

La existencia de múltiples ontologías definidas en distintos lenguajes ha supuesto el mismo problema de interoperabilidad que el que existe en la gestión de red integrada. Para solventarlo ha habido múltiples propuestas por parte de los distintos grupos de investigación que trabajan en este ámbito. Esta sección introduce el problema de la integración de ontologías y los distintos enfoques que se han planteado para su solución. Cuando se habla de integración de ontologías existen varias acepciones, incluidas en [Pinto99]. Este apartado utiliza la que entiende la integración como la *fusión de distintas ontologías sobre un mismo dominio en una que unifique todas ellas*. En [Klein01] se define la terminología que se maneja para la integración de ontologías:

- Combinación. Usar dos o más ontologías diferentes para una tarea en la que su relación mutua es relevante.
- Fusión, integración. Crear una nueva ontología a partir de dos o más que ya existen con partes solapadas.
- Alineamiento. Poner dos ontologías de mutuo acuerdo, haciéndolas consistentes y coherentes.
- Correspondencia. Relacionar conceptos o relaciones similares (de acuerdo a cierta métrica) que sean de distintas fuentes. Una correspondencia puede suponer una integración virtual.
- Articulación. Los puntos de enlace entre dos ontologías alineadas.
- Traducción. Cambiar el formalismo de representación de una ontología manteniendo su semántica.
- Transformación. Cambiar ligeramente la semántica de una ontología para que valga para propósitos distintos al original.
- Versión. El resultado de un cambio que puede existir junto al original.
- Versionado. Un método para mantener la relación entre ontologías recién creadas, las existentes y los datos que las conforman consistentemente.

Los problemas que aparecen debido a la heterogeneidad de los datos también son bien conocidos en el mundo de las Bases de Datos. Por ello, muchas aproximaciones para integrar ontologías se basan a su vez en estos trabajos. Un estudio bastante completo de las soluciones propuestas en este campo es [Rahm01], donde no sólo habla de ontologías, sino sobre todo de diagramas entidad-relación que modelan bases de datos. En este trabajo se

hace una taxonomía de las distintas posibilidades existentes para emparejar o combinar distintos modelos de información, según se muestra en la Figura 4.3, y luego clasifica a partir de las aproximaciones en las que se basan varias implementaciones que dan soporte automático para emparejar esquemas. Según este análisis, para emparejamientos individuales los criterios ortogonales de clasificación pueden ser:

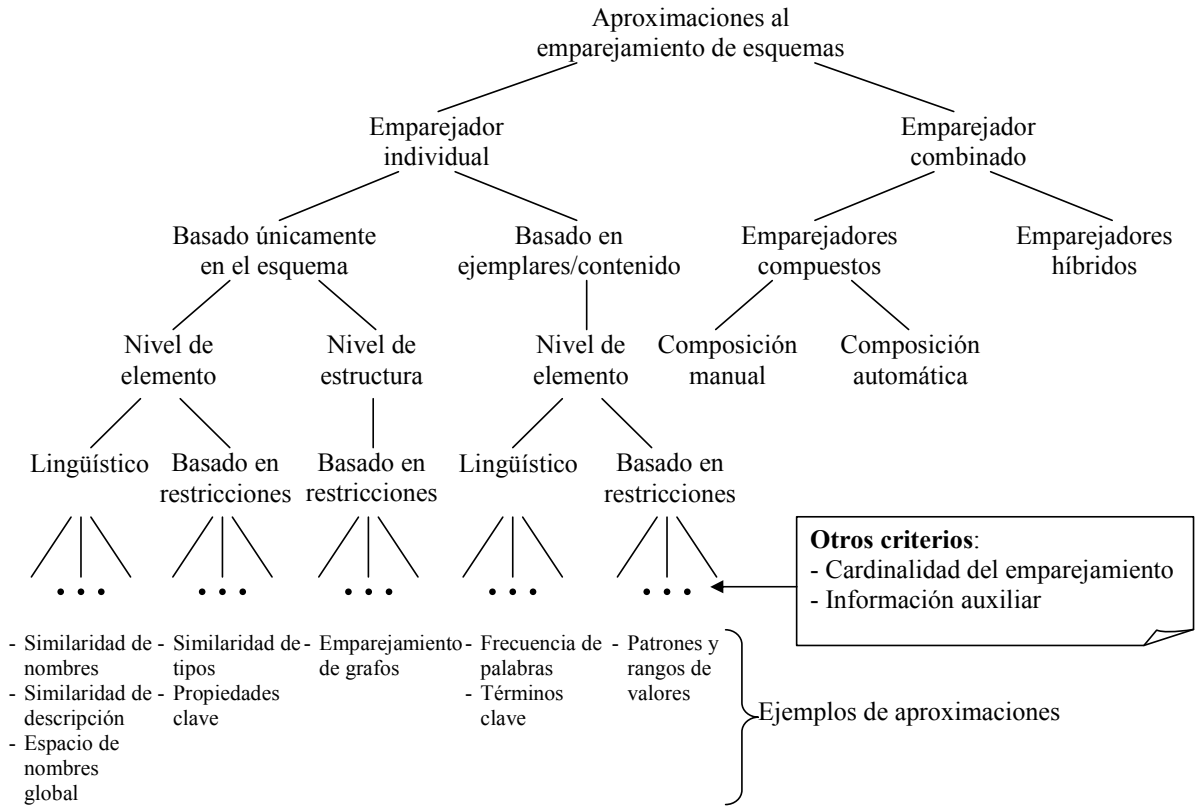

Figura 4.3. Aproximaciones de emparejamiento de esquemas según [Rahm01].

- Ejemplares frente a esquemas. Si se consideran los datos (ejemplares) o los esquemas que definen la información. En el caso de la gestión, no se poseen los datos a priori, sino únicamente las especificaciones de los mismos, por lo que los emparejamientos no se podrían hacer con ejemplares.

- Elementos frente a estructuras. Si se consideran los elementos individuales, como atributos, o la combinación de elementos que forman estructuras complejas, como clases. En el caso de la gestión, ambas cuestiones son aplicables.

- Lenguaje frente a restricción. Si se usa una aproximación basada en los nombres y descripciones de los elementos, o bien según sus atributos claves y relaciones. En gestión normalmente se utilizará la primera aproximación, pues cada modelo posee sus propias relaciones y atributos clave. No obstante, en algunos casos se podrá aprovechar alguna restricción como el tipo de datos empleado.

- Cardinalidad. Cada elemento de un esquema se puede corresponder con uno o más elementos del esquema a combinar, pudiendo existir relaciones 1:1, 1:n, n:1 y n:m. En gestión de red será común que existan distintas cardinalidades.
- Información auxiliar. Además de los esquemas a combinar, normalmente se utiliza información auxiliar como diccionarios, esquemas globales, decisiones de correspondencia previa e interacciones con el usuario. En gestión de red también podrá existir información auxiliar.

En este análisis además se propone el siguiente conjunto de pasos para realizar el emparejamiento, que también son válidos para la integración de información de gestión:

1. Identificar y caracterizar las relaciones entre los esquemas.
2. Una vez que se han identificado, los elementos emparejados se pueden unificar en una vista integrada.
3. Durante este proceso, se crean programas o consultas que permitan la traducción de datos de los esquemas originales a la representación integrada.

Por otro lado, pasando a estudios directamente relacionados con las ontologías, en [Wache01] se hace una descripción exhaustiva de la integración de ontologías, distinguiéndose tres posibles aproximaciones, basadas en:

- Una única ontología global que proporcione un vocabulario compartido. Esta solución puede ser inabordable si cada fuente de información posee una perspectiva distinta para un mismo dominio.
- Múltiples ontologías que describen cada fuente de información de manera independiente, como ocurre en [Mena00]. Para esto es necesario tener una correspondencia entre todas las ontologías, lo que puede ser muy difícil de definir.
- Una solución híbrida, que posee una ontología global y traducciones a las ontologías locales. De esta forma, se puede utilizar un mismo vocabulario. Por el contrario, hay que definir la traducción a cada ontología. Este caso puede ser parecido al de CIM, que está definido como un modelo de información común, con correspondencias con los modelos de gestión existentes.

En [Klein01] también se hace un estudio bastante completo del problema de la integración de ontologías y las soluciones existentes, distinguiendo en tres categorías los problemas a la hora de integrar ontologías:

1. Desemparejamiento (*mismatch*) entre ontologías, que ocurre cuando se trabaja con ontologías que se han desarrollado independientemente.
2. Versionado de ontologías, que ocurre cuando existen distintas versiones de una ontología.
3. Problemas prácticos, como la automatización de la integración.

La presente tesis es afín al primer y al tercer punto. Dentro del primer caso, distingue entre:

1. Desemparejamiento en el nivel del lenguaje. Son los que ocurren debido a la combinación de ontologías escritas con distintos lenguajes. Pueden darse varios casos según el grado de desemparejamiento:
    a. Sintaxis. Se refiere a que un mismo lenguaje tenga varias representaciones sintácticas. Por ejemplo, en el caso de CIM, se puede escribir en MOF y en XML.
    b. Representación lógica. Está relacionado con que algunos lenguajes no pueden expresar construcciones de una manera directa, aunque sí con construcciones más complejas. Por ejemplo, en SMIv2 no se pueden representar clases directamente, pero sí definiendo tablas.
    c. Semántica de primitivas. Está asociado a que una construcción tenga significados distintos en dos lenguajes. Este caso no es muy común en gestión de red.
    d. Expresividad del lenguaje. Se debe a que algunos lenguajes permiten definir ciertas construcciones y otros no. Por ejemplo, GDMO permite la herencia múltiple, y otros lenguajes no.
2. Desemparejamiento en el nivel de la ontología. Son los que ocurren cuando se combinan dos o más ontologías que describen dominios solapados, independientemente del lenguaje que utilicen para su definición. Pueden ser:
    a. Desemparejamiento de la conceptualización. Se deben a diferencias en la forma en que se interpretan los dominios:
        i. Ámbito. Aunque dos clases parecen representar el mismo concepto, no tienen exactamente los mismos ejemplares.
        ii. Granularidad y cobertura del modelo. Tiene que ver con el grado de detalle y las partes de un dominio que cubre un modelo.
    b. Desemparejamiento de modelado. Es debido a diferencias en la forma en que se especifica la conceptualización:
        i. Paradigma. Dos modelos similares se definen reutilizando ontologías generales distintas.
        ii. Descripción de conceptos. Los atributos que están definidos dentro de una clase en la otra ontología se definen en una tercera clase externa a la segunda; o bien, la jerarquía entre las clases es diferente en ambas ontologías.
    c. Desemparejamiento terminológico. Debido a diferencias en la terminología que se emplea:

i. Términos sinónimos. Dos términos tienen el mismo significado, pero distinto nombre. Pueden ser sinónimos de un mismo lenguaje, o un término escrito en dos lenguajes distintos (inglés y español).

ii. Términos homónimos. Dos términos tienen el mismo nombre, pero distinto significado.

d. Codificación de los valores. Se deben a cómo se codifican los valores, como distintas unidades de medida, formatos de fecha, etc.

Dentro de los problemas prácticos se identifica que es difícil encontrar los términos a alinear, que en general se debe hacer a mano, y que las consecuencias de una correspondencia específica son difíciles de prever. También presenta el problema de que las fuentes que se fusionan suelen evolucionar por su cuenta, con lo que hay que hacer fusiones posteriores con las ontologías revisadas.

Finalmente, en [Noy02b] también se ha evaluado cuáles son las características de herramientas de combinación de ontologías, incluyendo los siguientes criterios:

- Requisitos de entrada:
  - Elementos que se usan: clases, propiedades, jerarquía, ejemplares.
  - Elementos que se requieren: si es necesario que haya ejemplares o no.
  - Paradigma de representación: marcos, objetos, lógica descriptiva, etc.
- Nivel de interacción del usuario:
  - Proceso en segundo plano con presentación final de resultados, o análisis del usuario de los resultados intermedios.
- Tipo de salida:
  - Reglas de articulación.
  - Ontología.
  - Ejemplares de una ontología de correspondencia.
  - Lista de pares de conceptos.
- Contenido de la salida:
  - Qué elementos de la entrada se correlan en la salida. Esto incluye clases, ranuras, valores, ejemplares…

En lo que se refiere a la formalización del proceso de integración de ontologías, en [Mitra00] se define un álgebra de ontologías basada en un modelo orientado a grafos, con operadores unarios y binarios. Los primeros incluyen los operadores filtro y extracto, que permiten definir las áreas de interés de cada ontología en estudio. Los segundos pueden ser uniones, intersecciones y diferencia de ontologías.

Entre las distintas herramientas de soporte a la combinación de ontologías cabe mencionar Chimaera [McGuinness00], Prompt [Noy00], FCA-Merge [Stumme01], que permiten la fusión de ontologías, y ECIMF-ST [ECIMF01] y MAFRA [Maedche02], que generan ejemplares de ontologías de correspondencia. Otros sistemas que también han trabajado con la integración de ontologías son ONION [Mitra00], OBSERVER [Mena00] y GLUE [Doan02]. Al examinar estos trabajos se encuentra que los problemas existentes para integrar ontologías son similares a los de la integración de modelos de información de gestión, con la diferencia de que no se trata sólo la traducción sintáctica o reescritura en distintos lenguajes, sino otras cuestiones para permitir la interoperabilidad a un nivel semántico, en el que fusionan los modelos para obtener un modelo común, o bien se establecen correspondencias entre los modelos. Siguiendo este razonamiento, los siguientes apartados analizan por separado las técnicas de fusión y correspondencias de ontologías, que aunque son distintas poseen el objetivo común de integración.

4.4.1 Fusión y alineamiento de ontologías

Según la terminología explicada anteriormente, la fusión consiste en crear una nueva ontología a partir de dos o más que ya existen con partes solapadas. El alineamiento supone poner dos ontologías de mutuo acuerdo, haciéndolas consistentes y coherentes, es decir, ambas ontologías persisten con enlaces o articulaciones entre ellas. En [Noy99] se ilustra esta diferencia, según se ve en la Figura 4.4. A partir del ejemplo de la traducción de lenguajes naturales se pueden explicar los términos de fusión, alineamiento y correspondencia. En la correspondencia se estudia qué palabra en inglés se corresponde con otra en español. El esperanto se puede ver como una fusión de distintos lenguajes europeos. Finalmente, el alineamiento consiste en introducir en el lenguaje el vocabulario de un dominio como por ejemplo, la medicina o las telecomunicaciones.

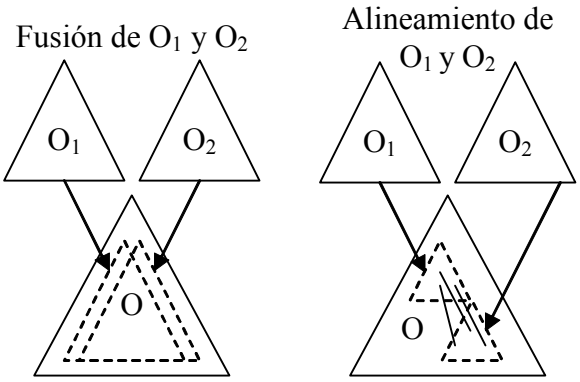

Figura 4.4. Comparación entre fusión y alineamiento de ontologías según [Noy99].

Realizar fusión o alineamiento de ontologías depende en gran medida de si las ontologías a integrar poseen partes solapadas o si por el contrario se refieren a información complementaria. En general, las ontologías a integrar suelen poseer ciertos solapamientos, por lo que el resultado final es un conjunto de fusión y alineamiento de dichas ontologías.

Han sido varios los trabajos en los que se proponen metodologías para la fusión y alineamiento de ontologías para obtener una ontología unificada. Por ejemplo, en [Hovy98], uno de los primeros estudios en este sentido, se define este conjunto de etapas:

- Identificación manual de términos principales.
- Fusión de contenido de los términos identificados.
- Alineamiento más amplio de términos, tomando los términos próximos a los ya fusionados.
- Resolución de inconsistencias.
- Repetición de los dos últimos pasos.

Este estudio también define un conjunto de heurísticos que permiten identificar con gran probabilidad conceptos semejantes, basados en:

- Emparejamiento de texto, según el nombre y según la definición. Si dos conceptos poseen el mismo nombre o una descripción similar, con gran probabilidad serán emparejables.
- Emparejamiento de jerarquía y taxonomía. Si dos conceptos cuyos conceptos padre estén emparejados, con gran probabilidad podrán emparejarse.
- Emparejamiento de restricciones. Si dos ranuras pertenecientes a dos conceptos emparejados poseen un mismo rango, podrán emparejarse.

También define heurísticos de validación del resultado, como la creación de ciclos, o violación de restricciones.

El grupo de investigación Stanford Medical Informatics[14] también ha realizado un análisis en profundidad acerca de la fusión y alineamiento de ontologías. Diversos artículos han ido presentando sus resultados, que son bastante completos. Por ejemplo, en [Noy99] se propone un algoritmo de fusión y alineamiento basado en un modelo de conocimiento compuesto de clases, ranuras, facetas y ejemplares. El algoritmo es el siguiente:

1. Se cargan dos ontologías y se decide si se sobrescribe con la ontología preferida, o pregunta qué se hace en cada caso.
2. Se genera la lista inicial de sugerencias. Para ello se miran los nombres de las clases buscando nombres lingüísticamente similares: sinónimos, subcadenas similares, prefijos o sufijos comunes.
3. Si el proceso es de fusión:

[14] http://smi.stanford.edu/

3.1. Para cada par de clases con nombres *idénticos*, se fusionan clases o se borra una de las clases.

3.2. Para cada par de clases con nombres similares lingüísticamente, se establece un enlace entre ellas (con un nivel más bajo de confianza que para el primer grupo).

4. Si el proceso es de alineado, una ontología será más general que otra y entonces:

4.1. Si hay una clase en la ontología particular con el mismo nombre que en la general, se fusionan las clases.

4.2. Si no, se busca una clase madre en la ontología general.

5. El usuario selecciona una operación, que puede ser de las sugeridas, o bien tomada por iniciativa propia.

6. La herramienta hace una actualización automática y crea nuevas sugerencias. Para los conceptos fusionados o alineados en el paso anterior, se consideran similitudes lingüísticas de los nombres de la clase y las ranuras, así como de la estructura de la ontología. Y de esta forma:

6.1. Se ejecutan cambios adicionales, determinados automáticamente por el sistema.

6.2. Se añade a la lista de conflictos el conjunto de conflictos que se generen. Los conflictos pueden deberse a un tipo de datos distintos en cada ontología.

6.3. Se añade a la lista de sugerencias aquéllas que se creen.

7. Se repiten los pasos anteriores hasta que las ontologías estén completamente fusionadas o alineadas.

Este trabajo también define el conjunto de operaciones básicas de fusión y alineamiento, que incluyen fusionar clases, borrar clase, borrar clase madre, añadir clase madre, renombrar clase, mover clase de una ontología a la otra (alinear), borrar ranura, mover ranura de clase, renombrar ranura, renombrar ranura en una clase y cambiar el tipo de datos de una ranura. También habla del papel que tienen las facetas en todo esto, pues definen rangos de valores que pueden causar conflictos.

Posteriormente, [Noy00] presenta la herramienta PROMPT como implementación del algoritmo anteriormente descrito. Entre las operaciones posibles ahora también incluye la fusión de ranuras, o las copias superficiales y en profundidad de una jerarquía de clases. También identifica distintos conflictos que pueden aparecer, como los debidos a la colisión de nombres, referencias a conceptos que no existen en la nueva ontología, la existencia de múltiples caminos de herencia entre dos clases o restricciones en el valor de una ranura que violan la herencia de clase.

Más adelante, con Anchor-Prompt [Noy01] se incluyen heurísticos basados en contexto no local. Dadas dos clases iniciales de dos ontologías y dos clases finales de dichas ontologías, correspondiéndose entre ellas en ambos casos, las clases que se encuentren en el camino jerárquico entre las clases iniciales y finales tendrán muchas posibilidades de corresponderse igualmente.

En cualquier caso, los métodos que se proponen para obtener un modelo común no incluyen reglas de correspondencia que permitan deshacer la fusión. El siguiente apartado introduce un conjunto de aproximaciones para definir dichas correspondencias.

4.4.2 Ontologías de correspondencia

Los mecanismos de fusión y alineamiento permiten obtener una ontología fusionada, con la que poder trabajar de manera integrada. Sin embargo, el proceso de fusión no permite obtener reglas de correspondencia para, a partir de ellas, poder traducir entre la ontología común y cada una de las ontologías particulares. Este apartado presenta distintas propuestas de ontologías de correspondencia que se han definido para resolver esta cuestión.

4.4.2.1 *Mapping Ontology*

La primera ontología que se conoce de este tipo es la que se explica en [Park97]. En este caso se trataba de aplicar correspondencias entre métodos de resolución de problemas (PSM) y ontologías de dominio. Para ello se definió la *mapping ontology* (ontología de correspondencia), expresada en UML en la Figura 4.5.

Como se puede ver, esta ontología está orientada a la traducción de ranuras mediante distintas fórmulas o `slot-maps`, estableciéndose que una relación de correspondencia se compone de un conjunto de correspondencias de ranuras. Ésta será de cuatro tipos, según las correspondencias de ranuras que posea:

- Renombrado (*Renaming-slot-map*). Sólo se cambia el nombre a la ranura.
- Constante (*Constant-slot-map*). Se le asigna un valor constante, pues no tiene correspondencia en el otro dominio.
- Léxico (*Lexical-slot-map*). Permite la concatenación de ranuras del otro dominio.
- Expresión Regular (*Regular-expression-slot-map*). Permite manipulación y modificación mediante expresiones regulares para la traducción.
- Expresión Numérica (*Numerical-expression-slot-map*). Permite el procesado numérico para la traducción.
- Función (*Functional-slot-map*). Permite el uso de funciones para transformaciones complejas que no se puedan hacer mediante expresiones regulares o numéricas.

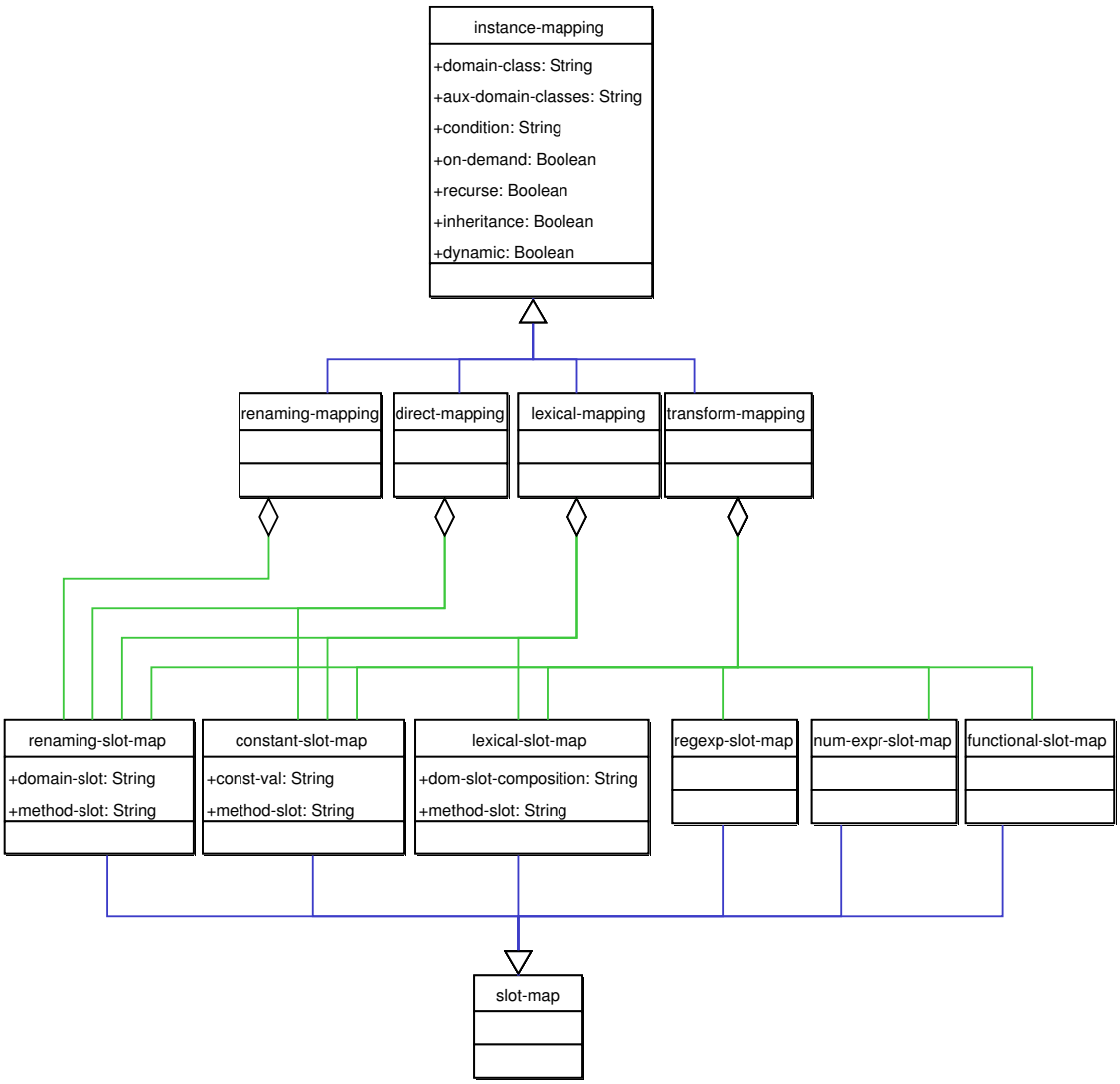

Figura 4.5. ***Mapping Ontology*.**

En este caso no se realiza ninguna traducción de conceptos o clases, debido a que las correspondencias son entre ontologías de dominio y de resolución de problemas. Sin embargo, la aproximación de definir los tipos de traducciones es válida para la transformación de ejemplares a distintas ontologías.

4.4.2.2 *Semantic Translation Ontology*

Posteriormente, dentro del proyecto ECIMF[15] (*E-Commerce Integration Meta-Framework*, Meta-Marco de Integración de Comercio Electrónico) se ha definido la ontología *Semantic Translation* (Traducción Semántica) para su uso en una herramienta de definición de reglas de correspondencia [ECIMF01]. El diagrama en UML que representa los elementos de la ontología es el que se muestra en la Figura 4.6.

[15] http://www.ecimf.org/

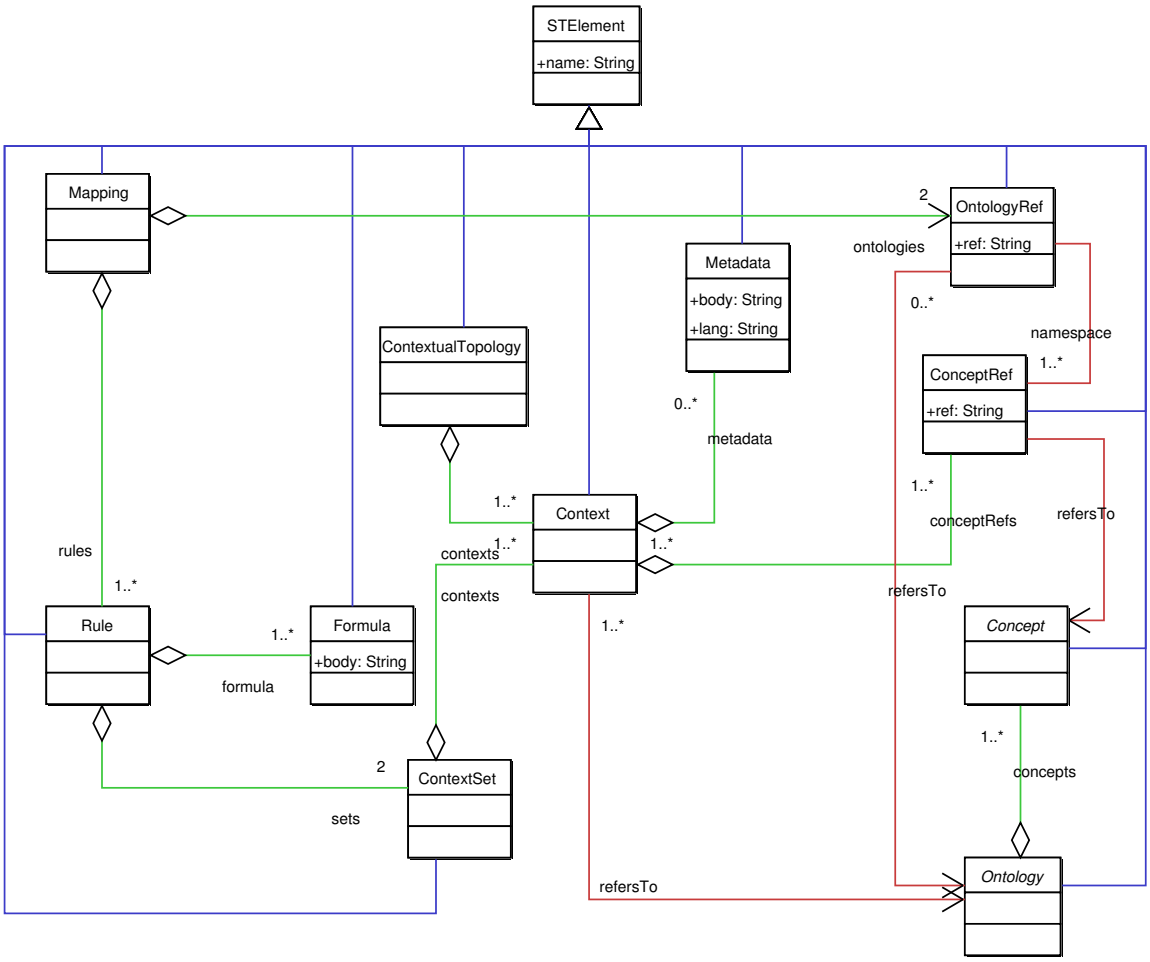

Figura 4.6. ***Semantic Translation Ontology*.**

Según la documentación extraída de la ontología, estos elementos actúan de la siguiente manera:

- Elemento de la Traducción Semántica (*STElement*). Es la clase madre del resto, que son elementos de la correspondencia de traducciones semánticas.
- Ontología (*Ontology*). La `Ontology` de dominio original (externa al proyecto). Clase definida como abstracta.
- Referencia a la Ontología (*Ontology Ref*). Es un nombre único que identifica la `Ontology` referida (posiblemente permitiendo acceder a ella remotamente).
- Concepto (*Concept*). Los `Concept` definidos en la `Ontology` original (externa al proyecto). Clase definida como abstracta.
- Referencia a los Conceptos (*Concept Ref*). Es una referencia dentro de un espacio de nombres relativa a conceptos individuales definidos en la `Ontology` original. Un nombre único, que posiblemente permite acceder remotamente a la definición del concepto en la ontología original.

- Correspondencia (*Mapping*). Es un contenedor de alto nivel para las reglas de correspondencia semántica, aplicable a un par de ontologías especificadas por las `OntologyRef`.
- Contextos (*Context*). Se crea sobre la base de la `Ontology` original. Consiste en conceptos relacionados representados por `ConceptRef`, que se consideran relevantes para una regla de transformación dada (la relación exacta y completa de los `Concept` se define en la ontología original – `Context` captura sólo el hecho de que están relacionados con propósito de establecer su correspondencia).
- Conjunto de Contextos (*Context Set*). Es un grupo de uno o más `Context`, referidos a la misma `Ontology`.
- Regla (*Rule*). Define cómo traducir entre los conceptos en un `ContextSet` de una `Ontology` a los conceptos correspondientes en un `ContextSet` de la otra `Ontology`. Una `Rule` contiene exactamente dos `ContextSets`, cada uno referido respectivamente a una de las `Ontology`, y un conjunto de fórmulas, que definen las transformaciones válidas de estos `ContextSet`.
- Fórmula (*Formula*). Es una expresión formal que define cómo se realiza la traducción entre los conceptos de un `ContextSet` a los del otro.

Esta ontología, al contrario que la anterior está orientada a conceptos. Sin embargo, en este caso no se indica cómo hacer corresponder las ranuras o propiedades.

4.4.2.3 *Bridging Ontology*

Más adelante se ha definido la *Bridging Ontology* (Ontología de Puentes) de MAFRA (*MApping FRAmework*, Marco de Correspondencia) [Maedche02], que es la más completa, aunque también la más compleja de todas. Los `slot-map` de la *Mapping Ontology* aquí se definen como *bridge* (puente), y no únicamente para ranuras, sino también para conceptos y relaciones. Asimismo define reglas que pueden ser de tres tipos.

Según sus autores, esta ontología trata de trabajar con las cinco dimensiones que han identificado para una correspondencia:

- Entidad. Refleja el tipo de entidades ontológicas (conceptos, relaciones, atributos) que están siendo puenteadas. Se dejan de lado las facetas de los atributos, a no ser que se incluyan en los *slot-bridge*.
- Cardinalidad. Refleja el número (1:1, 1:n, m:1) de entidades ontológicas que se están puenteando.
- Estructura. Refleja la forma en que los puentes elementales se pueden combinar en puentes más complejos, mediante especialización, abstracción, composición y relaciones alternativas.

- Restricción. Refleja las restricciones aplicadas durante la fase de ejecución a los ejemplares de la ontología de origen.
- Transformación. Refleja cómo los ejemplares de la ontología de origen se transforman durante el proceso de correspondencia.

Los elementos principales, representados en la Figura 4.7, son:

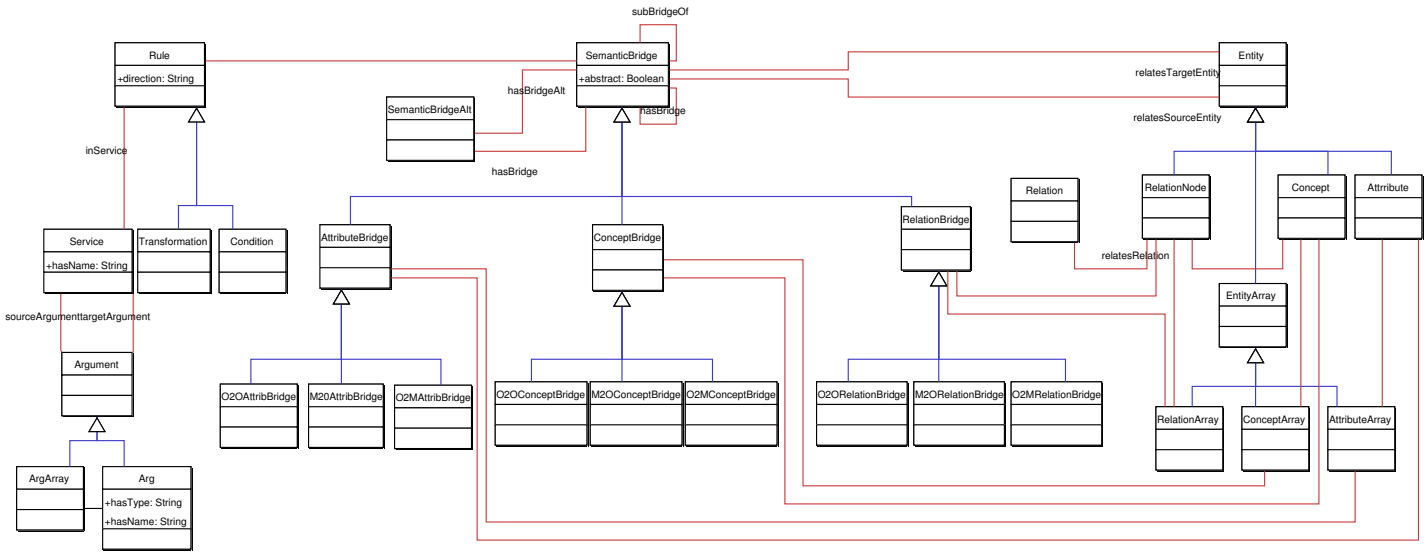

Figura 4.7. *Bridging Ontology*.

- Entidad (*Entity*). Las entidades pueden ser `Concepts` (Conceptos), `Relations` (Relaciones) y `Attributes` (Atributos).
- Puente semántico (*Semantic Bridge*). Puente genérico que define las relaciones entre las entidades origen y destino. Se especializa según el tipo de `Entity` y según la Cardinalidad. También permite realizar composiciones.
- Servicio (*Service*). Vale para referenciar recursos que son responsables de la conexión o de describir transformaciones, incluyendo nombre, interfaz y localización. `Argument` y sus subclases permiten describir estas características de una forma simple y directa.
- Regla (*Rule*). Es la clase general para la información de restricciones y transformaciones relevante.
- Transformación (*Transformation*). Obligatoria en los puentes salvo si es abstracta. Usa la relación `inService` para enlazar al procedimiento de transformación y cualquier motor de ejecución y atributos específicos de funciones para especificar requisitos extra.
- Condición (*Condition*). Representa la condición que se debe verificar para ejecutar el puente semántico. Es similar a la transformación, pues debe especificar los requisitos extra. Permite controlar transformaciones complejas.

- Puente Semántico Alternativo (*Semantic Bridge Alt*). Agrupa puentes mutuamente excluyentes, escogiéndose el que cumpla la condición de ejecución.

Como se puede ver tras analizar la figura con detenimiento, los conceptos situados en la parte de la izquierda se refieren al conjunto de reglas y fórmulas que se van a emplear para llevar a cabo las traducciones; los conceptos situados en la parte central se refieren a los tipos de puentes que se pueden definir, según la cardinalidad y los elementos que se correspondan; finalmente, en la parte derecha se sitúan los distintos elementos a corresponder, siendo esto una especie de meta-ontología que indica cómo deben estar definidas las ontologías a corresponderse.

4.4.2.4 Ontologías de la Web Semántica

Finalmente, y relacionando este estudio con la Web Semántica, se han identificado las construcciones de DAML+OIL definidas en este sentido. Se puede ver que este lenguaje de definición de ontologías posee distintas construcciones como `rdfs:subClassOf`, `rdfs:subPropertyOf`, `daml:sameClassAs` y `daml:samePropertyAs`, que permiten definir cierto grado de correspondencias entre clases y propiedades definidas en distintas ontologías. También hay construcciones para hacer corresponder ejemplares, como `daml:sameIndividualAs` y `daml:differentIndividualFrom`.

A partir de DAML+OIL, el proyecto CODIP ha definido la *Articulation Ontology*[16] (Ontología de Articulación), representada en la Figura 4.8 en UML. Dicha ontología posee tres clases:

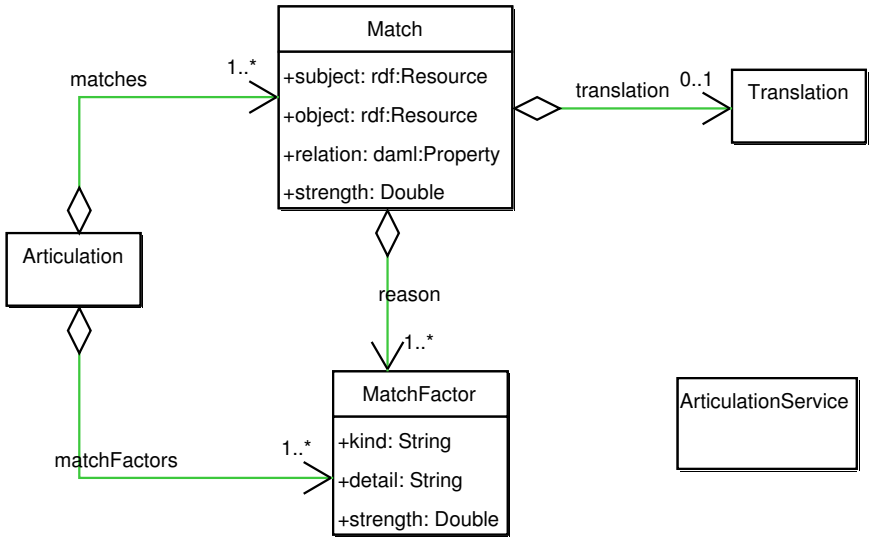

Figura 4.8. ***Articulation Ontology*.**

[16] http://codip.grci.com/Articulation.daml

- Articulación (*Articulation*). Es una especialización de una ontología para representar articulaciones. Posee las propiedades `matches`, de tipo `Match` y `matchFactors`, de tipo `MatchFactor`.
- Servicio de Articulación (*Articulation Service*). No tiene descripción, pero se entiende que es el servicio que lleva a cabo el proceso de articulación.
- Traducción (*Translation*). Es la firma descriptiva del servicio de traducción requerida para traducir ejemplares de sujetos en ejemplares de objetos.

