# De la Integración de Datos a la Integración de Servicios en los Sistemas Mediadores

# From Data Integration to Services Integration in Mediator Systems

Gloria Lucia Giraldo G.[1], Ph.D; Germán Urrego G.[2] Ph.D
1. Escuela de Sistemas, Facultad de Minas, Universidad Nacional de Colombia - Sede Medellín
2. Facultad de Ingeniería Universidad de Antioquia
glgiraldog@unalmed.edu.co; gaurrego@udea.edu.co

Recibido para revisión 04 de Abril de 2008, Aceptado 19 de Mayo de 2008, Versión final 20 de Mayo de 2008

*Resumen*— **Los sistemas mediadores han demostrado ser muy eficientes en la integración de información. El sistema mediador PICSEL (Producción de Interfaces de Conocimiento para Sistemas En Linea) fue concebido inicialmente para integrar fuentes de datos múltiples y heterogéneas. Dado el amplio desarrollo que tiene la Web y los cambios de paradigma que ella conlleva, se hace necesario extender la aplicación de los sistemas mediadores a la integración de servicios. Este cambio, lleva a adaptar los sistemas existentes a este nuevo contexto aplicativo. Se trata principalmente de adaptar el contenido de la ontología y de transformar las vistas que describen el contenido de las fuentes en vistas que describan las funcionalidades de los servicios ejecutables. En este artículo se presenta, desde el punto de vista del mediador PICSEL, las características de este cambio de paradigma y sus implicaciones.**

*Palabras Claves*: **Iintegración de Información, Integración de Servicios, Sistemas Mediadores.**

**Abstract—The mediator systems are efficient for information integration. The PICSEL mediator was created for integrating multiples heterogeneous data sources. The wide Web development and his inherent new paradigms, extend the application of these systems to the services domain. This change leads to adapt the ontology and transform the views describing sources content in order to describe the executable services functionalities. This paper presents the characteristics of this paradigm change and his implications using the PICSEL frame.**

*Keywords:* **Data Integration, Mediator Systems, Services Integration.**

## I. INTRODUCCIÓN

Existen dos grandes enfoques de integración de información, uno de ellos constituido por los Sistemas Mediadores y el otro por los Almacenes de Datos (Data Warehouse).

La mayoría de los Sistemas Mediadores desarrollados hasta ahora, poseen una arquitectura centralizada [2],[5],[9],[14] según la cual, el mediador se comunica solamente con las fuentes que él integra, con el fin de responder las consultas de los usuarios. En este tipo de arquitecturas, el mediador es construido sobre un conjunto de fuentes de datos predefinidas. La respuesta a una consulta es construida únicamente a partir de los elementos de esas fuentes de datos. Otros sistemas mediadores, tienen una arquitectura descentralizada, ellos son llamados, sistemas mediadores pares o de arquitectura par-a-par (P2P). Las propiedades de autonomía y distribución de los datos, son criterios claves para el diseño de este tipo de sistemas mediadores [4].

Los Sistemas Mediadores explotan vistas abstractas que describen el contenido de las diferentes fuentes que se van a integrar. El sistema mediador PICSEL (Producción de Interfaces de Conocimiento para Sistemas En Línea) [14] fue concebido inicialmente para integrar fuentes de datos múltiples y heterogéneas. En el proyecto PICSEL II [6] se buscó automatizar al máximo la construcción del sistema mediador PICSEL, con el propósito de utilizarlo en la Web. La parte más interesante para automatizar de un sistema mediador, es su base de conocimiento, ya que ésta es específica al dominio de las fuentes de información, por lo tanto es necesario construir una base de conocimiento para cada dominio. En cambio, el motor de consultas es genérico y por tanto, solo se construye una vez y es utilizable en todos los dominios. Las ontologías, son una parte esencial de la base de conocimiento de los sistemas mediadores. En [6] se propone un

Revista Avances en Sistemas e Informática, Vol.5 No.2, Junio de 2008, Medellín, ISSN 1657-7663

método para construir de manera semiautomática una ontología del dominio para sistemas mediadores, mostrando su aplicación al sistema mediador PICSEL.

Hace algunos años, los usuarios de la Web, comenzaron a solicitar la ejecución de servicios, en lugar de la simple búsqueda de información estática. Este cambio de paradigma y el propósito de utilizar PICSEL en la Web, hizo que se extendiera su aplicación a la integración de servicios. En este nuevo contexto, fue necesario interrogarse sobre la correspondencia de la integración de servicios con respecto a la integración de fuentes de datos y sobre las consecuencias de utilizar PICSEL en este nuevo contexto. El objetivo de este artículo es presentar el resultado de estos cuestionamientos.

La estructura de este artículo es la siguiente: en la sección 2, se introduce los sistemas mediadores y se presentan los lenguajes de vistas y de interrogación de PICSEL. Luego, en la sección 3, mostramos la integración de fuentes de datos en PICSEL y en la sección 4 se presenta la integración de servicios en PICSEL. Finalmente, presentamos la conclusión y el trabajo futuro, en la sección 5.

## II. SISTEMAS MEDIADORES

Los sistemas mediadores son una de las soluciones más eficientes al problema de integración de información. PICSEL es un sistema mediador de arquitectura centralizada, que permite interrogar fuentes de información múltiples y heterogéneas. Las fuentes son relativas a un dominio de aplicación dado. PICSEL posee un motor de consultas, genérico, utilizable en cualquier dominio de aplicación y una base de conocimiento, específica al dominio del mediador. La base de conocimiento se compone de un esquema global u ontología del dominio y de la descripción del contenido de las fuentes de información. La figura 1 muestra la arquitectura de PICSEL.

La representación del conocimiento en PICSEL se hace con el lenguaje CARIN [8],[10], el cual combina un lenguaje a base de reglas y un lenguaje de clases. CARIN posee dos componentes, la componente terminológica que utiliza el lenguaje de clases y la componente deductiva constituida por un conjunto de reglas. La ontología es representada con la ayuda de estas dos componentes [12], las cuales no son presentadas en el presente artículo, por limitaciones de espacio. PICSEL posee también un lenguaje de vistas y un lenguaje de consultas que permiten expresar en términos de la ontología, respectivamente, el contenido de las fuentes y las consultas de los usuarios.

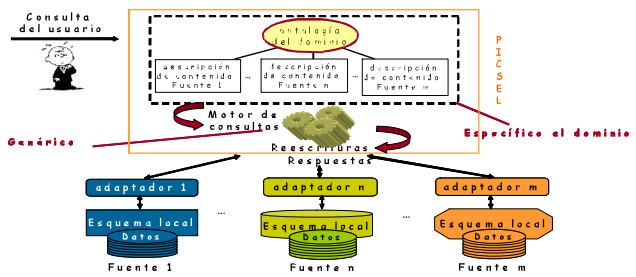


Figura. 1. Arquitectura de PICSEL

### A. El lenguaje de vistas

En PICSEL la descripción del contenido de una fuente es expresada en términos de vistas. Se trata de representar cada fuente S a partir del vocabulario VS constituido tanto de relaciones locales como de relaciones del dominio de las cuales se sabe que la fuente S provee instancias. Estas relaciones locales son llamadas vistas. La descripción del contenido de una fuente S en términos de esas vistas esta constituida por:

- Un conjunto de implicaciones lógicas de la forma $v_i(X) \rightarrow p(X)$ ligando cada vista vi a la relación del dominio p en la ontología.

- Un conjunto de restricciones de integridad y de tipo sobre las vistas, de la forma:

(a) $v \subseteq C$ donde C es una expresión de concepto, con, para cada vista v, al menos una restricción de esta forma:

(b) $l_1(\overline{X}_1),...,l_n(\overline{X}_n) \rightarrow \bot$[1] donde cada $l_i(\overline{X}_i)$ es una vista $v_i(\overline{X}_i)$ y

(c) $l_1(\overline{X}),\neg l_2(X) \rightarrow \bot$[2] donde $l_1$ y $l_2$ son vistas y $X \in \overline{X}$.

EJEMPLO: Sea la fuente de datos AirCompany que describe los vuelos, con su número, su aeropuerto de salida y de llegada. Las siguientes implicaciones expresan que esta fuente permite obtener instancias del concepto *FlightSegment* así como instancias de los roles *FlightNumberAssociate, DepartureAirportAssociate y ArrivalAirportAssociate.*

*IAirlineCompany :*

*AirlineCompany_FlightSegment(X)* $\rightarrow$ *FlightSegment (X)*

*AirlineCompany_FlightNumberAss(X,Y)* $\rightarrow$

*FlightNumberAssociate(X,Y)*

*AirlineCompany_DepartureAss(X,Y)* $\rightarrow$

*DepartureAirportAssociate(X,Y)*

*AirlineCompany_ArrivalAss(X,Y)* $\rightarrow$

*ArrivalAirportAssociate (X,Y)*

Las restricciones de integridad permiten expresar incompatibilidades. Por ejemplo, la restricción de integridad siguiente expresa que las instancias del concepto FlightSegment en la fuente AirCompany son disjuntas con respecto a las de la fuente FlightCompany.

*AirlineCompany_FlightSegment(X), FlightCompany_*

1 El constructor $\bot$ corresponde, en la notación alfanumérica, a bottom (concepto vacío).

2 El constructor ¬ corresponde, en la notación alfanumérica, a not (negación).

$FlightSegment(X) \rightarrow \bot$

Las restricciones de tipo, restringen ciertos argumentos de una vista a ser de un tipo dado. Así, las restricciones siguientes significan que las instancias de los aeropuertos, de salida (*AirlineCompany_DepartureAss*), de llegada (*AirlineCompany_ArrivalAss*) y los números de vuelo (*AirlineCompany_FlightNumberAss*) que se encuentran en la fuente AirlineCompany son concernientes a los vuelos (*FlightSegment*) que se encuentran en esa misma fuente.

$C_{AirlineCompany}$:

$AirlineCompany\_DepartureAss(X,Y), \neg AirlineCompany\_FlightSegment(X) \rightarrow \bot$

$AirlineCompany\_ArrivalAss(X,Y), \neg AirlineCompany\_FlightSegment(X) \rightarrow \bot$

$AirlineCompany\_FlightNumberAss(X,Y), \neg AirlineCompany\_FlightSegment(X) \rightarrow \bot$

Como puede apreciarse en el ejemplo, en PICSEL, los nombres de las vistas v son prefijados con el nombre de la fuente cuyo contenido es descrito.

B. El lenguaje de consultas

PICSEL permite a los usuarios hacer consultas utilizando los términos de la ontología. Luego, las fuentes serán interrogadas, una vez que los wrappers hagan la traducción de la pregunta. Las consultas son consultas conjuntivas de la forma:

$$Q(\overline{X}) \leftarrow p_1(\overline{X}_1, \overline{Y}_1, \overline{c_1}) \wedge \ldots \wedge p_n(\overline{X}_n, \overline{Y}_n, \overline{c_n})$$

Donde los $p_i$ son átomos-conceptos[3], átomos-roles[4] o átomos ordinarios[5]. Su conjunción constituye el cuerpo de la consulta y $Q(\overline{X})$ es llamada *la cabeza de la consulta*. Las variables del vector $\overline{X} = \overline{X}_1 \cup \ldots \cup \overline{X}_n$ son llamadas **variables distinguidas** de la consulta y representan las variables de las cuales el usuario desea obtener instancias, cuando él hace la consulta. Las variables del vector $\overline{Y} = \overline{Y}_1 \cup \ldots \cup \overline{Y}_n$ (variables **existenciales**) y las **constantes** $\overline{c_i}$ solo sirven para expresar restricciones sobre las variables distinguidas.

EJEMPLO: La consulta siguiente expresa la búsqueda de vuelos entre Londres y Los Angeles el 1ero de Agosto de 2008.

Q(X)←(FlightSegment)(X),(FlightNumberAssociate)(X,Y), (FlightNumber)(Y),(DepartureAirportAssociate)(X, london),

(ArrivalAirportAssociate)(X, losAngeles),

(DepartureDateTimeAssociate)(X, 01/08/2008)

En esta consulta existen:

- Una variable distinguida, X, que significa que el usuario desea obtener instancias del concepto *FlightSegment*
- Una variable existencial, Y, obligando a la segunda componente del rol *FlightNumberAssociate* a ser de tipo *FlightNumber*
- Tres constantes: *london*, *losAngeles* y 01/08/2008 que restringen respectivamente el aeropuerto de salida, el de llegada y la fecha del vuelo.

En esta consulta existen:

- Una variable distinguida, X, que significa que el usuario desea obtener instancias del concepto FlightSegment
- Una variable existencial, Y, obligando a la segunda componente del rol FlightNumberAssociate a ser de tipo FlightNumber
- Tres constantes: london, losAngeles y 01/08/2008 que restringen respectivamente el aeropuerto de salida, el de llegada y la fecha del vuelo.

## III. DE LA INTEGRACIÓN DE DATOS A LA INTEGRACIÓN DE SERVICIOS EN PICSEL

La utilización creciente de la Web ha favorecido la aparición de nuevas tecnologías de comunicación, el desarrollo del comercio electrónico y la implantación de servicios, para permitir al usuario la consulta y modificación de datos disponibles sobre la Web. Debemos tener presente esta evolución y permitir a los sistemas mediadores integrar servicios accesibles vía la Web, mientras que los trabajos efectuados hasta el momento en el dominio de la mediación, como SIMS [1], Information Manifold [9] y OBSERVER [11], entre otros, lo han hecho integrando fuentes de datos.

PICSEL ha sido concebido para integrar fuentes de datos heterogéneas. Su utilización para integrar servicios representa una modificación del contexto aplicativo y conlleva a cambios en la manera de utilizar el sistema, en la interpretación de los resultados producidos y en los razonamientos ejecutados para calcular los planes de consultas.

Por consiguiente la ontología debe ser adaptada. La ontología utilizada para integrar fuentes de datos es una descripción de conceptos y de relaciones entre conceptos del dominio. Cuando se trata de integrar servicios, la ontología debe modelar el dominio de los servicios en los que nos interesamos. Las modificaciones afectan igualmente a las vistas. Las vistas que describen el contenido de las fuentes deben ser transformadas en vistas que describan las funcionalidades de los servicios ejecutables. La formulación de las preguntas debe ser adaptada a la búsqueda de uno o varios servicios. Los planes de las

3 Un átomo-concepto es de la forma C(U) donde U es una variable y C es un nombre de concepto definido en la ontología.

4 Un átomo-role es de la forma r(U1,U2) donde U1 y U2 son variables y r es un nombre de role que aparece en definiciones de conceptos de la ontología.

5 Un átomo ordinario es de la forma p(U1, …,Un) donde (U1, …,Un) es un vector de variables y p es un predicado de aridad n cualquiera que no aparece en la ontología.

Revista Avances en Sistemas e Informática, Vol.5 No.2, Junio de 2008, Medellín, ISSN 1657-7663

consultas (o reescrituras) deben ser interpretados de manera diferente.

A. La integración de fuentes de datos en PICSEL

Cuando se integran datos, PICSEL toma como entrada la ontología del dominio, la descripción del contenido de las fuentes de datos y la consulta del usuario.

1) La ontología

La ontología utilizada por el motor de consultas de PICSEL es un esquema a base de clases tal como el obtenido por OntoMedia [6]. Ese modelo está constituido por clases que son ligadas entre ellas por relaciones de generalización/especialización (is-a) y por relaciones especificas al dominio. Además cada clase posee un conjunto de propiedades que la caracteriza. La ontología es representada en CARIN que es el lenguaje escogido para representar el conocimiento en PICSEL [10]. Allí las clases son representadas por relaciones unarias y las relaciones son representadas por medio de roles o de relaciones binarias.