Existen propiedades que están contenidas en la clase `Articulation`, pero que a su vez poseen otras propiedades, por lo que en la figura se han representado como clases:

- Emparejamiento (*Match*). Es una especialización de un `daml:ObjectProperty` que empareja un conjunto no vacío de sujetos con su conjunto no vacío de objetos. Posee las siguientes propiedades:
  - Sujeto (*subject*). El sujeto del emparejamiento, al menos uno.
  - Objeto (*object*). El objeto del emparejamiento, al menos uno.
  - Razón (*reason*). Son ejemplares de `MatchFactor` que apoyan la validez del emparejamiento.
  - Traducción (*translation*). Un emparejamiento (`Match`) puede tener hasta una traducción (opcional). La traducción es una referencia a un servicio o capacidad que puede traducir ejemplares del conjunto de sujetos a ejemplares equivalentes del conjunto de objetos.
  - Relación (*relation*). Un emparejamiento (`Match`) puede tener hasta una relación (opcional). La relación es una propiedad DAML+OIL que especifica la relación entre el conjunto de sujetos y el conjunto de objetos. La relación por defecto es `daml:equivalentTo`.
  - Fortaleza (*strength*). Media de las `strength` que proporcionan los distintos ejemplares de `MatchFactor` asociados al emparejamiento.
- Factor de Emparejamiento (*Match Factor*). Es una especialización de un `daml:ObjectProperty` que empareja sus sujetos con objetos. Los ejemplares de `MatchFactor` son los "ladrillos" de razonamiento que se usan para definir distintos `Match`. Posee las siguientes propiedades cuyo rango son tipos de datos:
  - Tipo (*kind*). Define el tipo de factor de emparejamiento. En un futuro será sustituido por especializaciones de `MatchFactor`. Es una cadena de caracteres.
  - Detalle (*detail*). Da información específica que depende de la clase de `MatchFactor`. Es una cadena de caracteres.

- Fortaleza (*strength*). Es un número real entre 0 y 10 que proporciona información acerca de cuánto se emparejan el sujeto y el objeto.

En general, para un caso práctico únicamente se usan construcciones `Match` y `MatchFactor`, que indican las correspondencias entre dos clases.

4.4.2.5 Tabla comparativa

El uso de una de estas ontologías o una nueva basada en las existentes puede ser útil para completar la integración de ontologías, pues permite traducir entre los ejemplares de una ontología fusionada según los métodos presentados en la sección anterior, y cada una de las ontologías particulares. La Tabla 4.1 resume las distintas ontologías de correspondencia estudiadas.

Tabla 4.1. Ontologías de correspondencia.

Nombre de la ontología	Ventajas	Inconvenientes
Mapping Ontology	Propone distintas clases de fórmulas de correspondencia.	Está orientado a correspondencia con PSM.
Semantic Translation Ontology	Orientada a la correspondencia de conceptos.	No indica cómo hacer la correspondencia entre propiedades.
Bridging Ontology	Orientada a todos los elementos de una ontología.	Demasiado compleja de manejar.
DAML+OIL	Incluye directamente algunas construcciones para corresponder elementos.	Únicamente permite hacer enlaces entre elementos, pero no permite establecer reglas de traducción
Articulation Ontology	Incluye distintos grados de correspondencia.	La traducción se realiza por un servicio que no define.

4.5 Conclusiones

El presente capítulo ha introducido el concepto de ontología, así como su relación con los modelos de información de gestión, pudiendo verse que éstos se pueden considerar como ontologías ligeras. Al igual que en el caso de la gestión de red, también existen múltiples lenguajes para definir la información: lenguajes tradicionales con notación prefija al estilo de LISP y con capacidad de definición de axiomas, lenguajes de la Web Semántica, con sintaxis XML, y la posibilidad de utilizar UML para modelar estas ontologías. Este hecho, junto a la existencia de múltiples bibliotecas ha supuesto tener que establecer mecanismos de integración de ontologías, de manera similar a lo que ocurre con los modelos de información de gestión, pero con una diferencia: no sólo se han tratado las traducciones sintácticas, sino que se ha llevado el problema al nivel semántico, existiendo distintas

aproximaciones que tratan de solventarlo mediante mecanismos de fusión y ontologías de correspondencia. Por tanto, según lo expuesto las ontologías pueden ayudar a realizar la integración de modelos de gestión de red de una manera más fácil e incluyendo información adicional:

- Para definir la información de gestión sería necesario un lenguaje de definición similar a los estudiados en este capítulo, cuya capacidad expresiva le permita expresar una semántica precisa, incluyendo restricciones sobre la información. Para ello, se pueden comparar los lenguajes de información de gestión en los mismos términos en los que se comparan los lenguajes de ontologías.
- Asimismo, a partir de dicho lenguaje, se podría definir una ontología de gestión de red, que defina los distintos conceptos actualmente contenidos en los distintos modelos existentes, a partir de su fusión. A la vez, sería útil definir a partir de una ontología de correspondencia las reglas para traducir entre la ontología anterior y cada uno de los modelos de información de gestión existentes.
- Para todo ello, sería necesario definir algún método que aplique los conceptos de fusión y correspondencia aquí explicados teniendo en cuenta las especificidades de la información de gestión.
- Sería útil, finalmente, que un sistema de gestión pudiera trabajar con esta ontología, pues esto le permitiría poder tener una vista única de todos los recursos gestionados, independientemente de que pertenezcan a uno u otro dominio de gestión. Asimismo, se pueden aprovechar las restricciones definidas sobre la información para modelar en cierta manera el comportamiento del sistema de gestión.

Los dos capítulos que se incluyen a continuación tratan estas cuestiones realizando dos propuestas. La primera propuesta, contemplada en el siguiente capítulo, se refiere a como aplicar las ontologías para la integración de modelos de información de gestión. Para ello, se estudian los primeros tres primeros puntos indicados anteriormente, es decir, comparación de la expresividad semántica los lenguajes de información de gestión, análisis de lenguajes de ontologías para especificar información de gestión, y diseño de mecanismos de integración semántica de los modelos de información de gestión existentes. La segunda propuesta, contemplada más adelante, trata el último punto, esto es, analizar la forma en que se puede incluir comportamiento a las especificaciones de información de gestión utilizando las ontologías, y a partir de ahí, definir como debe ser la arquitectura de un sistema de gestión que utilice estos modelos de información.

Capítulo 5 Propuesta de aplicación de las ontologías para la integración de información de gestión

5.1 Introducción

En los tres capítulos anteriores se han introducido diversas cuestiones que han mostrado de forma encadenada el problema en estudio, las limitaciones de las propuestas actuales y una posible vía de solución que resuelve dichos inconvenientes:

- Existen distintos modelos de gestión integrada. Cada uno posee un protocolo de intercambio de información y un lenguaje de definición de información con el que se modelan los recursos a gestionar.
- Ante esta heterogeneidad de modelos se han planteado diversas aproximaciones que proponen soluciones para que interoperen en mayor o menor medida. De estas soluciones cabe mencionar WBEM, un modelo de gestión integrada definido desde un comienzo con este propósito. Sin embargo, ninguna de las aproximaciones consigue llegar al nivel de la semántica de la información definida en cada uno de los modelos.
- Posteriormente se han estudiado las ontologías y se han comparado con los modelos de información de gestión. Asimismo, se han identificado los mecanismos que se emplean al trabajar con ontologías para tratar de solventar el problema de la interoperabilidad desde un punto de vista semántico, teniendo en cuenta tanto las construcciones con que se especifican los modelos, como el significado contenido en cada definición.

El presente capítulo contempla este problema de integración de los modelos de gestión desde la perspectiva de las ontologías y plantea una propuesta original para mejorar la interoperabilidad semántica de las distintas especificaciones de información. Para ello se han estudiado tres puntos, contenidos en los tres apartados siguientes:

- El primer punto se refiere a analizar y comparar los distintos lenguajes que se utilizan actualmente para definir información de gestión. Esto permitirá saber cuál es el que posee una mayor expresividad, lo cual es importante si se quiere obtener una integración no sólo de la sintaxis, sino de la semántica. Para ello se utiliza un

marco utilizado en el campo de las ontologías para comparar los lenguajes empleados en ese ámbito.

- El segundo punto analiza qué características sería necesario añadir a los lenguajes de definición de información de gestión para que un sistema automático pueda obtener la semántica de las definiciones. Asimismo, estudia la posibilidad de emplear un lenguaje de definición de ontologías para especificar información de gestión. Este segundo caso supondría tener que adaptar dicho lenguaje para que pueda expresar algunas construcciones típicas de los lenguajes de información de gestión.
- Finalmente, y partiendo de los puntos anteriores, se podrán aplicar de manera conjunta las técnicas de fusión y correspondencia definidas en el capítulo previo a la integración de los modelos de información de gestión. Para ello se define un método que integra ambas técnicas y las particulariza para el caso concreto de la información de gestión. Con esto se podrá obtener un nuevo modelo que realmente integre los ya existentes teniendo en cuenta la semántica contenida en los mismos, pudiendo solventar de esta forma el problema identificado anteriormente.

5.2 Análisis y comparación de la expresividad semántica de los modelos actuales de información de gestión

El primer paso a realizar para poder obtener una integración semántica de los modelos de información de gestión debe ser el estudio de la expresividad que proporcionan los distintos lenguajes con los que estos modelos se especifican. Para ello, esta sección utiliza las ontologías para estudiar y comparar diferentes lenguajes aplicados normalmente a la definición de información de gestión de red y sistemas. Analizar la capacidad expresiva según los términos de las ontologías tiene dos ventajas [LópezDeVergara03b]. La primera es que tiene en cuenta la forma en que los lenguajes de ontologías proporcionan la expresividad semántica. La segunda es que esta comparación se hace desde una perspectiva totalmente neutra a cualquiera de los lenguajes de información de gestión.

Para llevar a cabo este análisis se compararán las construcciones de los lenguajes de información de gestión, presentadas en el Capítulo 2 con los términos habitualmente empleados en las ontologías. Esto permitirá establecer la capacidad semántica que posee cada uno de ellos, y así facilitar los mecanismos de integración que se deban realizar posteriormente. Esta propuesta se basa en algunas aproximaciones existentes que comparan lenguajes de ontologías [Corcho00, Gómez02, Wache01], que evalúan los elementos que estos lenguajes pueden expresar.

Para realizar este análisis, primero se comentarán otros estudios acerca de los lenguajes de gestión existentes. Tras esto, se realizará un análisis de su expresividad, emparejando las

construcciones de estos lenguajes con las que están incluidas normalmente en los de ontologías. Finalmente, se mostrarán tablas que comparan los distintos lenguajes de gestión a partir de este análisis.

5.2.1 Otros análisis de lenguajes de información de gestión

Hasta la fecha ha habido diversos estudios que comparan lenguajes de información de gestión. Por ejemplo, hay trabajos que para buscar la interoperabilidad han realizado comparaciones similares aunque no tan completas como la que esta tesis presenta. En [Rivière96a] se han analizado las construcciones que utilizan GDMO, SMI y MIF. Con una aproximación similar, en [Ban97] se ha comparado GDMO, SMI e IDL en términos de su metamodelo. En [Festor99] se han confrontado las propiedades de CIM y GDMO según las siguientes características, que se basan en criterios propios e independientes de cualquier taxonomía de lenguajes:

- Lenguajes de especificación (GDMO y GRM frente a MOF)
- Unidad de especificación (módulos frente a esquemas)
- Herencia (múltiple frente a simple)
- Ámbito de los atributos (externos o internos al objeto)
- Tipo de datos de los atributos, parámetros y respuestas (todos los de ASN.1, incluyendo estructuras y secuencias, frente a tipos simples y *arrays* de estos)
- Nombrado (*distinguished name* frente a *keys*)
- Relación entre objetos (GRM frente a clases asociación)
- Repositorio de especificaciones (MIB dedicada frente a almacenamiento en el espacio de nombres)
- Protocolo de gestión (CMIP frente a HTTP-XML)
- Eventos (Servicio de eventos definido en GDMO).

Frente a los estudios anteriores, la lista de lenguajes que se comparan a continuación incluye los de los modelos de gestión integrada presentados en el Capítulo 2: GDMO, SMIv2 y SMIng, MIF, IDL, y MOF/CIM. Con ello se da una panorámica completa que contempla a todos los lenguajes. Además, no se basa en juicios subjetivos, sino que toma los criterios propuestos por expertos en el campo de las ontologías, con lo que se puede garantizar que la comparación será objetiva y neutral a todos los modelos de información existentes. Es más, al hacerse la comparación desde la perspectiva de las ontologías, ésta permitirá conocer la expresividad semántica de cada uno de estos lenguajes, lo que facilitará una posterior integración basada en el significado contenido en los modelos de información.

Por otro lado, en [Schönwälder01] se aporta un conjunto de ideas para extraer relaciones de herencia y composición en las MIBs de SNMP, tratando de verlas como un modelo orientado a objetos. De hecho, a partir de este estudio han implementado una herramienta llamada LIBSMI[1] que genera diagramas de clases UML a partir de MIBs definidas en cualquier versión de SMI. En [Schönwälder01] no se llega a comparar SMI desde este punto de vista con otros lenguajes de definición de información, pero estas ideas se tienen en cuenta en este apartado a la hora de realizar el análisis y comparación de SMI.

5.2.2 Análisis de la expresividad semántica de los lenguajes de información de gestión

Para realizar este análisis se utiliza un marco de comparación basado en algunas publicaciones referidas a la expresividad de lenguajes de ontologías [Corcho00, Gómez02, Wache01]. Las dos primeras referencias ya se han mencionado en el capítulo anterior precisamente para comparar los lenguajes de ontologías. Dichos trabajos propone una metodología para comparar estos lenguajes, proporcionando el conjunto de criterios necesarios para ello y estudiando su expresividad semántica. La tercera referencia, también mencionada en el capítulo anterior, especializa este marco para analizar precisamente la integración de ontologías. Partiendo de estas aproximaciones, se han dado los mismos pasos para los lenguajes de información de gestión en estudio, aplicando como base de comparación los razonamientos explicados en estas propuestas [LópezDeVergara03b].

Según este marco, las estructuras principales que se emplean para describir el conocimiento de un dominio son conceptos, taxonomías, relaciones, funciones, ejemplares y axiomas. A su vez, los conceptos poseen atributos con facetas y las taxonomías y relaciones pueden ser de varios tipos. A continuación se estudiará en detalle cada uno de estos términos y sus propiedades, comparando los lenguajes de información de gestión según estas características.

Como se explicó en la sección 4.3, cuando se trata con ontologías ocurre que hay múltiples terminologías dependiendo del paradigma empleado para la representación del conocimiento [Kiryakov01]: redes semánticas, marcos, lógica descriptiva y orientación a objetos son paradigmas que nombran de distinta manera los conceptos, sus propiedades, sus ejemplares y relaciones. Aunque las terminologías sean distintas, se pueden hacer correspondencias directas entre la mayoría de ellas, lo que también es útil para establecer otras similares con los lenguajes de información de gestión, cuyas características se analizan en la sección siguiente. En el caso de los lenguajes de información de gestión existen básicamente dos paradigmas: el de la orientación a objetos, utilizado en GDMO,

[1] http://www.ibr.cs.tu-bs.de/projects/libsmi/

IDL, MOF/CIM y SMIng, y el de tablas, empleado en SMIv2 y MIF. A su vez, el paradigma de tablas puede corresponderse con el de los objetos si se entiende, como se comentó en su momento, que una tabla es una clase, cuyas columnas son los atributos y cuyas filas son los ejemplares de dicha clase.

5.2.2.1 Conceptos

Los conceptos o clases son los elementos más importantes para definir información. Los siguientes puntos identifican la expresividad de un lenguaje a la hora de definir dichos conceptos:

- Metaclases. Trata la posibilidad de definir clases como ejemplares de otras clases. En GDMO, SMIv2, MIF o IDL esto no es posible, mientras que en MOF/CIM y SMIng se pueden especificar cláusulas con calificadores y extensiones respectivamente, lo que indirectamente posibilita la rdefinición de metaclases.
- Particiones. Este concepto, muy común en lenguajes de ontologías, está relacionado con la definición de conjuntos de clases disjuntas, lo cual no existe en los lenguajes de información de gestión.
- Atributos. Los conceptos normalmente poseen atributos, y todos los lenguajes de información de gestión permiten su definición.
    - Ámbito local. Los atributos se pueden especificar como internos o externos a una clase. Sólo GDMO define los atributos externamente al ámbito de una clase, dado que pertenecen a un paquete.
    - Ejemplares de atributos o plantillas (*templates*). Son atributos cuyo valor puede ser diferente para cada ejemplar del concepto. Todos los lenguajes de gestión definen este tipo de atributos.
    - Atributos propios o de clase. Son atributos cuyo valor, que puede modificarse, debe ser el mismo para todos los ejemplares de un concepto. En este caso, sólo MOF/CIM puede definir estos atributos, usando el calificador `Static`. En otros lenguajes (IDL, con la construcción `const`, o GDMO definiendo un atributo de sólo lectura con un valor inicial) es posible definir constantes, pero como su nombre indica, no pueden modificar su valor.
    - Atributos polimórficos. Son atributos con el mismo nombre y diferente comportamiento para distintos conceptos. En este caso, aquellos lenguajes que tienen un espacio de nombres de atributos interno al ámbito de una clase pueden tener este tipo de atributos. Esto incluye a MIF, IDL, MOF/CIM y SMIng. SMIv2 tiene un espacio de nombres único para todos los atributos, por lo que no se permite el polimorfismo de atributos. GDMO

posee el alomorfismo, con el que un atributo puede comportarse como el atributo de la clase madre, pero esto no concuerda exactamente con la definición presentada.

- Facetas. Los atributos normalmente poseen un conjunto de propiedades predefinidas llamadas facetas. El apartado 5.2.3 hace una comparación exhaustiva de ellas, lo que puede ser útil a la hora de especificar información de gestión. Facetas típicas son:
    - Valor por defecto. Todos los lenguajes salvo IDL pueden definir un valor por defecto para sus atributos.
    - Restricción del tipo de datos. Todos los lenguajes emplean tipos de datos para definir sus atributos. En general los tipos de datos son simples, excepto en GDMO, donde se pueden emplear todos los tipos ASN.1 posibles, e IDL que también puede definir estructuras y secuencias. MOF/CIM sólo permite arreglos o *arrays* de tipos simples. Además, también se pueden definir rangos de valores de esos datos.
    - Restricción de cardinalidad. Esta faceta restringe el número máximo y mínimo de valores para cada atributo. En este caso, sólo CIM e IDL pueden definir restricciones de este tipo, y sólo para *arrays* y secuencias respectivamente. CIM también permite definir un número máximo y mínimo de asociaciones, usando los calificadores `Max` y `Min`.
    - Documentación. Proporciona una descripción del atributo escrita en lenguaje natural. La mayoría de los lenguajes poseen una cláusula de descripción. En GDMO, la plantilla `BEHAVIOUR` se utiliza normalmente para esto. IDL no tiene este tipo de documentación, aunque se pueden emplear las marcas de comentario para ello.
    - Definición operacional. Ninguno de los lenguajes estudiados incluyen directamente esta faceta en la que se puede incluir la definición de una fórmula o regla que especifique cómo se obtiene el valor de un atributo.
    - Adición de nuevas facetas. Una vez más, sólo MOF/CIM y SMIng pueden definir nuevas facetas, empleando los calificadores y extensiones respectivamente. La cláusula `pragma` se podría emplear en MIF para definir alguna faceta particular.
    - Otras facetas. Todos los lenguajes de información de gestión analizados poseen otras facetas comunes, adicionales a las incluidas hasta ahora. A continuación se presentan las que se han considerado más relevantes, si bien cada lenguaje incluye otras particulares de cada modelo de información.

- Acceso. Indica la forma en que se puede acceder al atributo: lectura, escritura, creación, etc.
- Estado de implementación. Indica si el atributo está actualmente en las definiciones, o por el contrario obsoleto. También, la versión de la definición.
- Identificador único. Proporciona un identificador, normalmente de tipo numérico, único para cada atributo, asociado a su nombre.
- Distinción. Indica que ese atributo permite distinguir a cada uno de los ejemplares de la clase.
- Unidades. Proporciona la unidad de medida que utiliza el atributo.
- Referencia. Da una referencia cruzada a otro atributo con el que tiene relación.
- Redefinición. Indica si el atributo está redefinido respecto al de una clase madre.

5.2.2.2 Taxonomías

Los conceptos normalmente se organizan en taxonomías, con relaciones de generalización/especialización entre ellos. Algunas de estas características taxonómicas también se pueden analizar:

- Subclase de. Especializa conceptos generales en otros más específicos. Los lenguajes orientados a objetos tales como GDMO, IDL, MOF/CIM y SMIng permiten la definición de subclases, siendo GDMO e IDL los únicos que permiten la herencia múltiple. En SMIv2, el uso de un índice externo o la cláusula `AUGMENTS` se pueden considerar como cierto tipo de herencia [Schönwälder01].
- Descomposición disjunta. Define una partición entre subclases de una clase. Dos o más clases son disjuntas si son clases hermanas y no puede existir una clase hija de ambas a la vez. Ninguno de los lenguajes estudiados puede definir esta descomposición.
- Descomposición exhaustiva de subclases. Se define de forma que la clase madre es equivalente a la unión de todas sus subclases. Que una descomposición sea exhaustiva no quiere decir que sea disjunta. Al igual que en el caso anterior, ningún lenguaje está preparado para definir esta descomposición.
- No subclase de. Se declara que una clase no es la especialización de otra. De nuevo, ninguno de los lenguajes posee esta construcción. MOF/CIM define el calificador `Final` para evitar que una clase se pueda especializar, pero éste es un concepto distinto.

5.2.2.3 Relaciones y funciones

Las relaciones representan un tipo de interacción entre conceptos. Las funciones proporcionan un valor único a partir de una lista de argumentos valuados.

- Definición de relaciones /funciones. Los lenguajes orientados a objetos tales como GDMO, IDL, MOF/CIM y SMIng permiten la definición de relaciones. Las relaciones de GDMO son agregaciones, aunque se puede emplear el GRM (*Generic Relationship Model*, Modelo de Relaciones Genérico) para otro tipo de asociaciones. Si se emplea la aproximación de las asociaciones utilizada en MOF/CIM (reificación) en la que una clase con dos o más referencias es una asociación, se pueden definir relaciones en IDL y SMIng. En SMIv2 se puede crear una asociación entre dos clases/tablas usando índices externos. Respecto a las funciones, sólo GDMO, IDL y MOF/CIM permiten su definición, llamándolas respectivamente acciones, operaciones o métodos.
- Restricción del tipo de datos. Está referido a si el tipo de los argumentos está restringido. Dado que todos los lenguajes de información de gestión son tipados, los argumentos de aquéllos que permitan la definición de funciones estarán también restringidos. Es decir, en GDMO, IDL y MOF/CIM se restringe el tipo de datos de los argumentos de las funciones.
- Definiciones operacionales. Se especifica la forma en que se obtienen o infieren valores de argumentos con procedimientos o fórmulas, o definen su semántica mediante axiomas o reglas. Ésta es una de las principales característica de las que carecen los lenguajes de información de gestión. GDMO posee la construcción `BEHAVIOUR`, pero normalmente se emplea como descripción. Algunas propuestas han tratado de formalizar este lenguaje creando un lenguaje de comportamiento específico o usando otros lenguajes de especificación como SDL. El siguiente capítulo trata este tema con más profundidad.

5.2.2.4 Ejemplares

Los ejemplares representan elementos de un concepto, una relación o una aseveración.

- Ejemplares de conceptos. Salvo en los lenguajes normalizados por el DMTF, que permiten su definición de manera explícita, los ejemplares no se especifican en general como parte del lenguaje de definición de información sino como valores que se devuelven ante una operación de gestión.
- Hechos (*facts*). Son ejemplares de relaciones. En este caso, sólo pueden tener hechos aquellos lenguajes que permitan definir relaciones, y sólo se definen de manera explícita en CIM.

- Demandas (*claims*). Representan aseveraciones de un hecho por parte de un ejemplar. Este concepto, común en la Web Semántica, no se incluye en ninguno de los modelos de gestión en estudio.

5.2.2.5 Axiomas, reglas de producción y razonamiento

Los axiomas modelan expresiones que siempre son ciertas, y normalmente se usan para definir restricciones. Las reglas de producción siguen la estructura "si se cumple X, entonces Y", y se usan para expresar un conjunto de acciones. Los procesos de razonamiento se pueden llevar a cabo siguiendo las diferentes relaciones definidas en el conocimiento representado.

Los lenguajes de información de gestión no incluyen ninguno de estos elementos. Esto quiere decir que no es posible definir restricciones ni procedimientos con ellos, lo que sería muy útil para definir comportamientos. Como se comentaba anteriormente, GDMO incluye una plantilla de comportamiento, pero es algo totalmente abierto a cualquier tipo de definición. Otros lenguajes no incluyen siquiera dicha construcción, aunque los mecanismos de extensión existentes en MOF/CIM y SMIng podrían ser útiles para esto.

5.2.3 Tablas comparativas

El apartado anterior ha aplicado los conceptos típicos de las ontologías para analizar la expresividad semántica de los lenguajes de definición de información de gestión. A continuación se presentan dos tablas como resultados de la comparación.

La Tabla 5.1 muestra un resumen de todos los lenguajes comparados en términos de los elementos típicos de las ontologías. Se ha colocado un signo positivo (+) cuando el lenguaje posee ese elemento, y un signo negativo (-) cuando no. También se ha incluido un asterisco (*) cuando el lenguaje tiene una funcionalidad semejante, pero no se puede aplicar directamente.

Como se puede ver, MOF/CIM es el lenguaje que posee una mejor expresividad semántica, con 16 (+1*) elementos de definición de información. SMIng y GDMO se acercan a este nivel, con 10 (+3*) y 8 (+5*) elementos respectivamente.

Analizando la tabla en mayor profundidad se puede ver que hay filas en las que todos o casi todos los lenguajes no poseen ese elemento. Estas filas se refieren principalmente a relaciones taxonómicas y definición de procedimientos o reglas que describan algún tipo de comportamiento, cuestiones que suelen ser normales en los lenguajes de definición de ontologías. Aunque MOF/CIM es el lenguaje con mayor capacidad expresiva, también carece de estas construcciones. Por tanto, habría que incluir este tipo de elementos para añadir mayor expresividad a los lenguajes de información de gestión.

Tabla 5.1. Resumen de las características comparadas.

Lenguaje	GDMO	SMIv2	SMIng	MIF	IDL	MOF/CIM
Conceptos						
Metaclases	-	-	*	-	-	*
Particiones	-	-	-	-	-	-
Atributos						
Ámbito local	-	+	+	+	+	+
Ejemplares de atributo	+	+	+	+	+	+
Atributos de clase	-	-	-	-	-	+
Atributos polimórficos	-	-	+	+	+	+
Facetas						
Valor por defecto	+	+	+	+	-	+
Restricción de tipo	+	+	+	+	+	+
Cardinalidad	-	-	-	-	*	+
Documentación	*	+	+	+	-	+
Definición operacional	-	-	-	-	-	-
Nuevas facetas	-	-	+	*	-	+
Otras facetas, ver Tabla 5.2	+	+	+	+	+	+
Taxonomías						
Subclase de	+	*	+	-	+	+
Descomposición disjunta	-	-	-	-	-	-
Descomposición exhaustiva	-	-	-	-	-	-
No subclase de	-	-	-	-	-	-
Relaciones/funciones						
Relaciones N-arias	+	*	+	-	+	+
Funciones	+	-	-	-	+	+
Restricción de tipo	+	-	-	-	+	+
Definiciones operacionales	*	-	-	-	-	-
Ejemplares						
Ejemplares de conceptos	*	*	*	+	*	+
Hechos	*	*	*	-	*	+
Demandas	-	-	-	-	-	-
Axiomas, reglas	*	-	-	-	-	-

También se han identificado y comparado las facetas más comunes que poseen los lenguajes de información de gestión. La Tabla 5.2 muestra las construcciones empleadas en los lenguajes de información de gestión para incluir facetas de los atributos. Dicha tabla incluye tanto facetas mostradas en la tabla anterior como el resto de facetas que se han considerado comunes en este tipo de lenguajes. Para cada faceta se incluye la construcción

empleada en el lenguaje correspondiente. En aquellos casos en los que no existe para dicho lenguaje, se ha puesto "n/a" (no aplica), y si la faceta se da directamente, indicando su valor adjunto al atributo, se ha puesto "Adjunto".

En esta tabla se puede observar que IDL es el lenguaje peor dotado para expresar estas facetas, debido a que se creó para especificar interfaces de aplicaciones distribuidas y no para definir información de gestión. MOF/CIM, por el contrario, posee construcciones para representar la práctica totalidad de las facetas. El resto de lenguajes va más o menos a la par respecto a las facetas expuestas.

Tabla 5.2. Facetas típicas de los lenguajes de información de gestión.

Lenguaje	GDMO	SMIv2	SMIng	MIF	IDL	MOF/CIM
Valor por defecto	DEFAULT VALUE	DEFVAL	default	value, {}	n/a	=
Restricción de tipo de datos	WITH ATTRIBUTE SYNTAX	SYNTAX	Adjunto	type	Adjunto	Adjunto
Cardinalidad	n/a	n/a	n/a	n/a	<n>	[n], Max, Min
Documentación	BEHAVIOUR	DESCRIPTION	description	description	/* */	Description
Acceso	Adjunto	MAX-ACCESS	access	access	Adjunto	Read, Write
Estado de implementación	n/a	STATUS	status	class	n/a	Version, Revision
Identificador único	REGISTERED AS	::=	n/a	id, pragma snmp	n/a	MappingStrings, URLs
Distinción	WITH ATTRIBUTE	INDEX, AUGMENTS	unique	key	n/a	Key, Propagated, Weak
Unidades	n/a	UNITS	units	n/a	n/a	Units
Referencia	n/a	REFERENCE	reference	n/a	n/a	Model Correspondence
Redefinición	DERIVED FROM	n/a	n/a	n/a	n/a	Override

Una vez que se han analizado las capacidades semánticas de los lenguajes de información de gestión la siguiente sección estudiará qué lenguaje es el más apropiado para la definición de una ontología de información de gestión. Dicho lenguaje deberá, por un lado, poseer la mayor capacidad expresiva posible, y a la vez, modelar todas las facetas que se emplean habitualmente en gestión.

5.3 Mejora de la capacidad semántica de lenguajes de especificación de información de gestión

En la sección anterior se ha realizado un análisis y comparación de la expresividad semántica que aportan los lenguajes de definición de información de gestión. Esto ha permitido, por un lado, ver qué lenguajes son mejores desde este punto de vista, y por otro lado, identificar qué características no poseen que podrían proporcionar una semántica adicional. Asimismo, se han encontrado aquellas facetas que son más comunes en los lenguajes de información de gestión.

Una vez realizado el análisis previo, a continuación se aprovechará dicho estudio para mejorar la capacidad semántica de las especificaciones de información de gestión. Para ello se puede optar por utilizar algún lenguaje de ontologías, o bien un lenguaje de gestión de red, adaptándolo para que la expresividad semántica que posea sea similar a la de dichos lenguajes de ontologías. Se plantean por tanto dos alternativas:

- Los lenguajes de definición de información de gestión capturan las facetas que son útiles al campo de la gestión, pero como se vio en la sección anterior, adolecen de cierta falta de expresividad semántica.
- Por otro lado, los lenguajes de definición de ontologías suelen poseer mayor expresividad semántica, pero a cambio, no incluyen algunas de las facetas típicas y necesarias en el campo de la gestión, ya analizadas previamente.

Esta sección analiza ambas alternativas en los siguientes apartados, presentando en primer lugar otros trabajos relacionados con la formalización de lenguajes de información de gestión, y finalizando con un resumen de aquellas conclusiones que se obtengan de este análisis.

5.3.1 Trabajos relacionados con la formalización de lenguajes de información de gestión

La mayor diferencia entre los lenguajes de ontologías y los lenguajes de definición de información de gestión es su grado de formalización. En este sentido, varios artículos han tratado el tema de la formalización de lenguajes de gestión.

En [Bapat93] se define un modelo formal de relaciones entre clases de objetos gestionados GDMO, para aportar mayor semántica, pudiendo así tener bases de conocimiento de gestión. De esta forma, introduce conceptos tales como la multiplicidad, la conmutatividad, la transitividad y la distribución. Aunque trata de explicar estos conceptos mediante teoría de conjuntos, esta formalización se concreta únicamente en la definición de una plantilla para especificar estas relaciones, con lo que no existe una formalización real de las reglas para definir plantillas GDMO.

También por la misma época, en [Zhang93] se propone un modelo basado en el conocimiento para la gestión de red. Para ello, trata de formalizar con lenguaje matemático las actividades en que intervienen los objetos gestionados y los informes de eventos de un entorno de gestión OSI. Sin embargo, no amplía esta formalización a las construcciones de los objetos. Posteriormente, en [Zhang96a] se analiza la semántica de la información de gestión, definiendo restricciones específicas de cada contexto que pueden relacionarse de manera jerárquica. Este caso es similar al anterior, pues se basa en las actividades de gestión, por lo que no aporta ningún grado de formalización a la información en sí misma.

Finalmente, y citando otros trabajos que hacen uso de lenguajes de ontologías para la gestión, en [Ensel01] se propone usar RDF para describir, consultar y calcular dependencias entre servicios en un sistema distribuido que se gestione a través del Web. Aunque este trabajo habla de la integración de RDF con CIM, lo deja como trabajo futuro, con lo que no llega a aplicar esta solución a la definición de información de gestión. Además, se limita al uso de RDFS, sin utilizar un lenguaje más propio de ontologías como DAML+OIL, y tampoco realiza extensiones a RDFS para expresar facetas típicas de la gestión como el tipo de acceso a los atributos. En [Shen03] se propone una traducción entre SMIng y RDF para que lo puedan usar agentes inteligentes. Al mismo tiempo, propone una formalización muy resumida de SMIng basada en un análisis matemático de los árboles de OIDs, citando a [Shen01] como una ampliación de la misma. Este trabajo es bastante interesante, pero no hace referencia a otros lenguajes de información como MOF/CIM, que como se ha visto posee mejores características. Aunque en este caso define en RDF términos referidos a SMIng, no lo hace para el caso general de cualquier lenguaje de gestión. Por último, [Lavinal03] trata de integrar el metamodelo de CIM dentro de una herramienta de ontologías, que es una aproximación inversa a la de aprovechar los lenguajes de ontologías, que poseen mayor expresividad. La justificación de incluir este metamodelo en la herramienta de ontologías es poder utilizar la información definida en CIM para el intercambio de información entre agentes inteligentes usando OKBC. Sin embargo, esto no implica formalizar el metamodelo de CIM, y además, la integración que se consigue con esta aproximación es parcial, puesto que es de tipo técnico.

5.3.2 Propuesta de mejora de un lenguaje de información de gestión con características propias de los lenguajes de ontologías

En la sección anterior se concluyó que MOF/CIM era, de los lenguajes analizados, el que proporcionaba un mayor número de las características que suelen ir incluidas en lenguajes de ontologías. Por otro lado, al comparar la forma en que se define la información en CIM con respecto a las ontologías, se pudo ver en la Figura 4.1 que eran claramente similares, y que según la forma en que se estructuran los esquemas de CIM, podría considerarse una ontología de no ser por su falta de formalismo. Por otro lado, CIM emplea los diagramas de clase de UML para el modelado de información de gestión. Al mismo tiempo, el

apartado 4.3.3 presentaba diversos trabajos que identifican UML como un lenguaje válido para definir ontologías.

Este conjunto de razones hace ver a MOF/CIM como un buen candidato para definir la información de gestión desde un punto de vista semántico. Hay, sin embargo, un conjunto de problemas que hay que solventar para conseguir este objetivo:

- Al ser MOF/CIM un lenguaje no formal, sigue sin ser válido para definir ontologías pesadas. Por tanto, es necesario formalizarlo. Para ello, se puede emplear el lenguaje de restricciones empleado en UML, conocido como OCL (*Object Constraint Language*, Lenguaje de Restricciones de Objetos), rescribiendo las reglas que definen el metamodelo de CIM, evitando las ambigüedades existentes debidas a estar escritas en lenguaje natural.
- Las carencias detectadas en la sección anterior para este lenguaje se pueden solventar mediante la definición de calificadores adicionales a los que ya posee, que permitan definir, por ejemplo, a una clase como disjunta de otra.
- Finalmente, CIM también carece de una ontología general común, referida a tipos de datos o unidades de medida. Este tipo de ontologías puede ser útil para comparar dos conceptos similares, pero definidos con unidades diferentes (números enteros frente a decimales ó bits por segundo frente a Kilobits por segundo, por ejemplo).

5.3.2.1 Formalización de MOF/CIM como lenguaje de ontologías

Este apartado presenta como contribución original una especificación formal del metamodelo de CIM. Para ello se han definido en OCL el conjunto de reglas definidas en inglés en la especificación de CIM 2.2 [DMTF99]. Durante este proceso se encontraron algunas incongruencias entre las reglas existentes que han tratado de ser solventadas. Esta formalización no supone ninguna modificación en la información que ya existe definida en CIM, sino que por el contrario permite su validación a partir de las restricciones OCL.

La semántica de MOF/CIM está definida en su metamodelo o metaesquema CIM, que como se vio en el apartado 2.7.2 modela los elementos existentes en este lenguaje a partir de su representación con un diagrama de clases de UML y un conjunto de reglas definidas en lenguaje natural.

Podría pensarse que el metamodelo de CIM es directamente el de UML [OMG01b], pero esto no es así. Según se aprecia en la Tabla 5.3, que compara CIM con el modelo de cuatro niveles empleado en UML, el meta-metamodelo de CIM, o lenguaje empleado para definir el metaesquema de CIM, es directamente UML; el metamodelo, o lenguaje empleado para definir los modelos, es el Metaesquema CIM; los modelos son los esquemas CIM y los objetos de usuario son ejemplares de los esquemas CIM.

Tabla 5.3. Comparación entre niveles de UML y CIM.

Nivel	UML	CIM
Meta-metamodelo	Meta-metamodelo OMG-MOF	UML
Metamodelo	Metamodelo UML	Metaesquema CIM
Modelo	Modelos UML	Esquemas CIM
Objetos de usuarios	Ejemplares de Modelos UML	Ejemplares de clases.

Sin embargo, esto no quiere decir que el metamodelo de CIM no tenga cierto parecido con el de UML. La Figura 5.1 muestra un subconjunto del metamodelo de UML 1.4 que incluye un conjunto de elementos similares a los del metaesquema de CIM, presentado con anterioridad en la Figura 2.14. El número de elementos es claramente distinto, existiendo muchos más grados de especialización, si bien se sigue una estructura parecida.

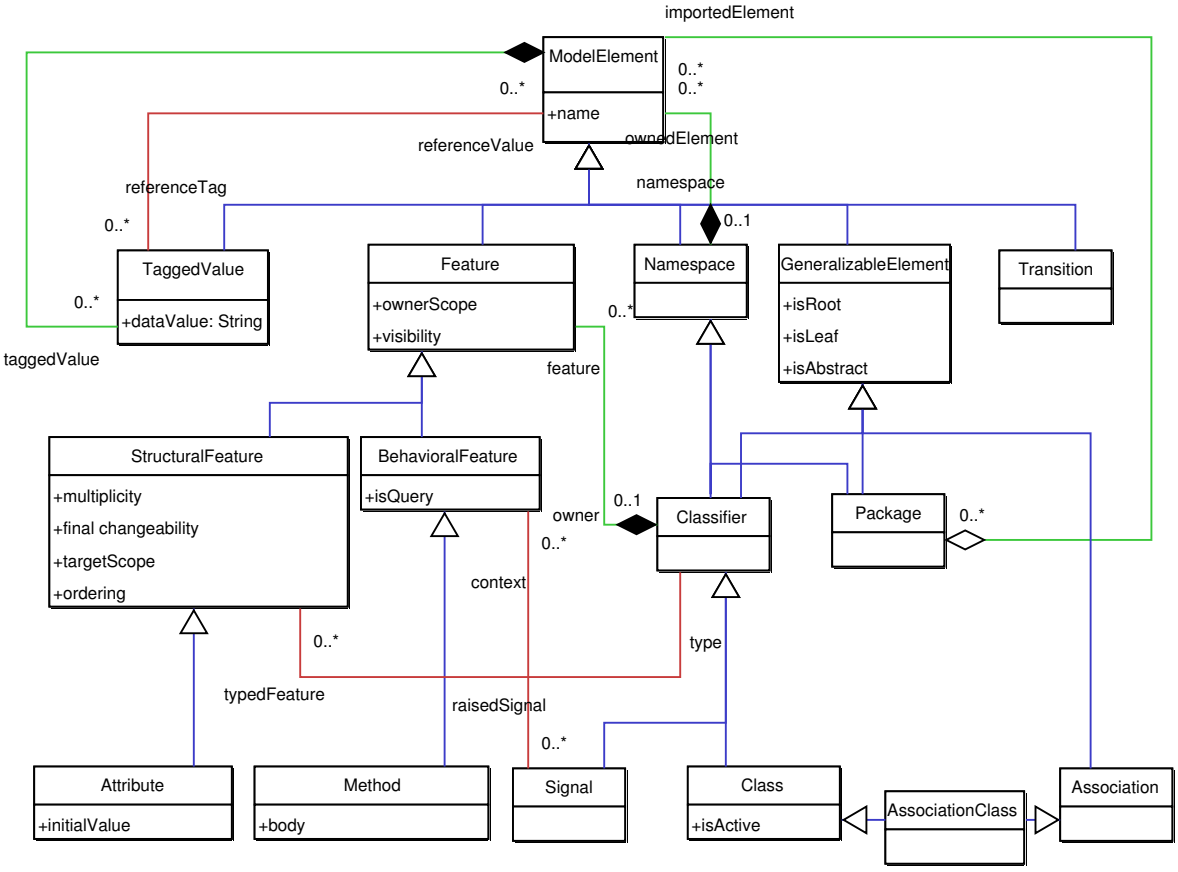

Figura 5.1. Subconjunto del metamodelo de UML 1.4.