La ontología descrita en CARIN es un elemento de la base de conocimiento del mediador PICSEL. Ella provee el vocabulario apropiado para formular las consultas y permite establecer una conexión entre las diferentes fuentes de datos que se van a integrar. Una experiencia de modelamiento y de representación de una ontología (en el dominio del turismo) se realizó en el proyecto PICSEL I [3],[13]. Allí la ontología se construyó de manera manual[6]. La figura 2 muestra un extracto de la ontología que ha sido simplificada, para un mejor entendimiento. Esta ontología posee dos tipos de jerarquía: una jerarquía principal, que representa todo lo que se puede vender en el dominio del turismo y que reagrupa el alojamiento y los vuelos, y unas jerarquías disjuntas que describen el conocimiento relativo a los subdominios del dominio del turismo. En la Fig. 2 solo se muestra una de éstas, la cual tiene el concepto Lugar como raíz.

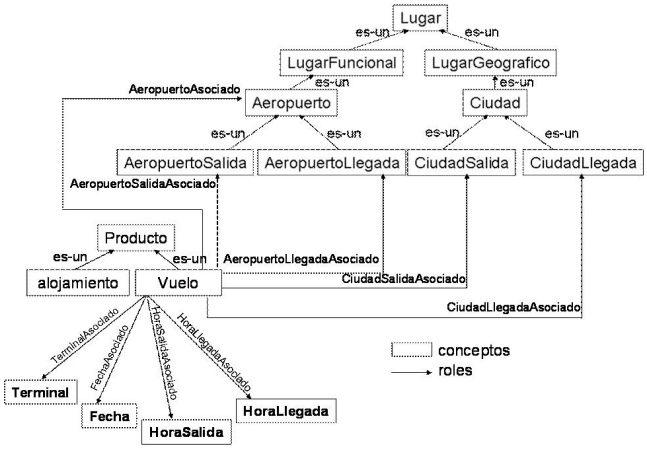


Figura. 2 Extracto de la ontología para la integración de fuentes de datos

Luego, la ontología es representada manualmente en CARIN y esto permite disponer de los siguientes conceptos y roles.

**Conceptos**: *Producto, Alojamiento, Vuelo, Terminal, Fecha, HoraSalida, HoraLlegada, Lugar, LugarFuncional, LugarGeografico, Aeropuerto, AeropuertoSalida, AeropuertoLlegada, Ciudad, CiudadSalida, CiudadLlegada.*

**Roles**[7]: *AeropuertoAsociado, AeropuertoSalidaAsociado, AeropuertoLlegadaAsociado, CiudadSalidaAsociado, CiudadLlegadaAsociado, TerminalAsociado, FechaAsociado, HoraSalidaAsociado, HoraLlegadaAsociado.*

A continuación ilustraremos con un ejemplo, utilizando un subconjunto de los conceptos y roles de esta ontología, la descripción en términos de vistas, del contenido de las fuentes de datos y la interpretación de la consulta del usuario y de los planes de consultas resultantes.

2) La descripción del contenido de las fuentes de datos

El esquema de una fuente de datos, o esquema local, representa las estructuras en las cuales, en esa fuente, se almacenan sus datos. En PICSEL la descripción del contenido de una fuente de datos se expresa en el lenguaje de vistas que presentamos anteriormente. Una vista representa las instancias que pueden ser obtenidas por el mediador a partir de esa fuente. Como mencionamos anteriormente, en PICSEL, los nombres de las vistas son prefijados con el nombre de la fuente que se está describiendo.

EJEMPLO: Supongamos la integración de dos fuentes AirFusion (AF) y Degretour (DGT) cuyos datos son representados como documentos XML. Sea el siguiente documento que contiene la programación de los vuelos de la compañía aérea AirFusion.

**Fuente AirFusion (AF)**

**a) documento XML representando los datos:**

```
<?xml version="1.0" encoding="UTF-8"?>
<AirFusion>
    <Programacion>
        <Vuelo>af2030 </Vuelo>
        <Aeropuerto>
            <Nombre>orly</Nombre>
            <Codigo>ORL</Codigo>
            <Terminal>T1</Terminal>
        </ Aeropuerto>
        < Aeropuerto>
            <Nombre>Jose Maria Cordova</Nombre>
            <Code>JMC</Code>
            <Terminal>C12</Terminal>
        </ Aeropuerto>
    </Programacion>
    …
</AirFusion>
```

6 Cabe anotar que esta ontología fue creada en francés, pero aquí se tradujo un extracto de ella, al español para un mejor entendimiento.

7 Por convención se decidió agregar la palabra "Asociado" al final de cada rol.

**b) Definición de las vistas:**

(1)AF_Vuelo(V) → Vuelo(V)
(2)AF_Terminal(T) → Terminal(T)
(3)AF_Aeropuerto(A) → Aeropuerto(A)
(4)AF_AeropuertoAs(X,A) → AeropuertoAsociado(X,A)
(5)AF_TerminalAs (X,T) → TerminalAsociado (X,T)

Esto significa que la fuente *AirFusion* permite obtener instancias de los conceptos *Vuelo, Terminal y Aeropuerto*, así como de los roles *AeropuertoAsociado y TerminalAsociado*.

Con el fin de simplificar la manera de presentar los datos extraídos de la fuente AirFusion, definimos el predicado VueloAeropuertoTerminal, que corresponde al vuelo V saliendo del aeropuerto A y cuyo terminal es *T*, con la siguiente regla:

VueloAeropuertoTerminal( *V,A, T*)←(Vuelo)( *V*), (AeropuertoAsociado)( *V,A*), (Aeropuerto)(*A*), TerminalAsociado)(*A, T*), (Terminal)( *T*)

Entonces las implicaciones lógicas (1) a (5) son completadas con:

(6)AF_Programacion(V,A,T)→ VueloAeropuertoTerminal(V,A,T)

**c) Algunos datos de la fuente AirFusion** (extensiones de la vista (6) AF_Programacion):

(A) AF_Programacion (af2030, orly,T1)

(B) AF_Programacion (af2030,JoseMariaCordoba,C12)

**d) Definición de restricciones:**

AF_AeropuertoAs(V,A) , ¬ AF_Vuelo(V) → ⊥
AF_ AeropuertoAs(V,A) , ¬ AF_Aeropuerto(A) → ⊥
AF_TerminalAs(V,T) , ¬ AF_Vuelo(V) → ⊥
AF_TerminalAs(V,T) , ¬ AF_Terminal(T) → ⊥

Estas restricciones significan que las instancias de Aeropuerto y de Terminal de la fuente AirFusion (AF) están relacionadas con los vuelos que se encuentran en esa misma fuente. Ahora consideremos otra fuente, la de la agencia de viajes Degretour (DGT).

**Fuente Degretour (DGT).**

**a1) documento XML representando los datos:**

```
<?xml version="1.0" encoding="UTF-8"?>
<AirFusion>
      <Programacion>
            <Vuelo>af2030 </Vuelo>
            <Aeropuerto>
                  <Nombre>orly</Nombre>
                  <Codigo>ORL</Codigo>
                  <Terminal>T1</Terminal>
            </ Aeropuerto>
            < Aeropuerto>
                  <Nombre>Jose Maria Cordova</Nombre>
                  <Code>JMC</Code>
                  <Terminal>C12</Terminal>
            </ Aeropuerto>
      </Programacion>
      …
</AirFusion>
```

b1) Definición de las vistas:

```
<?xml version="1.0" encoding="UTF-8"?>
<Vuelos>
      <Vuelo numero ="ac234P" >
            <Fecha>"12112008"</Fecha>
            <CiudadDeSalida>"Las Vegas"</CiudadDeSalida>
            <AeropuertoDeSalida>"Las Vegas Airport"</AeropuertoDeSalida>
            <HoraDeSalida> "12:55"</HoraDeSalida>
            <CiudadDeLlegada>"Philadelphia"</CiudadDeLlegada>
            <AeropuertoDeLlegada> "PHL Airport"</AeropuertoDeLlegada>
            <HoraDeLlegada> "20:44"</HoraDeLlegada>
      </Vuelo>
      <Vuelo numero ="af2030" >
            <Fecha>"12112008"</Fecha>
            <CiudadDeSalida>"Paris"</CiudadDeSalida>
            <AeropuertoDeSalida>"Orly"</AeropuertoDeSalida>
            <HoraDeSalida>"13:45"</HoraDeSalida>
            <CiudadDeLlegada>« Medellin"</CiudadDeLlegada>
            <AeropuertoDeLlegada>«Jose Maria Cordova"</AeropuertoDeLlegada>
            <HoraDeLlegada>"15:00"</HoraDeLlegada>
      </Vuelo>
      …
</Vuelos>
```

Estas implicaciones significan que en la fuente *Degretour* (DGT) podríamos encontrar instancias de los conceptos *Vuelo, Fecha, CiudadSalida, AeropuertoSalida, HoraSalida, CiudadLlegada, AeropuertoLlegada, HoraLlegada y de los roles FechaAsociado, CiudadSalidaAsociado, AeropuertoSalidaAsociado, HoraSalidaAsociado, CiudadLlegadaAsociado, AeropuertoLlegadaAsociado, HoraLlegadaAsociado.*

Los datos disponibles son las extensiones de las vistas que hemos definido anteriormente. De la misma manera que en la fuente AF, con el fin de simplificar, utilizamos el predicado de la ontología VuelosDegretour, construido a partir de la regla siguiente:

VuelosDegretour(V,D,VD,AD,HD,VA,AA,HA)←
(Vuelo)(V),(FechaAsociado)(V,D),
(CiudadSalidaAsociado)(V,VD),
(AeropuertoSalidaAsociado)(V,AD),
(HoraSalidaAsociado)(V,HD),
(CiudadLlegadaAsociado)(V,VA),
(AeropuertoLlegadaAsociado)(V,AA),
(HoraLlegadaAsociado)(V,HA)

Así, las implicaciones (1) a (15) son completadas con:

(16) DGT_Vuelos → VuelosDegretour

c1) Algunos datos de la fuente Degretour (extensiones de la vista (16) DGT_Vuelos):

(A1) DGT_Vuelos(“ae234P”,”12112008”,”LasVegas”, “Las Vegas Airport”,”12:55”,”Philadelphia”, “PHLAirport”, “20:44”),

(B1) DGT_Vuelos(“af2030”,”12112008”,”Paris”, “Orly”,”13:45”,”Medellin”,”Jose Maria Cordova”, “15:00”),

d1) Definición de restricciones
DGT_FechaAs(V,D) , ¬ DGT_Vuelo(V) → ⊥
DGT_FechaAs(V,D) , ¬ DGT_Fecha(D) → ⊥
DGT_CiudadDeSalidaAs(V,VD), ¬ DGT_Vuelo(V) → ⊥
DGT_CiudadDeSalidaAs (V,VD), ¬DGT_CiudadDeSalida (VD) → ⊥
DGT_AeropuertoDeSalidaAs(V, AD), ¬ DGT_Vuelo (V) → ⊥
DGT_AeropuertoDeSalidaAs(V ,AD),
¬ DGT_AeropuertoDeSalida(AD) → ⊥

Revista Avances en Sistemas e Informática, Vol.5 No.2, Junio de 2008, Medellín, ISSN 1657-7663

DGT_HoraDeSalidaAs(V ,HD), ¬ DGT_Vuelo (V) → ⊥
DGT_HoraDeSalidaAs(V ,HD),
¬ DGT_HoraDeSalida(HD) → ⊥

DGT_CiudadDeLlegadaAs(V ,VA), ¬DGT_Vuelo (V) → ⊥
DGT_CiudadDeLlegadaAs(V ,VA),
¬ DGT_CiudadDeLlegada(VA) → ⊥

DGT_AeropuertoDeLlegadaAs(V ,AA), ¬ DGT_Vuelo (V) → ⊥
DGT_AeropuertoDeLlegadaAs(V ,AA),
¬ DGT_AeropuertoDeLlegada(AA) → ⊥

DGT_HoraDeLlegadaAs (V, HA), ¬ DGT_Vuelo(V) → ⊥
DGT_HoraDeLlegadaAs (V, HA),
¬ DGT_HoraDeLlegada(HA) → ⊥

3) La consulta del usuario

La consulta del usuario se formula utilizando el lenguaje de consultas visto en la sección 2.2. Ejemplo: supongamos que un usuario se conecta al mediador con el fin de conocer los vuelos que salen de Paris hacia Medellín, el 12 de noviembre de 2008, así como los aeropuertos, las terminales de embarque y las horas de salida. Entonces él hace la siguiente consulta:

*q(V,A,T,H)← Vuelo(V) ∧ FechaAsociado(V,"12112008")*

*∧ HoraSalidaAsociado(V,H) ∧ CiudadSalidaAsociado(V,"paris") ∧ AeropuertoSalidaAsociado(V,A) ∧ VueloAeropuertoTerminal(V,A,T) ∧ CiudadLlegadaAsociado(V,"medellin")*

Donde *VueloAeropuertoTerminal* es el nuevo predicado creado a partir de las vistas definidas en la fuente *AirFusion*.

La consulta q es un ejemplo de consulta que no utiliza variables existenciales. En efecto, todas las variables del cuerpo de la consulta aparecen en el conjunto de variables de la cabeza de la misma (variables distinguidas).

4) La salida

La salida de PICSEL es un conjunto de planes de consultas que son luego traducidos por los wrappers, con el fin de obtener la respuesta a la consulta del usuario. En efecto, PICSEL no evalúa directamente las consultas, porque él solo dispone de vistas abstractas de datos que son almacenados de manera distribuida en fuentes independientes. Las vistas que posee el mediador son expresadas en función de la ontología. Ellas describen los datos que pueden ser obtenidos interrogando las fuentes, según sus esquemas locales y en el lenguaje de interrogación aceptado por cada una de ellas. Las respuestas a una consulta pueden ser calculadas por reescritura en términos de vistas. El trabajo del motor de consultas consiste en encontrar una consulta que solo utilice vistas y que según la decisión de concepción del mediador sea equivalente o implique lógicamente la consulta del usuario. Esta consulta es llamada "reescritura". El plan de consultas es entonces el conjunto de reescrituras encontradas por PICSEL equivalentes a la consulta del usuario [7]. Las respuestas a una consulta son obtenidas evaluando las reescrituras sobre las extensiones de las vistas. Un ejemplo de reescritura es el siguiente. Supongamos que entre los planes de consultas entregados por PICSEL para responder a la consulta q, encontramos la reescritura en términos de vistas $q_v$ :

$q_v$(V,A,T,H)←AF_Programacion(V,A,T) ∧ DGT_Vuelos(V, "12112008", "paris", A, H, "medellin", AA, HA)

La función del wrapper es entonces traducir la reescritura qv en el lenguaje propio de cada fuente. A continuación mostramos cómo la respuesta a la consulta q es obtenida evaluando la reescritura $q_v$ en las extensiones de las vistas (6) AF_Programacion y (16) DGT_Vuelos.

En un primer momento, el wrapper puede interrogar la fuente Degretour utilizando las constantes de la vista DGT_Vuelos de $q_v$ ("12112008","paris","medellin"). Así, se obtiene la respuesta (B1) (V= "af2030", A="orly", H="13:45", AA="Jose Maria Cordova", HA="15:00").

Luego él puede interrogar la fuente AirFusion con los valores de V y A encontrados en (B1) (V= "af2030", A="orly"). La respuesta obtenida es (A), con T="T1". La respuesta completa a la consulta q(V,A,T,H) es q("af2030","orly","T1","13:45"). Esta respuesta significa que existe un vuelo el 12 de noviembre de 2008 que sale de Paris y llega a Medellín. Se trata del vuelo af2030 que parte a las 13:45 del Terminal T1 del aeropuerto de Orly.

Con este ejemplo hemos mostrado como PICSEL integra dos fuentes de datos. La fuente Degretour permitió obtener una parte de la respuesta: el número del vuelo, el aeropuerto y la hora de salida. La fuente AirFusion permitió obtener la Terminal (T). La respuesta completa es obtenida combinando los datos extraídos de las dos fuentes. Esta combinación es muy importante, es un punto fuerte de los sistemas de integración.