La Figura 5.2 presenta la formalización realizada sobre el metaesquema de CIM.

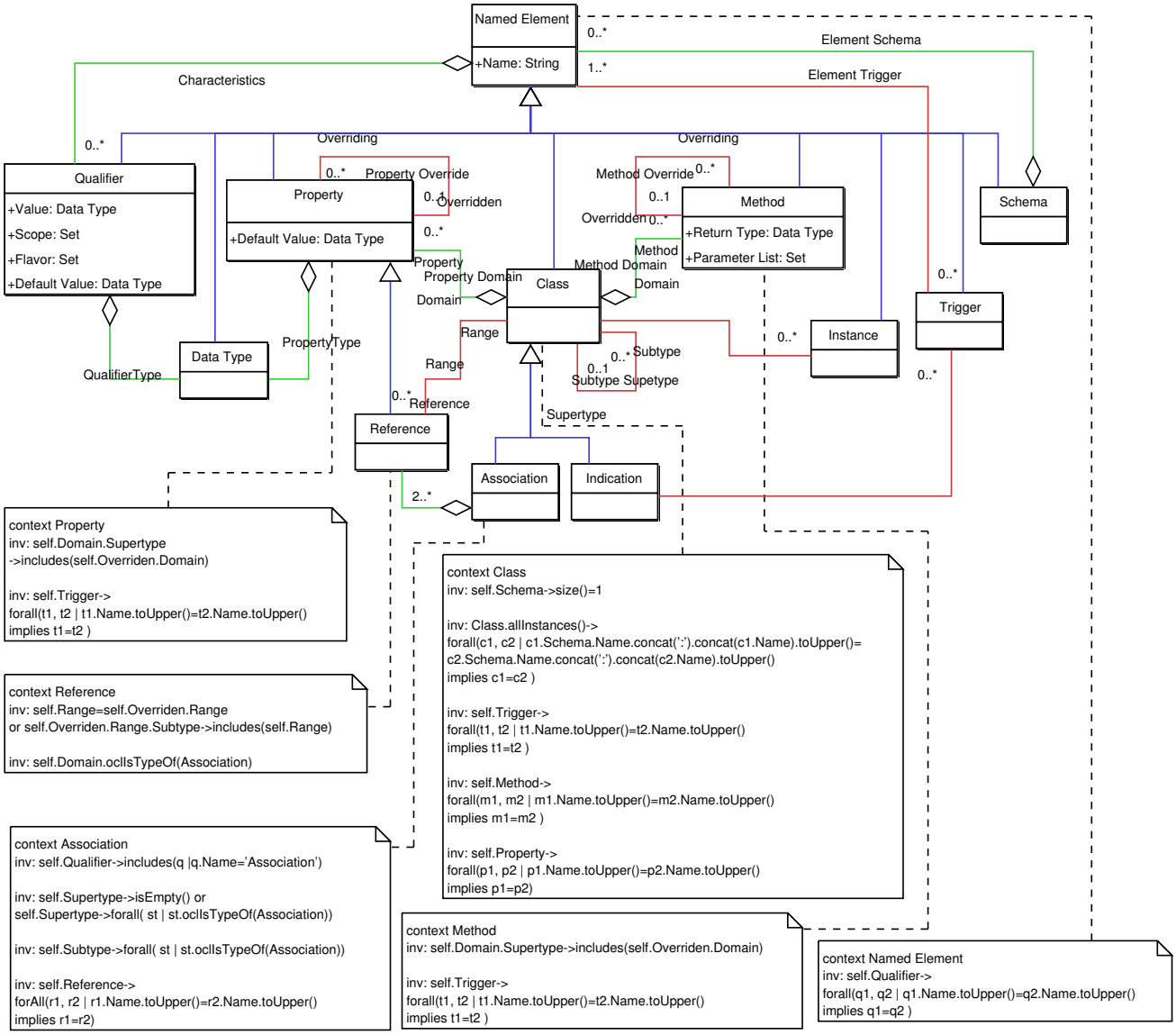

Figura 5.2. Metamodelo de CIM formalizado con OCL.

Para llevar a cabo esta formalización se ha tenido en cuenta toda la información que se aportaba en [DMTF99]. Dicho documento especifica por un lado, el metaesquema de CIM a partir de un diagrama de clases UML. Después, aporta un conjunto de reglas escritas en lenguaje natural (y por tanto, no formal). Finalmente especifica en ABNF (*Augmented Backus-Naur Form*, Forma Aumentada Backus-Naur) la sintaxis de MOF. Este documento necesitaría una revisión profunda por parte del DMTF, debido a los conflictos que se han encontrado en su especificación:

1. El diagrama UML que define los elementos resulta estar incompleto, no sólo por no incluir las restricciones que afectan a cada elemento, sino por carecer de algunos elementos nombrados en las reglas escritas en lenguaje natural o bien en la especificación de la sintaxis de MOF/CIM, como por ejemplo, los ejemplares.

2. Algunas de las reglas escritas en lenguaje natural son incongruentes entre sí, definiendo para algún elemento distintas propiedades en distintas reglas (estas incongruencias aparecen sobre todo en aquellas reglas referidas al elemento `Qualifier`).
3. Otras reglas son redundantes con respecto al diagrama de clases, por lo que no es necesaria su definición, como aquéllas que se refieren a la especialización de elementos, o a la cardinalidad de sus relaciones.

Así pues, el diagrama formalizado que se presenta incluye las siguientes cuestiones:

1. Se han añadido aquellos elementos que se ha creído necesario por aparecer tanto en las reglas como en la sintaxis de MOF/CIM. Éstos incluyen el tipo de datos, los ejemplares, y algunas asociaciones entre elementos no incluidas previamente en el diagrama.
2. Se han añadido nombres a las terminaciones de las asociaciones que empiezan y acaban en un mismo elemento, para mejorar la semántica del diagrama y facilitar la definición de reglas. Para el resto de asociaciones se entiende que el nombre de la terminación es el del elemento asociado, por lo que no ha sido necesario añadirlo.
3. Todas aquellas reglas definidas en inglés que eran formalizables han sido planteadas en OCL, mediante la definición de invariantes dentro del contexto de cada elemento. Otras reglas, que indicaban la utilidad de cada elemento, no se han podido formalizar.
4. En los casos de incongruencias, se ha tratado de buscar la mejor solución posible, si bien estas incongruencias deberían ser resueltas por el DMTF.
5. En los casos de redundancia se ha optado por no incluir dichas reglas en el diagrama pues no aporta nada su inclusión.

El Apéndice A incluye la definición en OCL de las reglas de formalización del metaesquema de CIM.

5.3.2.2 Calificadores adicionales

El análisis efectuado en la sección anterior señalaba que las carencias de MOF/CIM se debían sobre todo a la definición de taxonomías. También se apuntaba que el uso de calificadores podría permitir aumentar estas definiciones. Así, se podrían crear sendos calificadores para definir clases que posean descomposiciones disjuntas o exhaustivas, de la siguiente forma:

```
Qualifier DisjointDecomposition : boolean = false, Scope (class);
Qualifier ExhaustiveDecomposition: boolean = false, Scope (class);
```

Así, si una clase posee uno de estos calificadores, se entenderá que sus subclases son disjuntas o exhaustivas respectivamente.

El problema que supone esta definición es que la semántica de los calificadores no se formaliza, como sí ocurre, por ejemplo, con la Frame Ontology[2] de Ontolingua, en la que cada elemento del lenguaje está formalizado en KIF (*Knowledge Interchange Format*, Formato de Intercambio de Conocimiento). Por tanto, sería necesario llevar a cabo una formalización para cada uno de los calificadores al estilo de la realizada en el apartado anterior, estableciendo qué invariantes deben cumplirse para cada elemento que los posea.

Otras cuestiones, relativas a la definición de operaciones, axiomas y reglas, se verán en el capítulo siguiente al tratar la definición de comportamiento dentro de las definiciones de información de gestión.

5.3.2.3 Ontología general común

Cuando se comparó la arquitectura de ontologías con CIM en la Figura 4.1 se vio que CIM no poseía una ontología general común, entendida como aquella que define conceptos generales tales como tipos de datos, o unidades de medida.

En realidad, CIM sí define tipos de datos y unidades de medida, pero no lo hace de una manera formal, sino como una simple enumeración de nombres. Así, los tipos de datos posibles son:

- Enteros
    - con signo, de 8, 16, 32 y 64 bits de precisión (sint8, sint16, sint32, sint64)
    - sin signo, de 8, 16, 32 y 64 bits de precisión (uint8, uint16, uint32, uint64)
- Decimales de 4 y 8 octetos de precisión (real32, real 64)
- Booleanos (boolean)
- Fecha (datetime)
- Cadena de caracteres (string)
- Caracteres simples (char16).

Al mismo tiempo, las unidades de medida permitidas incluyen:

- Información. Bits, KiloBits, MegaBits, GigaBits
- Caudal de información. Bits per Second

[2] http://www-ksl-svc.stanford.edu:5915/FRAME-EDITOR/UID-174&sid=ANONYMOUS&user-id=ALIEN

- Memoria. Bytes, KiloBytes, MegaBytes, GigaBytes, Words, DoubleWords, QuadWords
- Temperatura. Degrees C, Tenths of Degrees C, Hundredths of Degrees C, Degrees F, Tenths of Degrees F, Hundredths of Degrees F, Degrees K, Tenths of Degrees K, Hundredths of Degrees K, Color Temperature
- Energía eléctrica. Volts, MilliVolts, Tenths of MilliVolts, Amps, MilliAmps, Tenths of MilliAmps, Watts, MilliWattHours
- Energía y fuerza. Joules, Coulombs, Newtons
- Luz. Lumen, Lux, Candelas
- Peso. Pounds, Pounds per Square Inch
- Revoluciones. Cycles, Revolutions, Revolutions per Minute, Revolutions per Second
- Tiempo. Minutes, Seconds, Tenths of Seconds, Hundredths of Seconds, MicroSeconds, MilliSeconds, NanoSeconds, Hours, Days, Weeks
- Frecuencia. Hertz, MegaHertz
- Imágenes. Pixels, Pixels per Inch
- Porcentajes. Percent, Tenths of Percent, Hundredths of Percent
- Longitudes. Meters, Centimeters, Millimeters, Cubic Meters, Cubic Centimeters, Cubic Millimeters
- Longitudes según el sistema anglosajón. Inches, Feet, Cubic Inches, Cubic Feet Ounces, Liters, Fluid Ounces
- Grados. Radians, Steradians, Degrees
- Campos electromagnéticos. Gauss, Gilberts, Henrys, MilliHenrys, Farads, MilliFarads, MicroFarads, PicoFarads
- Resistencia. Ohms, Siemens
- Concentración. Moles, Becquerels, Parts per Million
- Relación de potencia. Decibels, Tenths of Decibels

Formalizar estas definiciones es útil si se quieren comparar conceptos. Así, por ejemplo se podría establecer una correspondencia entre dos conceptos que midieran caudal de información, aunque uno lo mida en bits por segundo y el otro en Megabits por segundo, o aunque uno realice la medida con números enteros de 64 bits y otro con números decimales. Para ello se puede aprovechar gran parte de las definiciones de las bibliotecas

de ontologías existentes. Por ejemplo, la biblioteca de Ontolingua incluye una ontología de Números (KIF-NUMBERS[3]) y otra de Unidades Estándar (STANDARD-UNITS[4]). La ontología de Números de Ontolingua no sólo hace una clasificación de números, sino que también define relaciones y funciones. Por otro lado, la ontología de Unidades Estándar define las relaciones entre estas unidades (por ejemplo, que un Kilobyte son 1024 bytes).

5.3.3 Adaptación de un lenguaje de ontologías para definir información de gestión

Los lenguajes de ontologías actualmente en mayor uso y con mayor número de herramientas disponibles son aquéllos relacionados con la Web Semántica. Como ya se vio en el apartado 4.3.2.3, el W3C, uno de los organismos promotores de esta idea, ha definido DAML+OIL [Connolly01], y trabaja en el momento de escribir esta tesis en el borrador de OWL Web Ontology Language y OWL Lite [Dean02], siendo OWL una revisión de DAML+OIL, y OWL Lite una versión reducida de OWL. DAML+OIL queda bastante bien posicionado en la comparación expuesta en [Gómez02], donde se hace un análisis de los distintos lenguajes de ontologías definidos para la Web Semántica. Este hecho, junto a que DAML+OIL sea el lenguaje que proponen los organismos de estandarización hace que se plantee su estudio como ejemplo de adaptación de lenguaje de ontologías para definir información de gestión. Aunque este análisis también se podría realizar con otros lenguajes no orientados a la Web Semántica, se ha preferido hacerlo con éste por la actualidad del mismo y para llegar a la mayor audiencia posible.

La ventaja de este lenguaje frente a MOF/CIM, estudiado en el apartado anterior, es que DAML+OIL está formalizado en KIF [Fikes01], con lo que su semántica es unívoca y puede ser usada por sistemas inteligentes. Asimismo, existen múltiples bibliotecas de ontologías[5] que se pueden reutilizar para tener ontologías generales comunes. Por ejemplo, se utilizan los tipos de datos XSD (*XML Schema Datatyes*, Tipos de datos de Esquemas XML) [Biron01] para definir las propiedades tipadas. Por el contrario, DAML+OIL no es un lenguaje específico de gestión, por lo que no posee construcciones para definir todas las facetas típicas de gestión, presentadas en la Tabla 5.2, y además, existe gran cantidad de información de gestión, que habría que traducir a DAML+OIL. Por otro lado, tampoco permite definir métodos u operaciones, aunque en general, las bases de información de gestión no suelen tener muchas de estas construcciones.

[3] http://www-ksl-svc.stanford.edu:5915/FRAME-EDITOR/UID-185&sid=ANONYMOUS&user-id=ALIEN

[4] http://www-ksl-svc.stanford.edu:5915/FRAME-EDITOR/UID-203&sid=ANONYMOUS&user-id=ALIEN

[5] http://www.daml.org/ontologies/

El estudio que se muestra en este apartado se puede generalizar para otros lenguajes de definición de ontologías con características similares. Todos aquellos lenguajes que permitan definir clases y propiedades pueden valer para definir información de gestión, si bien es posible que se pierda claridad si estos lenguajes no poseen las facetas adecuadas, ni un mecanismo para definir nuevas facetas.

A continuación se estudia cómo se puede definir información de gestión con DAML+OIL, y más adelante se verá cómo expresar facetas típicas de gestión en este lenguaje.

5.3.3.1 Definición de conceptos y taxonomías de gestión con DAML+OIL

Según el estudio de [Gómez02], anteriormente comentado, DAML+OIL es un lenguaje de ontologías bastante completo, pues incluye:

- En lo que se refiere a conceptos:
  - La definición de particiones, así como documentación de la clase.
  - Se pueden definir atributos, aunque no es posible distinguir entre atributos de clase (que serían estáticos) y de ejemplar. Los atributos se pueden definir para un ámbito local (definiendo el dominio al que pertenecen) o global (si no se especifica dicho dominio).
  - Proporciona facetas tales como la restricción de tipo o de cardinalidad, así como la documentación de cada atributo. No incluye, sin embargo, un valor por defecto u otras facetas, que deberían añadirse como una extensión.
- En lo que se refiere a taxonomías, permite definir subclases, con herencia múltiple, así como descomposiciones exhaustivas y disjuntas. Asimismo, se puede definir una clase como no subclase de otra (definiéndola como complementaria).
- En lo que se refiere a la definición de relaciones y funciones, se pueden definir relaciones con restricción de rango. Sin embargo, no se pueden expresar funciones, y tampoco definiciones operacionales, cuestión esta última que también faltaba en CIM.
- Respecto a axiomas, DAML+OIL permite definir axiomas con lógica de primer orden respecto a propiedades algebraicas de relaciones (simetría, transitividad, unicidad). Además, se pueden definir restricciones de universalidad y existencia para las clases y propiedades.
- Finalmente, en lo que se refiere a ejemplares, DAML+OIL es capaz de definir ejemplares de conceptos, de relaciones (o hechos) y también demandas.

Así, en rasgos generales, este lenguaje de ontologías se puede emplear directamente para la definición de información de gestión, puesto que en general posee las construcciones habituales en los lenguajes de información de gestión.

Como ejemplo, a continuación se presenta una parte de un documento que contiene la traducción a DAML+OIL de la clase `CIM_ManagedSystemElement`, definida en el modelo nuclear de CIM [DMTF00], según se muestra en la Figura 5.3.

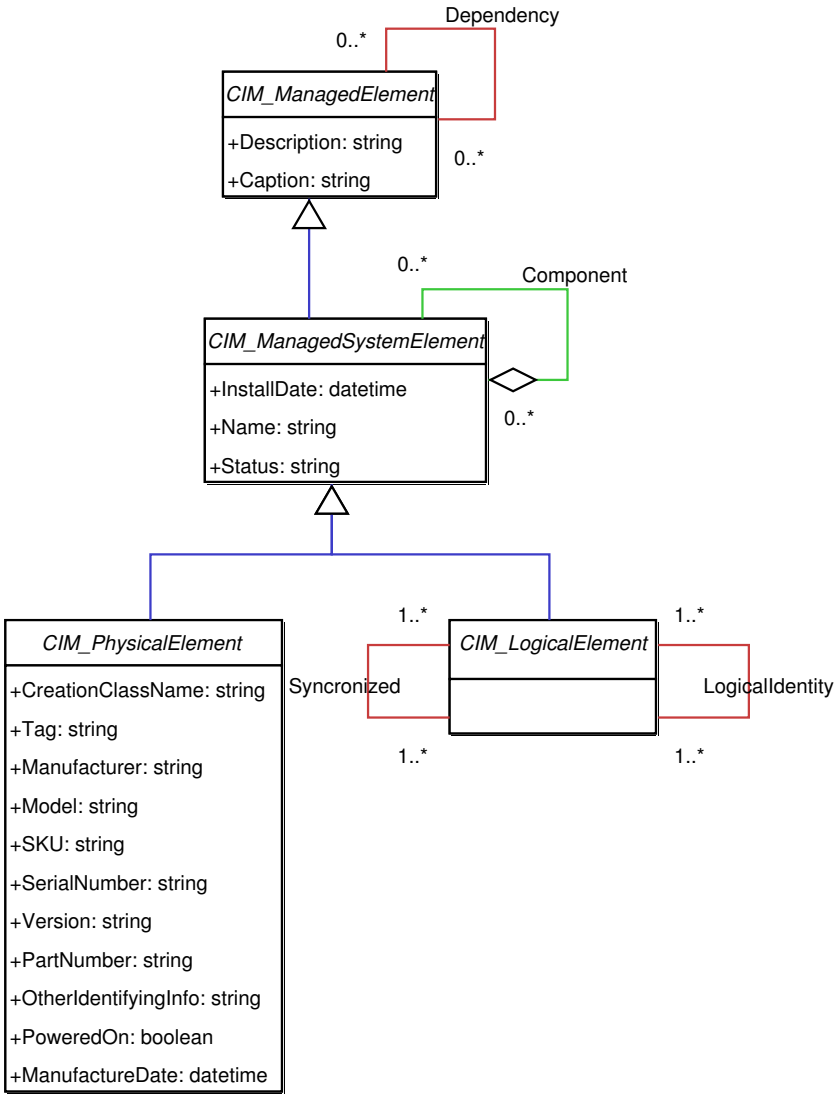

Figura 5.3. Subconjunto del Modelo Nuclear de CIM.

Primero, se definiría el documento XML con sintaxis RDF, y un conjunto de espacios de nombres para las distintas marcas o *tags*:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF
      xmlns:xsd ="http://www.w3.org/2000/10/XMLSchema#"
      xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:daml ="http://www.daml.org/2001/03/daml+oil#"
      xmlns:cim ="cim#"
      xmlns:rdfs ="http://www.w3.org/2000/01/rdf-schema#"
```

```
      xmlns ="cim#"
>
```

Tras esto, se definiría la cabecera con información general de la ontología:

```
<daml:Ontology rdf:ID="CIM_Core">
      <daml:versionInfo>2.6</daml:versionInfo>
      <rdfs:comment>The version 2.6 of CIM Core model</rdfs:comment>
</daml:Ontology>
```

A continuación, se definiría la clase, como subclase de `CIM_ManagedElement`, que también estaría definida en este documento:

```
<daml:Class rdf:ID="CIM_ManagedSystemElement">
      <rdfs:comment>
CIM_ManagedSystemElement is the base class for the System Element hierarchy. [...]
      </rdfs:comment>
      <rdfs:subClassOf rdf:resource="#CIM_ManagedElement"/>
</daml:Class>
```

Más adelante, se definirían las propiedades asociadas a la clase, con sus tipos de datos en XSD.

```
<daml:DatatypeProperty rdf:ID="InstallDate">
      <rdfs:comment>
A datetime value indicating when the object was installed. A lack of a value does not indicate that the object is not installed.
      </rdfs:comment>
      <daml:domain rdf:resource="#CIM_ManagedSystemElement"/>
      <daml:range
rdf:resource="http://www.w3.org/2000/10/XMLSchema#dateTime"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="Name">
      <rdfs:comment>
The Name property defines the label by which the object is known. When subclassed, the Name property can be overridden to be a Key property.
      </rdfs:comment>
      <daml:domain rdf:resource="#CIM_ManagedSystemElement"/>
      <daml:range
rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="Status">
      <rdfs:comment>
```

```
A string indicating the current status of the object. Various operational and non-operational statuses are defined. […]
      </rdfs:comment>
      <daml:domain rdf:resource="#CIM_ManagedSystemElement"/>
      <daml:range
rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml:DatatypeProperty>
```

Finalmente, se acabaría el documento RDF.

```
</rdf:RDF>
```

Esta traducción es generalizable a la totalidad de las clases definidas en el modelo nuclear de CIM. Sin embargo, no todos los calificadores de las propiedades de las clases se pueden expresar directamente en DAML+OIL. Por ejemplo, el atributo `Name` se define en MOF/CIM con una longitud máxima de 256 caracteres, cuestión que no queda aquí definida. El siguiente apartado trata este tema, explicando cómo se podrían expresar las facetas típicas de gestión en DAML+OIL.

5.3.3.2 Ampliación de DAML+OIL con facetas de gestión

El desarrollo de DAML+OIL ha seguido un modelo basado en capas. De hecho, al desarrollarse OIL, su predecesor, se propuso que éste pudiera tener varios niveles. Así, si se usa un sistema que sólo implementaba el nivel más básico de OIL, podría entender parte de la ontología definida con este lenguaje [Fensel01].

Siguiendo este razonamiento, para ampliar DAML+OIL con facetas de gestión se propone añadir un nivel a los ya existentes, según se muestra en la Figura 5.4, que amplíe la capacidad expresiva de este lenguaje con facetas de gestión. Una definición que emplee esta aproximación permite que la información sea lo más completa posible, y también que los sistemas que pueden interpretar DAML+OIL carguen información de gestión, aunque no entiendan toda la semántica contenida en el documento.

Como se indicaba en la Tabla 5.2, las facetas típicas de lenguaje de información de gestión incluyen: valor por defecto, restricción de tipo de datos, cardinalidad, documentación, acceso, estado de implementación, identificador único, distinción, unidades, referencia y redefinición. Para poder expresar dichas facetas, muchas veces bastará con usar las construcciones que posee DAML+OIL, mientras que para el resto de casos habrá que definir otras nuevas.

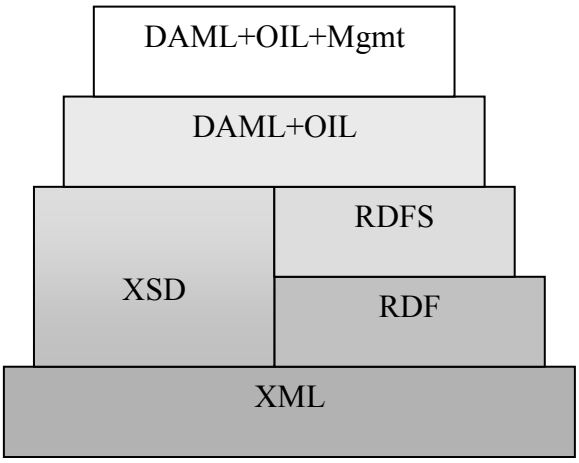

Figura 5.4. Capas de DAML+OIL.

A continuación se enumeran las facetas ya incluidas en DAML+OIL. Se ha supuesto que se usa el mismo espacio de nombres XML definido en el apartado anterior:

- Restricción de tipo de datos: `rdfs:range` permite restringir el conjunto de valores que puede tomar una propiedad. Además, las propiedades pueden ser `daml:ObjectProperty`, referidas a ejemplares de una clase, y `daml:DatatypeProperty`, referida a tipos de datos literales. Los tipos de datos de esta última clase de propiedades pueden ser tipos de datos XSD, como ya se pudo ver en la especificación mostrada en el apartado anterior. Además, esto supone que se puede restringir el rango de los tipos según las facetas que éstos posean. Por ejemplo, una cadena de caracteres cuya longitud máxima sea 64 podría definirse como:

```
<xsd:simpleType name="string64">
      <xsd:restriction base="xsd:string">
            <xsd:maxLength value="64" />
      </xsd:restriction>
</xsd:simpleType>
```

- Cardinalidad. DAML+OIL define varias construcciones relativas a la cardinalidad. Por ejemplo, `daml:cardinality`, `daml:maxCardinality`, `daml:minCardinality` definen cardinalidad exacta, máxima y mínima respectivamente.

- Documentación. `rdfs:comment` permite incluir una descripción del recurso que se está definiendo, como ya se vio en la definición del apartado anterior.

- Estado de implementación. DAML+OIL posee la marca `daml:versionInfo` que se coloca en la cabecera del documento que permite definir información relativa a la versión, como también se vio anteriormente. Esta marca no tiene restringido el dominio de aplicación, por lo que se podría utilizar para otros elementos.

- Identificador único. Viene dado por la etiqueta `rdf:ID`, que proporciona la referencia a un recurso dentro de un documento. Dicho documento normalmente poseerá un URI (*Uniform Resource Identifier*, Identificador Uniforme de Recursos), por lo que el URI concatenado con la referencia permitirá identificar unívocamente a clases y propiedades.
- Distinción. También se puede emplear la etiqueta `rdf:ID` para identificar unívocamente cada ejemplar, por lo que no es necesario establecer atributos de distinción. En cualquier caso, también se puede utilizar la construcción `daml:UnambigousProperty`, que identifica que el valor que tome una propiedad debe ser única para cada ejemplar.
- Referencia. `rdfs:seeAlso` permite especificar un recurso que proporcione información adicional, y podría entenderse como una referencia.
- Redefinición. Una propiedad se puede asociar a la etiqueta `rdfs:subPropertyOf` para indicar que especializa otra propiedad.

Otras facetas típicas de gestión se podrían definir como extensiones, como se adelantaba previamente. Se puede definir un documento en el que estén especificadas en RDFS aquellas construcciones necesarias para contener este tipo de facetas, al estilo en que está definido el propio DAML+OIL. A continuación se proponen definiciones de facetas no incluidas en DAML+OIL:

- Valor por defecto. Para definir el valor por defecto se puede definir una nueva propiedad RDF como la que se muestra a continuación, cuyo dominio sea cualquier tipo de propiedad, y sin restricción de rango, para poder expresar cualquier tipo de datos.

```
<rdf:Property rdf:ID="defaultValue">
      <rdfs:label>defaultValue</rdfs:label>
      <rdfs:comment>
It defines the default value of a property.
      </rdfs:comment>
      <rdfs:domain rdf:resource=
            "http://www.daml.org/2001/03/daml+oil#DatatypeProperty"/>
</rdf:Property>
```

- Acceso. Para definir el acceso, se procede como en el caso anterior, si bien aquí se restringe el rango a las distintas posibilidades de acceso. Para ello se define un nuevo tipo de datos, una restricción de cadenas de caracteres cuyo valor sea uno de una enumeración de accesos posibles.

```
<xsd:simpleType name="accessString">
      <xsd:restriction base="string">
```

```
            <xsd:enumeration value="read-only"/>
            <xsd:enumeration value="read-write"/>
            <xsd:enumeration value="read-create"/>
      </xsd:restriction>
</xsd:simpleType>

<rdf:Property rdf:ID="access">
      <rdfs:label>access</rdfs:label>
      <rdfs:comment>
It defines the access of a property, which can be read or write.
      </rdfs:comment>
      <rdfs:domain rdf:resource=
            "http://www.daml.org/2001/03/daml+oil#DatatypeProperty"/>
      <rdfs:range rdf:resource="#accessString"/>
</rdf:Property>
```

- Unidades. En este caso, el dominio sólo podrá estar formado por propiedades referidas a tipos de datos. Como rango se podría utilizar alguna ontología de unidades de medida como la *GNU Units Ontology*[6], contenida en la biblioteca de DAML. Los valores que pudieran tomar esta faceta serían ejemplares de la ontología de unidades.