B. La integración de servicios en PICSEL

Un servicio corresponde, en su forma más simple, a una búsqueda de información. Por ejemplo un usuario se puede conectar a la Web, para saber si hay cupo en un vuelo de Paris a Medellín el 15 de abril del 2008. En este caso se trata de una simple consulta de datos. Un servicio puede también corresponder a la ejecución de una funcionalidad que va a modificar los datos de la aplicación del proveedor como en el caso de la reservación de un vuelo.

Responder a la petición de un usuario, puede necesitar la ejecución de uno o varios servicios. Actualmente se encuentra en la Web una gran cantidad de servicios construidos de manera independiente los unos de los otros. Así, dos mensajes de petición de servicios con funcionalidades equivalentes, pueden tener una estructura diferente. En este artículo se consideran mensajes que utilizan protocolos de comunicación como SOAP (Simple Object Access Protocol) y HTTP (HyperText Transfer Protocol), pero cuya estructura interna y vocabulario, utilizados en la construcción de los mensajes, no obedecen a ningún estándar preestablecido. A estos mensajes se les llama "mensajes no estándares".

SOAP es uno de los protocolos más utilizados actualmente sobre la Web, él gestiona la comunicación entre el solicitante de un servicio y el proveedor del mismo. Es un mecanismo de transporte de mensajes Solicitud/Respuesta ("Requirement/ Response"). En SOAP los mensajes son escritos en XML (eXtensible Markup Language). Para establecer la comunicación, el solicitante emite un mensaje hacia el servidor del proveedor del servicio. Este mensaje contiene, por una parte, el nombre del procedimiento que dispara el servicio y por otra, el valor de los parámetros necesarios para ejecutarlo. En respuesta a la llamada efectuada por el solicitante, el proveedor ejecuta el tratamiento y retorna el valor resultante, igualmente bajo la forma de documento XML, indicando si la ejecución fue exitosa y precisando los valores resultantes.

En las siguientes subsecciones se presentan las entradas y salidas de PICSEL, en el caso de la integración servicios.

1) La ontología de servicios

En el contexto de la integración de servicios, igual que en el caso de integración de fuentes de datos, la ontología corresponde a un esquema a base de clases. Esta ontología posee los mismos tipos de relaciones que la ontología presentada en la sección III.A.1: relaciones entre clases (relaciones de generalización / especialización y relaciones especificas del dominio) y relaciones entre clases y propiedades (relaciones de caracterización). Ella es el soporte para la expresión de la solicitud de servicios y también permite conectar y combinar los diferentes servicios a los que se desea acceder. En este nuevo contexto la ontología modela los servicios relativos al dominio que nos interesa.

Conforme al protocolo de comunicación descrito anteriormente, a cada servicio está asociada una solicitud, a la cual el proveedor del servicio es capaz de responder y a la cual, corresponde un mensaje que el proveedor puede tratar. El proveedor responde a la solicitud de un usuario, tratando el mensaje que este último envía. Los mensajes de solicitud de un servicio, contienen los términos del dominio. Sus contenidos pueden ser explotados para construir la ontología, principalmente la red de relaciones entre las diferentes clases. Se considera que el nombre del mensaje es el nombre de la etiqueta raíz de los documentos XML, igualmente corresponde al nombre del servicio que se ofrece. Los proveedores pueden llamar de manera diferente los servicios que proveen y dar nombres diferentes a los mensajes, aunque se trate del mismo tipo de servicio.

Una descripción precisa del proceso de construcción de la ontología de un sistema de integración de servicios se presenta en [6]. El siguiente es un ejemplo de un extracto de una ontología para la integración de servicios. Supongamos el extracto de ontología de la Fig.3, allí los nombres de las clases busquedaDisponibilidadVuelo y busquedaHotel son nombres de servicios a los cuales se tiene acceso. La clase raíz Servicio Turístico representa el dominio de aplicación. Las relaciones entre la clase raíz y las dos clases anteriormente mencionadas son relaciones de generalización/especialización. Esta representación indica que hay dos servicios turísticos: busquedaDisponibilidadVuelo y busquedaHotel.

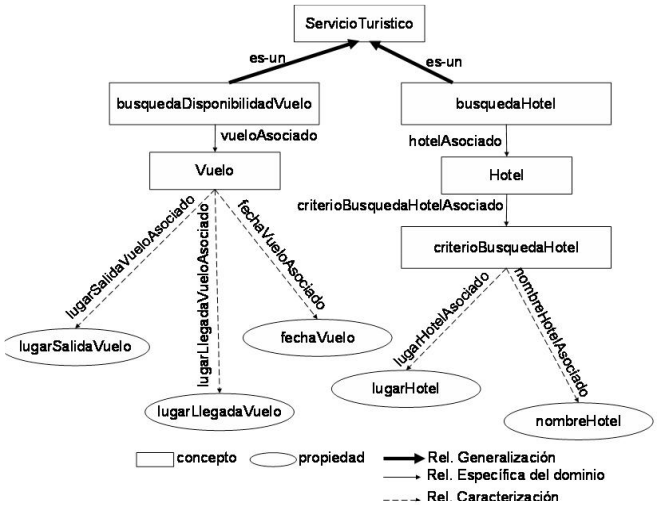


Figura 3. Extracto de una ontología de servicios turísticos

Las demás partes de la ontología describen los conceptos útiles para precisar los diferentes servicios buscados. Así, la clase busquedaDisponibilidadVuelo está ligada a la clase Vuelo por la relación específica al dominio vueloAsociado. La clase Vuelo esta ligada a las propiedades lugarSalidaVuelo, lugarLlegadaVuelo y fechaVuelo por las relaciones de caracterización lugarSalidaVueloAsociado, lugarLlegadaVueloAsociado y fechaVueloAsociado. La clase busquedaHotel está ligada a la clase Hotel por la relación especifica del domino hotelAsociado. La clase Hotel esta ligada a la clase criterioBusquedaHotel por la relación especifica del dominio criterioBusquedaHotelAsociado. La clase criterioBusquedaHotel esta ligada a las propiedades lugarHotel, nombreHotel por las relaciones de caracterización lugarHotelAsociado, nombreHotelAsociado

La traducción de esta ontología a CARIN permite obtener una representación en términos de los conceptos y roles.

**Conceptos**: ServicioTuristico, busquedaDisponibilidadVuelo, busquedaHotel, Vuelo, Hotel, criterioBusquedaHotel. Donde los tres primeros son conceptos primitivos.

**Roles**: vueloAsociado, hotelAsociado lugarSalidaVueloAsociado, lugarLlegadaVueloAsociado, fechaVueloAsociado, lugarHotelAsociado, nombreHotelAsociado y criterioBusquedaHotelAsociado.

Continuando con el ejemplo, la siguiente sección muestra cómo a partir de una ontología de servicios de este tipo, se pueden describir las funcionalidades de los servicios y solicitar su ejecución.

2) La descripción de las funcionalidades de los servicios

De la misma manera que para la descripción del contenido de las fuentes de datos, el lenguaje de vistas de PICSEL, es utilizado para describir las funcionalidades de los servicios. La descripción de las funcionalidades de un servicio está

Revista Avances en Sistemas e Informática, Vol.5 No.2, Junio de 2008, Medellín, ISSN 1657-7663

constituida por un conjunto de implicaciones lógicas que describen el servicio y por un conjunto de restricciones sobre las vistas. Como se verá en el ejemplo siguiente, describir la funcionalidad de un servicio en PICSEL, consiste primero que todo, en definir una implicación lógica, que asocia una vista, al nombre del concepto de la ontología que representa el servicio y luego definir otras implicaciones, que permiten precisarla. En el caso de la descripción de las funcionalidades de los servicios se decidió prefijar los nombres de las vistas con el nombre del proveedor del servicio asociado.

EJEMPLO: Supongamos que PICSEL integra servicios, provistos por dos proveedores:

- la agencia de viajes OTO ofrece dos servicios, los servicios de búsqueda de disponibilidad de vuelos busquedaDisponibilidadVuelo y el servicio de búsqueda de hoteles BusquedaHotel, cuyos mensajes corresponden a los documentos XML Nro. 1 y Nro. 2 que se muestran más adelante.

- Por otra parte, supongamos la compañía HOLIDAY ofrece el servicio de búsqueda de hoteles HotelFinder cuyo mensaje corresponde al documento XML Nro. 3

De la misma manera que en la integración de fuentes de datos, es necesario escribir el contenido de cada fuente, acá es necesario describir las funcionalidades de los servicios asociados a cada uno de los proveedores de servicios. Así, en nuestro ejemplo, es necesario describir la funcionalidad del servicio, de solicitud de disponibilidad de vuelos, para la agencia de viajes OTO y para la compañía HOLIDAY, así como la descripción de la funcionalidad del servicio de búsqueda de hoteles para HOLIDAY.

Proveedor OTO

a) documentos XML representando los mensajes de solicitud de los servicios

```
<?xml version="1.0" encoding="UTF-8"?>
<solicitudBusquedaDisponibilidadVuelo>
        <InformacionVuelo>
        <Salida>"..."</Salida>
        <Llegada>"..."</Llegada>
        <Fecha>"..."</Fecha>
        </InformacionVuelo>
</ solicitudBusquedaDisponibilidadVuelo>
```

Documento XML Nro.1

```
<?xml version="1.0" encoding="UTF-
<solicitudBusquedaHotel >
    <criterioBusqueda>
        <Lugar>"..."</ Lugar >
        <Nombre>"..."</ Nombr
    < criterioBusqueda >
</ solicitudBusquedaHotel >
```

Documento XML Nro.2

b) Descripción de la funcionalidad del servicio de búsqueda de disponibilidad de vuelos.

- Definición de las vistas:

(I1)OTO_solicitudBusquedaDisponibilidadVuelo(X) → busquedaDisponibilidadVuelo(X)

(I2)OTO_informacionVuelo(X) →Vuelo (X)

(I3)OTO_informacionVueloAs(X,Y) → vueloAsociado(X,Y)

(I4)OTO_salidaAs(X,Y) → lugarSalidaVueloAsociado(X,Y)

(I5)OTO_llegadaAs(X,Y) → lugarLlegadaVueloAsociado(X,Y)

(I6)OTO_fechaAs(X,Y)→fechaVueloAsociado(X,Y)

La implicación I1 expresa que el proveedor OTO es capaz de responder a la solicitud del servicio de búsqueda de disponibilidad de vuelos (busquedaDisponibilidadVuelo). Las implicaciones I2 a I6 son definidas con el fin de que el usuario pueda precisar el servicio buscado. Por ejemplo, el usuario debe indicar el lugar de salida, el lugar de llegada y la fecha del vuelo deseado. Por otra parte las vistas definidas en estas implicaciones (I2 a I6) deben ser definidas, para que le motor de consultas, tenga en cuenta estas informaciones, en el momento del cálculo de las reescrituras. Se trata de vistas asociadas, a todas las relaciones de la ontología que están implicadas en el proceso de especificación de la solicitud del servicio. Por lo tanto, es necesario definir tantas vistas como elementos compongan el mensaje de solicitud asociado al servicio ofrecido. Así, las implicaciones I2 hasta I6, expresan que el servicio de búsqueda de disponibilidad de vuelos (busquedaDisponibilidadVuelo) requiere información relativa al vuelo (vueloAsociado): lugar de salida (lugarSalidaVueloAsociado), lugar de llegada (lugarLlegadaVueloAsociado) y fecha del vuelo (fechaVueloAsociado).

Definición de las restricciones:

(R1)OTO_informacionVueloAs(X,Y), ¬ OTO_solicitudBusquedaDisponibilidadVuelo(X) → ⊥

(R2)OTO_informacionVueloAs(X,Y), ¬OTO_informacionVuelo(Y) → ⊥

(R3)OTO_salidaAs(X,Y),¬ OTO_informacionVuelo (X) → ⊥

(R4)OTO_llegadaAs(X,Y),¬ OTO_informacionVuelo(X) → ⊥

(R5)OTO_fechaAs(X,Y),¬ OTO_informacionVuelo(X) →

Las restricciones R1 y R2 se interpretan de la siguiente manera: R1 indica que, en el servidor de la agencia de viajes OTO, el primer argumento de la vista informacionVueloAs debe ser de tipo solicitudBusquedaDisponibilidadVuelo. Se trata de una solicitud de disponibilidad de vuelos al servidor que contiene los servicios ofrecidos por OTO. La restricción R2 indica que en el servidor de OTO, el segundo argumento de la vista informacionVueloAs debe ser de tipo informacionVuelo.

Cabe anotar que en CARIN no es necesario definir el tipo del segundo argumento cuando se definen las restricciones

c) Descripción de las funcionalidades del servicio de búsqueda de hotel.

- Definición de las vistas:

(I1) OTO_BuscarHotel(X) → busquedaHotel(X)

(I2) OTO_criterio(X) → criterioBusquedaHotel(X)

(I3) OTO_criterioAs(X,Y) → criterioBusquedaAsociado(X,Y)

(I4) OTO_lugarAs(X,Y) → lugarHotelAsociado (X,Y)

(I5) OTO_nombreAs(X,Y) → nomHotelAsociado(X,Y)

- Definición de las restricciones:

(C1) OTO_criterioAs(X,Y),¬ OTO_buscarHotel(X) → ⊥

(C2) OTO_criterioAs(X,Y),¬ OTO_criterio(Y) → ⊥

(C3) OTO_lugarAs(X,Y), ¬ OTO_criterio(X) → ⊥

(C4) OTO_nombreAs(X,Y), ¬ OTO_criterio(X) → ⊥

Las restricciones C1 y C2 se interpretan de la siguiente manera. C1 indica que, en la agencia de viajes OTO, el primer argumento de la vista criterioAs debe ser de tipo buscarHotel. Se trata de una solicitud de búsqueda de hoteles en OTO. La restricción C2 indica que, para OTO, el segundo argumento de la vista criterioAs debe ser de tipo criterio. La restricción C3 indica que, en OTO, el primer argumento de la vista lugarAs debe ser de tipo criterio. La restricción C4 indica que, en OTO, el primer argumento de la vista nombreAs debe ser de tipo criterio.

Proveedor HOLYDAY

A1) documento XML representando el mensaje de solicitud del servicio de búsqueda de hoteles

```
<?xml version="1.0" encoding="UTF-8"?>
<HotelFinder>
    <SearchCriterion>
        <Place>"..."</Place>
    </ SearchCriterion>
</ HotelFinder>
```

b1) Descripción de las funcionalidades del servicio de búsqueda de hoteles

- Definición de las vistas:

(I1) HOLIDAY_indagarHotel(X)→ busquedaHotel (X)

(I2) HOLIDAY_criterio (X)→ criterioBusqueda (X)

(I3) HOLIDAY_criterioAs(X,Y)→ criterioBusquedaAsociado(X,Y)

(I4) HOLIDAY_lugarAs(X,Y)→lugarHotelAsociado (X,Y)

Las implicaciones I1 a I4 expresan que el proveedor HOLIDAY es capaz de responder a la solicitud del servicio de búsqueda de hotel (busquedaHotel) y que él propone un criterio de búsqueda correspondiente a los lugares donde se encuentran los hoteles (lugarHotelAsociado).

- Definición de las restricciones

(C1)HOLIDAY_criterioAs(X,Y),¬HOLIDAY_indagaHotel(X) → ⊥

(C2)HOLIDAY_criterioAs(X,Y), ¬ HOLIDAY_criterio(Y) → ⊥

(C3)HOLIDAY_lugarAs(X,Y), ¬ HOLIDAY_criterio (X) → ⊥

La interpretación de estas restricciones se hace de manera similar a como se hizo para el proveedor OTO.