```
<rdf:Property rdf:ID="units">
      <rdfs:label>units</rdfs:label>
      <rdfs:comment>
It defines the measure units of a property.
      </rdfs:comment>
      <rdfs:domain rdf:resource=
            "http://www.daml.org/2001/03/daml+oil#DatatypeProperty"/>
      <rdfs:range rdf:resource=
            "http://www.daml.org/2002/10/units/units-ont.daml#Unit"/>
</rdf:Property>
```

En el caso en que fuera necesario definir otros calificadores, se procedería de manera similar a la aquí expuesta.

5.3.4 Análisis de las posibles soluciones

Esta sección ha presentado dos alternativas para especificar información de gestión aprovechando las características semánticas de los lenguajes de ontologías, lo que es útil a la hora de integrar la información de distintos dominios de gestión.

[6] http://www.daml.org/2002/10/units/units-ont.daml

La primera posibilidad estudiada ha consistido en adaptar un lenguaje de definición de información de gestión, tratando de corregir todos los defectos que le impidieran ser un lenguaje de ontologías válido. Así, se han practicado las siguientes mejoras a MOF/CIM, lenguaje que en la sección anterior ha resultado mejor posicionado en la comparación realizada:

- Formalización de su metamodelo, mediante la especificación en OCL de las reglas que tenía definidas en lenguaje natural.
- Definición de nuevos calificadores, que permitirán realizar una especificación más completa de las taxonomías.
- Asimismo, se ha propuesto el uso de ontologías generales comunes, que son más formales que la lista de tipos de datos y unidades existente.

Por otro lado, la segunda alternativa ha analizado cómo se podría especificar información de gestión con un lenguaje de definición de ontologías. Para ello se ha utilizado DAML+OIL, lenguaje de plena actualidad por su aplicación en la Web Semántica. Dos han sido los pasos realizados en este punto:

- Comprobar, mediante el estudio del lenguaje y un ejemplo, que se puede utilizar para definir información de gestión.
- Definir cómo se construirían los calificadores típicos en los lenguajes de gestión para DAML+OIL.

Cada una de las posibilidades presentadas puede ser válida dependiendo del ámbito de aplicación. Ambas proporcionan distintas ventajas e inconvenientes que se deben tener en cuenta a la hora de escoger el lenguaje más adecuado para cada caso:

- Usar la versión formal de MOF/CIM permite una transición suave desde el dominio de la gestión de red al de las ontologías. Además, al haberse formalizado el metamodelo, no es necesario traducir a otro lenguaje la información contenida en los esquemas CIM, que ahora se podrá validar a partir de las reglas definidas en OCL. Sin embargo, no existen en la actualidad herramientas que permitan trabajar con este lenguaje desde un punto de vista semántico.
- Usar un lenguaje de ontologías como DAML+OIL proporciona toda la capacidad expresiva de este tipo de lenguajes, existiendo asimismo múltiples herramientas desarrolladas para su uso y validación. Sin embargo, su mayor inconveniente es tener que traducir a este lenguaje toda la información de gestión definida actualmente. Además, DAML+OIL no permite la definición de métodos sobre objetos, con lo que se perdería parte de la información actualmente definida.

Una vez que se han analizado las posibilidades para mejorar la capacidad semántica de los lenguajes de especificación de información de gestión, la sección siguiente tratará sobre

cómo llevar a cabo una integración semántica de todos los modelos de información de gestión existentes.

5.4 Diseño y propuesta de mecanismos de integración semántica de modelos de información

Gracias a los estudios existentes en el campo de las ontologías, en las secciones anteriores se ha podido hacer una comparación de la expresividad de los lenguajes de información de gestión, definiendo el conjunto de términos que permiten definir dicha información. Además, se ha analizado cómo se podría definir información de gestión mediante lenguajes de ontologías o mejorar los lenguajes de información de gestión añadiéndole a éstos las características de aquéllos.

A continuación se abordará, como contribución original de esta tesis doctoral, cómo integrar los actuales modelos de información de gestión aprovechando el conjunto de técnicas propuestas para el campo de las ontologías. Para ello, se utilizarán los estudios realizados en las dos secciones anteriores, pues será necesario tener un lenguaje único que pueda tener las definiciones existentes y que tenga la suficiente capacidad expresiva que permita realizar esta integración de la manera más sencilla posible. Con esto, se establecerán los mecanismos que posibiliten la integración semántica entre los distintos modelos de información de gestión existentes, permitiendo la fusión de modelos pertenecientes a distintos dominios de gestión, y estableciendo de manera unívoca las correspondencias necesarias para que un sistema de gestión pueda obtener la información de dichos dominios a partir del modelo fusionado.

Como ya se ha explicado anteriormente, en Gestión de Red existen múltiples modelos integrados que definen diferentes protocolos y lenguajes, y que poseen múltiples agentes desplegados. A pesar de esta heterogeneidad es necesario llevar a cabo una gestión unificada de dichos modelos para que un sistema que gestione todos los recursos pueda relacionar la información que obtiene de cada uno de los recursos que administra, pues esto redunda en una visión más clara del estado general de la red, sistema, aplicación o servicio gestionado. Por lo tanto, es preciso poseer un modelo común a todos, escrito en un lenguaje único. A la vez, existe la necesidad de poder hacer una traducción semántica o de dominio (frente a traducciones sintácticas) de los modelos que implementan los agentes que hay desplegados. Para ello, a la hora de realizar la integración de modelos de información (entendiendo dichos modelos como bases de información de gestión), se propone llevar a cabo un conjunto de pasos:

1. El paso previo a la integración semántica es realizar una traducción sintáctica. De esta forma, se podrá definir toda la información de gestión con el mismo lenguaje.

2. Una vez que se utiliza un mismo lenguaje, se puede llevar a cabo la integración semántica, que estará basada en dos pasos simultáneos:
    a. El primero de ellos consiste en efectuar una fusión de modelos, teniendo en cuenta la semántica de la información que hay definida en cada uno de ellos. El resultado de esta fusión será un modelo común unificado.
    b. El segundo consiste en establecer una correspondencia entre los modelos iniciales y el modelo fusionado. Esto permitirá una posterior traducción del modelo común a los modelos que poseen los agentes desplegados cuando sea necesario obtener la información que dichos agentes mantienen.
3. Por último, se podría completar el modelo fusionado resultante incluyendo las restricciones que sean necesarias para dotar de una mayor semántica y poder definir algunas reglas de comportamiento, como se mostrará en la sección 6.2 del capítulo siguiente.

Por tanto, la información que se genere debe dar en paralelo un modelo común o fusionado, y un conjunto de reglas de correspondencias para poder pasar del modelo unificado a cada uno de los modelos particulares que existan, tal y como muestra la Figura 5.5. La forma en que luego se aplicarán tanto el modelo común como las reglas de correspondencia se verá en la sección 6.3 del capítulo siguiente, ilustrándose asimismo el sistema que los utilice en la Figura 6.1.

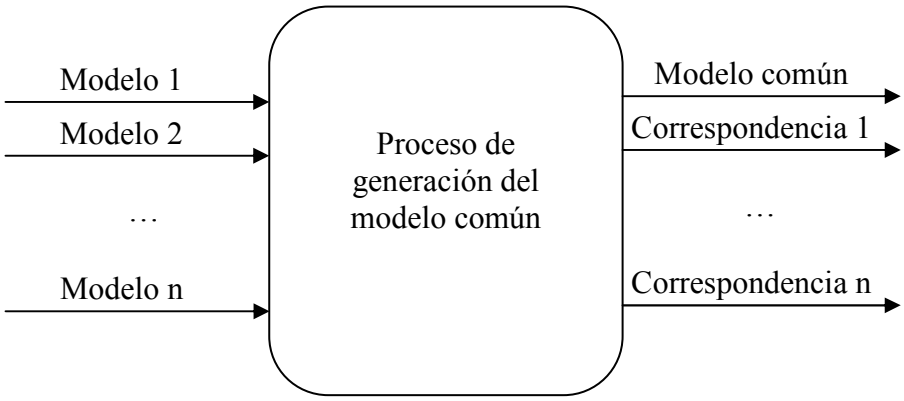

Figura 5.5. Sistema generador del modelo común.

Tras el estudio realizado de los trabajos en el campo de las ontologías, se ha encontrado que en general éstos han abordado exclusivamente la fusión o la correspondencia, pero no ambas cuestiones en paralelo. Por otro lado, tampoco ha habido ninguna aproximación similar en el campo de la gestión, dado que aunque el modelo propuesto por el DMTF para CIM/WBEM identifica el problema, no es capaz de resolverlo eficazmente. En este sentido, se puede considerar que esta propuesta es novedosa.

Los siguientes apartados describen los pasos aquí expuestos. Primero se hablará acerca de las traducciones sintácticas y su importancia. Tras ello, se explicarán las distintas

posibilidades que existen para obtener un modelo común. A continuación se detallan los métodos de fusión y correspondencia, proponiendo seguidamente un método que hace uso de ambos y que se podría implementar en el sistema que generara dicho modelo común. Finaliza la sección un análisis de la propuesta, identificando aquellas ideas que se han considerado más importantes.

5.4.1 Traducción sintáctica

Aunque en esta tesis doctoral se ha resaltado la importancia de realizar traducciones de tipo semántico, también es cierto que es necesario llevar a cabo una traducción previa de tipo sintáctico. Esto es debido a que si se va a trabajar con un mismo modelo, común y unificado, será necesario que dicho modelo tenga una sintaxis homogénea en todas sus definiciones. Así, será más fácil su manejo por parte de un gestor, que no tendrá que trabajar con distintas sintaxis ni diferentes tipos de datos. Por tanto, si se va a proceder a fusionar modelos, será necesario que antes de dicha fusión se realice una traducción sintáctica.

Estas traducciones sintácticas son las que en la literatura se conocen como de tipo *recast* o reescritura, como se vio en el apartado 3.3.2, y son las que se han empleado habitualmente para realizar una integración de los modelos de gestión. Así, tanto IIMC como JIDM proporcionan mecanismos de traducción estándar entre varios de los lenguajes estudiados. Sin embargo, en el caso que ocupa, tras realizar un análisis en la sección previa, se han identificado dos posibles lenguajes que permiten la definición de información incluyendo aspectos semánticos. Por un lado, se podría emplear MOF/CIM con la formalización que se han descrito en el apartado 5.3.2, y por el otro, existe la posibilidad de usar DAML+OIL con las extensiones definidas en el apartado 5.3.3.

Para realizar traducciones de otros lenguajes a CIM, se pueden aprovechar los trabajos de [Festor99] (GDMO), [Pablos01] (IDL), [DMTF99] (MIF) y [Microsoft02, Sun02] (SMI), como se comentó en el apartado 3.4.5.

Por otro lado, traducir a DAML+OIL no debe tener mayores complicaciones. Como se comentó en el capítulo anterior, existen trabajos que permiten la definición de este lenguaje en UML [Backlawski01]. La única cuestión que hay que tener en consideración es que las propiedades en DAML y RDF son *first class* (primera clase), es decir, que tienen entidad en sí mismas y se pueden definir independientemente de cualquier clase a la que estén asociados. Sin embargo, si la traducción se hace primero desde los lenguajes de información de gestión, todas las propiedades siempre pertenecerán al dominio de una clase. En [Backlawski01] también se define desde un punto de vista formal la traducción sintáctica, indicando que es complicado definir una función inversa de la traducción, y que lo más que se puede conseguir es que la función de traducción no expanda el resultado.

En general, las cuestiones que hay que establecer para realizar las traducciones sintácticas serán:

- Definición de módulos o esquemas. En general, todos los conjuntos de clases o tablas están agrupados en módulos.
- Definición de propiedades y métodos, teniendo en cuenta una correspondencia de tipos de datos. En este caso, no se podrán traducir los métodos u operaciones de GDMO, IDL o CIM a DAML+OIL, por lo que puede ser preferible utilizar MOF/CIM como lenguaje común.

Además, si se está traduciendo un modelo orientado a objetos, habrá que establecer correspondencias en lo que se refiera a la definición de clase, incluyendo cuestiones relativas a la herencia. Si hay que traducir de un modelo con herencia múltiple (posible en GDMO e IDL) a otro con herencia simple (como es el caso de CIM), la herencia se haría definiendo asociaciones de tipo `CIM_LogicalIdentity` [DMTF00].

Si el modelo no es orientado a objetos, sino a tablas y grupos de variables, como ocurre en SMI y MIF, la correspondencia habría que establecerla entre clases y tablas y grupos de variables. Asimismo, se pueden aprovechar los algoritmos de LIBSMI [Schönwälder01] para detectar posibles relaciones de herencia entre las tablas para el caso de SMI. Dado que MIF no tiene capacidad para definir índices externos, para este caso no se establecerá este tipo de relaciones.

Una vez que todos los modelos han sido traducidos a una sintaxis común se puede proceder a obtener el modelo de información común.

5.4.2 Obtención de un modelo común: método M&M

A continuación se propondrá la forma en que se puede obtener un modelo común de información de gestión, explicando las ideas generales en las que se basa. Los siguientes apartados detallarán cada parte del proceso.

La traducción entre varios modelos de gestión se necesita normalmente en aquellos escenarios donde se usan para la administración de un mismo conjunto de recursos, de forma que se pueda aplicar una política de gestión coordinada sobre los mismos. Para ello hay dos posibilidades. La primera es definir traducciones entre todos los modelos, dos a dos. La segunda es definir un modelo de información que contenga los existentes. La segunda aproximación es mejor si el número de modelos es alto, probándose que sólo se necesitan $2 \cdot n$ traducciones, en vez de $n^2 - n$ [LópezDeVergara03b].

Por tanto, el modelo de información común deberá ser fruto de un proceso de fusión de los modelos existentes de cada dominio de gestión. Así, en un caso ideal, según se ve en la Figura 5.6, ocurrirá que todos los modelos tienen el mismo peso y el mismo nivel de solapamiento con los otros modelos. Por lo tanto, en este caso convendría crear un modelo

final, suma de todos los modelos, que sea independiente de los mismos (trazo de puntos y rayas en la figura). Esta independencia se traduce en que la definición de la información no se basará en ninguno de los modelos iniciales. Por ejemplo, la información que tuviera este modelo común en aquellas partes con información solapada (trazo punteado en la figura) sería independiente de los primeros.

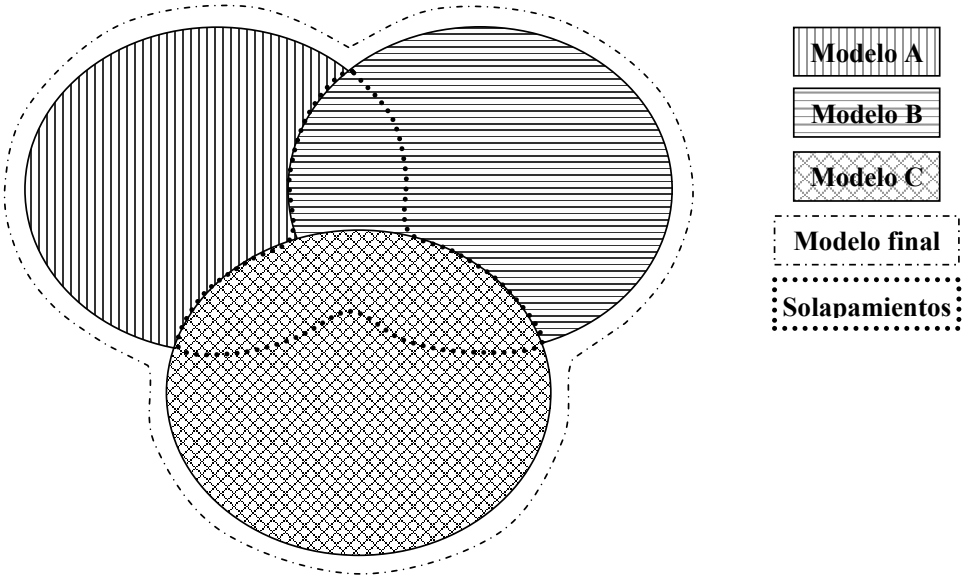

Figura 5.6. Diagrama que muestra la fusión de modelos.

También, al mismo tiempo que se define el modelo final, habría que definir la correspondencia entre el modelo final y cada uno de los modelos existentes, que en aquellas partes que no se encuentran solapadas podría ser directa.

Sin embargo, como se ve en la Figura 5.7, en un caso real no existe la simetría de la Figura 5.6, por lo que es más lógico que el modelo final se cree a partir del modelo mayor al que se le añaden o alinean aquellas partes de los otros modelos que no se encuentran solapadas (trazo grueso discontinuo en la figura). Así se evita tener que definir las reglas de correspondencia de gran parte de la información con el modelo mayor, pues éstas serían directas. Es decir, en este caso la información del modelo final será dependiente de la que se haya definido para el modelo dominante.

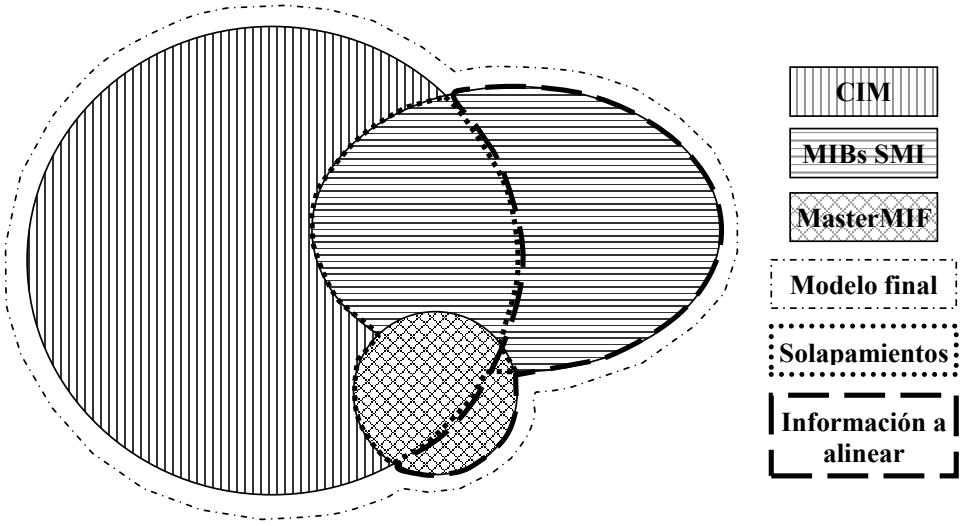

Figura 5.7. Caso real: CIM, MIBs SMI y Master MIF.

Así, el diagrama definido en la Figura 5.5 pasaría ahora a tener este aspecto:

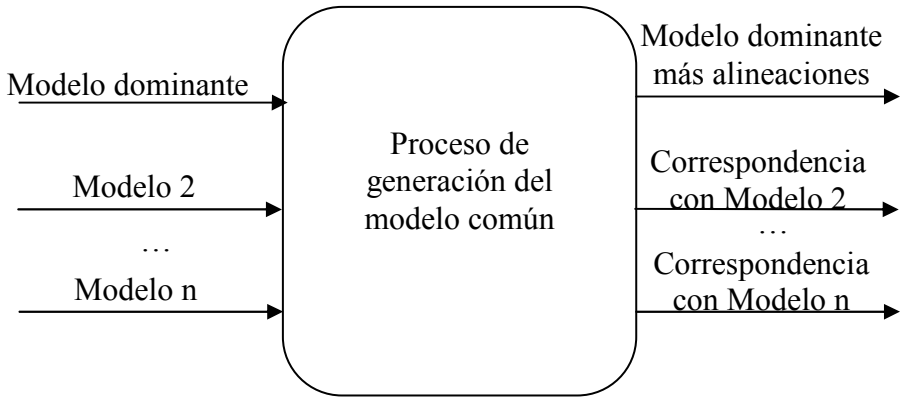

Figura 5.8. Sistema generador del modelo común a partir de modelos asimétricos.

Según esto, es inteligente escoger como modelo dominante a los esquemas CIM. La definición de estos esquemas en WBEM suponen una buena alternativa como un modelo de información común: define la información para la mayor parte de los dominios de gestión, incluyendo un subconjunto de otros modelos de gestión como muchas MIBs de Internet y la Master MIF. La ventaja de elegir CIM como modelo dominante es poder reutilizar al máximo la información ya definida y no *reinventar la rueda*. Las deficiencias de CIM identificadas anteriormente se refieren a la falta de capacidad para definir correspondencias, pero no a que la información que haya definido sea incorrecta. Además, la correspondencia que se generase del modelo común con CIM sería directa, y la traducción sintáctica vista en el apartado anterior tampoco sería necesaria para la información definida en CIM, reduciendo el tiempo del proceso de integración.

Como ejemplo a todo esto, se propone un caso de estudio en el que se fusiona la información definida en distintos dominios para gestionar sistemas. Para ello, a lo largo del apartado se irán presentando ejemplos acerca de la integración de la HOST-RESOURCES-

MIB [Waldbusser00a], escrita en SMIv2 y representada aplicándole los algoritmos de LIBSMI en la Figura 5.9, con un subconjunto de los esquemas CIM [Bumpus00], mostrado a su vez en la Figura 5.10.

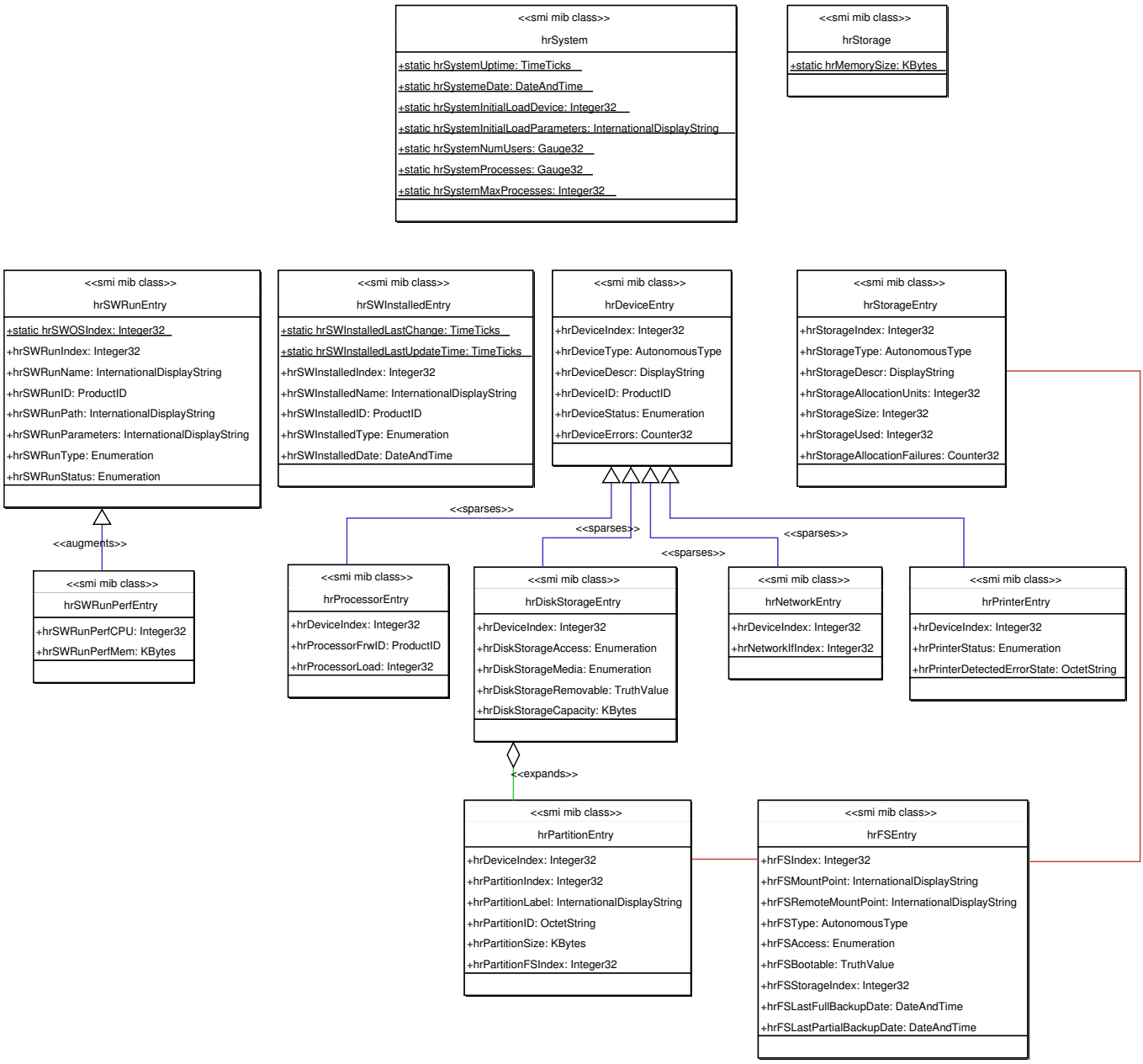

Figura 5.9. Representación de HOST-RESOURCES-MIB en UML.

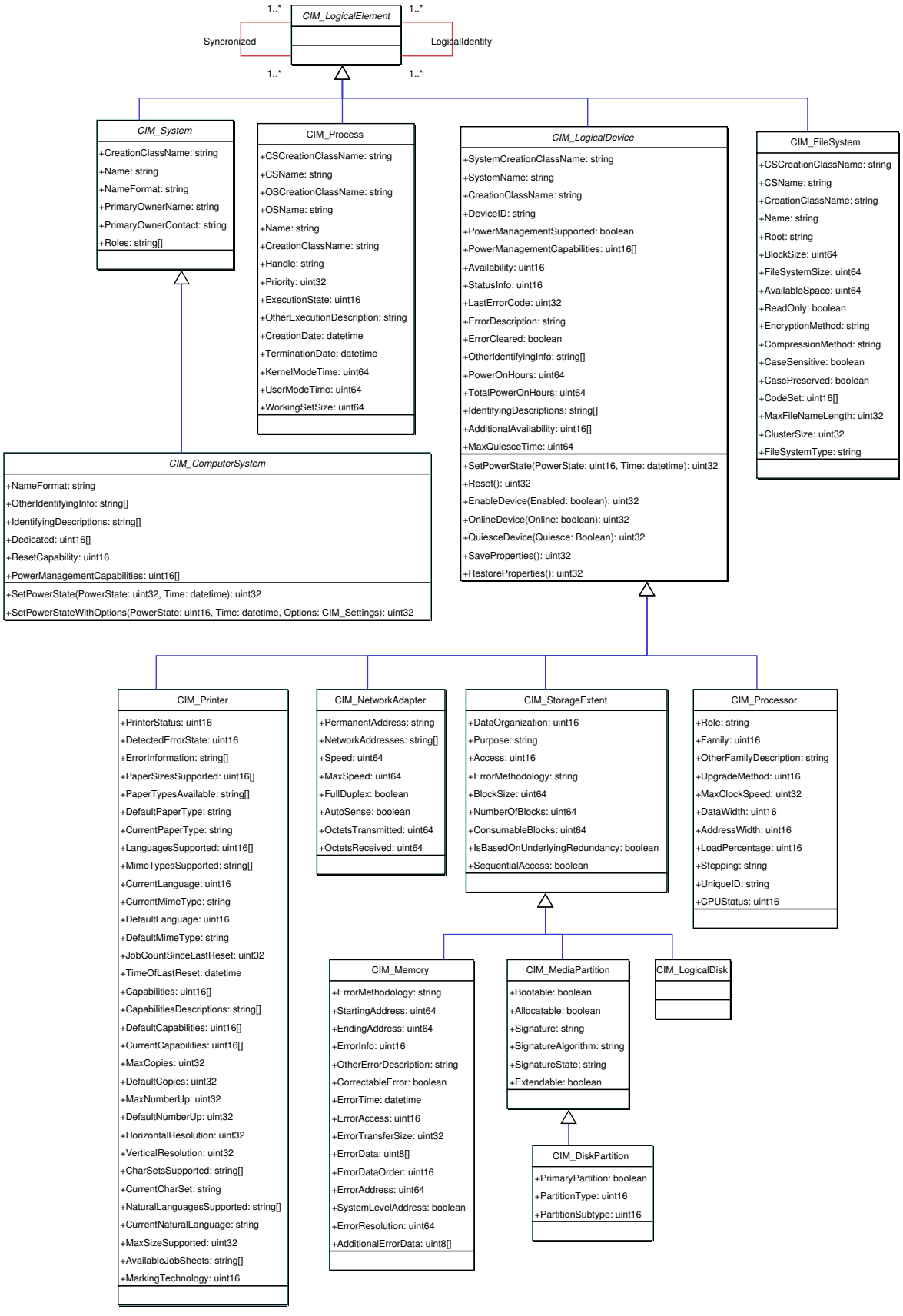

Figura 5.10. Subconjunto del esquema de CIM con información similar al anterior.

Para el caso ideal, la información de las zonas solapadas (Por ejemplo, `hrDeviceEntry` y `CIM_LogicalDevice`), e incluso las no solapadas, sería independiente de los dos modelos (los dispositivos se definirían de una manera independiente a los anteriores, por ejemplo `GenericDevice`). En el segundo caso, la información de las zonas solapadas sería directamente la del modelo dominante, usándose la información que se ha definido para CIM, añadiéndole aquélla que falte.

Por otro lado, este método de fusión es similar al utilizado en [Noy00], en el que se escoge una ontología como principal y a ésta se le va añadiendo la información de la ontología secundaria. Sin embargo, dicho método no incluye la definición en paralelo de reglas de correspondencia.

A todo lo anterior se le puede sumar el que la información definida en CIM es la más rica semánticamente, al estar definida con MOF/CIM, pues se ha visto que es el lenguaje de definición de información de gestión con mayor capacidad semántica de los existentes. Y a la vez, CIM es un modelo estructurado al estilo de las ontologías, como se mostraba en la Figura 4.1.

Con todo, concretando el diagrama de la Figura 5.8 a modelos concretos de información de gestión, quedaría de la siguiente manera:

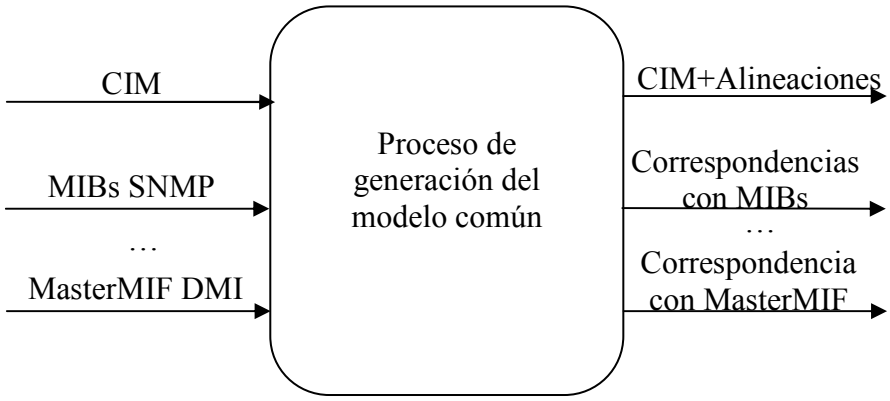

Figura 5.11. Sistema final para la generación del modelo común de información de gestión.

Entonces, dadas las ventajas expuestas, para realizar todo el proceso de fusión y correspondencia, se partirá de los esquemas de CIM como punto de representación común, y tras esto se fusionarán todos los modelos que se precisen con CIM. Además, para cada fusión que se realice se detallarán las reglas de correspondencia necesarias para traducir entre el modelo común y cada modelo particular. Con esto, se podrá trabajar con un modelo unificado, y acceder a la información que pertenezca a cada dominio mediante las reglas de correspondencia. Las reglas deberán estar definidas con cierta formalidad, de manera que se puedan cargar en un sistema que las interprete.

Los siguientes subapartados estudiarán los procesos de fusión y correspondencia desde el punto de vista particular de la información de gestión, definiendo finalmente el método M&M, que combina ambos procesos para obtener ambos resultados de manera simultánea.

5.4.2.1 Estudio de fusión de modelos de información de gestión

Como se explicaba anteriormente, el motivo de fusionar los modelos de información existentes viene dado por la necesidad que tiene un sistema gestor de manejar un modelo de información unificado. Por tanto, toda la información deberá estar definida de manera conjunta, e independientemente de los dominios de gestión a los que pertenezcan los recursos gestionados. Realizar este proceso de fusión es importante para poder tener una visión común de todos los recursos que se están gestionando. El mero hecho de pertenecer a un dominio de gestión distinto no debe ser la causa de aislar la información relativa a unos recursos respecto de otros si existe algún tipo de relación entre ellos.

A continuación se describen los mecanismos de fusión que se pueden emplear para obtener el modelo común de información de gestión. Cuando se habla de fusión, realmente se debería hablar de fusión y alineamiento, puesto que no sólo se trata de fusionar la información del resto de modelos dentro del modelo principal o dominante, sino que se añaden nuevas clases a la jerarquía dicho modelo dominante. Sin embargo, se da la circunstancia de que el proceso más trabajoso es precisamente el de fusionar la información, puesto que para el caso de alineamiento únicamente hay que buscar cuál sería la clase madre de la que se añade al modelo.

Por tanto, los mecanismos de fusión deben incluir, para clases:

- Identificación de clases candidatas para fusionarlas con clases del modelo común.
- Identificación de clases únicas para añadirlas (alinearlas) al modelo fusionado.
- Identificación de relaciones de herencia.

Y de la misma forma, para las propiedades:

- Identificación de propiedades similares para fusionarlas.
- Identificación de propiedades únicas para añadirlas (alinearlas) al modelo fusionado.

Para implementar estas cuestiones se pueden aprovechar los trabajos existentes en el campo de las ontologías explicados en el apartado 4.4.1, seleccionando aquellas técnicas que sean útiles en la Gestión de Red. Por otro lado, se puede aprovechar la forma en que está definida la información de gestión para añadir algún método a los ya existentes.

No todos los trabajos de fusión se pueden usar para este caso. Por ejemplo, la fusión basada en la correspondencia por los valores de los ejemplares, utilizada en las herramientas FCA-Merge [Stumme01] y GLUE [Doan02], nombradas anteriormente, no es aplicable pues en un sistema de gestión los ejemplares se obtienen a posteriori y no a priori. Por tanto, habrá que emplear métodos en los que se fusionen las clases y propiedades, sin tener en cuenta los valores de los ejemplares. Este método es el que está implementado en la herramienta Prompt [Noy00]. Sin embargo, no es válido el caso de

Chimaera [McGuinness00], pues sólo compara clases sin tener en cuenta las propiedades. ONION [Mitra00] tampoco sirve al no generar un modelo común, sino reglas de articulación.

La identificación de clases y propiedades se puede semiautomatizar, asistiendo a la persona que realice el proceso mediante la aplicación de heurísticos que identifiquen las clases y propiedades candidatas a fusionar. Así, los heurísticos más útiles para el caso que se estudia son:

- De correspondencia por similitud en cadenas de caracteres [Hovy98, Noy99]. Para ello, se pueden analizar aquéllas que estén incluidas en identificadores de clases o propiedades o en su documentación. Esto permite proponer como candidatos de fusión a clases o propiedades con nombres similares. Por ejemplo, `hrDeviceEntry` y `CIM_LogicalDevice` comparten la subcadena `Device`, lo que podría significar que modelan conceptos similares. Este heurístico se puede emplear tanto para clases como para propiedades, y se puede ampliar a comparaciones con sinónimos de las palabras encontradas. Aunque los lenguajes de ontologías suelen permitir nombres compuestos y separados por caracteres de espacio (" "), en el caso particular de la información de gestión, los nombres de las clases y propiedades son nombres compuestos de varias palabras concatenadas (sin caracteres de separación). Para separarlos se suele indicar cada palabra que compone el nombre poniendo su inicial en mayúscula, o bien con el símbolo de subrayado ("_"). Esta particularidad se puede aprovechar para hacer una comparación de subcadenas más rápidamente. También, las definiciones de gestión suelen tener un prefijo, como `CIM_` o `hr` en el ejemplo anterior, que suele ser similar para un conjunto de definiciones. Un sistema que busca candidatos a la fusión debería aprender a ignorarlos al encontrarlos repetidas veces.
- De correspondencia por similares jerarquías de herencia [Noy01]. En muchos casos, las clases hijas de una dada suelen ser similares a las clases hijas de la clase fusionada o alineada con la primera. Por ejemplo, si se fusiona `hrDeviceEntry` con `CIM_LogicalDevice`, con gran probabilidad se podrán fusionar sus subclases (`hrProcessorEntry` con `CIM_Processor`, `hrNetworkEntry` con `CIM_NetworkAdapter`, `hrPrinterEntry` con `CIM_Printer`…).
- De correspondencia por dominio de propiedades (I). En los lenguajes de información de gestión, a diferencia de los empleados habitualmente en las ontologías, las propiedades siempre pertenecen al dominio de una clase. Esto supone una ventaja, puesto que la fusión de modelos de información es más directa. Sabiendo que una clase se corresponde con otra, será normal que las propiedades de la primera clase también se correspondan con las de la segunda. Por ejemplo, si `hrPrinterEntry` se corresponde con `CIM_Printer`, entonces las propiedades de la

primera clase posiblemente se correspondan con propiedades de la segunda, restringiendo el dominio de búsqueda de propiedades candidatas.

- De correspondencia por dominio de propiedades (II). Siguiendo con el razonamiento de la regla anterior, y restringiéndose al caso de los lenguajes de información de gestión, también se podría decir, dado que las propiedades siempre pertenecen al dominio de una clase, que si una propiedad *x* de una clase *w* se corresponde con la propiedad *y* de una clase *z*, la clase *w* con gran probabilidad se corresponde con la clase z. Por ejemplo, si `hrPrinterStatus` de la clase `hrPrinterEntry` se corresponde con `PrinterStatus` de la clase `CIM_Printer`, posiblemente `hrPrinterEntry` se corresponda con `CIM_Printer`. Este heurístico se puede combinar con el anterior, y así, obtener que con gran probabilidad dos propiedades se corresponden porque otras propiedades de los mismos dominios se corresponden.

También se puede obtener una nota a partir de la suma ponderada de estos heurísticos, como se propone en [Hovy98] para identificar los candidatos que con mayor posibilidad se puedan fusionar.

Otras ideas a tener en cuenta en el proceso de fusión son las siguientes:

- La fusión supone definir un nuevo espacio de nombres, con lo que es mejor usar el espacio de nombres del modelo más grande a añadir un nuevo espacio de nombres independiente, según lo explicado en el apartado anterior.
- Si una clase tiene correspondencia con dos del otro modelo que tienen una relación de herencia (madre e hija), la clase del primer modelo se debería corresponder con la hija del segundo, puesto que dicha clase hija contiene toda la información de la clase madre. Por ejemplo, si la clase `hrDiskStorageEntry` se corresponde con `CIM_StorageExtent` y su hija `CIM_LogicalDisk`, entonces la fusión se debe hacer con la última clase, pues es la que contiene la semántica de la madre y la propia.
- Además de las operaciones de fusión y alineamiento también habría que hacer una revisión de las inconsistencias que se generen. En casos de conflictos debidos a tipos de datos, prevalece el de la ontología dominante. Los mecanismos de correspondencia se encargarán de adaptar la información de cada uno de los dominios a la ontología general.
- También se debe registrar el proceso de las fusiones realizadas, por si hubiera que reproducirlas [Klein01, Noy02a]. Por ejemplo: cada vez que salga una nueva versión de los esquemas CIM, habría que realizar un nuevo proceso de fusión. De esta manera, si se tiene un registro de las fusiones que se realizaron con el esquema anterior, no habrá que repetir gran parte del trabajo.

A continuación se presenta un análisis sobre la correspondencia de modelos de gestión, proceso que complementará al de fusión en la obtención de un modelo común.

5.4.2.2 Estudio de correspondencia de modelos de información de gestión

Si la causa de fusionar los modelos de información existentes viene dada, según se explicaba en el apartado anterior, por la necesidad que tiene un sistema gestor para manejar un modelo de información unificado, el motivo de traducir la información que maneja el gestor a la que hay definida en cada dominio de gestión es debido a la necesidad de que dichos dominios interoperen.

Para ello, este apartado propone la definición en paralelo a la fusión de algún mecanismo de correspondencia que permita realizar esta traducción interdominio. A diferencia de la traducción sintáctica, comentada anteriormente, los métodos de correspondencia que se exponen a continuación pretenden realizar una traducción del significado de la información que se está manejando. Por tanto, afecta a los ejemplares del modelo (que poseen valores para las propiedades), aunque también supone una traducción de otras cuestiones, como el nombre de los elementos a corresponder o incluso ciertas facetas de los mismos.

Definir un método de correspondencia es importante para poder desarrollar una pasarela genérica para cada dominio, independiente del contenido de la información que se defina. Dichas pasarelas (o proveedor en el caso de WBEM) poseerán la capacidad necesaria para pasar las solicitudes de información entre las distintas pilas de protocolos, y únicamente tendrán que cargar las reglas de correspondencia para cada conjunto definiciones que deba soportar, sin necesidad de tener, por ejemplo, una distinta para cada conjunto de definiciones (o MIB). Estas pasarelas no sólo se encargarán de adaptar los tipos de datos y su codificación, sino que también serán capaces de obtener el valor de una propiedad en un dominio a partir de las reglas de correspondencia que se hayan definido con las propiedades del otro dominio de gestión. La sección 6.3 del capítulo siguiente explica en mayor detalle cómo deben ser estas pasarelas basadas en reglas de correspondencia, pues forman parte de la arquitectura de gestión ilustrada en la Figura 6.1.

Por todo esto, al mismo tiempo que se realiza el proceso de fusión hay que establecer algún mecanismo que permita crear reglas de correspondencia cada vez que se fusione un elemento con el modelo común. La correspondencia puede ser de varios tipos y siguiendo relaciones 1:1, 1:n, o n:m, pudiéndose ver estas últimas como generalización de 1:n [Maedche02]. Además, podrán existir distintos tipos de correspondencia como se indicaba en [Park97]. Por ejemplo:

- Directa. Es una correspondencia 1:1 en donde el valor de una propiedad es igual en uno y otro dominio, por lo que sólo hay que tomar dicho valor. El nombre que tenga la propiedad puede variar entre uno y otro dominio. El calificador `MappingStrings` de MOF/CIM es un ejemplo de este tipo de correspondencia.

- Correspondencia de valores. Es una correspondencia 1:1 en la que para un conjunto de valores en un dominio existe otro conjunto de valores distinto en el otro dominio. Esta regla puede valer para establecer correspondencias entre enumeraciones de valores que siguen distinto orden numérico, o distinta codificación (números y cadenas de caracteres). Para ello se puede crear una función C de correspondencia entre dos conjuntos de valores X e Y, con lo que si en un modelo una propiedad vale x_i, entonces en el otro modelo, vale y_i. Puede ocurrir que la función no pueda establecer los elementos y_i para todos los valores x_i, pero en gestión, los enumerados suelen tener algún valor de tipo desconocido, que se podría emplear para estos casos.
- Correspondencias de cambio de tipos de datos. Es una correspondencia 1:1 para aquellos casos en los que haya que traducir entre números y cadenas de caracteres y no sea necesario establecer la función C del punto anterior pero sí cambiar la codificación.
- Correspondencia mediante operación aritmética sobre una variable. Es una correspondencia 1:1 en la que para obtener el valor de una propiedad en un dominio se realiza una operación sobre esta propiedad en el otro dominio. Esta regla es útil para cambiar de unidades de medida, en las que para obtener un valor basta con multiplicar o dividir por una constantes.
- Correspondencia mediante operación aritmética sobre un conjunto de variables. A diferencia del caso anterior es una correspondencia 1:n en la que intervienen varias propiedades de un dominio para obtener el valor del otro. Para ello es necesario operar sobre el conjunto de propiedades. Esta correspondencia es aplicable cuando la propiedad de un dominio se obtiene como resultado de operaciones aritméticas sobre varias propiedades del otro dominio de gestión.
- Correspondencias mediante operación sobre cadenas de caracteres. Las correspondencias anteriores se referían sobre todo a valores numéricos, pero también es posible establecer correspondencias que se refieran a cadenas de caracteres, que en general tendrán una relación 1:n. Éstas podrán ser cuestiones tan simples como la concatenación de varias propiedades de un dominio en una propiedad del otro o la división de una propiedad en subcadenas a partir de un caracter de división. También, pueden aplicarse expresiones regulares que realicen otro tipo de modificaciones, incluyendo conversión entre mayúsculas y minúsculas.
- Finalmente, se pueden establecer reglas de correspondencia múltiples en las que haya que realizar varias de las anteriores para obtener el valor final.

A continuación se presentan ciertas pautas que se han identificado a la hora de realizar el proceso de correspondencia. Basándose en las mismas, un sistema podría sugerir la regla a aplicar para cada caso.

- Los tipos de correspondencia analizados se refieren a propiedades, pero también puede haber reglas de correspondencia de clases que indiquen si una clase es similar a otra, o bien es una particularización/generalización de otra.
- La correspondencia está limitada a aquella información que existe en cada dominio. Puede ocurrir que al completar la fusión modelos, una parte del modelo común generado no esté presente en un dominio de gestión concreto. Por tanto, no será posible establecer correspondencias para dicho caso. Es decir, sólo se generarán reglas de correspondencia para aquella información que haya sido previamente fusionada.
- Las reglas de correspondencia se establecen entre el modelo común y el modelo particular, de forma que se pueda pasar de uno a otro y viceversa. De esta forma, en el caso de que existan dos modelos distintos al modelo general que se solapen, se podría pasar de la información del primero a la del segundo en dos pasos, debido a que, como se comentaba anteriormente, esta aproximación supone la definición de $2 \cdot n$ reglas de correspondencia en vez de n^2-n, lo cual es mejor para $n \geq 4$.
- En los casos de alineamiento en qué simplemente se incluye la información de un modelo concreto al modelo general, las reglas de correspondencia serán en general directas. Sin embargo, en los casos de fusión las reglas de correspondencia podrán ser de cualquiera de los tipos anteriores, aunque dado que la información de gestión se define al detalle [MartinFlatin01], normalmente habrá una relación 1:1 entre los atributos de distintos modelos.
- La definición de tipos de datos o unidades de medida puede ser útil para definir las reglas de correspondencia entre las propiedades. Así, si en el proceso de fusión se plantea un conflicto debido a tipos de datos o de unidades de medida, se utiliza el del modelo común frente al del particular, y se sugiere el establecimiento de una regla de correspondencia no directa que permita realizar el cambio.
- Finalmente, en caso de no haber conflictos, si la correspondencia es 1:1, entonces en general se puede proponer establecer una correspondencia directa.

Para definir las reglas de correspondencia sería útil definir una ontología que describiera los elementos de una correspondencia para los modelos de información de gestión. Para ello se pueden plantear las siguientes preguntas, siguiendo la misma estrategia que se utiliza en otros campos:

- ¿Qué elementos se van a corresponder? En general van a ser clases y sus propiedades.

- ¿Cómo se van a corresponder? Mediante reglas que traduzcan las clases y propiedades de un modelo al otro, teniendo en cuenta sus facetas (tipo de datos, unidades…)
- ¿Cómo se van a plasmar estas reglas? Con un conjunto de correspondencias que poseerán los elementos origen y destino a corresponder, así como la fórmula para realizar dicha correspondencia.
- ¿Qué problemas se pueden plantear? Por un lado, será necesario definir de alguna forma la descripción de las fórmulas. Por otro lado, también habrá que buscar la forma en que se referencien los elementos a corresponder.

En cualquier caso, parece claro que esta información no cabe en el calificador `MappingStrings` de MOF/CIM, que únicamente contiene identificadores que permiten establecer correspondencias directas.

Al igual que para el caso de la fusión, se pueden aprovechar los trabajos existentes en el campo de las ontologías que se explicaron en el apartado 4.4.2, seleccionando aquellos aspectos de una ontología de correspondencia que sean útiles en la Gestión de Red. Así, como se comentó en dicho apartado al analizar el caso de de MAFRA, una ontología de correspondencias se puede expresar mediante tres conceptos:

- Correspondencia (*Mapping*). El término o concepto Correspondencia es fundamental en una ontología de este tipo, pues representa la relación entre dos entidades de información. A su vez, posee los dos términos siguientes, así como varios atributos, que pueden indicar el tipo de correspondencia.
- Elemento a Corresponder (*Element*). Una correspondencia se establecerá entre uno o más Elementos origen y destino, teniendo en cuenta su cardinalidad, dado que la relación entre los elementos puede ser 1:1, 1:n ó n:m. Estos elementos pueden referirse a clases o propiedades de clase, o relaciones, como apuntaba la ontología anterior. Los atributos que tenga este concepto pueden ser:
    - Tipo de elemento, que indique si se trata de una clase, propiedad, etc.
    - Referencia al modelo que pertenece, que identifique el nombre del modelo y del elemento. Dicha referencia podría tener una sintaxis similar a la del calificador `MappingStrings` de MOF/CIM.
- Fórmula (*Formula*). La Fórmula indica la forma en que se establece la correspondencia de los elementos del modelo de origen con de destino o viceversa, mediante traducciones de nombre, concatenaciones o expresiones regulares, como ya se ha analizado. Las propiedades de esta clase incluyen:
    - Referencias a los Elementos origen y destino, para poder establecer las reglas a aplicar a cada uno de ellos.

- Expresión de una secuencia de reglas de transformación. Puede ser una cadena de caracteres que contenga código escrito en un lenguaje concreto.
- Lenguaje de las reglas, que indique el lenguaje en que está escrita la regla de transformación.

La Figura 5.12 representa en UML la ontología definida en las líneas anteriores.

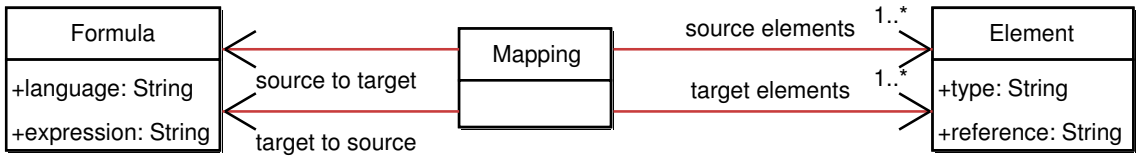

Figura 5.12. Primer esbozo de ontología de correspondencias.

Además, pueden existir conceptos adicionales, que valgan para agrupar o subdividir los anteriores, como el de Esquema de correspondencia, que sería la colección de todas las correspondencias definidas para traducir un modelo concreto.

Sin embargo, desde un punto de vista pragmático, pensando en una pasarela que traduzca de un dominio a otro, lo importante es obtener el elemento solicitado mediante la composición de los respectivos elementos del otro dominio. En este sentido, lo importante no es la clase Correspondencia, sino los Elementos en sí mismos y su relación con las Fórmulas. De esta forma, a partir de la solicitud de un elemento se busca y se aplica la fórmula para componerlo a partir de los elementos del otro dominio. Por ello, la ontología quedaría simplificada según se muestra en la Figura 5.13, cuya definición en DAML+OIL está contenida en el Apéndice B.

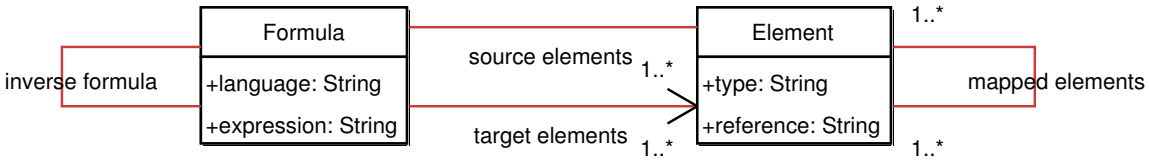

Figura 5.13. Segundo esbozo de ontología de correspondencias.

En este caso, un Elemento está relacionado con una Fórmula, que a su vez está relacionado con este elemento como origen y los elementos destino. Asimismo, se define una asociación entre Fórmulas que permita identificar la fórmula inversa a una dada, y otra asociación entre Elementos que permite identificar los elementos con los que se corresponde.

La mecánica para utilizar esta ontología sería la siguiente:

1. Se solicita a la pasarela el Elemento A del modelo común.
2. Los ejemplares contenidos en la ontología indican que el Elemento A se consigue a partir del Elemento (o Elementos) B del modelo particular.
3. Se obtiene el Elemento (o Elementos) B.

4. Se encuentra la Fórmula para obtener el Elemento A a partir del B.
5. Se aplica la Expresión contenida en la Fórmula que relaciona A con B y se devuelve el resultado como Elemento A.

En otros términos, y volviendo a utilizar el caso de estudio introducido anteriormente, supuestas las propiedades `hrSWRunEntry.hrSWRunStatus` (B) y `CIM_Process.ExecutionState` (A) que especifican el estado en el que se encuentra un proceso de un sistema operativo, se darían estos pasos:

1. La pasarela recibe la solicitud del valor de `CIM_Process.ExecutionState` (A).
2. Se encuentra que se corresponde con `hrSWRunEntry.hrSWRunStatus` (B).
3. Se obtiene el valor de `hrSWRunEntry.hrSWRunStatus` (B) en el dominio de SNMP.
4. Se encuentra la fórmula para obtener `CIM_Process.ExecutionState` (A) a partir de `hrSWRunEntry.hrSWRunStatus` (B). Dicha fórmula hará los cambios de valor y tipos de datos necesarios según las definiciones de ambas propiedades.
5. Se aplica la expresión de la fórmula y se devuelve el valor obtenido para `CIM_Process.ExecutionState` (A).

Una vez se han estudiado los mecanismos de correspondencia a aplicar para la interoperabilidad de dominios de gestión de red, se puede proceder a definir un método que permita implementar un sistema para asistir y semiautomatizar el proceso de fusión y correspondencia de modelos de información de gestión.

5.4.2.3 Definición del método M&M

Como se ha comentado anteriormente, los procesos de fusión y correspondencia son complementarios, y por tanto, sería interesante llevarlos a cabo de manera paralela. Para ello, este apartado presenta el método bautizado como M&M (*Merge and Map*, Fusión y Correspondencia), que trata de integrar ambas cuestiones. Este método propone un conjunto de pasos que ayude a obtener simultáneamente el modelo común y las reglas de correspondencia. Está basado en el método de fusión de [Noy99] adaptándolo al caso particular de la gestión de red, y añadiendo todo lo que se refiere a la definición de reglas de correspondencia, con lo que por cada elemento que se fusione o alinee se añade una regla que relaciona los elementos fusionados en la ontología de correspondencia.

Dado que se usan los heurísticos identificados en los apartados anteriores tanto para la fusión como para la correspondencia, este método no genera un resultado de manera automática. Es más bien un método que asiste en el proceso de fusión y correspondencia a la persona que deba realizar esta labor, y por tanto, está pensado para su implementación en un sistema que maneje múltiples ontologías.

En resumen, el método M&M consiste en identificar primero clases similares (con mismo nombre, o descripción similar), y a través de ellas, fusionar los distintos atributos que hay en ellas. Al mismo tiempo, para cada elemento a fusionar o alinear se define dicho elemento en la ontología de correspondencia, y luego se van definiendo las fórmulas asociadas y elementos que se corresponden con éste.

La Figura 5.14 muestra el diagrama de actividad que describe el método M&M. Las actividades con fondo gris son aquéllas que realizaría el usuario, mientras que las de fondo blanco serían efectuadas por el sistema.

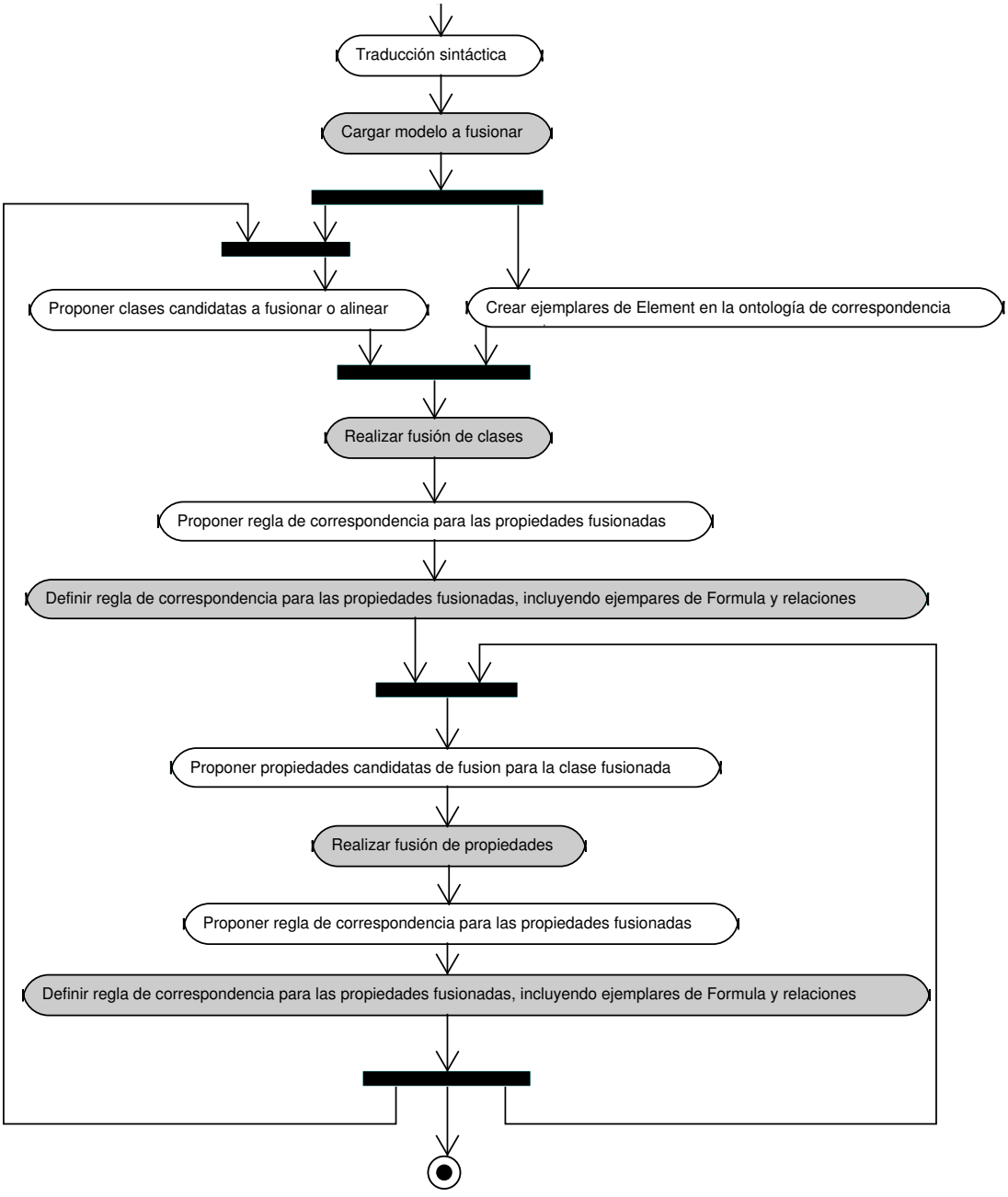

Figura 5.14. Diagrama de actividad del método M&M.

Así, los pasos que componen este método son los siguientes:

1. Previamente al proceso que se va a realizar se ha efectuado una traducción sintáctica para trabajar con una sintaxis común.
2. El usuario carga el modelo a fusionar, teniendo en cuenta cuál es el modelo dominante, y por tanto, base del modelo común. En este caso concreto, siguiendo los razonamientos previos, se ha decidido que el modelo dominante es el conjunto de esquemas CIM. Por tanto, se establecerá como modelo común a la clase `CIM_ManagedElement`, raíz del los esquemas CIM, y todos sus descendientes. En general, el proceso será una combinación de fusión y alineado con dicho modelo dominante. Existe información definida que dado su carácter básico es fusionable, mientras que en otros casos es información muy específica de un dominio, por lo que posiblemente se realizará un alineado.
3. Una vez están cargadas ambas ontologías se registran todos los elementos de ambos modelos en la ontología de correspondencia, para ir estableciendo posteriormente las reglas de traducción.

   A la vez se genera la lista inicial de sugerencias, a partir de las clases que existan en los dos modelos. Identificar dos clases similares (como `CIM_Processor` y `hrProcessor`) puede no ser interesante si el conjunto de propiedades que está contenido en cada clase es diferente. Sin embargo, según el heurístico de correspondencia por dominio de propiedades, se supone que dos clases similares también poseerán atributos similares, por lo que éste puede ser un buen punto de comienzo. Para buscar clases similares se emplean los heurísticos de correspondencia por similitud en cadenas de caracteres explicados en el apartado 5.4.2.1. Los identificadores que se manejen normalmente tendrán prefijos y sufijos, como `CIM_` o `Entry`, por lo que será extraño encontrar nombres idénticos, y únicamente habrá subcadenas similares. Así, según los resultados de esta búsqueda se tendrán dos opciones:

   a. Para cada par de clases con nombres idénticos o con subcadenas similares, se propondrá establecer una fusión entre ellas.

   b. Si no se encuentran nombres similares, se propondrá alinear esta clase junto con sus atributos en el modelo general, siendo necesario buscar una clase madre a la que alinearla.
4. Con todo esto, el usuario realiza una operación, que puede ser una de las fusiones sugeridas, o bien tomada por iniciativa propia. Cuando se fusionan dos clases, las propiedades que poseen se incluyen en la clase fusionada, y en este caso hay que determinar qué propiedades son nuevas y cuáles poseen correspondencia entre ellas, y por tanto, están repetidas. Por otro lado, si lo que se ha hecho ha sido un

alineamiento, tanto la clase como sus propiedades serán nuevas en el modelo común.

Sea cual sea la operación realizada por el usuario, el sistema debe proponer una regla de correspondencia que asocie el elemento del modelo fusionado con el modelo general, incluyendo una posible fórmula. Si se han fusionado dos clases, se propone una fórmula de correspondencia directa entre ambas, y posteriormente se definirán las fórmulas para las propiedades. Si la operación ha sido de alineamiento, se propone una fórmula de correspondencia directa tanto para la clase como para las propiedades, dado que esta información es exactamente la que hay en el modelo a fusionar. Además, se definirá una regla que indique la relación de particularización de esta clase con respecto a de la que hereda en el modelo general.

5. Una vez se ha fusionado o alineado una clase, se crean nuevas sugerencias basadas en el conjunto de heurísticos definidos en el apartado 5.4.2.1, incluyendo la correspondencia por similitud de cadenas de caracteres, jerarquía de herencia y dominio de las propiedades. Y así:

    a. Se buscan clases que tengan relación de herencia con las que se han fusionado para añadirlas a la lista de sugerencia de fusión de clases.

    b. Se buscan propiedades que pertenezcan a las clases que se han fusionado y se puedan corresponder para proceder a su fusión. Para cada fusión entre propiedades, también se establece la correspondiente regla de correspondencia. En este caso, la regla puede ser de cualquier tipo, con las distintas operaciones que se han identificado en la correspondencia de información, pero para la mayoría de los casos será una relación directa, si no ha habido ningún conflicto.

    c. El conjunto de conflictos que se generen también serán analizados. Los conflictos pueden deberse a un tipo de datos distintos en cada modelo de información. En este caso, la regla de correspondencia no podrá ser directa, sino una en la que, por ejemplo, se cambie el tipo de datos mediante correspondencia de valores. Si estuviera definida una faceta de los atributos acerca de las unidades de medida, se podría proponer una correspondencia mediante operación aritmética que permitiera la traducción a la unidad de medida del modelo general.

        Los conflictos pueden deberse también a la existencia de referencias a clases que todavía no se han fusionado. Es decir, propiedades cuyo valor es un ejemplar de clase, y cuya clase no está en la ontología fusionada. En estos casos se aguardará a que se resuelva el conflicto para proponer la regla de correspondencia adecuada.

6. Los pasos anteriores se repiten hasta que los modelos de información estén completamente fusionados o alineados. Dado que tras cada fusión o alineamiento se han definido reglas de correspondencia, cuando el proceso haya concluido existirá por un lado un modelo común de información de gestión, y por otro, el conjunto de ejemplares de la ontología de correspondencia que se refiere al modelo particular con respecto a dicho modelo común.

Para ayudar a comprender mejor este método el Apéndice C completa el caso de estudio analizado a lo largo de la sección acerca de la integración de la HOST-RESOURCES-MIB en los esquemas CIM, incluyendo detalles del proceso por el que se fusionan y corresponden las tablas y clases.

5.4.3 Análisis de la propuesta

Esta sección ha presentado una contribución original de esta tesis, consistente en una propuesta de mecanismos que permitan una integración semántica de los actuales modelos de información de gestión. Para ello, se ha aprovechado el conjunto de técnicas propuestas para el campo de las ontologías, así como los estudios que se han efectuado en las secciones anteriores. El resultado de este método permite obtener, por un lado, una ontología de gestión fruto de la fusión de la información de gestión existente, y por otro lado, una ontología de correspondencia para realizar las traducciones semánticas entre la ontología global y la información de cada dominio de gestión.

Para realizar la integración de modelos de información se propone llevar a cabo un conjunto de pasos, de los cuales el segundo es el más importante:

1. Traducir todos los modelos a un mismo lenguaje de definición de información para trabajar con una sintaxis común. Para esto se utilizará aquel lenguaje mejor dotado desde el punto de vista de la semántica.
2. Método M&M de fusión y correspondencia en paralelo de la información de gestión. Para ello, se utiliza un modelo común a todos, y se realizan fusiones y reglas de correspondencia con dicho modelo, siguiendo un conjunto de pautas y heurísticos que se han detallado para el caso particular de los modelos de información de gestión.
3. Adición de información relativa al comportamiento según se explicará en el capítulo siguiente, obteniendo una ontología de gestión completa.

Estos pasos están pensados para ser aplicados a la gestión interdominio, de forma que se pueda gestionar simultáneamente y desde un mismo punto elementos de red que pertenecen a distintos dominios de gestión. Por ejemplo, una central telefónica en la que el ADSL se gestiona con SNMP y la telefonía conmutada tradicional con CMIP.

También se puede aplicar a la gestión intradominio, para aquellos casos en la que hay definidos varios modelos de información para resolver un mismo problema en un mismo dominio. Por ejemplo, en el caso de SNMP, existen distintas MIBs que definen los recursos de un sistema. La MIB estándar es la HOST-RESOURCES-MIB, pero también se usa mucho la HP-UNIX, y la SUN-SNMP, que tienen información muy similar. Usando este método, se podrían integrar en un modelo común, y utilizando las reglas de correspondencia se podría acceder a la información que proporciona cada uno de los modelos mencionados.

Es más, la aplicación de esta propuesta podría valer para gestionar dominios no estándar. Por ejemplo, la gestión de aplicaciones comunes como pueden ser los distintos servidores (correo, web, archivos, impresión…). Dependiendo de cada implementación, la configuración de una aplicación con funciones similares tendrá distinta sintaxis y distinto conjunto de parámetros. Utilizando este método se puede definir un modelo común para realizar la administración de todas las implementaciones que se utilicen, dado que se haría a un nivel más alto, independiente de cada dominio de aplicación.

5.5 Conclusiones

El presente capítulo ha introducido varias contribuciones originales para la integración de información de gestión mediante la aplicación de las ontologías. Con ello, se ha tratado de solventar el problema que existe cuando es necesario llevar a cabo una gestión unificada de todos los recursos de un sistema cuyos métodos de acceso y modelos de información pertenecen a distintos dominios de gestión. La aplicación de las ontologías a estos modelos de información ha sido múltiple:

- En primer lugar, se ha tomado un marco empleado para comparar los lenguajes de ontologías para analizar la expresividad semántica que los lenguajes de información de gestión son capaces de aportar. De esta comparación, MOF/CIM ha sido el lenguaje que ha obtenido el mejor resultado. Al mismo tiempo, se han identificado las facetas comunes en los atributos de los lenguajes de gestión.
- En segundo lugar se ha tratado la mejora de la capacidad semántica de un lenguaje de definición de información de gestión. Para ello se han propuesto dos posibilidades:
    - La primera ha consistido en formalizar MOF/CIM, el lenguaje mejor clasificado en la comparación anterior. Asimismo se han indicado otras posibles mejoras, como calificadores que añadan cuestiones relativas a la definición de jerarquías y la posibilidad de usar ontologías generales comunes.
    - La segunda ha consistido en analizar cómo se podría aprovechar un lenguaje de definición de ontologías como DAML+OIL, tomando las

construcciones que ya posee y definiendo nuevas estructuras para aquellos casos en que sea necesario expresar alguna faceta común en los lenguajes de información de gestión.

- Por último, se han aprovechado las técnicas de fusión y correspondencia existentes en las ontologías para posibilitar una integración semántica de modelos de información de gestión.
    - Dado que es mejor poseer un modelo de información común, se puede aprovechar el definido en CIM, pues esto aporta ventajas a la hora de realizar la integración con otros modelos que se fusionen y correspondan con éste.
    - En lo que se refiere al proceso de fusión, se han adaptado los heurísticos para buscar candidatos al caso concreto de los modelos de información de gestión, aprovechando las características que éstos poseen.
    - En lo que se refiere a correspondencia, se han clasificado los tipos de correspondencia que se pueden dar en el caso de modelos de información de gestión, definiendo una pequeña ontología de correspondencia que cumple con los requisitos necesarios para traducir entre distintos modelos de información.
    - Finalmente, se ha descrito el método M&M que expone como realizar este procedimiento de una manera ordenada e implementable.

Una vez que se ha propuesto como obtener este modelo común y sus ontologías de correspondencia asociadas, sería útil ver cómo introducirlas en un sistema de gestión. Para ello, el capítulo siguiente propone una arquitectura de gestión basada en el uso de ontologías. En él, se tratará de explicar cómo añadir aspectos que especifiquen el comportamiento del gestor en la información de gestión, haciendo uso de los lenguajes presentados en este capítulo. La expresividad que proporcione cada uno de ellos será determinante para poder expresar dicho comportamiento. Además, se mostrará cómo debería ser un sistema de gestión que maneje las características expuestas en este capítulo, pudiendo igualmente tratar con cuestiones relativas al comportamiento.

Capítulo 6 Propuesta de arquitectura para un sistema de gestión basado en ontologías

6.1 Introducción

El capítulo anterior ha presentado una propuesta de integración de los distintos modelos de información de gestión mediante el uso de ontologías. Éstas han sido fundamentales a la hora de comparar la capacidad para expresar semántica de los lenguajes de información de gestión. Asimismo, se ha estudiado la posibilidad de que esta información de gestión se pueda definir con un lenguaje de ontologías que proporcione la semántica necesaria para trabajar a ese nivel. Finalmente, las técnicas empleadas en las ontologías para llevar a cabo tareas de fusión y correspondencia han sido vitales para definir un método que permita obtener un modelo común a partir de las distintas especificaciones de información. Con esto, se puede llevar a cabo una gestión con un único modelo de información, independientemente de los dominios de gestión a los que pertenezcan los distintos recursos, alcanzando la interoperabilidad semántica necesaria para ello.

A partir de la propuesta anterior, es necesario que un sistema de gestión pueda trabajar con este modelo común basado en ontologías, pues esto le permitirá poder tener una vista única de todos los recursos gestionados, independientemente de que pertenezcan a uno u otro dominio de gestión. Además, la solución presentada en el capítulo anterior no resulta completa si no puede aplicarse a un sistema de gestión.

Asimismo, se puede aprovechar la capacidad de formalización de los lenguajes de ontologías para definir restricciones sobre la información que modela los recursos gestionados. Esto permitirá especificar en cierta manera el comportamiento del sistema de gestión, automatizando en gran medida la operación del mismo, sumando una ventaja adicional a las ya expuestas para utilizar lenguajes de ontologías en la definición de información de gestión.

Las siguientes secciones presentan esta propuesta de arquitectura para un sistema de gestión basado en ontologías. Para ello, como contribuciones originales de esta tesis doctoral:

- Se analiza la forma de incluir en las especificaciones de información de gestión el comportamiento de un gestor aprovechando las características de formalización que proporcionan los lenguajes de ontologías, pudiendo integrar dichas especificaciones

en el modelo común generado a partir del capítulo anterior para poder tener modelos de información completos como ocurre con las ontologías pesadas.

- Se propone un diseño de arquitectura para un gestor que acceda a los distintos dominios de gestión integrada desde una interfaz unificada e integrada semánticamente. Para ello, este gestor deberá ser capaz de manejar el modelo común basado en ontologías que se genere a partir de los procedimientos indicados en el capítulo anterior y que se mejore con las reglas de comportamiento que se especifiquen.

6.2 Análisis de lenguajes de especificación de comportamiento desde el punto de vista de su inclusión en las definiciones ontológicas

En la sección 0 del capítulo anterior se presentaron distintas alternativas para definir información de gestión aprovechando las características semánticas de los lenguajes de ontologías. Por un lado, se propuso la posibilidad de formalizar MOF/CIM, un lenguaje de gestión, y por otro, se analizó la capacidad para definir información de gestión de DAML+OIL, un lenguaje de ontologías.

Al mismo tiempo, ninguno de los modelos de información de gestión estudiados integra las reglas que gobiernan el comportamiento de un gestor inteligente que sea capaz de resolver consultas. Esta sección complementa esta cuestión, mostrando cómo los lenguajes presentados anteriormente pueden especificar comportamiento. Estas especificaciones se basan principalmente en la posibilidad de poder definir restricciones sobre la información definida. A partir de estas restricciones se pueden realizar razonamientos sobre la información, cuestión que no sería posible sin las mismas. Por ejemplo, el sistema que tuviera estas reglas cargadas debería comprobar que se cumplen cada vez que acceda a los valores de las distintas propiedades de los ejemplares. Si dejan de cumplirse, se generaría una alarma en el sistema gestor para informar de la incidencia.

Dichas restricciones, también llamadas axiomas en el dominio de las ontologías, muchas veces se encuentran descritas de manera *implícita* en los campos de descripción de las clases y atributos de información de gestión, utilizando lenguaje natural. Sin embargo, esta falta de formalidad impide que un sistema pueda interpretarlas. Estas restricciones implícitas siempre se deben cumplir en un estado de operación normal, porque así es como se ha definido la información. Su utilidad está en que permiten comprobar que los recursos se encuentran configurados correctamente.

Otras veces, es el gestor el que necesita restringir de forma *explícita* los valores que pueda tomar la información previamente definida para conseguir que la red trabaje dentro de rangos operativos, siguiendo una política concreta. La definición de estas restricciones explícitas puede ser tan útil para monitorizar el rendimiento de los recursos como para

gestionar la configuración de los mismos, estableciendo los parámetros máximos o mínimos que deben cumplir los recursos que se estén configurando, como por ejemplo el número de usuarios, carga de la máquina, uso de recursos (memoria, disco…).

A continuación, se presentan las distintas propuestas existentes en este sentido y se analizan distintas posibilidades, desde el punto de vista de los lenguajes de gestión y desde el de los de ontologías.

6.2.1 Propuestas existentes

A la hora de hablar de especificación de comportamiento en lenguajes de gestión conviene recordar que esta cuestión no es nueva. Ya en su momento, al definirse GDMO, se definió una plantilla de comportamiento `BEHAVIOUR`, aunque ésta no aportaba más que una descripción textual pues su contenido puede ser totalmente arbitrario.

Para mejorar la definición de comportamiento en GDMO ha habido varias propuestas. Por un lado, aquéllas que definían un lenguaje para ello, y por el otro, aquéllas que reutilizaban técnicas de descripción formal como SDL (*Specification and Description Language*, Lenguaje de Especificación y Descripción) [ITUT02].

En el primer caso se encuentra la aproximación de [Keller95], que define un conjunto de nuevas plantillas de comportamiento que se integran perfectamente en GDMO. Cada una de ellas se refiere al resto de plantillas, pudiendo definir, con cláusulas *si-entonces* y un conjunto de operaciones de comparación de valores, las condiciones en que se emite una notificación, o que se solicita una acción. Esta aportación ha sido considerada para su inclusión en la recomendación de GDMO, aunque finalmente no forma parte del estándar. Por otro lado, en [Hasselmeyer99] se propone el uso de un lenguaje ad-hoc que resuelve la distribución de las definiciones de comportamiento, incluidas en cada una de las plantillas que componen una clase de objeto.

En el segundo caso se encuentra [Rodríguez99a,Rodríguez99b] que a su vez basa sus trabajos en [Bartocci95]. Estas aproximaciones definen una correspondencia entre las plantillas GDMO y las construcciones de SDL, con lo que pueden definir directamente en este lenguaje el comportamiento de gestores y agentes con respecto a la información que están manejando. Por otro lado, existe una enmienda a GDMO [ITUT97c] que indica como formalizar el comportamiento de las clases de objetos gestionados con Z.

En [Duarte99] se sigue una filosofía parecida para MIBs SNMP, añadiendo reglas definidas con *Action Semantics*[1] en el campo `DESCRIPTION` de cada objeto.

[1] http://www.brics.dk/Projects/AS/

Las ideas contenidas en estos trabajos son en general válidas. El gran inconveniente de estas propuestas es precisamente que utilizan GDMO y SMI, que son lenguajes con menos capacidad expresiva que MOF/CIM, y mucho menos si a este último lenguaje se le añaden las características formales y semánticas presentadas en el 5.3.2.

En lo que se refiere al modelo de WBEM, en [Tosic99] se propone extender MOF/CIM para mejorar la forma en que se describen los métodos. En [Tosic98] se explica en detalle esta idea que se basa en la definición de nuevos calificadores en los que se añaden precondiciones, poscondiciones e invariantes a los métodos que se definan. Sin embargo, no amplía esta extensión para definir comportamiento de las clases en general, que también podrían tener invariantes como se verá a continuación.

La gestión basada en políticas [Sloman02] es otra forma de proporcionar reglas que gobiernen el comportamiento, usándose actualmente para implementar sistemas de gestión de una manera flexible y adaptativa. Un problema de la gestión basada en políticas es que las reglas se definen de manera independiente a la información de gestión, aunque un grupo conjunto del IETF e DMTF trabaja en integrar dichas reglas como una parte de los esquemas de CIM [Moore01]. Sin embargo, esta integración se limita a definir clases que describan políticas, y no la asociación de políticas a clases de los esquemas CIM.

Finalmente, también existen otras propuestas, como la descrita en [Bénech99], que utiliza un lenguaje declarativo como Prolog para especificar el comportamiento de un gestor. Aunque esta alternativa interopera con SNMP y DMI, tiene como inconveniente que define las reglas de manera independiente a la información de gestión, lo que aísla su relación con dichos modelos. Las propuestas que se presentan en los apartados siguientes sí que integran las reglas de comportamiento dentro de las especificaciones de información, siendo el objetivo principal que el modelo unificado, obtenido a partir de los resultados del capítulo anterior, posea dicha formalización.

6.2.2 Propuesta de especificación del comportamiento en lenguajes de información de gestión

Este apartado está íntimamente ligado al que proponía el uso de lenguajes de información de gestión para definir ontologías (5.3.2), pues toca de lleno la formalización, aunque en este caso es del comportamiento. Es decir, en este caso no se formaliza el metamodelo, sino los modelos de información de gestión.

Aunque ha habido varias propuestas para formalizar la plantilla de comportamiento de GDMO, esta definición no ha sido extensiva al resto de lenguajes de información de gestión. Por ejemplo, MOF/CIM, lenguaje mejor dotado semánticamente de los analizados previamente, no posee ningún calificador para definir este tipo de cuestiones. Se podría definir, por tanto, un calificador a este efecto de la siguiente manera:

```
Qualifier Constraint : string = null, Scope (any);
```

Esto es, una restricción vendrá definida por una cadena de caracteres que por defecto será nula y cuyo ámbito serán todas las construcciones posibles de MOF/CIM.

Al mismo tiempo, se puede definir un calificador adicional que indique el lenguaje empleado para definir esa restricción. Por ejemplo, si se supone que el lenguaje de restricciones empleado por defecto es OCL [OMG01c], se podría definir de la siguiente manera:

```
Qualifier ConstraintLanguage : string = "OCL", Scope (any);
```

Se ha elegido OCL como lenguaje por defecto debido a su fácil integración en CIM, dado que es el lenguaje de restricciones de UML. Si en el capítulo anterior se empleaba para definir las reglas que gobiernan el metamodelo de CIM, ahora se utiliza para definir en otro nivel de abstracción distinto las restricciones de los modelos CIM.

OCL permite definir precondiciones, poscondiciones e invariantes. Las precondiciones son útiles para restringir los valores que se pueden pasar a un método y las poscondiciones para restringir el valor que dicho método puede devolver. Las condiciones invariantes valen para definir restricciones sobre atributos.

Así, se podrían definir restricciones como las incluidas en las descripciones de las clases de manera implícita. Por ejemplo, siguiendo con el caso de estudio presentado en el capítulo anterior, la clase `CIM_Printer` representada en la Figura 5.10, contiene en la descripción de su propiedad `DefaultLanguage` que "`a language that is used as a default by the Printer should also be listed in LanguagesSupported`". De esta manera, se puede definir una regla que permita comprobar que esto es cierto. Dicha regla se incluiría en un calificador `Constraint` aplicado a la clase cuya codificación en OCL sería:

```
[ Constraint (
"context CIM_Printer "
"inv: self.LanguagesSupported->includes(self.DefaultLanguage)" ) ]
```

Esto es, para el contexto de la clase que modela impresoras, todos sus ejemplares incluirán el lenguaje por defecto en el conjunto de lenguajes soportados.

También se pueden definir restricciones de forma explícita para que los recursos funcionen dentro de un rango, siguiendo una política concreta. Por ejemplo, si se quiere restringir que el espacio disponible en un sistema de archivos no baje del 10% del total, se podría definir la siguiente restricción para el atributo `AvailableSpace` de la clase `CIM_FileSystem`, también representada en la Figura 5.10:

```
[ Constraint (
"context CIM_FileSystem "
"inv: self.AvailableSpace > self.FileSystemSize * 0.10" ) ]
```

Es decir, para el contexto de la clase que modela sistemas de archivos, el espacio disponible será mayor que 0,10 veces el tamaño de cada sistema.

Como se comentaba anteriormente, las restricciones se pueden definir en otro lenguaje distinto a OCL, aunque en ese caso habría que indicarlo con el calificador `ConstraintLanguage`. Por ejemplo, se podría emplear un lenguaje similar a KIF como PAL (*Protégé Axiom Language*, Lenguaje de Axiomas de Protégé) [Crubézy02], que permitiría definir las reglas anteriores como se describe a continuación.

Para el caso de la impresora, la regla se definiría:

```
[ ConstraintLanguaje ("PAL"),
Constraint (
"(defrange ?printer :FRAME CIM_Printer)"
"(forall ?printer "
"     (element-of (DefaultLanguage ?printer) "
"                 (LanguagesSupported ?printer) ) )" )]
```

Y en el caso del sistema de archivos, se podría expresar:

```
[ ConstraintLanguaje ("PAL"),
Constraint (
"( defrange ?fs :FRAME CIM_FileSystem ) "
"( forall ?fs "
"     ( > ( AvailableSpace ?fs ) ( * 0.10 ( FileSystemSize ?fs ) ) ) )"
)]
```

La elección de uno u otro lenguaje vendrá dada por aquél que entienda el motor de inferencias que se emplee para comprobar las restricciones.

6.2.3 Especificación del comportamiento con lenguajes de ontologías

Siguiendo el razonamiento empleado en el apartado anterior de analizar la inclusión de información de comportamiento en un lenguaje de gestión como CIM, éste pretende analizar qué tipo de información de comportamiento se puede añadir utilizando un lenguaje de definición de ontologías.

Si el estudio se restringe a DAML+OIL, se puede ver que la definición de restricciones en este lenguaje es bastante limitada. En general sólo se pueden definir restricciones de rango y de cardinalidad de las propiedades, que pueden ser tanto de tipo `daml:ObjectProperty` como `daml:DatatypeProperty`. Las marcas empleadas para dichas restricciones son las siguientes:

- Para definir una restricción se emplea `daml:Restriction`, que define una clase anónima que incluya todos los ejemplares que satisfagan la restricción. Existe una diferencia semántica entre la definición de restricciones en OCL y en los lenguajes de ontologías. En la primera, las restricciones se violan, y en la segunda, se cumplen y de esto se deduce conocimiento [Backlawski01]. Es decir, aquellos objetos que no satisfagan la restricción en este caso simplemente no pertenecen a esa subclase. Entonces, si se definieran restricciones de este tipo, un sistema de

gestión debería estar pendiente de que no existan ejemplares que no pertenezcan a dicha clase anónima.

- Para indicar a qué propiedades se aplica la restricción se emplea la marca `daml:onProperty`, a la que se añadirá el identificador `rdf:ID`.
- Para indicar una restricción de universalidad de rango (todas las propiedades serán ejemplares de una clase), se emplea `daml:toClass`, incluyendo el identificador de la clase.
- Para indicar una restricción de existencialidad de rango existen las construcciones `daml:hasValue` y `daml:hasClass`, que indican respectivamente que la propiedad debe tener al menos un valor, o un ejemplar que sea de la clase concreta.
- Para indicar una cardinalidad concreta, existen las marcas `daml:cardinality`, `daml:maxCardinality`, `daml:minCardinality`, así como estas mismas acabadas con la letra Q en caso de que la cardinalidad esté calificada.

Este tipo de restricciones funciona relativamente bien cuando se trabaja con `daml:ObjectProperty`, pues permiten indicar los ejemplares que pueden existir. Esto se debe a que las restricciones están orientadas a hacer razonamientos de tipo clasificativo. Sin embargo, las restricciones no se pueden aplicar tan bien para `daml:DatatypeProperty`, pues las restricciones típicas sobre datos definen rangos de valores especificados mediante operaciones aritméticas sobre las propiedades. En DAML+OIL no se pueden definir dichas operaciones, y por tanto, no es posible especificar restricciones como la que se planteaba de manera explícita en el apartado anterior acerca del espacio disponible en un sistema de archivos. Lo que sí es posible es redefinir un tipo de datos XSD y utilizarlo para restringir los valores máximos o mínimos que una propiedad puede tomar. Por ejemplo, si el espacio disponible se define en porcentaje del espacio total, en vez de kilobytes, se podría plantear la siguiente restricción:

```
<xsd:simpleType name="over10">
      <xsd:restriction base="xsd:positiveInteger">
            <xsd:minInclusive value="10"/>
            <!-- Como es un porcentaje se podría incluir una restricción
de máximo -->
            <xsd:maxInclusive value="100"/>
      </xsd:restriction>
</xsd:simpleType>

<daml:Class rdf:about="#CIM_FileSystem">
      <daml:Restriction>
            <daml:onProperty rdf:resource="#AvailableSpace"/>
            <daml:toClass rdf:resource="#over10"/>
```

```
      </daml:Restriction>
</daml:Class>
```

Así, se definiría un tipo de datos cuyo valor menor fuera 10, y se aplicaría la restricción para que la propiedad deba tener dicho tipo de datos. Sin embargo, esto no permite definir la regla como se hacía anteriormente, puesto que no se pueden aplicar restricciones de tipos de datos a los elementos del dominio de los objetos [Horrocks02].

Por otro lado, la primera regla definida en el apartado anterior, relativa de forma implícita a la clase `CIM_Printer`, se podría especificar de la siguiente manera:

```
<daml:Class rdf:about"#CIM_Printer">
      <daml:Restriction>
            <daml:onProperty rdf:resource="#LangaugesSupported"/>
            <daml:hasClass rdf:resource="#DefaultLanguage"/>
      </daml:Restriction>
</daml:Class>
```

La cuestión que se puede presentar en este caso es que se utiliza una propiedad (`DefaultLanguage`) como una clase, si bien el rango de `daml:hasClass` no está definido como `daml:Class`, sino como `rdfs:Class`, y `daml:DatatypeProperty` es una clase `rdfs:Class`. Esta posible incongruencia puede deberse a que la definición de DAML+OIL no emplea el modelo de cuatro niveles de UML, sino que existe una mezcla de niveles que provoca este tipo de cuestiones. Por esto, en [Pan01] se propone una definición DAML+OIL con distintos niveles para separar claramente los meta-metalenguajes de los metalenguajes y los lenguajes.

A la hora de definir restricciones, entendidas como subconjuntos de clases, también se puede jugar con las operaciones de conjunción, disyunción, igualdad y complemento, definidas como `daml:intersectioOf`, `daml:unionOf`, `daml:sameClassAs` y `daml:complementOf`, así como las clases `daml:Thing` y `daml:Nothing`, que representan al conjunto de todas las clases y al conjunto vacío respectivamente. Utilizando estas construcciones se podría indicar que aquellas clases que posean una propiedad con un valor concreto sean complementarias a la clase `daml:Thing` o igual a la clase `daml:Nothing`, es decir, que pertenecen al conjunto vacío.

Como se ve, la definición de restricciones en este lenguaje de la Web Semántica no es tan completa como en el caso de los lenguajes que se empleaban en el apartado anterior. Cuando se comenzó el desarrollo de lo que ahora es DAML+OIL, se pensó en crear un lenguaje que permitiera definir construcciones lógicas como las aquí vistas. Dicho lenguaje se llamaría DAML-L (*DAML Logic*, DAML Lógico) [Hendler00] pero hasta la fecha no ha salido a la luz ninguna especificación del mismo. Actualmente se trabaja en otra iniciativa[2]

[2] http://www.daml.org/rules/

para integrar en DAML+OIL el lenguaje de reglas llamado RuleML (*Rule Markup Language*, Lenguaje de Marcas de Reglas) [Grosof02].

Por otro lado, esta carencia no implica que sea extensible al resto de lenguajes de ontologías y de hecho, Ontolingua, basado en KIF permite definir restricciones del mismo tipo que se han visto en esta sección. De hecho, la formalización de DAML+OIL no está hecha con ningún lenguaje derivado de XML, sino con KIF, y además, PAL también está basado en KIF.

6.2.4 Resumen de las soluciones propuestas

Esta sección ha analizado la forma en que se puede especificar comportamiento utilizando los lenguajes de definición de información analizados previamente mediante reglas que impongan restricciones sobre dicha información. Estas reglas permitirían a un sistema gestor razonar y determinar si los sistemas gestionados están operando correctamente a partir de la información que se maneja, o por el contrario si algún parámetro no entra dentro de las restricciones que se han planteado. Se han analizado dos posibilidades:

- Como contribución original de la tesis, para el caso de lenguajes de información de gestión, se ha seguido empleando MOF/CIM definiéndose un calificador que permita la inclusión de restricciones en un lenguaje al efecto, como pueda ser OCL, aunque sin limitar el uso de otros, como KIF o su derivado PAL.
- Para el caso de lenguajes de ontologías, se han analizado las posibilidades que proporcionan las construcciones de DAML+OIL, constatándose que aunque es posible definir alguna restricción sobre la información definida, no se ofrece el mismo nivel expresivo que poseen otros lenguajes de definición de ontologías.

A continuación se propone la arquitectura que deberá poseer un sistema de gestión que utilice el modelo de información generado el capítulo anterior, completado, según se ha visto en esta sección, con un conjunto de restricciones que le permitan establecer pautas de comportamiento.

6.3 Diseño y propuesta de la arquitectura de un gestor de red basado en ontologías

El capítulo anterior presentaba una propuesta para integrar la información de gestión mediante la fusión de los modelos existentes y la definición de reglas de correspondencia, habiéndose propuesto en la sección previa añadir otra información a este modelo integrado que permita especificar el comportamiento de los sistemas gestores con respecto a dicha información. A continuación se detalla cómo aplicar las propuestas anteriores a un sistema de gestión de red que sea capaz de manejar las ontologías que se generen. Este estudio es necesario dado que aunque existen muchas similitudes entre las ontologías y los modelos de información de gestión, también son muchas las características contrapuestas. Por

ejemplo, si en un sistema que maneja ontologías el razonamiento se realiza con ejemplares conocidos, en un sistema de gestión, esta información no se conoce a priori.

La integración de ambos paradigmas haría del gestor un motor ontológico que podría realizar la gestión basándose en las ontologías definidas. Para esto hay que proponer la arquitectura que defina dicho sistema de gestión, que por un lado se comportaría en función de las reglas especificadas, y por otro lado, se aprovecharía de la ontología de correspondencia para obtener la información de cualquier dominio de manera automática.

Al mismo tiempo, de la misma forma que se han utilizado los lenguajes y modelos de información de gestión, es conveniente aprovechar las arquitecturas que se han definido para interoperar entre distintos dominios gestión de red. Esto supondrá la adaptación de la que hasta ahora ha tenido más éxito en este sentido, esto es, WBEM.

A continuación se presentan otros trabajos relacionados, para pasar a exponer tras esto aquellas características que habría que incluir en un sistema de gestión para que utilice ontologías. Más tarde, el siguiente apartado expone las ideas que tratan de incluir el manejo de ontologías en la arquitectura de WBEM. Finaliza la sección un análisis de esta propuesta.

6.3.1 Otros trabajos relacionados

Hasta el momento existen pocos trabajos que presenten gestores que hagan uso de las ontologías. Sin embargo, sí que se han estudiado algunas propuestas referentes a gestores inteligentes que trabajan con modelos de gestión integrada.

En este sentido, se podrían comentar el trabajo contenido en [Zhang96b], íntimamente relacionado con los artículos referentes a la formalización de los mismos autores. En dicha aproximación se emplea la formalización de GDMO para definir la base de conocimiento que emplee un sistema multiagente. Sin embargo, este trabajo no maneja ontologías, ni tampoco está referida a otros dominios de gestión distintos a OSI-SM. También otros trabajos relacionados con la formalización han tenido su origen en poder definir modelos de información que puedan manejar agentes inteligentes. Tal es el caso de [Lavinal03] y [Shen03]. Estos trabajos sí que utilizan ontologías (usando OKBC en el primer caso y RDFS en el segundo) pero en ninguno de ellos se trabaja con distintos modelos de información de gestión integrada, estando el primero relacionado con CIM y el segundo con SMIng.

Por otro lado, en [Bénech99] se presenta un sistema de gestión inteligente basado en un motor de inferencias Prolog con características de distribución y cooperación que es capaz de manejar modelos de información de SNMP y DMI. Sin embargo, y pesar de que usa CORBA para la implementación del sistema, no es capaz de acceder a otros dominios de gestión, y además, no usa ontologías para su funcionamiento. En [Steff99] se propone el uso de KQML (*Knowledge Query and Manipulation Language*, Lenguaje de Manipulación

y Consulta de Conocimientos) como lenguaje de diálogo entre gestores que colaboren, pero ni siquiera se menciona el acceso a modelos de gestión integrada.

En definitiva, ninguno de los trabajos estudiados sobre sistemas de gestión inteligentes es capaz de llevar a cabo una gestión de todos los modelos de gestión integrada existentes, trabajando con ontologías que permiten utilizar un modelo de información unificado e integrado semánticamente que además contiene reglas de comportamiento, como se propone en los siguientes apartados.

6.3.2 Adición de características de ontologías a sistemas de gestión

A continuación se presentan algunas ideas acerca de la forma en que se pueden integrar las ontologías en los sistemas de gestión, dadas las diferencias funcionales y estructurales que existen entre ambos. Ninguno de los sistemas de gestión existentes en el mercado es capaz de manejar directamente ontologías ni poder inferir conocimiento a partir de ellas. Será necesario adaptarlo según se describe en el resto del apartado.

Para razonar a partir de las reglas de comportamiento y poder realizar la gestión basándose en ellas, el sistema gestor debe poseer un motor de inferencias que utilice la ontología de gestión junto con el conjunto de restricciones que se definieran sobre ella. Como se ha comentado anteriormente, esto puede suponer una diferencia de funcionamiento con respecto a los sistemas que manejan ontologías. En éstos el razonamiento se realiza a partir de una base de conocimientos con ejemplares conocidos, dado que no se puede razonar sin información. Sin embargo, la información relativa a los recursos gestionados no se conoce a priori en un gestor, puesto que esta información está contenida en los agentes y es necesario realizar tantas solicitudes como sea necesario a dichos agentes para obtener los distintos ejemplares de la información a gestionar.

Por otro lado, los sistemas de gestión normalmente realizan sondeos periódicos de la información de gestión necesaria y la van almacenando en sus bases de datos. Por ello, si el razonamiento se efectúa sobre los datos que el gestor va recolectando en cada sondeo se puede solventar este problema. La base de conocimientos sobre la que trabaje el motor de inferencias se generará con los datos obtenidos por el gestor, razonando a partir de la información sondeada en cada momento. También hay que tener presente que la información que aportan los agentes varía con el tiempo, por lo que los ejemplares de la base de conocimiento se deberán actualizar según lo haga la información de los recursos.

Otra cuestión que se debe tener en cuenta para que un gestor maneje los modelos de información basados en ontologías es la que se refiere a la forma en que se apliquen las reglas de correspondencia. Las plataformas de gestión normalmente acceden a un dominio concreto, o acceden a varios pero sin ningún tipo de transparencia (las aplicaciones deben conocer el dominio con el que se va a trabajar), no siendo válidas para el problema en estudio. En este caso hay que pasar claramente a un modelo de gestión paraguas como el propuesto en WBEM. Sin embargo, la mayor diferencia con este último es precisamente el

uso de la ontología de correspondencia definida en el capítulo anterior que permite una traducción semántica. Desde el punto de vista de las ontologías, proyectos como Observer [Mena00] proponen que la traducción entre modelos de información se resuelva mediante un sistema de intermediación. Así, en este sistema habría un conjunto de proveedores a los que se accedería para obtener la información de cada dominio concreto. Si se utilizan ontologías de correspondencia que proporcionan las reglas que debe aplicar para traducir semánticamente los ejemplares de un dominio específico a un dominio común, entonces estos proveedores pueden ser genéricos para cada dominio de gestión. Para ello, cada proveedor debería cumplir al menos estos dos requisitos:

1. Manejar los ejemplares de la ontología de correspondencia referidos a su dominio para así poder traducir semánticamente la petición de la información que se le solicita.
2. Implementar las llamadas al protocolo de intercambio propio de su dominio y así poder actuar de pasarela con el sistema de gestión.

El siguiente apartado muestra como aplicar estas características a una arquitectura de gestión concreta.

6.3.3 Aplicación a la arquitectura de gestión basada en Web

A continuación se verá la aplicación de las características expuestas en el apartado anterior a la arquitectura de gestión basada en Web conocida como WBEM. Esta propuesta se realiza sobre WBEM por varios motivos:

1. Es conveniente aprovechar los estándares ya definidos en vez de proponer algo completamente nuevo, pues ello redundará en una adopción más rápida por parte de los organismos implicados.
2. WBEM es una arquitectura pensada para la gestión multidominio, por lo que su utilidad está íntimamente relacionada con los objetivos aquí planteados.
3. Tanto el modelo de información común como el lenguaje de definición de la información que se han propuesto en esta tesis se basan en estándares relacionados con WBEM, por lo que su uso será más sencillo en este caso.

Por tanto, si se quieren aplicar las características anteriormente expuestas a WBEM, se podría actuar de la siguiente manera:

- El CIMOM se encargaría de mantener el modelo fusionado y comprobar mediante un motor de inferencias que las restricciones indicadas en el modelo se cumplen para todos los ejemplares que va almacenando. El hecho de que los CIMOM mantengan los calificadores permite que se trabaje con ellos, incluyendo a los calificadores que impongan restricciones.

- El CIMOM debe saber a qué proveedor redirigir las solicitudes de información. Al mismo tiempo, y para poder realizar las inferencias sobre el conocimiento de gestión, el CIMOM tendría que tener establecido un mecanismo de sondeo de información que actualmente no tiene definido, si bien, sí que existe el repositorio de objetos CIM.
- Las pasarelas a los distintos dominios de gestión, que en WBEM casualmente se llaman proveedores, se encargarían de recibir las peticiones del CIMOM y realizar las solicitudes adecuadas según las correspondencias definidas entre el modelo común manejado por el CIMOM y el modelo propio de cada dominio. De hecho, el Capítulo 7 de [DMTF99] propone que en el repositorio de CIM se mantengan anotaciones para traducir entre dominios, si bien propone únicamente el uso de calificadores como `MappingStrings`.

Otra pregunta que surge al proponer esta arquitectura es cómo sabe el CIMOM qué proveedor proporciona la información que necesita. Para ello existen varias posibilidades:

1. Si el CIMOM conoce las correspondencias para cada elemento puede saber a qué dominio se corresponde cada uno. Si una clase posee correspondencias con un único dominio, redirigirá al proveedor de dicho dominio la solicitud. Si posee varios dominios, la redirigirá a todos ellos. El problema que se puede plantear es que se obtengan varios ejemplares similares (por ejemplo, de un recurso que tiene un agente SNMP y otro DMI que proporcionan la misma información). Para evitar esto, se puede establecer un orden de preferencia. Si un dominio devuelve información, se toma como correcta y si no, se pregunta al siguiente dominio y así sucesivamente.
2. Otra posibilidad consiste en que sean los proveedores los que, conociendo las traducciones que pueden realizar, se registren en el CIMOM indicando que pueden proporcionar la información correspondiente a los mismos. Esta solución es más elegante, pues el CIMOM no tiene por qué saber de correspondencias, sino de proveedores. Además, es coherente con la definición del calificador `Provider`, que vale para indicar qué proveedor puede proporcionar una información. Sin embargo, sigue existiendo el problema de que al haber varios dominios pueda replicarse una información. El CIMOM debería encargarse de comprobar la unicidad de las claves, y ante la aparición de un mismo ejemplar por parte de dos proveedores, desechar uno de ellos (usando reglas basadas, por ejemplo, en el que menos información proporcione).
3. También se pueden hacer solicitudes por inundación. El proveedor que posea la correspondencia responderá, y el que no, no responderá o responderá que le falta la de regla de correspondencia de ese elemento. Este algoritmo permite aprender al CIMOM cuáles son los proveedores de cada clase del modelo común.

Si se estudia qué se hace en algunas implementaciones, ocurre que hay distintos casos:

- En el caso de Microsoft WMI (*Windows Management Instrumentation*, Instrumentación de Gestión de Windows) [Microsoft02], cada proveedor se registra para decir qué clases puede proporcionar. Por ejemplo, el proveedor SNMP se registra en propiedades estáticas de clase `__Namespace`, que a su vez es una clase especial.
- En el caso de Sun WBEM [Sun02], se puede registrar un proveedor indicándolo con el calificador estándar `Provider` comentado anteriormente. El calificador se puede aplicar tanto a la clase como a atributos o métodos individuales de manera independiente. Por tanto, se podría considerar que puede ser buena la opción de que los proveedores estén indicados con este calificador en la información que maneja el gestor.

Con todo, la aplicación de ontologías a la arquitectura elegida quedaría como muestra la Figura 6.1:

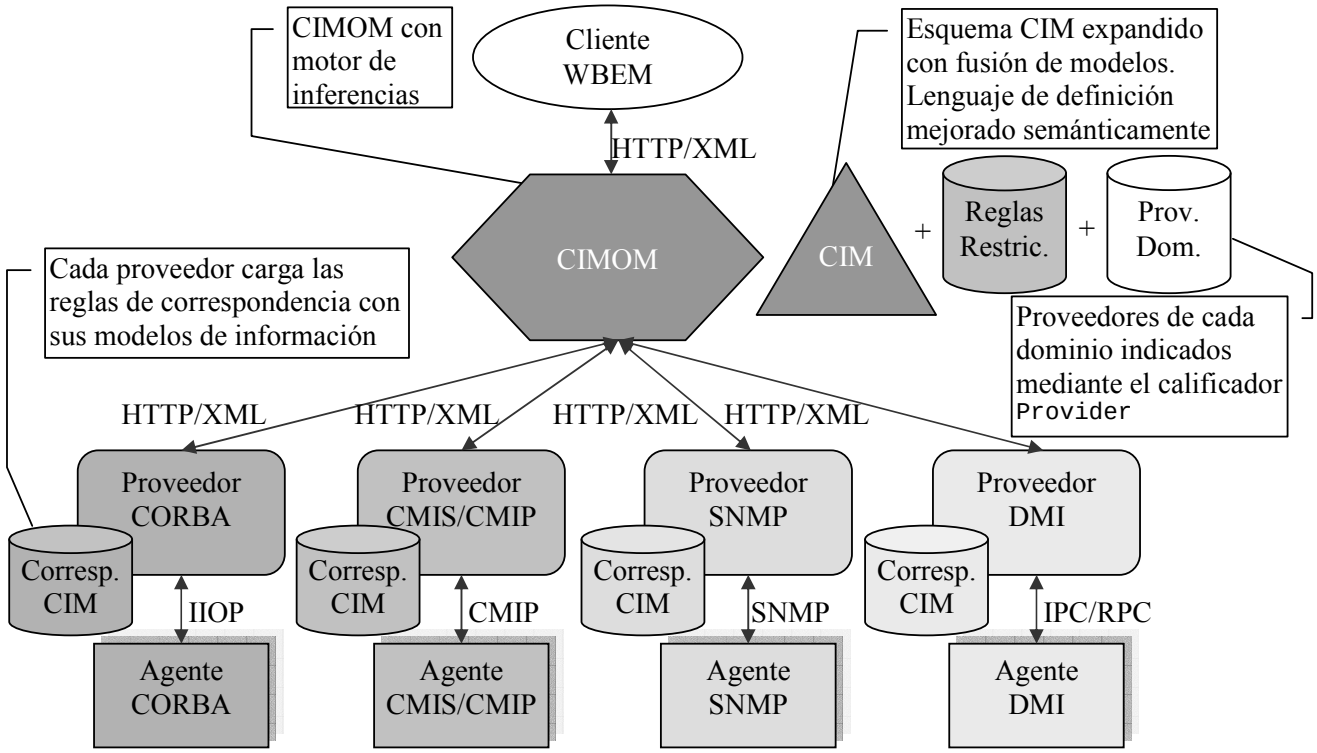

Figura 6.1. Propuesta de arquitectura de gestión web semántica.

6.3.4 Análisis de la propuesta

Esta sección ha tratado de describir cómo aplicar los métodos propuestos para la utilización de ontologías en sistemas de gestión. Para ello primero se han estudiado los problemas que pueden existir desde un punto de vista general, como la forma en que se aplican las restricciones o las reglas de correspondencia y a continuación se han tratado de aplicar al caso concreto de la gestión basada en Web, en el que también se ha estudiado el

problema de conocer cuál es el proveedor que proporciona la información que necesita el gestor de objetos.

Las ideas aquí contenidas valdrían no sólo para implementar un gestor centralizado, sino también para sistemas multiagente, que comparten una misma ontología, como se propone en [Lavinal03].

6.4 Conclusiones

El presente capítulo ha presentado una propuesta para aplicar la información de gestión fusionada mediante el uso de ontologías a gestores que puedan manejarla. Para ello, se ha expuesto una primera sección que complementa dicha información con la definición de reglas de restricción que permiten describir el comportamiento del gestor que las maneje, presentándose la forma en que se deberían definir tanto si se usase un lenguaje de gestión como MOF/CIM, como un lenguaje de ontologías como DAML+OIL. Tras esto, se ha presentado una arquitectura de un gestor que manejara estas ontologías, tratando de incluirlo en el contexto de la gestión basada en Web.

El capítulo siguiente presenta las conclusiones finales de la tesis. Para ello se dará una vista general del conjunto de soluciones propuestas durante el desarrollo de la misma, resumiendo las principales aportaciones y proponiendo líneas futuras de trabajo.

Capítulo 7 Conclusiones y trabajos futuros

7.1 Resumen

El principal objetivo de esta tesis doctoral ha sido proponer soluciones que permitan mejorar la interoperabilidad en lo que se refiere a los modelos de información de gestión, analizando los requisitos necesarios para llevar a cabo la traducción semántica. Debido a la amplitud de este tema y tras un repaso de los aspectos más importantes del estado del arte en lo que se refiere a la gestión de red integrada, los mecanismos de integración de información de gestión existentes y las ontologías como técnica de representación del conocimiento, la tesis se ha centrado en dos problemas concretos y estrechamente relacionados: la utilización de las ontologías para la integración de información de gestión y la definición de una arquitectura para un sistema de gestión basado en ontologías.

Después de haber descrito y comparado las principales propuestas existentes relacionadas con la interoperabilidad de modelos de gestión, se ha visto que éstas únicamente proporcionan traducciones sintácticas de la información que se maneja. Para conseguir una traducción semántica que permita a un gestor trabajar con todos los modelos desde una vista común, se ha propuesto la aplicación de las ontologías a este problema, dado que estas técnicas de representación del conocimiento se enfocan precisamente en la semántica de la información.

Así, en esta tesis se han empleado los conceptos y principios de las ontologías para la definición de información de gestión. Esta aproximación ha resultado ser muy útil en cada una de las contribuciones planteadas:

- Se ha analizado y comparado exhaustivamente los distintos lenguajes de definición de información de gestión existentes (GDMO, SMIv2, SMIng, MIF, IDL, MOF/CIM), estudiando sus capacidades y carencias expresivas a partir de las construcciones que posee cada uno. De este trabajo se concluye que MOF/CIM es el que tiene una mejor capacidad expresiva, con lo que la información que esté definida en este lenguaje será más fácilmente integrable semánticamente.
- Se ha propuesto mejorar la capacidad expresiva de los lenguajes de definición de información de gestión, proponiendo la aplicación de las características que poseen los lenguajes de ontologías. Para ello se han contemplado dos opciones:
    - Mejorar el lenguaje que obtuvo una mejor valoración en el punto anterior. Para ello se ha formalizado el metamodelo de MOF/CIM mediante reglas

OCL, para que las especificaciones definidas con este lenguaje posean también un grado de formalidad similar al de las ontologías. Asimismo, se ha planteado la adición de nuevos calificadores que mejoren la expresión de taxonomías, así como el uso de ontologías generales comunes para expresar tipos de datos y unidades de medida.

  - Ampliar un lenguaje de ontologías con facetas propias de lenguajes de información de gestión para poder definir dicha información, aprovechando las ventajas que desde un punto de vista semántico esto supone. Así, se han definido nuevas construcciones a incluir en el lenguaje DAML+OIL, ampliamente utilizado para la Web Semántica, de forma que pueda expresar facetas tales como el valor por defecto, el tipo de acceso o las unidades de medida.

- Se han aprovechado los métodos existentes de fusión y correspondencia de ontologías, particularizándolos para su aplicación a la integración de modelos de información de gestión integrada. El método resultante, llamado M&M., combina ambos procesos de fusión y correspondencia para obtener un modelo común, basado en los esquemas de CIM junto con los modelos que se le fusionen, y un conjunto de ontologías de correspondencia que contienen las reglas para traducir semánticamente entre este modelo común y los modelos de cada dominio de gestión particular. Este método se ha aplicado en un caso de estudio que integra la HOST-RESOURCES-MIB de SNMP dentro de los esquemas CIM.

- Se ha propuesto añadir a los modelos de información características referentes a su comportamiento, mediante la inserción de reglas de restricción que suelen incluirse en las ontologías. Con esto se puede formalizar dicha información, lo que a su vez permite que se interprete de manera automática. Siguiendo la aproximación de la mejora de los lenguajes de definición de información, se han contemplado nuevamente dos opciones:

  - En el caso de utilizar el lenguaje MOF/CIM mejorado, habría que añadir calificadores que contengan las restricciones del modelo, para poder formalizar la información definida, permitiendo añadir de esta manera reglas de comportamiento. Estas restricciones se pueden definir en OCL, u otro lenguaje como KIF.

  - Para el caso del lenguaje de ontologías DAML+OIL, se ha estudiado el tipo de restricciones que permite, comprobándose que su expresividad en este sentido no es tan completa como otros lenguajes de ontologías más clásicos, como Ontolingua.

- Se ha definido la arquitectura de un gestor que maneje dichas ontologías, aprovechando el modelo de información común y las reglas de correspondencia y de comportamiento obtenidas en los puntos anteriores. Dicho gestor podrá,

finalmente, realizar una gestión desde un punto de vista común y neutro a los dominios de gestión en los que se encuentran los recursos gestionados. A la vez, las restricciones definidas sobre la información podrían ayudar a automatizar las tareas de vigilancia y control de dicho gestor. Esta arquitectura se ha particularizado para el caso concreto de WBEM, indicando qué modificaciones se deberían incluir para que posea esta funcionalidad.

7.2 Valoración y análisis del trabajo realizado

Existen múltiples aspectos positivos relacionados con las contribuciones de la tesis, algunos de los cuales han sido expuestos en los siguientes artículos:

- Jorge E. López de Vergara, Víctor A. Villagrá, Julio Berrocal, ***Semantic Management: advantages of using an ontology-based management information meta-model***, en *Proceedings of the HP Openview University Association Ninth Plenary Workshop (HP-OVUA'2002)*, Böblingen, Alemania, 11-13 de junio de 2002.
- Jorge E. López de Vergara, Víctor A. Villagrá, Julio Berrocal, Juan I. Asensio, Roney Pignaton, ***Semantic Management: Application of Ontologies for the Integration of Management Information Models*****,** en *Proceedings of the Eighth IFIP/IEEE International Symposium on Integrated Network Management (IM'2003)*, Colorado Springs, Colorado, EE. UU. A., 24-28 de marzo de 2003.
- Jorge E. López de Vergara, Víctor A. Villagrá, Juan I. Asensio, Julio Berrocal, ***Ontologies: Giving Semantics to Network Management Models***, aceptado para su publicación en *IEEE Network*, special issue on Network Management, Volume 17, Number 3, mayo/junio de 2003.
- Jorge E. López de Vergara, Víctor A. Villagrá, Julio Berrocal, ***An ontology-based method to merge and map management information models***, aceptado para su publicación en *Proceedings of the HP Openview University Association Tenth Plenary Workshop*, Ginebra, Suiza, 6-9 de julio 2003.

Asimismo, las ideas aquí presentadas están siendo de gran utilidad para el desarrollo del proyecto GESEMAN (Definición y Desarrollo de Técnicas Basadas en el Conocimiento para su Aplicación a la Gestión de Redes y Servicios: Gestión Semántica) con número de referencia TIC2002-00934 del Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica del Ministerio de Ciencia y Tecnología.

Respecto al resto de trabajos estudiados, las contribuciones aportadas en esta tesis permiten mejorar el estado del arte en múltiples aspectos:

- Ninguno de los trabajos anteriores había realizado una comparación tan extensa sobre los lenguajes de información de gestión, ni había tenido en cuenta la

expresividad semántica de los mismos, con lo que no era posible comparar la información definida por todos los modelos de gestión integrada en su conjunto.

- Las formalizaciones de lenguajes de información de gestión no se habían aplicado hasta la fecha a MOF/CIM, que es el lenguaje con una mayor capacidad expresiva. El uso de este lenguaje con las mejoras propuestas en esta tesis permite que un intérprete de OCL pueda validar los modelos de información, cuestión que no era posible hasta la fecha.
- Trabajar con lenguajes de ontologías estándar para definir la información de gestión permite aprovechar las herramientas desarrolladas hasta la fecha en este ámbito. La ampliación de DAML+OIL para que pueda expresar facetas típicas de gestión evita además que se pierda información en la traducción, permitiendo a la vez que herramientas que entiendan DAML+OIL estándar puedan trabajar con aquella parte de la información que contiene construcciones no específicas de gestión.
- Los estudios que existían hasta ahora respecto de la integración de información trataban ésta únicamente desde el punto de vista sintáctico, o con alguna posibilidad de correspondencia semántica muy limitada. La aplicación del método M&M permite realizar esta integración desde el nivel del significado contenido en la información, lo que permite que un gestor pueda manejar un único modelo, con total transparencia de los dominios de gestión subyacentes. Esto redunda en una mejora de las aplicaciones de gestión, que pueden relacionar datos que hasta el momento no tenían relación directa al pertenecer a distintos dominios de gestión.
- Los trabajos que hasta la fecha formalizaban el comportamiento de la información de gestión tenían varias limitaciones: o se circunscribían a un modelo de gestión concreto, o bien no integraban este comportamiento dentro de la propia información de gestión. Al aprovechar las características de los lenguajes de ontologías, la definición de restricciones se obtiene por añadidura. Con esto, se puede obtener una especificación de información de un modelo unificado e independiente del dominio de gestión, cuyos conceptos además posean reglas que gobiernen su comportamiento.
- La gestión inteligente que se puede realizar a partir de la arquitectura propuesta, tiene como ventaja sobre otros gestores de este tipo que no está restringida a ningún dominio concreto de gestión integrada, con lo que es aplicable para realizar una gestión paraguas. Al mismo tiempo, se mejora la arquitectura de WBEM, al indicar la forma en que el CIMOM y los proveedores deben comportarse ante una operación de gestión, utilizando el modelo unificado con las reglas de comportamiento y las ontologías de correspondencia, respectivamente.

No obstante, también es necesario señalar algunos aspectos de las contribuciones propuestas en esta tesis doctoral que podrían ser mejorados:

- Aunque la solución propuesta para la fusión y correspondencia de los modelos de información no impone ningún tipo de restricción, debido a la intervención manual para comprobar la validez de las reglas propuestas, puede llevar bastante tiempo su aplicación a modelos de información de gran tamaño. Sin embargo, este tiempo será menor que realizando esta tarea a mano.
- Algunos modelos de información, como SMI, incluyen algunas operaciones dentro de las definiciones (como las columnas de tipo `RowStatus`, por ejemplo). Estas cuestiones no se han tenido en cuenta en lo que respecta a la fusión y correspondencia de modelos.
- Las descripciones incluidas en la información de gestión normalmente incluyen definiciones de restricciones sobre dicha información. Al utilizar calificadores de restricción o restricciones propias del lenguaje de definición no se obtienen automáticamente dichas reglas, por lo que este trabajo debe hacerse a mano.
- Los actuales sistemas de gestión no poseen motores de inferencia que interpreten las reglas de comportamiento que se han añadido al modelo de información, por lo que sería necesario adaptar los actuales para esta funcionalidad. No obstante, el uso de lenguajes estándar permite aprovechar las implementaciones existentes.
- La arquitectura propuesta se podría considerar excesivamente pesada, al tener que poseer un motor de inferencias y distintos proveedores que manejen ontologías de correspondencia. Sin embargo, el beneficio obtenido es también importante, al poder gestionar cualquier recurso independientemente del dominio en que se haya definido su interfaz de gestión, y poderse automatizar las comprobaciones referidas a los parámetros de operación de dicho recurso, al haberse definido restricciones interpretables por dicho gestor.

Todas estas deficiencias se plasman en la siguiente sección en un conjunto de líneas de trabajo futuro complementadas con otras derivadas de la posible extensión de los resultados de esta tesis a otros ámbitos.

7.3 Líneas futuras de investigación

El trabajo desarrollado en esta tesis doctoral constituye el punto de partida para otras líneas de investigación relacionadas con la integración de modelos de gestión. Algunas de ellas están relacionadas con posibles mejoras de las contribuciones propuestas en esta tesis y otras identifican posibles nuevos campos de aplicación de las ideas en ella desarrolladas, o incluso, nuevas tecnologías susceptibles de ser utilizadas como soporte de nuevas vías con las que alcanzar los objetivos planteados en esta tesis.

- Como se vio en la sección anterior, existen algunos aspectos a mejorar de las contribuciones propuestas, relativos a la automatización, dado que es necesaria la intervención humana para validar los resultados de fusión y correspondencia, así como la definición de las reglas de comportamiento. Para mejorar esto, se pueden acometer varias cuestiones.
    - Por un lado, buscar heurísticos adicionales que faciliten en mayor medida el proceso de fusión y correspondencia así como la generación automática de reglas de comportamiento básicas para cada pieza de información. Por ejemplo, si la descripción contiene la cadena "`if`", entonces seguramente exista una condición que haya que hacer cumplir. También se puede incorporar un heurístico que aproveche la información contenida en los calificadores `MappingStrings`, para identificar puntos iniciales de correspondencia directa.
    - Por otro lado, se puede pasar a utilizar otras tecnologías de Inteligencia Artificial, entre las que cabe destacar el Procesado de Lenguaje Natural (NLP, *Natural Language Processing*), de forma que se pueda obtener información mediante un análisis del lenguaje natural contenido en las facetas de descripción. Al mismo tiempo, y en este sentido, también sería de utilidad el uso de ontologías como WordNet y EuroWordNet [Vossen98], que define los términos empleados normalmente en las distintas lenguas europeas.
- La tarea de traducción sintáctica no se ha abordado en detalle en esta tesis, entendiendo que ya existen suficientes estudios sobre este tema. Sin embargo, es conveniente estudiar la manera de traducir las columnas operacionales de SMI (RowStatus, y similares) a un lenguaje con más capacidad expresiva, dado que este tipo de columnas contienen una semántica adicional asociada que va más allá de la mera traducción sintáctica. Dado el acoplamiento que existe en este caso entre el modelo de información y el método de acceso a la información, sería interesante estudiar la forma de adaptar la ontología de correspondencia a este tipo de problemas.
- Asimismo, para acceder a una información concreta, suele ser necesario aportar al agente el identificador concreto del ejemplar solicitado. La ontología de correspondencia definida no tiene en cuenta la existencia de distintos espacios de nombres, por lo que será necesario estudiar con mayor detenimiento la forma de corresponder los espacios de nombres de ejemplares del modelo unificado y los modelos subyacentes.
- Finalmente, aunque esta tesis ha sido validada utilizando un caso de estudio determinado, como continuación del trabajo desarrollado sería necesaria la aplicación del método de fusión y correspondencia M&M de una manera más

amplia a diversos escenarios concretos, abarcando otros modelos contenidos en distintos dominios de gestión integrada. De esta forma, se puede obtener de manera completa un modelo común unificado semánticamente, en el que también se incluyan definiciones formales de comportamiento, con el que un gestor integrado pueda razonar y acceder unificadamente a los recursos gestionados.

Apéndice A Definición de las reglas del metamodelo de CIM en OCL

A.1 Introducción

A continuación se presentan las reglas presentadas en el diagrama de la Figura 5.2, que formalizan el metamodelo de CIM empleando OCL.

Para ello se ha tomado el conjunto de reglas definidas en [DMTF99] y se ha expresado de manera formal en OCL. También se identifican los casos en que se han encontrado incongruencias o redundancias.

A.2 Definición de las reglas

A continuación se incluyen en *cursiva* las reglas de la especificación del metamodelo de CIM, junto con los comentarios, en letra normal, y el código OCL, dentro de cuadros en letra `Courier`. Las reglas están en su versión original en inglés para evitar perder semántica en la traducción. El diagrama de clases al que se hace referencia es el de la Figura 2.14.

Todas las reglas en OCL están referidas al metaesquema de CIM, por lo que se comienza con esta cláusula:

```
package CIM_MetaModel
```

1. *Every meta construct is expressed as a descendent of a Named Element.*

   Dado que todo lo que esté expresado en el diagrama de clases es consistente, esta regla es redundante. Dicho diagrama ya contempla que todos los elementos descienden de `NamedElement`. Sin embargo, si se quisiera expresar en OCL se podría hacer de la siguiente manera:

```
context Qualifier
   inv: self.oclIsKindOf(NamedElement)
context Property
   inv: self.oclIsKindOf(NamedElement)
context Reference
   inv: self.oclIsKindOf(NamedElement)
context Class
   inv: self.oclIsKindOf(NamedElement)
context Association
```

```
   inv: self.oclIsKindOf(NamedElement)
context Indication
   inv: self.oclIsKindOf(NamedElement)
context Method
   inv: self.oclIsKindOf(NamedElement)
context Trigger
   inv: self.oclIsKindOf(NamedElement)
context Schema
   inv: self.oclIsKindOf(NamedElement)
context QualifierType
   inv: self.oclIsKindOf(NamedElement)
```

2. *A Named Element has zero or more Characteristics. A Characteristic is a Qualifier that characterizes a Named Element.*

   Esta regla también es redundante al estar contenida en el diagrama, existiendo la multiplicidad `0..*` en la relación de agregación entre las clases `NamedElement` y `Qualifier`. En cualquier caso, también se puede indicar en OCL de la siguiente manera:

```
context NamedElement
   inv: self.Qualifier->size()>=0
   inv: self.Qualifier->forAll( q |
              q.oclIsKindOf(Qualifier))
```

3. *A Named Element can trigger zero or more Indications.*

   Al igual que en los casos anteriores, la multiplicidad ya está incluida en la relación entre `NamedElement` y `Trigger`. Aquí, además, ocurre el caso de que la regla parece referirse a `Indication`, pero no existe la relación entre `NamedElement` e `Indication`, por lo que la regla que se incluye se refiere a `Trigger`.

```
context NamedElement
   inv: self.Trigger->size()>=0
```

4. *A Schema is a Named Element and can contain zero or more classes. A Class must belong to only one schema.*

   Esta regla no es congruente con el diagrama de clases, puesto que la relación de agregación no es entre `Schema` y `Class`, sino entre `Schema` y `NamedElement`. La regla que se incluye a continuación está basada en el diagrama.

```
context Schema
   inv: self.oclIsKindOf(NamedElement)
   inv:  self.NamedElement->size()>=0
context Class
   inv: self.Schema->size()=1
```

5. *A Qualifier Type (not shown in* Figura 2.14*) is a Named Element and must be used to supply a type for a Qualifier (that is, a Qualifier must have a Qualifier Type). A Qualifier Type can be used to type zero or more Qualifiers.*

   Esta regla incluye un elemento que no está en el diagrama de clases, lo cual es una incongruencia, pues debería estar incluido.

```
context QualifierType
   inv: self.oclIsKindOf(NamedElement)
   inv: self.Qualifier->size()>=0
context Qualifier
   inv: self.QualifierType->size()=1
```

6. *A Qualifier is a Named Element and has a Name, a Type (intrinsic data type), a Value of this type, a Scope, a Flavor and a default Value. The type of the Qualifier Value must agree with the type of the Qualifier Type.*

   Al igual que en el caso anterior, se incluyen elementos no presentados en el diagrama, lo que supone una incongruencia.

```
context Qualifier
   inv: self.oclIsKindOf(NamedElement)
   inv:  self.attributes()->=  Set  {  'Name',  'Type',  'Value',
'Scope', 'Flavor', 'DefaultValue' }
   inv:  self.Value.oclIsTypeOf(self.Type)
```

7. *A Property is a Named Element and has only one Domain: the Class that owns the Property.*

   Esta regla también es redundante con el diagrama de clases. Dado que nombra a `Class` como `Domain`, se han incluido estos roles en la asociación `PropertyDomain`.

```
context Property
   inv: self.oclIsKindOf(NamedElement)
   inv:  self.Domain->size()=1
```

8. *A Property can have an Override relationship with another Property from a different class. The Domain of the overridden Property must be a supertype of the Domain of the overriding Property.*

   También esta regla es redundante en lo que se refiere a la definición y multiplicidad de la relación `Override`.

   Al igual que la regla anterior, `Domain` se refiere a `Class`. Aquí se utiliza la asociación `SubtypeSupertype` de `Class`. Según la regla 12, sólo puede haber un supertipo, aunque según lo que aparece aquí, podría haber varios (dice "*a supertype*"). Para evitar confusiones, se nombran los papeles de la asociación `Override` como `Overriding` y `Overridden`.

```
context Property
   inv: self.Overridden->size()>=0
         and self.Overridden->size()<=1
   inv: self.Domain.Supertype
         ->includes(self.Overriden.Domain)
```

9. *The Class referenced by the Range association (Figure 2-4) of an overriding Reference must be the same as, or a subtype of, the Class referenced by the Range associations of the Reference being overridden.*

   Como en los casos anteriores, se ha nombrado a los extremos de la relación `Range` con los nombres `Reference` y `Range`.

```
context Reference
inv: self.Range=self.Overridden.Range
         or self.Overriden.Range.Subtype
               ->includes(self.Range)
```

10. *The Domain of a Reference must be an Association.*

```
context Reference
   inv: self.Domain.oclIsTypeOf(Association)
```

11. *A Class is a type of Named Element. A Class can have instances (not shown on the diagram) and is the Domain for zero or more Properties. A Class is the Domain for zero or more Methods.*

    Vuelve a aparecer un elemento que no existe en el diagrama de clases.

```
context Class
   inv: self.oclIsKindOf(NamedElement)
   inv: self.Instance.size()>=0
   inv: self.Property.size()>=0
   inv: self.Method.size()>=0
```

12. *A Class can have zero or one supertype, and zero or more subtypes.*

    Aquí, para referirse a herencia simple, definen esta regla, que como en casos anteriores también está contenida en el diagrama. Sin embargo, esto no permite decir que una clase abuela es supertipo de su nieta. Además, entra en conflicto con la regla 8, que habla de supertipos en general. Se definen los papeles de la asociación como `Subtype` y `Supertype`.

```
context Class
   inv: self.Supertype.size()=0
         or self.Supertype.size()=1
   inv:  self.Subtype.size()>=0
```

13. *An Association is a type of Class. Associations are classes with an Association qualifier.*

    La primera regla es redundante con el diagrama de clases.

```
context Association
   inv: self.oclIsKindOf(Class)
   inv: self.Qualifier->includes(q |q.Name='Association')
```

14. *An Association must have two or more References.*

    Esta regla también es redundante con el diagrama.

```
context Association
   inv: self.attributes()
         ->select( r | r.oclIsTypeOf(Reference))
         ->size()>=2
```

15. *An Association cannot inherit from a non-association Class.*

```
context Association
   inv: self.Supertype->isEmpty() or
         self.Supertype->forall( st |
               st.oclIsTypeOf(Association))
```

16. *Any subclass of an Association is an association.*

```
context Association
   inv: self.Subtype->forall( st |
                st.oclIsTypeOf(Association))
```

17. *A Method is a Named Element and has only one Domain: the Class that owns the Method.*

    Esta regla es también redundante con el modelo. Al igual que para el caso de `Property`, se ha definido el papel `Domain` para `Class`.

```
context Method
   inv: self.oclIsKindOf(NamedElement)
   inv: self.Class->size()=1
```

18. *A Method can have an Override relationship with another Method from a different Class. The Domain of the overridden Method must be a superclass of the Domain of the overriding Method.*

    La primera regla es redundante con el modelo. Para la segunda, y al igual que para `Property`, se han definido los papeles `Overriding` y `Overridden`.

```
context Method
   inv: self.Overridden->size()>=0
         and self.Overridden->size()<=1
   inv: self.Domain.Supertype->includes(self.Overridden.Domain)
```

19. *A Trigger is an operation that is invoked on any state change, such as object creation, deletion, modification or access, or on property modification or access. Qualifiers, Qualifier Types and Schemas may not have triggers. The changes that invoke a trigger are specified as a Qualifier.*

Para esta regla no se ha definido ninguna restricción. Quizás se podría hacer uso de la cláusula `oclInState`, pero como no están definidos los estados de los Elementos, no se puede usar.

20. *An Indication is a type of Class and has an association with zero or more Named Triggers that can create instances of the Indication.*

    Se hace mención a elemento y una asociación que no está en el diagrama de clases.

```
context Indication
   inv: self.oclIsKindOf(Class)
   inv: self.Trigger.size()>=0
```

21. *Every meta-schema object is a descendent of a Named Element and, as such, has a Name. All names are case-insensitive. The rules applicable to Name vary, depending on the creation type of the object.*

    La primera regla es redundante con el diagrama de clases, pue es implícito a la herencia que todos los elementos tengan la propiedad `Name`. La aplicación de la segunda regla se hace en los subapartados, pues puede haber una propiedad que tenga el mismo nombre que un método, por ejemplo.

    a. *Fully-qualified Class Names (that is, the Class name prefixed by the schema name) are unique within the schema. (See the discussion of schemas later in this section).*

```
context Class
      inv: Class.allInstances()->forall(c1, c2 |
      c1.Schema.Name.concat(':').concat(c1.Name).toUpper()=
      c2.Schema.Name.concat(':').concat(c2.Name).toUpper()
            implies c1=c2 )
```

    b. *Fully-qualified Association and Indication Names are unique within the schema (implied by the fact that Associations and Indications are subtypes of Class).*

       Esta regla está implícita en la anterior, al ser `Association` e `Indication` subclases de `Class`.

```
context Association
      inv: Association.allInstances()->forall(a1, a2 |
      a1.Schema.Name.concat(a1.Name).toUpper()=
      a2.Schema.Name.concat(a2.Name).toUpper()
            implies a1=a2 )
context Indication
      inv: Indication.allInstances()->forall(i1, i2 |
      i1.Schema.Name.concat(i1.Name).toUpper()=
      i2.Schema.Name.concat(i2.Name).toUpper()
            implies i1=i2 )
```

c. *Implicitly-defined Qualifier Names are unique within the scope of the characterized object (that is, a Named Element may not have two Characteristics with the same Name). Explicitly-defined Qualifier Names are unique within the defining Schema. An implicitly-defined Qualifier must agree in type, scope and flavor with any explicitly-defined Qualifier of the same name.*

Esta regla no indica la diferencia entre calificador implícito y explícito. Únicamente queda clara la regla para los calificadores implícitos.

```
context NamedElement
      inv: self.Qualifier->forall(q1, q2 |
      q1.Name.toUpper()=q2.Name.toUpper() implies q1=q2 )
```

Para el segundo caso, puede que estén hablando del QualifierType definido en la regla 5, pero dado que aquí utilizan otra nomenclatura, no queda claro a qué se refiere. Por ejemplo, podría ser:

```
context QualifierType
      inv: QualifierType.allInstances()->forall(qt1, qt2 |
      qt1.Name.toUpper()=qt2.Name.toUpper() implies qt1=qt2 )
```

d. *Trigger names must be unique within the Property, Class or Method to which the Trigger applies.*

```
context Property
      inv: self.Trigger->forall(t1, t2 |
      t1.Name.toUpper()=t2.Name.toUpper() implies t1=t2 )
context Class
      inv: self.Trigger->forall(t1, t2 |
      t1.Name.toUpper()=t2.Name.toUpper() implies t1=t2 )
context Method
      inv: self.Trigger->forall(t1, t2 |
t1.Name.toUpper()=t2.Name.toUpper() implies t1=t2 )
```

e. *Method and Property names must be unique within the Domain Class. A Class can inherit more than one Property or Method with the same name. Property and Method names can be qualified using the name of the declaring Class.*

Aquí hay una incongruencia. Si la herencia es simple, debería ser "*Classes cannot inherit*", puesto que si una `Class` sólo hereda de otra, no puede heredar varias `Property` con el mismo nombre. Todo esto, a no ser que se refiera que puede existir una `Property` cuyo `Name` sea igual al de un `Method`.

```
context Class
      inv: self.Method->forall(m1, m2 |
      m1.Name.toUpper()=m2.Name.toUpper() implies m1=m2 )
```

```
      inv: self.Property->forall(p1, p2 |
      p1.Name.toUpper()=p2.Name.toUpper() implies p1=p2)
```

f. Reference Names must be unique within the scope of their defining Association. Reference Names obey the same rules as Property Names. Note that Reference names are not required to be unique within the scope of the related Class. In such a scope, the Reference provides the name of the Class within the context defined by the Association.

It is legal for the class System to be related to Service by two independent Associations (Dependency and Hosted Services, each with roles System and Service). It would not be legal for Hosted Services to define another Reference Service to the Service class, since a single association would then contain two references called Service.

```
context Association
      inv: self.Reference->forAll(r1, r2 |
      r1.Name.toUpper()=r2.Name.toUpper() implies r1=r2)
```

22. *Qualifiers are Characteristics of Named Elements. A Qualifier has a Name (inherited from Named Element) and a Value. The Value is used to define the characteristics of the Named Element. For example, a Class might have a Qualifier with the Name "Description," the Value of which is the description for the Class. A Property might have a Qualifier with the Name "Units," which has Values such as "Bytes" or "KiloBytes." The Value can be thought of as a variant (that is, a value plus a type).*

Esta regla es redundante y en parte incongruente con la regla 6, al no hablar de los otros atributos que tiene un calificador.

```
context Qualifier
   inv: self.oclIsKindOf(NamedElement)
   inv:  self.attributes()->includesAll(Set {'Name', 'Value'})
```

23. *Association and Indication are types of Class; as such, they can be the Domain for Methods, Properties and References (that is, Associations and Indications can have Properties and Methods in the same way as a Class does). Associations and Indications can have instances. The instance of an Association has a set of references that relate one or more objects. An instance of an Indication represents the occurrence of an event, and is created because of that occurrence—usually a Trigger. Indications are not required to have keys. Typically, Indications are very short-lived objects used to communicate information to an event consumer.*

Esta regla es redundante con todo lo anterior en general. Otras cuestiones no se pueden escribir en OCL.

24. *A Reference has a range that represents the type of the Reference. For example, in the model of PhysicalElements and PhysicalPackages, there are two References: ContainedElement, which has PhysicalElement as its range and Container as its domain, and ContainingElement, which has PhysicalPackage as its range and Container as its domain.*

    Esta regla también es redundante. El rango de una referencia será siempre una clase, pues está definido así en el modelo.

25. *A Class has a subtype-supertype association that represents substitutability relationships between the Named Elements involved in the relationship. The association implies that any instance of a subtype can be substituted for any instance of the supertype in an expression, without invalidating the expression.*

    Revisiting the Container example: Card is a Subtype of PhysicalPackage. Therefore, Card can be used as a value for the Reference ContainingElement (that is, an instance of Card can be used as a substitute for an instance of PhysicalPackage).

    A similar relationship can exist between Properties. For example, given that PhysicalPackage has a Name property (which is a simple alphanumeric string), Card Overrides Name to a name of alpha-only characters.

    The same idea applies to Methods. A Method that overrides another Method must support the same signature as the original Method and, most importantly, must be substitutable for the original Method in all cases.

    Igual que en los casos anteriores, añaden redundancia a lo definido en el diagrama. Otras cuestiones no se pueden definir en OCL.

26. *The Override relationship is used to indicate the substitution relationship between a property or method of a subclass and the overridden property or method inherited from the superclass. This is the opposite of the C++ convention in which the superclass property or method is specified as virtual, with overriding occurring thereafter as a side effect of declaring a feature with the same signature as the inherited virtual feature.*

    Exactamente igual que en los casos anteriores.

27. *The number of references in an Association class defines the arity of the Association. An Association containing two references is a binary Association. An Association containing three references is a ternary association. Unary Associations (Associations containing one reference) are not meaningful. Arrays of references are not allowed. When an association is sub-classed, its arity cannot change.*

    Exactamente igual que anteriormente.

28. *Schemas provide a mechanism that allows ownership of portions of the overall model by individuals and organizations who are responsible for managing the evolution of the schema. In any given installation, all classes are mutually visible, regardless of schema ownership. Schemas have a universally unique name. The schema name is considered part of the class name. The full class name (that is, class name plus owning schema name) is unique within the namespace and is referred to as the fully-qualified name (see Section 2.4).*

    Exactamente igual que en los casos anteriores.

Las reglas acaban con la cláusula

```
endpackage
```

Apéndice B Definición de la ontología de correspondencia en DAML+OIL

B.1 Introducción

Este apéndice incluye la definición en DAML+OIL de una ontología simple para corresponder modelos de gestión. Cada posible elemento de cada modelo (clases, atributos o propiedades y relaciones) posee una fórmula de traducción. A la vez, cada elemento posee una referencia a su definición (como un OID, por ejemplo), y cada fórmula posee el conjunto de elementos origen y destinto que toman parte en la misma, y la expresión que se usa para traducir de los elementos origen a los elementos destino. También se incluyen asociaciones a los elementos correspondidos, así como las fórmulas inversas.

Un gestor que se base en una ontología de gestión global y esta ontología de correspondencia funcionaría de la siguiente manera: si se necesita obtener todos los ejemplares de cierto elemento de la ontología global, buscaría dicho elemento en la ontología de correspondencia, encontrando también la fórmula y los elementos correspondidos relativos a los modelos fusionados. La expresión contenida en la fórmula se podría aplicar para traducir los elementos de los modelos fusionados para que encajaran en la ontología global, obteniendo los ejemplares deseados.

B.2 Definición en DAML+OIL

A continuación se incluye la definición en DAML+OIL de la ontología de correspondencia, siguiendo la estructura mostrada en la Figura 5.13.

```
<rdf:RDF
  xmlns:xsd ="http://www.w3.org/2000/10/XMLSchema#"
  xmlns:map ="map#"
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:daml ="http://www.daml.org/2001/03/daml+oil#"
  xmlns:rdfs ="http://www.w3.org/2000/01/rdf-schema#"
  xmlns ="map#"
  >

<daml:Ontology rdf:ID="MappingOntology">
  <daml:versionInfo>1.0</daml:versionInfo>
```

```
  <rdfs:comment>An ontology to define mappings between elements</rdfs:comment>
</daml:Ontology>

<daml:Class rdf:ID="Element">
  <rdfs:comment>It represents the elements to be mapped</rdfs:comment>
</daml:Class>

<daml:DatatypeProperty rdf:ID="type">
  <rdfs:comment>The type of element</rdfs:comment>
  <daml:domain rdf:resource="#Element"/>
  <daml:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="reference">
  <rdfs:comment>The identifier of the element in its source domain</rdfs:comment>
  <daml:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
  <daml:domain rdf:resource="#Element"/>
</daml:DatatypeProperty>

<daml:ObjectProperty rdf:ID="formula">
  <rdfs:comment>The formula to obtain the mapped elements of this
element</rdfs:comment>
  <daml:domain rdf:resource="#Element"/>
  <daml:range rdf:resource="#Formula"/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID="mappedElements">
  <rdfs:comment>The mapped elements of this element</rdfs:comment>
  <daml:domain rdf:resource="#Element"/>
  <daml:range rdf:resource="#Element"/>
</daml:ObjectProperty>

<daml:Class rdf:ID="Formula">
  <rdfs:comment>A formula to generate the mapped elements of another
element</rdfs:comment>
</daml:Class>

<daml:DatatypeProperty rdf:ID="language">
  <rdfs:comment>The language used to specify the expression</rdfs:comment>
  <daml:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
  <daml:domain rdf:resource="#Formula"/>
</daml:DatatypeProperty>
```

```
<daml:ObjectProperty rdf:ID="expression">
  <rdfs:comment>An expression to map a source element in a target element</rdfs:comment>
  <daml:domain rdf:resource="#Formula"/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID="sourceElements">
  <rdfs:comment>The mapping elements</rdfs:comment>
  <daml:range rdf:resource="#Element"/>
  <daml:domain rdf:resource="#Formula"/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID="targetElements">
  <rdfs:comment>The mapped elements</rdfs:comment>
  <daml:range rdf:resource="#Element"/>
  <daml:domain rdf:resource="#Formula"/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID="inverseFormula">
  <rdfs:comment>A formula that it is the inverse of this one</rdfs:comment>
  <daml:range rdf:resource="#Formula"/>
  <daml:domain rdf:resource="#Formula"/>
</daml:ObjectProperty>

</rdf:RDF>
```

Apéndice C Caso de estudio

C.1 Introducción

Este apéndice complementa el caso de estudio que sirve como ejemplo para aplicar el método explicado en la sección 5.4. De esta manera, este apéndice completa aspectos relativos a la fusión y correspondencia de la HOST-RESOURCES-MIB [Waldbusser00a], escrita en SMI y representada en la Figura 5.9, con un subconjunto de los esquemas CIM [Bumpus00], mostrado a su vez en la Figura 5.10. La HOST-RESOURCES-MIB se nombra en varios calificadores `MappingStrings` de los esquemas CIM, que han ayudado en parte a realizar este proceso. También ha sido de utilidad la traducción a UML basada en los algoritmos de LIBSMI [Schönwälder01].

A continuación se incluyen sendos ejemplos de fusión y de correspondencia entre clases de ambos modelos de información, siguiendo el método M&M.

C.2 Ejemplo de fusión

Según el método M&M, el sistema que asista en el proceso de fusión y correspondencia debe ser capaz de buscar y proponer clases y propiedades candidatas a ser fusionadas. Para ello, se aplican los heurísticos estudiados en el subapartado 5.4.2.1. Con ellos se obtendrán los resultados que se indican a continuación.

Por similitud de nombres de identificadores, se corresponden los siguientes grupos de objetos o entradas de tabla de la HOST-RESOURCES-MIB con clases de CIM:

- `hrSystem` con `CIM_System` con `CIM_ComputerSystem`, por compartir la subcadena `System`.
- `hrDeviceEntry` con `CIM_Device`, por compartir la subcadena `Device`.
- `hrProcessorEntry` con `CIM_Processor`, por compartir la subcadena `Processor`.
- `hrDiskStorageEntry` con `CIM_StorageExtent`, también en el caso de `hrStorage` y `hrStorageEntry`, por compartir la subcadena `Storage`.
- `hrPartitionEntry` con `CIM_MediaPartition` y `CIM_DiskPartition`, por compartir la subcadena `Partition`.

- `hrNetworkEntry` con `CIM_NetworkAdapter`, por compartir la subcadena `Network`.
- `hrPrinterEntry` con `CIM_Printer`, por compartir la subcadena `Printer`.
- No hay similitud de nombres para `hrSWRunEntry`, `hrSWRunPerfEntry`, `hrSWInstalledEntry` y `hrFSEntry`.

Por similitud de jerarquía de herencia se corresponden las siguientes clases:

- Las hijas de `hrDeviceEntry` con las hijas de `CIM_Device`.
    - `hrProcessorEntry` con `CIM_Processor`.
    - `hrDiskStorageEntry` con `CIM_StorageExtent`.
    - `hrNetworkEntry` con `CIM_NetworkAdapter`.
    - `hrPrinterEntry` con `CIM_Printer`.

Por dominio de propiedades (I) y similitud de nombres de identificadores de las propiedades se corresponden:

- Las propiedades de `hrSystem` con las de `CIM_System` y `CIM_ComputerSystem`.
    - No se encuentran propiedades con nombres similares.
- Las propiedades de `hrDeviceEntry` con las de `CIM_Device`.
    - `hrDeviceID` con `DeviceID`.
    - `hrDeviceStatus` con `StatusInfo`.
    - `hrDeviceErrors` con `LastErrorCode`, `ErrorDescription` y `ErrorCleared`.
- Las propiedades de `hrProcessorEntry` con `CIM_Processor`.
    - `hrProcessorFwdID` con `UniqueID`.
    - `hrProcessorLoad` con `LoadPercentage`.
- Las propiedades de `hrDiskStorageEntry`, `hrStorage` y `hrStorageEntry` con `CIM_StorageExtent`.
    - `hrDiskStorageAccess` con `Access`.
- Las propiedades de `hrPartitionEntry` con `CIM_MediaPartition` y `CIM_DiskPartition`.
    - No se encuentran propiedades con nombres similares.
- Las propiedades de `hrNetworkEntry` con `CIM_NetworkAdapter`.
    - No se encuentran propiedades con nombres similares.
- Las propiedades de `hrPrinterEntry` con `CIM_Printer`.

- `hrPrinterStatus` con `PrinterStatus`.
- `hrPrinterDetectedErrorState` con `DetectedErrorState`.

El heurístico que identifica clases candidatas por dominio de propiedades (II) no es tan bueno como el resto, pues existen muchas propiedades con nombre similar en clases diferentes que pueden dar muchos falsos candidatos. Por este motivo, no se va a emplear en este caso, aunque podría ser útil en otras ocasiones.

A partir de estas clases y propiedades candidatas, la persona que realice las operaciones puede completar el proceso de fusión, decidiendo si los candidatos son válidos o no. Dicha persona también puede realizar otras operaciones sobre elementos que no han sido propuestos aunque no cumplan ninguno de los heurísticos. Por ejemplo, en la sección siguiente se ve como se realiza la correspondencia entre las clases `hrSWRunEntry` y `CIM_Process`, que se pueden fusionar aunque no se hayan identificado anteriormente como candidatas.

C.3 Ejemplo de correspondencia

Como ejemplo de correspondencia se ha tomado, por un lado, la clase `CIM_Process`, y por otro lado, las entradas de tabla `hrSWRunEntry` y `hrSWRunPerfEntry`. A continuación se describen esta clase y estas tablas, para facilitar el proceso posterior de correspondencia.

La descripción de `CIM_Process`, contenida en el calificador `Description` de su especificación es el siguiente: "*Each instance of the CIM_Process class represents a single instance of a running program. A user of the OperatingSystem will typically see a Process as an application or task. Within an OperatingSystem, a Process is defined by a workspace of memory resources and environmental settings that are allocated to it. On a multitasking System, this workspace prevents intrusion of resources by other Processes. Additionally, a Process can execute as multiple Threads, all which run within the same workspace.*"

La clase `CIM_Process` posee propiedades (sin incluir las heredadas) mostradas en la Tabla C.1.

Tabla C.1. Definición de la clase `CIM_Process`.

Nombre de la propiedad	Documentación original (en inglés)	Tipo
`CSCreationClassName`	*The scoping ComputerSystem's CreationClassName.*	`string`
`CSName`	*The scoping ComputerSystem's Name.*	`string`
`OSCreationClassName`	*The scoping OperatingSystem's CreationClassName.*	`string`
`OSName`	*The scoping OperatingSystem's Name.*	`string`
`Name`	*The name of the process.*	`string`
`CreationClassName`	*CreationClassName indicates the name of*	`string`

	the class or the subclass used in the creation of an instance. When used with the other key properties of this class, this property allows all instances of this class and its subclasses to be uniquely identified.	
`Handle`	*A string used to identify the Process. A Process ID is a kind of Process Handle.*	`string`
`Priority`	*Priority indicates the urgency or importance of execution of a Process. If a priority is not defined for a Process, a value of 0 should be used.*	`uint32`
`ExecutionState`	*Indicates the current operating condition of the Process. Values include ready (2), running (3), and blocked (4), among others.{Unknown(0), Other(1), Ready(2), Running(3), Blocked(4), Suspended Blocked(5), Suspended Ready(6), Terminated(7), Stopped(8), Growing(9)}*	`uint16`
`OtherExecutionDescription`	*A string describing the state - used when the instance's ExecutionState property is set to 1 ("Other"). OtherExecutionDescription should be set to NULL when the ExecutionState property is any value other than 1.*	`string`
`CreationDate`	*Time that the Process began executing.*	`datetime`
`TerminationDate`	*Time that the Process was stopped or terminated.*	`datetime`
`KernelModeTime`	*Time in kernel mode, in milliseconds. If this information is not available, a value of 0 should be used.*	`uint64`
`UserModeTime`	*Time in user mode, in milliseconds. If this information is not available, a value of 0 should be used.*	`uint64`
`WorkingSetSize`	*The amount of memory in bytes that a Process needs to execute efficiently, for an OperatingSystem that uses page-based memory management. If an insufficient amount of memory is available (< working set size), thrashing will occur. If this information is not known, NULL or 0*	`uint64`

	should be entered. If this data is provided, it could be monitored to understand a Process' changing memory requirements as execution proceeds.	

Al mismo tiempo, la descripción de entrada de tabla `hrSWRunEntry` contenida en la cláusula `DESCRIPTION` es la siguiente: "*A (conceptual) entry for one piece of software running on the host Note that because the installed software table only contains information for software stored locally on this host, not every piece of running software will be found in the installed software table. This is true of software that was loaded and run from a non-local source, such as a network-mounted file system. As an example of how objects in this table are named, an instance of the hrSWRunName object might be named hrSWRunName.1287*". Por su parte, la descripción de `hrSWRunPerfEntry` es: "*A (conceptual) entry containing software performance metrics. As an example, an instance of the hrSWRunPerfCPU object might be named hrSWRunPerfCPU.1287.*"A partir de estas entradas de tabla se han definido sendas clases con las propiedades contenidas en la Tabla C.2 y Tabla C.3 respectivamente.

Tabla C.2. Definición de la clase `hrSWRunEntry`.

Nombre de la propiedad	Documentación original (en inglés)	Tipo
`hrSWRunIndex`	*A unique value for each piece of software running on the host. Wherever possible, this should be the system's native, unique identification number.*	`Integer`
`hrSWRunName`	*A textual description of this running piece of software, including the manufacturer, revision, and the name by which it is commonly known. If this software was installed locally, this should be the same string as used in the corresponding hrSWInstalledName.*	`DisplayString`
`hrSWRunID`	*The product ID of this running piece of software.*	`ProductID (OBJECT IDENTIFIER)`
`hrSWRunPath`	*A description of the location on long-term storage (e.g. a disk drive) from which this software was loaded.*	`DisplayString`
`hrSWRunParameters`	*A description of the parameters supplied to this software when it was initially loaded.*	`DisplayString`
`hrSWRunType`	*The type of this software.{ unknown(1), operatingSystem(2),*	`Integer`

	deviceDriver(3), application(4)}	
hrSWRunStatus	*The status of this running piece of software. Setting this value to invalid(4) shall cause this software to stop running and to be unloaded.{ running(1), runnable(2) (waiting for resource (CPU, memory, IO)), notRunnable(3) (loaded but waiting for event), invalid(4) (not loaded)}*	Integer

Tabla C.3. Definición de la clase `hrSWRunPerfEntry`.

Nombre de la propiedad	Documentación original (en inglés)	Tipo
hrSWRunPerfCPU	*The number of centi-seconds of the total system's CPU resources consumed by this process. Note that on a multi-processor system, this value may increment by more than one centi-second in one centi-second of real (wall clock) time.*	Integer
hrSWRunPerfMem	*The total amount of real system memory allocated to this process.*	KBytes (Integer)

Uno de los primeros pasos del método M&M es la creación de los elementos en la ontología de correspondencia definida en el Apéndice B. Los elementos que se generarían automáticamente son los siguientes:

```
<Element rdf:ID="CIM_Process"
      rdfs:label="CIM_Process"
      type="class"
      reference="DMTF|CIM|CIM_Process" />

<Element rdf:ID="CSCreationClassName"
      rdfs:label="CIM_Process.CSCreationClassName"
      type="property"
      reference="DMTF|CIM|CIM_Process.CSCreationClassName" />
<Element rdf:ID="CSName"
      rdfs:label="CIM_Process.CSName"
      type="property"
      reference="DMTF|CIM|CIM_Process.CSName" />
<Element rdf:ID="OSCreationClassName"
      rdfs:label="CIM_Process.OSCreationClassName"
      type="property"
      reference="DMTF|CIM_Process.OSCreationClassName" />
<Element rdf:ID="OSName"
```

```
      rdfs:label="CIM_Process.OSName"
      type="property"
      reference="DMTF|CIM_Process.OSName" />
<Element rdf:ID="Name"
      rdfs:label="CIM_Process.Name"
      type="property"
      reference="DMTF|CIM_Process.Name" />
<Element rdf:ID="CreationClassName"
      rdfs:label="CIM_Process.CreationClassName"
      type="property"
      reference="DMTF|CIM_Process.CreationClassName" />
<Element rdf:ID="Handle"
      rdfs:label="CIM_Process.Handle"
      type="property"
      reference="DMTF|CIM_Process.Handle" />
<Element rdf:ID="Priority">
      rdfs:label="CIM_Process.Priority"
      type="property"
      reference="DMTF|CIM_Process.Priority" />
<Element rdf:ID="ExecutionState"
      rdfs:label="CIM_Process.ExecutionState"
      type="property"
      reference="DMTF|CIM_Process.ExecutionState" />
<Element rdf:ID="OtherExecutionDescription"
      rdfs:label="CIM_Process.OtherExecutionDescription"
      type="property"
      reference="DMTF|CIM_Process.OtherExecutionDescription" />
<Element rdf:ID="CreationDate"
      rdfs:label="CIM_Process.CreationDate"
      type="property"
      reference="DMTF|CIM_Process.CreationDate" />
<Element rdf:ID="TerminationDate"
      rdfs:label="CIM_Process.TerminationDate"
      type="property"
      reference="DMTF|CIM_Process.TerminationDate" />
<Element rdf:ID="KernelModeTime"
      rdfs:label="CIM_Process.KernelModeTime"
      type="property"
      reference="DMTF|CIM_Process.KernelModeTime" />
<Element rdf:ID="UserModeTime"
```

```
      rdfs:label="CIM_Process.UserModeTime"
      type="property"
      reference="DMTF|CIM_Process.UserModeTime" />
<Element rdf:ID="WorkingSetSize"
      rdfs:label="CIM_Process.WorkingSetSize"
      type="property"
      reference="DMTF|CIM_Process.WorkingSetSize" />

<Element rdf:ID="hrSWRunEntry"
      rdfs:label="hrSWRunEntry"
      type="class"
      reference="IETF|HOST-RESOURCES-MIB|hrSWRunEntry" />

<Element rdf:ID="hrSWRunIndex"
      rdfs:label="hrSWRunEntry.hrSWRunIndex"
      type="property"
      reference="IETF|HOST-RESOURCES-MIB|hrSWRunEntry.hrSWRunIndex" />
<Element rdf:ID="hrSWRunName"
      rdfs:label="hrSWRunEntry.hrSWRunName"
      type="property"
      reference=
            "IETF|HOST-RESOURCES-MIB|hrSWRunEntry.hrSWRunName" />
<Element rdf:ID="hrSWRunID"
      rdfs:label="hrSWRunEntry.hrSWRunID"
      type="property"
      reference="IETF|HOST-RESOURCES-MIB|hrSWRunEntry.hrSWRunID" />
<Element rdf:ID="hrSWRunPath"
      rdfs:label="hrSWRunEntry.hrSWRunPath"
      type="property"
      reference="IETF|HOST-RESOURCES-MIB|hrSWRunEntry.hrSWRunPath" />
<Element rdf:ID="hrSWRunParameters"
      rdfs:label="hrSWRunEntry.hrSWRunParameters"
      type="property"
      reference="
            IETF|HOST-RESOURCES-MIB|hrSWRunEntry.hrSWRunParameters" />
<Element rdf:ID="hrSWRunType"
      rdfs:label="hrSWRunEntry.hrSWRunType"
      type="property"
      reference="IETF|HOST-RESOURCES-MIB|hrSWRunEntry.hrSWRunType" />
<Element rdf:ID="hrSWRunStatus"
      rdfs:label="hrSWRunEntry.hrSWRunStatus"
```

```
      type="property"
      reference=
            "IETF|HOST-RESOURCES-MIB|hrSWRunEntry.hrSWRunStatus" />

<Element rdf:ID="hrSWRunPerfEntry"
      rdfs:label="hrSWRunPerfEntry"
      type="class"
      reference="IETF|HOST-RESOURCES-MIB|hrSWRunPerfEntry" />

<Element rdf:ID="hrSWRunPerfCPU"
      rdfs:label="hrSWRunPerfEntry.hrSWRunPerfCPU"
      type="property"
      reference=
            "IETF|HOST-RESOURCES-MIB|hrSWRunPerfEntry.hrSWRunPerfCPU"
/>
<Element rdf:ID="hrSWRunPerfMem"
      rdfs:label="hrSWRunPerfEntry.hrSWRunPerfMem"
      type="property"
      reference=
            "IETF|HOST-RESOURCES-MIB|hrSWRunPerfEntry.hrSWRunPerfMem"
/>
```

Al establecerse que ambas clases se corresponden, el método M&M indica la creación de fórmulas de correspondencia, definiéndose sendas fórmulas en ambos sentidos, aunque no se añada ninguna expresión, que deberá ser incluida por el usuario si es necesario.

```
<Formula rdf:id="CIM_Process->hrSWRunEntry,hrSWRunPerfEntry" />
      <sourceElements rdf:resource="#CIM_Process" />
      <targetElements rdf:resource="#hrSWRunEntry" />
      <targetElements rdf:resource="#hrSWRunPerfEntry" />
      <inverseFormula
            rdf:resource="#hrSWRunEntry,hrSWRunPerfEntry->CIM_Process" />
</Formula>
<Formula rdf:id="hrSWRunEntry,hrSWRunPerfEntry->CIM_Process">
      <sourceElements rdf:resource="#hrSWRunEntry" />
      <sourceElements rdf:resource="#hrSWRunPerfEntry" />
      <targetElements rdf:resource="#CIM_Process" />
      <inverseFormula
            rdf:resource="#CIM_Process->hrSWRunEntry,hrSWRunPerfEntry" />
</Formula>
```

A la vez, se completará automáticamente la definición de los elementos incluidos anteriormente en la ontología referenciando a las fórmulas creadas.

```
<Element rdf:about="#CIM_Process">
      <formula rdf:resource="#CIM_Process->hrSWRunPerfEntry" />
      <mappedElement rdf:resource="#hrSWRunEntry" />
      <mappedElement rdf:resource="#hrSWRunPerfEntry" />
</Element>
<Element rdf:about="#hrSWRunEntry">
      <formula rdf:resource=
            "#hrSWRunEntry,hrSWRunPerfEntry->CIM_Process" />
      <mappedElement rdf:resource="#CIM_Process" />
</Element>
<Element rdf:about="#hrSWRunPerfEntry">
      <formula rdf:resource=
            "#hrSWRunEntry,hrSWRunPerfEntry->CIM_Process" />
      <mappedElement rdf:resource="#CIM_Process" />
</Element>
```

Siguiendo el método M&M se procede de igual manera para las distintas propiedades de ambas clases. Para escribir las expresiones se ha empleado un pseudo-lenguaje llamado MapTrans, cuya funcionalidad se aclara en la sección siguiente. Por ejemplo, en un primer paso el usuario establece que existe correspondencia directa entre `Name` y `hrSWRunID`. En este caso, se ha supuesto una correspondencia directa entre ambas propiedades, que será la que se proponga por defecto.

```
<Formula rdf:id="Name->hrSWRunID">
      <language>MapTrans</language>
      <sourceElements rdf:resource="#Name" />
      <targetElements rdf:resource="#hrSWRunID" />
      <inverseFormula rdf:resource="#hrSWRunID->Name" />
      <expression>
            #hrSWRunID=#Name;
      </expression>
</Formula>
<Formula rdf:id="hrSWRunID->Name">
      <language>MapTrans</language>
      <sourceElements rdf:resource="#hrSWRunID" />
      <targetElements rdf:resource="#Name" />
      <inverseFormula rdf:resource="#Name->hrSWRunID" />
      <expression>
            #Name=#hrSWRunID;
      </expression>
</Formula>
```

Al igual que antes, y como ocurrirá para el resto de elementos que intervengan en una fórmula, se incluyen referencias a la fórmula y los elementos que se corresponden de manera automática.

```
<Element rdf:about="#Name">
      <formula rdf:resource="#Name->hrSWRunID" />
      <mappedElement rdf:resource="#hrSWRunID" />
</Element>
<Element rdf:about="#hrSWRunID">
      <formula rdf:resource="#Name" />
      <mappedElement rdf:resource="#hrSWRunID->Name" />
</Element>
```

El usuario establece a continuación la correspondencia entre `Handle` y `hrSWRunIndex`, y el sistema identifica que hay que hacer conversión de tipo de datos, puesto que la primera propiedad es una cadena de caracteres, y la segunda, un número entero.

```
<Formula rdf:id="Handle->hrSWRunIndex">
      <language>MapTrans</language>
      <sourceElements rdf:resource="#Handle" />
      <targetElements rdf:resource="#hrSWRunIndex" />
      <inverseFormula rdf:resource="#hrSWRunIndex->Handle" />
      <expression>
            hrSWRunIndex =(Integer)Handle;
      </expression>
</Formula>
<Formula rdf:id="hrSWRunIndex->Handle">
      <language>MapTrans</language>
      <sourceElements rdf:resource="#hrSWRunIndex" />
      <targetElements rdf:resource="#Handle" />
      <inverseFormula rdf:resource="#Handle->hrSWRunIndex" />
      <expression>
            Handle = (String)hrSWRunIndex;
      </expression>
</Formula>
<Element rdf:about="#Handle">
      <formula rdf:resource="#Handle->hrSWRunIndex" />
      <mappedElement rdf:resource="#hrSWRunIndex" />
</Element>
<Element rdf:about="#hrSWRunIndex">
      <formula rdf:resource="#hrSWRunIndex->Handle" />
      <mappedElement rdf:resource="#Handle" />
</Element>
```

A continuación se incluyen dos correspondencias más complejas, pues intervienen más de dos propiedades. Para estos casos, el usuario únicamente deberá definir la expresión de conversión. En el primer caso se resuelve con una operación aritmética que suma dos propiedades junto con otra que adapta las unidades de medida:

```
<Formula rdf:id="KernelModeTime,UserModeTime->hrSWRunPerfCPU">
      <language>MapTrans</language>
      <sourceElements rdf:resource="#KernelModeTime" />
      <sourceElements rdf:resource="#UserModeTime" />
      <targetElements rdf:resource="#hrSWRunPerfCPU" />
      <inverseFormula
            rdf:resource="#hrSWRunPerfCPU->KernelModeTime,UserModeTime"
/>
      <expression>
<!-- expresión establecida por el usuario -->
            hrSWRunPerfCPU = (KernelModeTime +UserModeTime)*10;
      </expression>
</Formula>
<Formula rdf:id="hrSWRunPerfCPU->KernelModeTime,UserModeTime">
      <language>MapTrans</language>
      <sourceElements rdf:resource="#hrSWRunPerfCPU" />
      <targetElements rdf:resource="#KernelModeTime" />
      <targetElements rdf:resource="#UserModeTime" />
      <inverseFormula
            rdf:resource="#KernelModeTime,UserModeTime->hrSWRunPerfCPU"
/>
      <expression>
<!-- expresión establecida por el usuario -->
            KernelModeTime =0;
            UserModeTime =hrSWRunPerfCPU /10;
      </expression>
</Formula>
<Element rdf:about="#KernelModeTime">
      <formula rdf:resource=
            "#KernelModeTime,UserModeTime->hrSWRunPerfCPU" />
      <mappedElements rdf:resource="#hrSWRunPerfCPU" />
</Element>
<Element rdf:about="#UserModeTime">
      <formula rdf:resource=
            "#KernelModeTime,UserModeTime->hrSWRunPerfCPU" />
      <mappedElements rdf:resource="#hrSWRunPerfCPU" />
</Element>
<Element rdf:about="#hrSWRunPerfCPU">
      <formula rdf:resource=
```

```
            "#hrSWRunPerfCPU->KernelModeTime,UserModeTime" />
      <mappedElements rdf:resource="#KernelModeTime" />
      <mappedElements rdf:resource="#UserModeTime" />
</Element>
```

En este segundo caso, es necesario indicar distintos casos de correspondencia de valores.

```
<Formula rdf:id="ExecutionState,OtherExecutionDescription->hrSWRunStatus">
      <language>MapTrans</language>
      <sourceElements rdf:resource="#ExecutionState" />
      <sourceElements rdf:resource="#OtherExecutionDescription" />
      <targetElements rdf:resource="#hrSWRunStatus" />
      <inverseFormula rdf:resource=
            "#hrSWRunStatus->ExecutionState,OtherExecutionDescription" />
      <expression>
<!-- expresión establecida por el usuario -->
            switch (ExecutionState) {
                  case 3: hrSWRunStatus =2; break;
                  case 4: hrSWRunStatus =1; break;
                  case 5: hrSWRunStatus =3; break;
                  case 8: hrSWRunStatus =4; break;
                  default: hrSWRunStatus =0;
            }
      </expression>
</Formula>
<Formula rdf:id="hrSWRunStatus->ExecutionState,OtherExecutionDescription">
      <language>MapTrans</language>
      <sourceElements rdf:resource="#hrSWRunStatus" />
      <targetElements rdf:resource="#ExecutionState" />
      <targetElements rdf:resource="#OtherExecutionDescription" />
      <inverseFormula rdf:resource=
            "#ExecutionState,OtherExecutionDescription->hrSWRunStatus" />
      <expression>
<!-- expresión establecida por el usuario -->
            switch (hrSWRunStatus.value) {
                  case 1: ExecutionState =4; break;
                  case 2: ExecutionState =3; break;
                  case 3: ExecutionState =5; break;
                  case 4: ExecutionState =8; break;
                  default: ExecutionState =1;
                        OtherExecutionDescription ="The state was" +
                              (String)hrSWRunStatus;
            }
```