3) La demanda de un servicio

En el contexto de integración de servicios en PICSEL, una consulta corresponde a la solicitud de búsqueda de los servicios que realicen las funcionalidades deseadas. Un usuario formula una consulta, la cual puede corresponder a la solicitud de un servicio simple, como por ejemplo la disponibilidad de tiquetes de avión. En ese caso la consulta seria:

Q(X) ← busquedaDisponibilidadVuelo(X)

La solicitud del usuario puede también corresponder a un servicio complejo y para realizarlo completamente puede ser necesario ejecutar varios servicios elementales, sin embargo el usuario no tiene que formular varias consultas, en ese caso la composición del servicio solicitado debe ser descrita en el cuerpo de la consulta. Supongamos por ejemplo que un usuario desea conocer la disponibilidad de vuelos y de hoteles, esta consulta seria:

Q(X) ← busquedaDisponibilidadVuelo(X), busquedaHotel(X)

Donde X corresponde a la solicitud del usuario. Esta solicitud es referente a la búsqueda de vuelos disponibles y de hoteles. En los dos casos (solicitud de un servicio simple y de un servicio complejo) solo hay una variable distinguida "X", porque "X" representa la solicitud del usuario.

Normalmente un usuario no desea conocer la disponibilidad de vuelos y hoteles en toda su generalidad, sino que se interesa, por ejemplo, en un vuelo con unas características precisas. La consulta anterior Q(X) no representa estas características. Para ello, es necesario incluir informaciones sobre el origen, el destino y la fecha del vuelo, así como la ciudad

donde se encuentra el hotel. Por ejemplo « Paris-Atenas » el «15/11/2008 ». Así, por ejemplo, la consulta Q'(X) expresa la solicitud de disponibilidad de vuelos que vayan de Paris a Atenas el 15 de noviembre de 2008 y la búsqueda de un hotel en Atenas. Esta consulta se escribe de la siguiente forma:

Q'(X) ←

(busquedaDisponibilidadVuelo)(X), (vueloAsociado)(X,Y),
(lugarSalidaVueloAsociado)(Y, "paris"), (lugarLlegadaVueloAsociado)(Y, "atenas"),
(fechaVueloAsociado)(Y, "151108"),
(busquedaHotel)(X), (criterioBusquedaAsociado)(X,Z)
(lugarAsociado)(Z, "atenas").

Esta consulta puede interpretarse como la solicitud de un servicio de búsqueda de disponibilidad en un vuelo que tiene por origen Paris y por destino Atenas, el 15 de noviembre de 2008 y de un hotel situado en Atenas, la ciudad de destino del vuelo.

Revista Avances en Sistemas e Informática, Vol.5 No.2, Junio de 2008, Medellín, ISSN 1657-7663

4) La salida

Las respuestas a la consulta hecha al mediador PICSEL, cuando está integrando servicios, de la misma manera que cuando esta integrando fuentes de datos, son calculadas por reescrituras de la consulta, en términos de vistas.

En el contexto de la integración de servicios, los planes de consultas entregados por el motor de PICSEL, en respuesta a la solicitud de un servicio (pudiendo ser complejo), permiten identificar los proveedores de los servicios solicitados e indicar la estructura de los mensajes a enviar a cada uno de los implicados, permitiendo combinarlos para obtener respuestas completas. El siguiente es un ejemplo de reescrituras entregadas por PICSEL en respuesta a Q'(X)

```
Plan 1 :
dans la source OTO utiliser la vue solicitudBusquedaDisponibilidadVuelo(X490)
dans la source OTO utiliser la vue informacionVueloAs (X490,X491)
dans la source OTO utiliser la vue salidaAs (X491, paris)                    } M1
dans la source OTO utiliser la vue llegadaAs (X491, atenas)
dans la source OTO utiliser la vue fechaAs (X491, 151108)

dans la source OTO utiliser la vue buscarHotel (X490)
dans la source OTO utiliser la vue criterioAs (X490, X492)                   } M2
dans la source OTO utiliser la vue lugarAs(X492, atenas)

Plan 2 :
dans la source OTO utiliser la vue solicitudBusquedaDisponibilidadVuelo(X490)
dans la source OTO utiliser la vue informacionVueloAs (X490,X491)
dans la source OTO utiliser la vue salidaAs (X491, paris)                    } M3
dans la source OTO utiliser la vue llegadaAs (X491, atenas)
dans la source OTO utiliser la vue fechaAs (X491, 151108)

dans la source HOLYDAY utiliser la vue indagarHotel (X490)
dans la source HOLYDAY utiliser la vue criterioAs (X490, X492)               } M4
dans la source HOLYDAY utiliser la vue lugarAs(X492, atenas)
```

En este ejemplo, el plan 1 indica que es necesario enviar dos mensajes (M1 y M2) a la agencia OTO con el fin de responder a la consulta Q'(X). El primer mensaje (M1) concerniente a la solicitud de búsqueda de vuelos disponibles (Solicitud BusquedaDisponibilidadVuelo(X490)), que está asociado a (informacionVueloAs(X490,X491)) informaciones relativas al lugar de salida (salidaAs(X491,paris)), al lugar de llegada (llegadaAs(X491,atenas)) y a la fecha del viaje (fechaAs (X491,151108)). El segundo mensaje (M2) concerniente a la búsqueda de hoteles (BuscarHotel (X490)) que tiene un criterio de búsqueda asociado (CriterioAs (X490, X492) ), el cual es el lugar donde se encuentra ubicado el hotel (lugarAs(X492, atenas)).

El plan 2 permite interrogar al proveedor OTO con respecto a la disponibilidad de cupos en un vuelo (M3) y al proveedor HOLIDAY con respecto a la búsqueda de hotel (M4). Este plan muestra la integración de servicios, puesto que PICSEL combina dos proveedores diferentes para responder a la solicitud de un servicio complejo.

Dado que cada proveedor utiliza un vocabulario propio, los nombres son muy diversos. Se puede ver en el ejemplo anterior la heterogeneidad en los nombres de los servicios de los diferentes proveedores, en la ontología, en las vistas asociadas y en el documento XML que representa el mensaje de solicitud del servicio (ver Tabla 1).

En el ejemplo de la sección precedente, el nombre del servicio de búsqueda de disponibilidad de vuelos en OTO (busquedaDisponibilidadVuelo) es el mismo que el nombre del concepto correspondiente en la ontología. Esto es casualidad y no es el caso más corriente. En la mayoría de los casos, estos nombres son diferentes. Por ejemplo, el concepto de la ontología que corresponde al servicio de búsqueda de hoteles es busquedaHotel. Ese servicio se llama HotelFinder para el proveedor HOLIDAY.

Como se mencionó anteriormente, los proveedores pueden llamar diferentemente los servicios ofrecidos y la etiqueta raíz del documento XML correspondiente al mensaje de solicitud del servicio. Por ejemplo, el proveedor OTO declara proveer el servicio busquedqDisponibilidadVuelo y la etiqueta raíz se llama solicitudBusquedqDisponibilidadVuelo. Además la designación de los nombres de las vistas es un trabajo colaborativo entre el proveedor, quien conoce bien el servicio, y el administrador del sistema, quien conoce bien el lenguaje de vistas. Por lo tanto, los nombres de vistas asociadas a un mismo servicio pueden ser diferentes. Por ejemplo, al servicio busquedaHotel son así asociadas las vistas BuscarHotel para OTO y IndagarHotel para HOLIDAY.

**Tabla 1:** Heterogeneidad de nombres de servicios, de vistas y de etiquetas raíces

| Nombres de los servicios de los proveedores | Nombres de los servicios en la ontología | Nombres de la vista asociada | Nombre de la etiqueta raíz del documento XML representando el mensaje de solicitud de los servicios |
|---|---|---|---|
| busquedaDisponibilidadVuelo en el servidor de "OTO" | busquedaDisponibilidadVuelo | solicitudBusquedaDisponibilidadVuelo para "OTO" | solicitudBusquedaDisponibilidadVuelo en el servidor de "OTO" |
| busquedaHotel en el servidor de "OTO"<br>HotelFinder en el servidor de "HOLIDAY" | busquedaHotel | BuscarHotel, para "OTO"<br><br>IndagarHotel para "HOLIDAY" | solicitudBusquedaHotel en el servidor de "OTO"<br>HotelFinder en el servidor de "HOLIDAY" |

Existe también heterogeneidad a nivel de la estructura de los mensajes. En nuestro ejemplo, los documentos XML Nro. 2 y Nro. 3 que representan la demanda del servicio de búsqueda de hoteles, provistos respectivamente por OTO y por HOLIDAY, no tienen la misma estructura. Para OTO, el criterio de búsqueda de un hotel esta compuesto por un lugar y por un nombre. En cambio para HOLIDAY, ese criterio no contiene sino el lugar donde esta ubicado el hotel. Se podría incluso imaginar que en el caso de OTO, el lugar sea descompuesto en: nombre de calle, código postal y ciudad, mientras que en el caso de HOLIDAY, esto corresponde a una cadena de caracteres. Además, los nombres de las etiquetas del documento XML que definen el mensaje de solicitud del servicio de búsqueda de hoteles son en español para OTO y en ingles para HOLIDAY.