```
      </expression>
</Formula>
<Element rdf:about="#ExecutionState">
      <formula rdf:resource=
            "#ExecutionState,OtherExecutionDescription->hrSWRunStatus" />
      <mappedElement rdf:resource="#hrSWRunStatus" />
</Element>
<Element rdf:about="#OtherExecutionDescription">
      <formula rdf:resource=
            "#ExecutionState,OtherExecutionDescription->hrSWRunStatus" />
      <mappedElement rdf:resource="#hrSWRunStatus" />
</Element>
<Element rdf:about="#hrSWRunStatus">
      <formula rdf:resource=
            "#hrSWRunStatus->ExecutionState,OtherExecutionDescription" />
      <mappedElement rdf:resource="#ExecutionState" />
      <mappedElement rdf:resource="#OtherExecutionDescription" />
</Element>
```

C.4 Lenguaje MapTrans

El lenguaje a emplear en las fórmulas se deja abierto, pero para definir las expresiones de traducción del apartado anterior se ha empleado un pseudo-lenguaje ficticio llamado MapTrans (*Mapping Translation*, Traducción de Correspondencia). Dicho lenguaje es muy simple, con una sintaxis similar a JavaScript, teniéndose que cada variable que se emplee es uno de los recursos contenidos en las marcas `sourceElements` y `targetElements`. Además, existen funciones algebraicas para sumar, restar, multiplicar y dividir, así como de traducción entre diferentes tipos de datos. Por último, también será conveniente la existencia de funciones que permitan traducir entre casos concretos, lo que en el ejemplo anterior se ha solucionado con una cláusula `switch`.

Referencias

A

[Asensio99] Juan I. Asensio, Víctor A. Villagrá, Jorge E. López de Vergara, Julio J. Berrocal, ***Experiences with SNMP-based integrated management of a CORBA-based electronic commerce application***, en *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management (IM'99)*, Boston, Massachusetts, EE. UU. A., mayo de 1999.

[Asensio00] Juan Ignacio Asensio Pérez, ***Contribución a la Especificación y Gestión Integrada de la Calidad de Servicio en Aplicaciones de Objetos Distribuidos***, Tesis Doctoral, Universidad de Valladolid, junio de 2000.

B

[Backlawski01] Kenneth Backawski, Mieczyslaw K. Kokar, Paul A. Kogut, Lewis Hart, Jeffrey Smith, Williams S. Holmes III, Jerzy Letkowski, Michael L. Aronson, ***Extending UML to Support Ontology Engineering for the Semantic Web***, en *Proceedings of the Fourth International Conference on UML (UML2001)*, Toronto, Canadá, octubre de 2001.

[Ban97] Bela Ban, ***A Generic Management Model for CORBA, CMIP and SNMP***, Ph. D. Thesis, Wirtschaftswissenschaftlichen Fakultaet, University of Zurich, Suiza, diciembre de 1997.

[Bapat93] Subodh Bapat, ***Towards Richer Relationship Modeling Semantics***, en *IEEE Journal on Selected Areas in Communications*, Volume 11, Number 9, diciembre de 1993.

[Bartocci95] A. Bartocci, A. Ferrero, ***Integrated Use of SDL and GDMO***, en *Proceedings of the Seventh SDL Forum (SDL'95)*, Oslo, Noruega, septiembre de 1995.

[Bénech99] Dominique Bénech, ***Interaction Frameworks for Distributed and Cooperative Paradigms of Intelligent Systems and Networks Management***, Tesis Doctoral, Universidad Paul Sabatier de Toulouse III, Francia, noviembre de 1999.

[Bénech00] Dominique Bénech, François Jocteur-Monrozier, Anne-Isabelle Rivière, ***Supervision of the CORBA environment with SUMO: a WBEM/CIM-based management framework***, en *Proceedings of the International Symposium on Distributed Objects and Applications (DOA'00)*, septiembre de 2000.

[Biron01] Paul V. Biron, Ashok Malhtra, ***XML Schema Part 2: Datatypes***, W3C Recommendation, 2 de mayo de 2001.

[BernersLee01] Tim Berners-Lee, James Hendler, Ora Lassila, ***The Semantic Web***, en *Scientific American*, mayo de 2001.

[Blumenthal02] U. Blumenthal, B. Wijnen, ***User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)***, IETF Standard 62, Request For Comments 3414, diciembre de 2002.

[Bray00] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, ***Extensible Markup Language (XML) 1.0 (Second Edition)***, W3C Recommendation, octubre de 2000.

[Brickley03] Dan Brickley, R.V. Guha, ***RDF Vocabulary Description Language 1.0: RDF Schema***, W3C Working Draft, 23 de enero de 2003.

[Bumpus00] Winston Bumpus, John W. Sweitzer, Patrick Thompson, Andrea R. Westerinen, Raymond C. Williams, ***Common Information Model***, John Wiley & Sons, Inc. Nueva York, EE. UU. A., 2000.

C

[Case02] J. Case, D. Harrington, R. Presuhn, B. Wijnen, ***Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)***, IETF Standard 62, Request For Comments 3412, diciembre de 2002.

[Castillo02] Sergio Castillo Castelblanco, ***Composición de Servicios Mediante el Modelo de los Agentes Móviles***, Tesis Doctoral, Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, octubre de 2002.

[Chaudhri98] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, James P. Rice, ***OKBC: A Programmatic Foundation for Knowledge Base Interoperability***, en *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98)*, Madison, Wisconsin, EE. UU. A., julio de 1998.

[Cherry02] Steven M. Cherry, ***Weaving a Web of Ideas***, en *IEEE Spectrum*, Volume 39, Number 9, septiembre de 2002.

[Connolly01] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuiness, Peter F. Patel-Schneider, Lynn Andrea Stein, ***DAML+OIL (March 2001) Reference Description***, W3C Notes, 18 de diciembre de 2001.

[Corcho99a] Óscar Corcho, Asunción Gómez-Pérez, ***Direct and Inverse Translators between Flogic and Ontolingua in the context of ODE and the (KA)2 Initiative: a case of Study***, en *Proceedings of the International Workshop on Ontological Engineering on the Global Information Infrastructure, (EKAW'99)*, Dagsthul, Alemania, mayo de 1999.

[Corcho99b] Óscar Corcho, Asunción Gómez-Pérez, ***Guidelines to Study Differences in Expressiveness between Ontology Specification Languages: A Case Of Study***, en *Proceedings of the Twelfth Banff Knowledge Acquisition for Knowledge-Based systems (KAW'99)*, Banff, Alberta, Canadá, octubre de 1999.

[Corcho00] Óscar Corcho, Asunción Gómez-Pérez, ***A Roadmap to Ontology Specification Languages***, en *Proceedings of the Twelfth International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000)*, Juan-les-Pins, Francia, octubre de 2000.

[Cranefield99] Stephen Cranefield, Martin Purvis, ***UML as an Ontology Modelling Language***, en *Proceedings of the Workshop on Intelligent Information Integration, Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, Estocolmo, Suecia, julio de 1999.

[Cranefield01] Stephen Cranefield, ***UML and the Semantic Web***, en en *Proceedings of the International Semantic Web Working Symposium (SWWS'01)*, Stanford University, California, EE. UU. A., julio de 2001.

[Crubézy02] Monica Crubézy, ***The Protégé Axiom Language and Toolset ("PAL")***, Protégé Project, Stanford University, abril de 2002. Disponible en http://protege.stanford.edu/plugins/paltabs/pal-documentation/index.html

D

[Dean02] Mike Dean, Dan Connolly, Frank van Harmelen, James Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, Lynn Andrea Stein, ***Web Ontology Language (OWL) Reference Version 1.0***, W3C Working Draft, 12 de noviembre de 2002.

[Deri96] Luca Deri, ***Network Management for the 90s***, en *Proceedings of the European Conference on Object-Oriented Programming (ECOOP'96)*, Linz, Austria, julio de 1996.

[Deri97] Luca Deri, Bela Ban, ***Static vs. Dynamic CMIP/SNMP Network Management Using CORBA***, en *Proceedings of the Fourth International Conference on Intelligence in Services and Networks (IS&N'97)*, Como, Milán, Italia, mayo de 1997.

[DMTF97] Desktop Management Task Force, Inc., ***DMI to SNMP Mapping Specification***, Version 1.0, DMTF Standard DSP0002, noviembre de 1997.

[DMTF99] Distributed Management Task Force, Inc., ***Common Information Model Specification***, Version 2.2, DMTF Standard DSP0004, junio de 1999.

[DMTF00] Distributed Management Task Force, Inc., ***Common Information Model (CIM) Core Model***, Version 2.4, DMTF Whitepaper DSP0111, agosto de 2000.

[DMTF02a] Distributed Management Task Force, Inc., ***Specification for the Representation of CIM in XML***, Versión 2.1, DMTF Standard DSP0201, mayo de 2002.

[DMTF02b] Distributed Management Task Force, Inc., ***Master MIF***, Version 020507, mayo de 2002.

[DMTF02c] Distributed Management Task Force, Inc., ***System Management BIOS (SMBIOS) Reference Specification***, Version 2.3.4, DMTF Standard DSP0134, diciembre de 2002.

[DMTF03a] Distributed Management Task Force, Inc., ***Desktop Management Interface Specification***, Version 2.0.1s, DMTF Standard DSP0005, enero de 2003.

[DMTF03b] Distributed Management Task Force, Inc., ***Specification for CIM Operations over HTTP***, Version 1.1, DMTF Standard DSP0200, enero de 2003.

[Doan02] AnHai Doan, Jayana Madhavan, Pedro Domingos, Alon Halevy, ***Learning to Map between Ontologies on the Semantic Web***, en *Proceedings of the*

Eleventh International World Wide Web Conference (WWW2002), Honolulu, Hawaii, EE. UU. A., mayo de 2002.

[Duarte99] Elias Procópio Duarte Jr., Martin A. Musicante, ***Formal Specification of SNMP MIB's Using Action Semantics: The Routing Proxy Case Study***, in *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management (IM'99)*, Boston, Massachusetts, EE. UU. A., mayo de 1999.

E

[ECIMF01] E-Commerce Integration Meta-Framework (ECIMF) Project, ***ECIMF Semantic Translation tool***, noviembre de 2001, disponible en http://www.ecimf.org/software.html

[Elliot01] C. Elliott, D. Harrington, J. Jason, J. Schoenwaelder, F. Strauss, W. Weiss, ***SMIng Objectives***, IETF Request For Comments 3216, diciembre de 2001.

[Ensel01] Christian Ensel, Alexander Keller, ***Managing Application Service Dependencies with XML and the Resource Description Framework***, en *Proceedings of the Seventh IEEE/IFIP International Symposium on Integrated Network Management (IM'2001)*, Seattle, Washington, EE. UU. A., mayo de 2001.

[Evans99] Andy Evans, Stuart Kent, ***Core Meta-Modelling Semantics of UML: The pUML Approach***, en *Proceedings of the 2nd International Conference on the Unified Modeling Language*, Fort Collins, Colorado, EE. UU. A., octubre de 1999.

F

[Fensel00] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann, M. Klein, ***OIL in a nutshell***, en *Proceedings of the Twelfth European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW'00)*, Juan-les-Pins, Francia, octubre de 2000.

[Fensel01] Dieter Fensel, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, ***OIL: An Ontology Infrastructure for the Semantic Web***, en *IEEE Intelligent Systems*, marzo/abril 2001.

[Fernández01] Gregorio Fernández Fernández, ***Representación del conocimiento en sistemas inteligentes***, Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, octubre de 2001. Ciberlibro disponible en http://www.gsi.dit.upm.es/~gfer/ssii/rcsi/.

[Festor99] O. Festor, P. Festor, Laurent Andrey, N. Ben Youssef, ***Integration of WBEM-based Management Agents in the OSI Framework***, en *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management (IM'99)*, Boston, Massachusetts, EE. UU. A, mayo de 1999.

[Fielding99] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. T. Berners-Lee, ***Hypertext Transfer Protocol -- HTTP/1.1***, IETF Request For Comments 2616, junio de 1999.

[Fikes01] Richard Fikes, Deborah McGuinness, ***An Axiomatic Semantics for RDF, RDF-S, and DAML+OIL (March 2001)***, W3C Note, 18 de diciembre de 2001.

G

[Goad01] Chris Goad, ***Describing Computation within RDF***, en *Proceedings of the International Semantic Web Working Symposium (SWWS'01)*, Stanford University, California, EE. UU. A., julio de 2001.

[Gómez99a] Asunción Gómez Pérez, V. Richard Benjamins, ***Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods***, en *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5)*, Estocolmo, Suecia, agosto de 1999.

[Gómez99b] Asunción Gómez-Pérez, ***Ontological Engineering: A State of the Art***, en *Expert Update*, Volume 2, Number 3, otoño de 1999.

[Gómez02] Asunción Gómez Pérez, Óscar Corcho, ***Ontology Languages for the Semantic Web***, en *IEEE Intelligent Systems*, Volume 17, Issue 1, enero/febrero de 2002.

[Grosof02] Benjamin Grosof, Mahesh D. Gandhe, Timothy W. Finin, ***SweetJess: Translating DamlRuleML to Jess***, en *Proceedings of the International Workshop on Rule Markup Languages for Business Rules on the Semantic Web*, Cerdeña, Italia, junio de 2002.

[Gruber93] Thomas R. Gruber, ***A Translation Approach to Portable Ontology Specification***, en *Knowledge Acquisition*, Volume 5, Issue 2, junio de 1993.

H

[Harrington02] D. Harrington, R. Presuhn, B. Wijnen, ***An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks***, IETF Standard 62, Request For Comments 3411, diciembre de 2002.

[Hasselmeyer99] P. Hasselmeyer, ***A Methodology for Formalizing GDMO Behavior Descriptions***, en *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management (IM'99)*, Boston, Massachusetts, EE. UU. A., mayo de 1999.

[Heflin99] Jeff Heflin, James Hendler, Sean Luke, ***Applying Ontology to the Web: A Case Study***, en *Proceedings of the International Work-Conference on Artificial and Natural Neural Networks (IWANN'99)*, Volume II, Alicante, España, junio de 1999.

[Hegering99] Heinz-Gerd Hegering, Sebastian Abeck, Bernhard Neumair, ***Integrated Management of Networked Systems***. Morgan Kaufmann, 1999.

[Hendler00] James Hendler, Deborah L. McGuinness, ***The DARPA Agent Markup Language***, en Dieter Fensel, Editor, ***The semantic Web and its languages***, en *IEEE Intelligent Systems*, Volume 15, Issue 6, noviembre/diciembre de 2000.

[Horrocks02] Ian Horrocks, ***DAML+OIL: a Reason-able Web Ontology Language***, en *Proceedings of the 8th Conference on Extending Database Technology (EDBT 2002)*, Praga, República Checa, marzo de 2002.

[Hovy98] Eduard Hovy, ***Combining and Standardizing Large-Scale, Practical Ontologies for Machine Translation and Other Uses***, en *Proceedings of the 1st International Conference on Language Resources and Evaluation (LREC)*, Granada, España, mayo de 1998.

I

[ITUT92a] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***Information technology - Open Systems Interconnection - Structure of Management Information: Guidelines for the definition of managed objects***, Recomendación X.722, enero de 1992.

[ITUT92b] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***Information technology - Open Systems Interconnection - Structure of management information: Definition of management information***, Recomendación X.721, febrero de 1992.

[ITUT95a] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***Generic network information model***, Recomendación M.3100, julio de 1995.

[ITUT95b] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***Information technology - Open Systems Interconnection - Structure of management information: General Relationship Model***, Recomendación X.725, noviembre de 1995.

[ITUT97a] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***Upper layer protocol profiles for the Q3 and X interfaces***, Recomendación Q.812, junio de 1997.

[ITUT97b] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***Information technology - Open Systems Interconnection - Systems management overview***, Recomendación X.701, agosto de 1997.

[ITUT97c] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***Guidelines for the use of Z in formalizing the behaviour of managed objects***, Recomendación X.722, Enmienda 3, agosto de 1997.

[ITUT97d] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***Information technology - Open distributed processing - Reference Model: Overview***, Recomendación X.901, agosto de 1997.

[ITUT97e] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***Information technology – Open Systems Interconnection – Common Management Information service***, Recomendación X.710, octubre de 1997.

[ITUT97f] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***Information technology – Open Systems Interconnection – Common Management Information Protocol: Specification***, Recomendación X.711, octubre de 1997.

[ITUT00a] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***Overview of TMN Recommendations***, Recomendación M.3000, febrero de 2000.

[ITUT00b] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***Principles for a Telecommunication Management Network***, Recomendación M.3010, febrero de 2000.

[ITUT00c] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***TMN Interface Specification Metodology***, Recomendación M.3020, febrero de 2000.

[ITUT00d] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***TMN management functions***, Recomendación M.3400, febrero de 2000.

[ITUT01a] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***CORBA-based TMN services***, Recomendación Q.816, enero de 2001.

[ITUT01b] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***TMN guidelines for defining CORBA managed objects***, Recomendación X.780, enero de 2001.

[ITUT01c] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***CORBA generic network and network element level information model***, Recomendación M.3120, octubre de 2001.

[ITUT02] International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ***Specification and description language (SDL)*** , Recomendación Z.100, agosto de 2002.

K

[Kalyanasundaram93] Pramod Kalyanasundaram, Adarshpal S. Sethi, ***An Application Gateway Design for OSI-Internet Management***, in *Proceedings of the Third IFIP/IEEE International Symposium on Integrated Network Management (IM'93)*, San Francisco, California, EE. UU. A, abril de 1993.

[Kalyanasundaram94] Pramod Kalyanasundaram, Adarshpal S. Sethi, ***Interoperability Issues in Heterogeneous Network Management***, en *Journal of Network and Systems Management*, Volume 2, No. 2, junio de 1994.

[Karp99] Peter D. Karp, Vinay K. Chaudhri, Jerome Thomere, ***XOL: An XML-Based Ontology Exchange Language***, Technical Report, Artificial Intelligence Center, SRI International, agosto de 1999.

[Keller95] J. Keller, ***An extension of GDMO for formalizing managed objects behaviour***, en *Proceedings of the Eighth IFIP TC6 International Conference on Formal Description Techniques (FORTE'95)*, Montreal, Canadá, octubre de 1995.

[Keller99] A. Keller, ***Managing the Management: CORBA-based Instrumentation of Management Systems***, en *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management (IM'99)*, Boston, Massachusetts, EE. UU. A., mayo de 1999.

[Kent00] Robert E. Kent, ***Conceptual Knowledge Markup Language: An Introduction***, en *Netnomics: Economic research and electronic networking*. Special Issue on Information and Communication Middleware, Volume 2, Issue 2, 2000.

[Kifer95] Michael Kifer, Georg Lausen, James Wu, ***Logical foundations of object-oriented and frame-based languages***, en *Journal of the ACM*, Volume 42, Number 4, julio de 1995.

[Kiryakov01] Atanas K. Kiryakov, Kiril Iv. Simov, Marin Dimitrov, ***OntoMap - the Guide to the Upper-Level***, en *Proceedings of the International Semantic Web Working Symposium (SWWS)*, Stanford University, California, EE. UU. A., julio de 2001.

[Klein01] Michel Klein, ***Combining and relating ontologies: an analysis of problems and solutions***, en *Proceedings of the Workshop on Ontologies and Information Sharing (IJCAI'01)*, Seattle, Washington, EE. UU. A, agosto de 2001.

[Kogut02] Paul Kogut, Stephen Cranefield, Lewis Hart, Kenneth Baclawski, Mieczyslaw Kokar, Jeffrey Smith, ***UML for Ontology Development***, en *Knowledge Engineering Review Journal*, Special Issue on Ontologies in Agent Systems, Volume 17, Issue 1, marzo de 2002.

L

[Lassila99] Ora Lassila, Ralph R. Swick, editores, ***Resource Description Framework (RDF) Model and Syntax Specification***, W3C Recommendation, febrero de 1999.

[Lavinal03] Emmanuel Lavinal, Thierry Desprats, Yves Raynaud, ***A Conceptual Framework for Building CIM-Based Ontologies***¸ en *Proceedings of the Eighth IFIP/IEEE International Symposium on Integrated Network Management (IM'2003)*, Colorado Springs, Colorado, EE. UU. A., marzo de 2003.

[Levi02] D. Levi, P. Meyer, B. Stewart, ***Simple Network Management Protocol (SNMP) Applications***, IETF Standard 62, Request For Comments 3411, diciembre de 2002.

[LópezDeVergara01] Jorge E. López de Vergara, Víctor A. Villagrá, Juan I. Asensio, Julio Berrocal, ***Análisis y comparativa de las alternativas propuestas para la Gestión Basada en Web***, en *Actas de las III Jornadas de Ingeniería Telemática, Jitel'01*, Barcelona, 19-21 de septiembre de 2001.

[LópezDeVergara02] Jorge E. López de Vergara, Víctor A. Villagrá, Julio Berrocal, ***Semantic Management: advantages of using an ontology-based management information meta-model***, en *Proceedings of the HP Openview University Association Ninth Plenary Workshop (HP-OVUA'2002)*, Böblingen, Alemania, junio de 2002.

[LópezDeVergara03a] Jorge E. López de Vergara, Víctor A. Villagrá, Julio Berrocal, Juan I. Asensio, Roney Pignaton, ***Semantic Management: Application of Ontologies for the Integration of Management Information Models***, en *Proceedings of the Eighth IFIP/IEEE International Symposium on Integrated Network Management (IM'2003)*, Colorado Springs, Colorado, EE. UU. A., marzo de 2003.

[LópezDeVergara03b] Jorge E. López de Vergara, Víctor A. Villagrá, Juan I. Asensio, Julio Berrocal, ***Ontologies: Giving Semantics to Network Management Models***, aceptado para su publicación en *IEEE Network*, special issue on Network Management, Volume 17, Number 3, mayo/junio de 2003.

M

[MacGregor91] Robert M. MacGregor, ***Inside the LOOM Description Classifier***, en *SIGART Bulletin*, Volume 2, Number 3, junio de 1991.

[Maedche02] Alexander Maedche, Boris Motik, Nuno Silva, Raphael Volz, ***MAFRA – A MApping FRAmework for Distributed Ontologies***, en *Proceedings of the Thirteenth European Conference on Knowledge Engineering and Knowledge Management (EKAW'02)*, Madrid, España, octubre de 2002.

[MartinFlatin01] J.P. Martin-Flatin, ***Toward Universal Information Models in Enterprise Management***, en *Proceedings of the VLDB 2001 Workshop on Databases in Telecommunications (DBTel 2001)*, Roma, Italia, septiembre de 2001.

[McCloghrie91] K. McCloghrie M. T. Rose, ***Management Information Base for Network Management of TCP/IP-based internets: MIB-II***, IETF Standard 17, Request For Comments 1213, marzo de 1991.

[McCloghrie99a] K. McCloghrie, D. Perkins, J. Schoenwaelder, J. Case, M. Rose, S. Waldbusser, ***Structure of Management Information Version 2 (SMIv2)***, IETF Standard 58, Request For Comments 2578, abril de 1999.

[McCloghrie99b] K. McCloghrie, D. Perkins, J. Schoenwaelder, J. Case, M. Rose, S. Waldbusser, ***Textual Conventions for SMIv2***, IETF Standard 58, Request For Comments 2579, abril de 1999.

[McCloghrie99c] K. McCloghrie, D. Perkins, J. Schoenwaelder, J. Case, M. Rose, S. Waldbusser, ***Conformance Statements for SMIv2***, IETF Standard 58, Request For Comments 2580, abril de1999.

[McCloghrie99d] K. McCloghrie, A. Bierman, ***Entity MIB (Version 2)***, IETF Request For Comments 2737, diciembre de 1999.

[McGuinness00] Deborah L. McGuinness, Richard Fikes, James Rice, Steve Wilder, ***An Environment for Merging and Testing Large Ontologies***, en *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, Breckenridge, Colorado, EE. UU. A., abril de 2000.

[McGuinness02] Deborah L. McGuiness, Richard Fikes, James Hendler, Lynn Andrea Stein, ***DAML+OIL: An Ontology Language for the Semantic Web***, en *IEEE Intelligent Systems*, Volume 17, Number 5, septiembre/octubre de 2002.

[Mena00] Eduardo Mena, Arantza Illarramendi, Vipul Kashyap, Amit P. Sheth, ***OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across pre-existing Ontologies***, en *International Journal on Distributed And Parallel Databases (DAPD)*, Volume 8, Number 3, abril de 2000.

[Microsoft02] Microsoft Corporation, ***SNMP Provider***, Platform SDK Release: octubre de 2002, disponible en http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/snmp_provider.asp

[Mitra00] Prasenjit Mitra, Gio Wiederhold , Martin Kersten, ***A Graph-Oriented Model for Articulation of Ontology Interdependencies***, en *Proceedings of the Conference on Extending Database Technology 2000 (EDBT'2000)*, Konstanz, Alemania, marzo de 2000.

[Mitra02] Nilo Mitra, ***SOAP Version 1.2 Part 0: Primer***, W3C Candidate Recommendation, 19 de diciembre de 2002.

[Moore01] B. Moore, E. Ellesson, J. Strassner, A. Westerinen, ***Policy Core Information Model -- Version 1 Specification***, IETF Request for Comments 3060, febrero de 2001.

[Motta99] Enrico Motta, ***Reusable Components for Knowledge Modelling: Principles and Case Studies in Parametric Design***, IOS Press, Ámsterdam, Holanda, 1999.

N

[Neches91] Robert Neches, Richard Fikes, Tim Finin, Ramesh Patil, Ted Senator, William R. Swartout, ***Enabling Technology For Knowledge Sharing***, en *AI Magazine*, Volume 12, Number 3, otoño de 1991.

[Neumair98] Bernard Neumair, ***Distributed Applications Management based on ODP Viewpoint Concepts and CORBA***, en *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS'98)*, New Orleans, EE. UU. A., febrero de 1998.

[Nielsen00] H. Nielsen, P. Leach, S. Lawrence, ***An HTTP Extension Framework***, IETF Request For Comments 2774, febrero de 2000.

[NMF93a] Network Management Forum, ***Translation of Internet MIBs to ISO/CCITT GDMO MIBs***, Issue 1.0, Forum 026, octubre de 1993.

[NMF93b] Network Management Forum, ***ISO/CCITT to Internet Management Security***, Issue 1.0, Forum 027, octubre de 1993.

[NMF93c] Network Management Forum, ***ISO/CCITT to Internet Management Proxy***, Issue 1.0, Forum 028, octubre de 1993.

[NMF93d] Network Management Forum, ***Translation of Internet MIB-II (RFC 1213) TO ISO/CCITT GDMO MIB***, Issue 1.0, Forum 029, octubre de 1993.

[NMF93e] Network Management Forum, ***Translation of ISO/CCITT GDMO MIBs to Internet MIBs***, Issue 1.0, Forum 030, octubre de 1993.

[Noy99] Natalya Fridman Noy, Mark A. Musen, ***An Algorithm for Merging and Aligning Ontologies: Automation and Tool Support***, en *Proceedings of the Workshop on Ontology Management, Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, Orlando, Florida, EE. UU. A., julio de 1999.

[Noy00] N. F. Noy, M. A. Musen, ***PROMPT: Algorithm and tool for automated ontology merging and alignment***. En *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI'00)*. Austin, Texas, EE. UU. A., julio-agosto de 2000.

[Noy01] Natalya F. Noy, Mark A. Musen, ***Anchor-PROMPT: Using Non-Local Context for Semantic Matching***, en *Proceedings of the Workshop on Ontologies and Information Sharing at the Seventeenth International Joint*

Conference on Artificial Intelligence (IJCAI-2001), Seattle, Washington, EE. UU. A., agosto de 2001.

[Noy02a] Natalya F. Noy, Mark A. Musen, ***PromptDiff: A Fixed-Point Algorithm for Comparing Ontology Versions***, en *Proceedings of the Eighteenth National Conferencen Artificial Intelligence (AAAI-02)*, Edmonton, Alberta. AAAI Press, agosto de 2002.

[Noy02b] Natalya F. Noy, Mark A. Musen, ***Evaluating Ontology-Mapping Tools: Requirements and Experience***, en *Proceedings of the Workshop on Evaluation of Ontology Tools (EON2002) at the Thirteenth International Conference on Knowledge Engineering and Knowledge Management (EKAW'02)*, Sigüenza, España, septiembre de 2002.

O

[O'Leary98] Daniel E. O'Leary, ***Using AI in Knowlege Management: Knowledge Bases and Ontologies***, en *IEEE Intelligent Systems*, Volume 13, Issue 3, mayo/junio de 1998.

[OMG01a] Object Management Group, ***OMG Telecom Task Force Roadmap White Paper V1.1***, OMG document telecom/01-05-01, mayo de 2001.

[OMG01b] Object Management Group, ***Unified Modeling Language (UML), version 1.4, UML Summary***, OMG document formal/01-09-72, septiembre de 2001.

[OMG01c] Object Management Group, ***Unified Modeling Language (UML), version 1.4, Object Constraint Language Specification***, OMG document formal/01-09-77, septiembre de 2001.

[OMG02a] Object Management Group, ***UML™ Profile for CORBA™***, OMG document formal/02-04-01, abril de 2002.

[OMG02b] Object Management Group, ***Meta-Object Facility Specification***, version 1.4, OMG document formal/02-04-04, abril de 2002.

[OMG02c] Object Management Group, ***CORBA 3.0.2 Specification, CORBA Overview***, OMG document formal/02-06-38, junio de 2002.

[OMG02d] Object Management Group, ***CORBA 3.0.2 Specification, IDL Syntax & Semantics***, OMG document formal/02-06-39, junio de 2002.

[OMG02e] Object Management Group, ***CORBA 3.0.2 Specification, General Inter-ORB Protocol***, OMG document formal/02-06-51, junio de 2002.

[OMG02f] Object Management Group, ***Notification Service Specification***, version 1.0.1, OMG document formal/02-08-04, agosto de 2002.

[OMG02g] Object Management Group, ***Naming Service Specification***, version 1.2, OMG document formal/02-09-02, septiembre de 2002.

[OMG02h] Object Management Group, ***Telecom Log Service Specification***, version 1.1, OMG document formal/02-11-12, noviembre de 2002.

[OntoWeb01] OntoWeb Consortium, ***Technical Roadmap***, version 1.0, Deliverable 1.1, noviembre de 2001.

[OntoWeb02a] OntoWeb Consortium, ***Ontology Language Standardisation Efforts***, version 1, Deliverable 4.0, enero de 2002.

[OntoWeb02b] OntoWeb Consortium, ***Requirements of Ontology Languages***, version 1.2, Deliverable 4.1, marzo de 2002.

[Open00] The Open Group, ***Inter-Domain Management: Specification and Interaction Translation***, Open Group Document C802, enero de 2000.

P

[Pablos01] Rolando Pablos Sánchez, ***Análisis y Evaluación de Sistemas de Gestión Basada en Web para su Aplicación en Servicios de Intermediación Electrónica***, Proyecto Fin de Carrera, E.T.S.I. de Telecomunicación, Universidad Politécnica de Madrid, febrero de 2001.

[Pan01] Jeff Z. Pan, Ian Horrocks, ***Metamodeling architecture of web ontology languages***, en *Proceedings of the International Semantic Web Working Symposium (SWWS'01)*, Stanford University, California, EE. UU. A., julio de 2001.

[Park97] John Y. Park, John H. Gennari, Mark A. Musen, ***Mappings for Reuse in Knowledge-Based Systems***, Technical Report SMI-97-0697, Stanford University, 1997.

[Pavlou97] G. Pavlov, ***From Protocol-based to Distributed Object-based Management Architectures***, en *Proceedings of the 4th Workshop of the Open View University Association (OVUA'97)*, Madrid, España, abril de 1997.

[Pinto99] H. Sofia Pinto, Asunción Gómez-Pérez, João P. Martins***, Some Issues on Ontology Integration***, en *Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5)*, Estocolmo, Suecia, agosto de 1999.

[Poirier95] Stéphane Poirier, Colin Ashford, ***Semantics: the key to interoperability***, en *Proceedings of the Business Object Design and Implementation Workshop (OOPSLA'95)*, Austin, Texas, EE. UU. A., octubre de 1995.

[Presuhn02] R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser, ***Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)***, IETF Standard 62, Request For Comments 3418, diciembre de 2002.

R

[Rahm01] Erhard Rahm, Phillip A. Bernstein, ***A survey of approaches to automatic schema matching***, en *The Very Large Data Bases Journal (VLDB)*, Volume 10, Number 4, , diciembre de 2001.

[Rivière95] Anne-Isabelle Rivière, Michelle Sibilla, Daniel Marquié, Yves Raynaud, Simon Towers, ***Network and System Management: Integration of Standardized Syntactic Descriptions***, en *Proceedings of the Second HP OpenView University Association Plenary Meeting (HP-OVUA'95)*, Munich, Alemania, mayo de 1995.

[Rivière96a] Anne-Isabelle Rivière, Adrian Pell, Simon Towers, Michelle Sibilla, Daniel Marquié, ***Integration of heterogeneous network and system management models***, en *Proceedings of the Third HP OpenView University Association Plenary Meeting (HP-OVUA'96)*, Toulouse, Francia, marzo de 1996.

[Rivière96b] Anne-Isabelle Rivière, Adrian Pell, Michelle Sibilla, ***Network Management Information: From Protocols to Information Integration***, en *Proceedings of the Seventh IFIP/IEEE Workshop on Distributed Systems: Operations and Management (DSOM'96)*, L'Aquila, Italia, octubre de 1996.

[Rivière98] Anne-Isabelle Rivière, Michelle Sibilla, ***Management Information Models Integration: From Existing Approaches to new Unifying Guidelines***, en *Journal of Network and Systems Management*, Volume 6, Number 3, septiembre de 1998.

[Rodríguez99a] Manuel Rodríguez Cayetano, ***Contribución a la Especificación Formal de Sistemas TMN a Partir del Modelo de Información de Gestión***, Tesis Doctoral, Universidad de Valladolid, 1999.

[Rodríguez99b] M. Rodríguez, R. Calmeau, E. Fernández, ***Application of SDL-92 for the specification of OSI management systems***, en *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management (IM'99)*, Boston, Massachusetts, EE. UU. A., mayo de 1999.

S

[Shen01] Jun Shen, ***Research on Multi-Agent System Based Network Management Models***, Ph.D. Thesis, Southeast University, Nanjing, China, 2001.

[Shen03] Jun Shen, Yun Yang, ***RDF-Based Knowledge Models for Network Management***, en *Proceedings of the Eighth IFIP/IEEE International Symposium on Integrated Network Management (IM'2003)*, Colorado Springs, Colorado, EE. UU. A., marzo de 2003.

[Schönwälder99] J. Schönwälder, F. Strauß, ***Next Generation Structure of Management Information for the Internet***, en *Proceedings of the Tenth IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM'99)*, Zürich, Alemania, octubre de 1999.

[Schönwälder01] J. Schönwälder, A. Müller, ***Reverse Engineering Internet MIBs***, en *Proceedings of the Seventh IFIP/IEEE International Symposium on Integrated Network Management*, Seattle, Washington, EE. UU. A., mayo de 2001.

[Schott02] J. Schott, A. Westerinen, J. P. Martin-Flatin, P. Rivera, ***Common Information vs. Information Overload***, en *Proceedings of the Network Operations and Management Symposium (NOMS'2002)*, Florencia, Italia, abril de 2002.

[Sloman02] Morris Sloman, Emil Lupu, ***Security and Management Policy Specification***, en *IEEE Network*, special issue on Policy-Based Networking, Volume 16, Number 2, marzo/abril 2002.

[Steff99] Yann Steff, Thierry Desprats, Daniel Marquié, ***New Languages for Interoperable Co-operative Management Platforms***, en *Proceedings of the Sixth HP OpenView University Association Plenary Meeting (HP-OVUA'99)*, Bolonia, Italia, junio de 1999.

[Studer98] R. Studer, V.R. Benjamins, D. Fensel, ***Knowledge Engineering: Principles and Methods***, en *Data & Knowledge Engineering*. 25: 161-197, 1998.

[Stumme01] Gerd Stumme, Alexander Maedche, ***FCA-MERGE: Bottom-Up Merging of Ontologies***, en *Proceedings of the Seventeenth International Joint*

Conference on Artificial Intelligence (IJCAI 2001), Seattle, Washington, EE. UU. A., agosto de 2001.

[Sun02] Sun Microsystems, Inc, ***Solaris WBEM Services Administration Guide***, Part No. 806-6827-10, mayo de 2002.

[Swartout99] William Swartout, Austin Tate, ***Ontologies***, Guest Editors' Introduction en *IEEE Intelligent Systems*, Volume 14, Issue 1, enero/febrero 1999.

T

[T101] Standards Committee T1 Telecommunications, ***Telecom Glossary 2000***, American National Standard T1.523-2001, febrero de 2001.

[Tosic98] Vladimir Tosic, ***On Object-Oriented Information Specification in Network and System Management***, Magisterium Thesis, Faculty of Electronic Engineering, Nis, Yugoslavia, 1998.

[Tosic99] Vladimir Tosic, Slobodanka Djordjevic-Kajan, ***The Common Information Model (CIM) Standard – An Analysis of Features and Open Issues***, en *Proceedings of the Fourth International Conference on Telecommunications in Modern Satellite, Cable, and Broadcasting Services (TELSIKS'99)*, Nis, Yugoslavia, octubre de 1999.

V

[Valera01] Francisco Valera, Jorge E. López de Vergara, José I. Moreno, Víctor A. Villagrá , Julio Berrocal, ***Communication and Management Experiences in an E-Commerce MAS-Based Environment***, en *Communications of the Association for Computing Machinery (CACM)*. Volume 44, Number 4, abril de 2001.

[Villagrá02] Víctor A. Villagrá, Juan I. Asensio, Jorge E. López de Vergara, Julio J. Berrocal, Roney Pignaton, ***An approach to the transparent management instrumentation of distributed applications***, en *Proceedings of the Eighth IEEE/IFIP Network Operations and Management Symposium (NOMS'2002),* Florencia, Italia, abril de 2002.

[Vossen98] P. Vossen, (ed.), ***EuroWordNet: A Multilingual Database with Lexical Semantic Networks***, Kluwer Academic Publishers, Dordrecht, 1998.

W

[Wache01] H. Wache, T. Vgele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner, ***Ontology-based integration of information - a survey of existing approaches***, en *Proceedings of the Workshop Ontologies and Information Sharing (IJCAI'2001*), Seattle, Washington, EE. UU. A., agosto de 2001.

[Waldbusser00a] S. Waldbusser, P. Grillo, ***Host Resources MIB***, IETF Request For Comments 2790, marzo de 2000.

[Waldbusser00b] S. Waldbusser, ***Remote Network Monitoring Management Information Base***, IETF Request For Comments 2819, mayo de 2000.

[Wang02] Xin Wang, Christine W. Chan, Howard J. Hamiltion, ***Design of Knowledge-Based Systems with the Ontology-Domain-System Approach***, en *Proceedings of the Fourteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'02)*, Ischia, Italia, julio de 2002.

[Westerinen01] Andrea Westerinen, ***CIM – The Common Information Model DMTF Tutorial***, en *Eighth Annual DMTF Developers' Conference*, San José, California, EE. UU. A., junio de 2001.

[Wijnen02] B. Wijnen, R. Presuhn, K. McCloghrie, ***View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)***, IETF Standard 62, Request For Comments 3415, diciembre de 2002.

Z

[Zhang93] Tianning Zhang, Panos-Gavriil Tsigaridas, ***A Knowledge-based Model for Network Service Management***, en *Proceedings of the First IEEE Symposium Global Data Networking*, diciembre de 1993.

[Zhang96a] Tianning Zhang, Stefan Covaci, ***The Semantics of Network Management Information***, en *Proceedings of the Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '96)*, San Francisco, California, EE. UU. A., marzo de 1996.

[Zhang96b] Tianning Zhang, Stefan Covaci, Radu Popescu-Zeletin, ***Intelligent Agents in Network and Service Management***, en *Proceedings of the Global Telecommunications Conference (GLOBECOM '96)*, Londres, Reino Unido, noviembre de 1996.

Abreviaturas y acrónimos

A

ABNF *Augmented Backus-Naur Form*, Forma Backus-Naur Aumentada.

ANSI *American National Standards Institute*, Instituto Americano de Normalización Nacional.

ASN.1 *Abstract Syntax Notation 1*, Notación de Sintaxis Abstracta 1.

B

BIOS *Basic Input Output System*, Sistema Básico de Entrada y Salida.

C

CASE *Computer Aided Software Engineering*, Ingeniería del Software Asistida por Ordenador.

CBC *Cipher Block Chain*, Encadenado de Bloques Cifrados.

CCITT *Comité Consultatif International Téléphonique et Télégraphique*, Comité Consultivo Internacional de Telefonía y Telegrafía.

CIM Common Information Model, Modelo de Información Común.

CIMOM *CIM Object Manager*, Gestor de Objetos CIM.

CMIP *Common Management Information Protocol*, Protocolo Común de Información de Gestión.

CMIS *Common Management Information Service*, Servicio Común de Información de Gestión.

CODIP *Components for Ontology Driven Information Push*, Componentes para Promover la Información Mediante Ontologías.

CORBA *Common Object Request Broker Architecture*, Arquitectura Común de Intermediarios de Peticiones de Objetos.

D

DAML *DARPA Agent Markup Language*, Lenguaje de Marcas de Agentes de DARPA.

DAML-L *DAML Logic*, DAML Lógico.

DAML+OIL Fusión de los lenguajes DAML y OIL.

DARPA *Defense Advanced Research Projects Agency*, Agencia de Proyectos de Investigación Avanzada para la Defensa.

DES *Data Encryption Standard*, Estándar de Cifrado de Datos.

DISMAN *Distributed Management Workgroup*, Grupo de Gestión Distribuida del IETF.

DMI *Desktop Management Interface*, Interfaz de Gestión de Equipos de Sobremesa.

DMTF Antiguamente *Desktop Management Task Force*, Grupo de Trabajo de Gestión de Equipos de Sobremesa

En la actualidad *Distributed Management Task Force*, Grupo de Trabajo de Gestión Distribuida.

DTD *Document Type Definition*, Definición de Tipos de Documento.

DUET *DAML UML Enhanced Tool*, Herramienta UML Mejorada para DAML.

E

ECIMF *E-Commerce Integration Meta-Framework*, Meta-Marco de Integración de Comercio Electrónico.

EFD *Event Forwarding Discriminator*, Discriminador de Reenvío de Eventos.

ETSI *European Telecommunications Standards Institute*, Instituto Europeo de Estándares de Telecomunicación.

F

FCAPS *Fault, Configuration, Accounting, Performance and Security*, Fallos, Configuración, Contabilidad, Rendimiento y Seguridad.

F-Logic *Frame Logic*, Lógica de Marcos.

G

GDMO *Guidelines for the Definition of Managed Object*, Directrices para la Definición de Objetos Gestionados.

GESEMAN GEstión SEMÁNtica.

GOM *Generic Object Model*, Modelo de Objeto Genérico.

GNU *GNU's Not UNIX*, GNU no es UNIX.

GRM *General Relationship Model*, Modelo General de Relaciones.

H

HTML *Hyper-Text Markup Language*, Lenguaje de Marcas de Hipertexto.

HTTP *Hyper-Text Transfer Protocol*, Protocolo de Transferencia de Hipertexto.

I

IDL *Interface Definition Language*, Lenguaje de Definición de Interfaces.

IEEE *Institute of Electrical and Electronics Engineers*, Instituto de Ingenieros Eléctricos y Electrónicos.

IETF *Internet Engineering Task Force*, Grupo de Trabajo de Ingeniería para Internet

IIMC *ISO-Internet Management Coexistence*, Coexistencia de las Gestiones ISO e Internet.

IIOP *Internet Inter-ORB Protocol*, Protocolo Inter-ORB de Internet.

IOR *Interoperable Object Reference*, Referencia Interoperable de Objeto.

IRIT *Institut de Recherche en Informatique de Toulouse*, Instituto de Investigación en Informática de Toulouse.

IRTF *Internet Research Task Force*, Grupo de Trabajo de Investigación de Internet.

ISO *International Organization for Standardization*, Organización Internacional para la Estandarización.

IST *Information Society Technologies*, Tecnologías de la Sociedad de la Información.

ITU *International Telecommunication Union*, Unión Internacional de Telecomunicación.

J

JIDM *Joint Inter-Domain Management*, Gestión Inter-Dominios Unificada.

JIDM-IT *JIDM – Interaction Translation*, Traducción de Interacciones de JIDM.

JIDM-ST *JIDM – Specification Translation*, Traducción de Especificaciones de JIDM.

K

KIF *Knowledge Interchange Format*, Formato de Intercambio de Conocimiento.

KQML *Knowledge Query and Manipulation Language*, Lenguaje de Manipulación y Consulta de Conocimientos.

KSL *Knowledge Systems Laboratory*, Laboratorio de Sistemas de Conocimiento.

L

LISP *LISt Processor*, Procesador de Listas.

M

MAFRA *MApping FRAmework*, Marco de Correspondencia.

M&M *Merge and Map*, Fusión y Correspondencia.

MD.5 *Message Digest 5*, Resumen de Mensaje 5.

MIB *Management Information Base*, Base de Información de Gestión.

MIF *Managed Information Format*, Formato de la Información Gestionada.

MOF En el contexto del DMTF: *Managed Object Format*, Formato de Objetos Gestionados.

En el contexto de OMG: *Meta Object Facility*, Facilidad de Meta Objetos.

N

NLP *Natural Language Processing*, Procesado de Lenguaje Natural.

O

OCL *Object Constraint Language*, Lenguaje de Restricciones de Objetos.

OCML *Operational Conceptual Modeling Language*, Lenguaje de Modelado Conceptual y Operacional.

ODP *Open Distributed Processing,* Procesamiento Distribuido y Abierto.

OID *Object IDentifier*, Identificador de Objeto.

OIL *Ontology Inference Layer*, Capa de Inferencias de Ontologías; también *Ontology Interchange Language*, Lenguaje de Intercambio de Ontologías.

OKBC *Open Knowledge Base Connectivity*, Conectividad de Bases de Conocimiento Abiertas.

OMG *Object Management Group*, Grupo de Gestión de Objetos.

OML *Ontology Markup Language*, Lenguaje de Marcas para Ontologías.

ORB *Object Request Broker*, Intermediario de Peticiones a Objetos.

OSI *Open System Interconnection*, Interconexión de Sistemas Abiertos.

OSI-MS *OSI – Management System*, Sistema de Gestión OSI.

OWL *Web Ontology Language*, Lenguaje de Ontologías para la Web.

P

PAL *Protégé Axiom Language*, Lenguaje de Axiomas de Protégé.

PC *Personal Computer*, Ordenador Personal.

PDU *Protocol Data Unit*, Unidad de Datos del Protocolo.

PSM *Problem Solving Methods*, Métodos de Resolución de Problemas.

pUML *Precise UML*, UML preciso.

R

RDF *Resource Description Framework*, Marco de Descripción de Recursos.

RDFS *RDF Schema*, Esquema RDF.

RDFSFA *RDFS Fixed metamodeling Architecture*, Arquitectura de Metamodelado Fijo de RDFS

RFC *Request for Comments*, Solicitud de Comentarios.

RM-ODP *ODP Reference Model*, Modelo de Referencia ODP.

RMON *Remote MONitoring*, Monitorización Remota.

RPC *Remote Procedure Calls*, Llamadas a Procedimientos Remotos.

RuleML *Rule Markup Language*, Lenguaje de Marcas de Reglas.

S

SDL *Specification and Description Language*, Lenguaje de Especificación y Descripción.

SGMP *Simple Gateway Management Protocol*, Protocolo Simple de Gestión de Pasarelas.

SHOE *Simple HTML Ontology Extensions*, Extensiones Simples de HTML para Ontologías.

SMBIOS *System Management BIOS*, BIOS de Gestión de Sistemas.

SMI *Structure of Management Information*, Estructura de la Información de Gestión.

SMIng *SMI next generation*, Siguiente Generación de SMI.

SMIv2 *SMI version two*, Segunda Versión de SMI.

SNMP *Simple Network Management Protocol*, Protocolo Simple de Gestión de Red.

SOAP *Simple Object Adaptor Protocol*, Protocolo Simple de Adaptación de Objetos.

SPPI *Structure of Policy Provisioning Information*, Estructura de la Información de Provisión de Políticas.

T

TMN *Telecommunication Management Network*, Red de Gestión de Telecomunicaciones.

TTF *Telecom Task Force*, Grupo de Trabajo de Telecomunicaciones de OMG.

U

UBOT *UML Based Ontology Toolset*, Herramientas de Ontologías Basadas en UML

UML *Unified Modeling Language*, Lenguaje de Modelado Unificado.

URI *Uniform Resource Identifier*, Identificador Uniforme de Recursos.

URL *Uniform Resource Locator*, Localizador Uniforme de Recursos.

UTRAD *Unified TMN Requirements, Analysis and Design*, Requisitos, Análisis y Diseño de TMN Unificados.

W

W3C *World Wide Web Consortium*, Consorcio de la World Wide Web.

WBEM *Web-Based Enterprise Management*, Gestión de Empresa Basada en Web.

WMI *Windows Management Instrumentation*, Instrumentación de Gestión de Windows

X

XMI *XML Metadata Interchange*, Intercambio de Metadatos en XML.

XML *eXtensible Mark-up Language*, Lenguaje de Marcas Extensible.

XOL *XML-based Ontology exchange Language*, Lenguaje de Intercambio de Ontologías Basado en XML.

XSD *XML Schema Data types*, Tipos de Datos de Esquemas XML.

Curriculum Vitae

- En la actualidad:

  Beca de investigación del Departamento de Ingeniería de Sistemas Telemáticos de la E.T.S.I. de Telecomunicación (Universidad Politécnica de Madrid).

- Desde enero de 1999 hasta diciembre de 2002:

  Beca del Programa de Formación del Profesorado Universitario del Ministerio de Educación, Cultura y Deporte.

- Desde julio de 2000 hasta enero de 2001:

  Estancia en los Laboratorios de Investigación de Hewlett-Packard en Bristol, Reino Unido.

- Desde septiembre de 1998 a diciembre de 1998:

  Beca de investigación del Departamento de Ingeniería de Sistemas Telemáticos de la E.T.S.I. de Telecomunicación (Universidad Politécnica de Madrid).

- Septiembre de 1998:

  Título de Ingeniero de Telecomunicación por la Escuela Técnica Superior de Ingenieros de Telecomunicación (Universidad Politécnica de Madrid).

- Desde septiembre de 1997 a septiembre de 1998:

  Proyecto Fin de Carrera: *Diseño e Implementación de un Sistema para la Gestión de una Aplicación Distribuida de Comercio Electrónico*. Calificación: Matrícula de Honor. Premio del Colegio Oficial de Ingenieros de Telecomunicación al Mejor Proyecto Fin de Carrera de Ingeniería Telemática.

  Beca del Departamento de Ingeniería de Sistemas Telemáticos de la E.T.S.I. de Telecomunicación (Universidad Politécnica de Madrid).