Frente a esta heterogeneidad, algunos organismos han propuesto estándares para la construcción de mensajes utilizados en transacciones comerciales.

## IV. CONCLUSIÓNES Y TRABAJO FUTURO

Utilizar sistemas mediadores para integrar servicios implica eliminar ciertos obstáculos. Uno de ellos tiene que ver con

la construcción de la ontología como soporte de servicios múltiples y heterogéneos. La construcción de ontologías es central en el desarrollo de sistemas mediadores. La construcción manual de una ontología, inclusive asistida por herramientas computarizadas, es un trabajo de modelamiento largo y difícil. Por ello, es necesario buscar soluciones que permitan automatizar su construcción. Igualmente, debe ser posible integrar gran cantidad de servicios, pues, dado el dinamismo de la Web, ellos aparecen y desaparecen rápidamente, entonces se debe facilitar la descripción de las funcionalidades ofrecidas por cada uno de ellos. Una manera, seria automatizar también la construcción de las vistas sobre los servicios. Y hacer la construcción de la ontología independiente de los servicios que se deseen integrar.

En este artículo se presentó, mediante un ejemplo aplicado al mediador PICSEL, lo que significa el paso de la integración de fuentes de datos heterogéneas a la integración de servicios igualmente heterogéneos, en los sistemas mediadores. Se utilizaron, el lenguaje de vistas y el lenguaje de consultas de PICSEL, para mostrar las diferentes interpretaciones, tanto a nivel de entradas al sistema, como de salidas, en cada uno de los casos de integración. Se mostró también cómo la integración de servicios en PICSEL, permite combinar servicios de diferentes proveedores para responder a la solicitud de servicios complejos.

Se identificaron diferentes tipos de heterogeneidad, que dificultan verdaderamente la automatización de la construcción de los sistemas mediadores, particularmente la base de conocimiento de PICSEL, a la cual pertenecen, la ontología de servicios y la descripción de las funcionalidades de los servicios que se van a integrar.

Frente a esta heterogeneidad, que dificulta además la comunicación entre los proveedores de servicios, en ciertos dominios del comercio electrónico, se están conformando organismos que ayudan en el proceso de estandarización. Estos organismos agrupan, por sectores productivos, a los expertos de cada dominio, con el propósito de definir un vocabulario y una estructura estándar, para la descripción de transacciones comerciales propias de cada dominio.

Un trabajo futuro bastante interesante, consiste en integrar servicios estándares, en el contexto de la mediación y analizar sus implicaciones, principalmente a nivel de automatización de la base de conocimiento de los sistemas mediadores.

## REFERENCIAS

[1] Arens Y., Knoblock C.A., and Shen W. Query reformulation for dynamic information integration. Journal of Intelligent Information Systems, Vol. 6, pp. 99-130, 1996.

[2] Chawathe S., Garcia-Molina H., Hammer J., Ireland K., Papakonstantinou Y., Ullman J. D., Widom J. The TSIMMIS project: Integration of heterogeneous information sources. In 16th Meeting of the Information Processing Society of Japan, p. 7-18, Tokyo, 1994.

[3] Gagliardi H., Reynaud C., Safar B., La conception d'une interface de serveur d'information basée sur une terminologie CARIN. 12èmes Journées Francophones d'Ingénierie des Connaissances, IC'01, Grenoble, 25-27 Junio 2001.

[4] Gardarin G., Dragan F. Yeh L. P2P Semantic Mediation of Web Sources. ICEIS 2006 - Proceedings of the Eighth International Conference on Enterprise Information Systems: Databases and Information Systems Integration, Paphos, Cyprus, Mayo 23-27, 2006

[5] Genesereth M. R., Keller A. M., Duschka O. M., Infomaster: an information integration system. In Joan M. Peckman, editor, Proceedings, ACM SIGMOD International Conference on Management of Data, SIGMOD 1997: May 13-15, 1997, Tucson, Arizona, USA, volume 26(2) of SIGMOD Record (ACM Special Interest Group on Management of Data), p. 539-542, New-York, NY 10036, USA. ACM Press, 1997

[6] Giraldo, Gloria Lucia. Construction automatisée de l'ontologie de systèmes médiateurs: application à des systèmes intégrant des services standards accessibles via le Web, Tésis doctoral, Université Paris Sud XI, Orsay, Francia, 2005.

[7] Goasdoue F. Réécriture de requêtes en termes de vues dans CARIN et intégration d'informations. Université de Paris XI – Orsay, noviembre 2001. Tesis doctoral, 2001.

[8] Goasdoue, F., Lattes, V., Rousset, M-C. The use of CARIN language and algorithms for information integration: the PICSEL system. In the International Journal on Cooperative Information Systems, 2000.

[9] Kirk T., Levy A. Y., Sagiv Y., Srivastava D. The Information Manifold. In C. Knoblock and A. Levy, editors, Information Gathering from Heterogeneous, Distributed Environments, AAAI Spring Symposium Series, Stanford University, Stanford, California, 1995.

[10] Levy A., Rousset M-C., Combining Horn rules and description logics in CARIN. Artificial Intelligence, 104: 165-209, 1998.

[11] Mena E., Illarramendi A., Kashyap V. et Sheth A. P. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. International Journal on Distributed and Parallel Databases, Vol. 8, No. 2, p. 223-271, Abril 2000.

[12] Reynaud Chantal, Giraldo Gloria. An Application of the Mediator Approach to Services over the Web, Special track "Data Integration in Engineering", Proceedings of CE'2003, p. 209-216, 26-30 Julio, Portugal.

[13] Reynaud Chantal, Safar Brigitte. Representation of Ontologies for Information Integration, EKAW'02, International Conference on Knowledge Engineering and Knowledge Management. 1-4 Octubre 2002, Siguenza, España

[14] Rousset M.-C., Bidault A., Froidevaux C., Gagliardi H., Goasdoue F., Reynaud Ch., Safar B., Construction de médiateurs pour intégrer des sources d'information multiples et hétérogènes : le projet PICSEL, Revue I3, Volumen 2, Número 1, p 9-59, 2002.

Revista Avances en Sistemas e Informática, Vol.5 No.2, Junio de 2008, Medellín, ISSN 1657-7663

Gloria Lucía Giraldo Gómez. Ingeniera de Sistemas de la Universidad de Antioquia, Colombia. Especialista en Ciencias Electrónicas e Informática de la Universidad de Antioquia (1997). Magíster en Teoría e Ingeniería de bases de datos de la Universidad Paris I Panteón – Sorbona, Francia (2000). Doctora en Informática de la Universidad Paris XI, Orsay Francia (2005). Actualmente se desempeña como profesora Asistente en la Escuela de Sistemas de la Facultad de Minas, de la Universidad Nacional de Colombia, sede Medellín.

Urrego Giraldo. Ingeniero Civil de la Universidad Nacional de Colombia, sede Medellín. Magíster en Informática Aplicada de la Universidad de Karlsruhe, Alemania. Magíster en Teoría e Ingeniería de bases de datos de la Universidad Paris I Panteón – Sorbona, Francia (2000). Doctor en Informática de la Universidad Paris I Panteón – Sorbona, Francia (2005). Actualmente es profesor Titular en la Facultad de Ingeniería, Departamento de Ingeniería de Sistemas de la Universidad de Antioquia. Colombia.