INFORMACIÓN SEMÁNTICA PARA LA INTEGRACIÓN DE SERVICIOS DE DATOS SOBRE LA WEB

Autores: J.F Aldana Montes
jfam@lcc.uma.es

M.M. Roldán García
mmar@lcc.uma.es

A.C. Gómez Lora
cesar@lcc.uma.es
E.T.S. de Ingeniería Informática. Universidad de Málaga. Campus de Teatinos.Málaga (España)

Resumen: En este articulo presentamos un Framework para la integración y consulta on-line de Fuentes de Datos heterogéneas en la Web, que permite construir servicios electrónicos de datos. Este Framework soporta varios tipos de interrelación entre los servicios que se construyen. Utilizamos esquemas XML y RDF para definir los distintos esquemas que representan la información sobre la semántica del servicio.

Palabras Clave: Servicios; Consultas Web; Integración Semántica;

1 Introducción

Estamos presenciando un rápido incremento de las fuentes de información estructuradas que están disponibles on-line, especialmente en la Web. Actualmente, la interacción con estas grandes colecciones de datos se hace manualmente. El usuario debe considerar las distintas fuentes de datos disponibles, decidir a cuales acceder, interactuar con cada una individualmente y combinar manualmente los resultados obtenidos. Esto es problemático: existen numerosas fuentes que varían en el tipo de información que contienen y en la interfaz que presentan al usuario. Algunas fuentes contienen documentos de texto y soportan modelos sencillos de consultas, donde éstas son simplemente una lista de palabras clave. Otras fuentes contienen datos más estructurados y suministran interfaces de consultas al estilo de los lenguajes de consulta de bases de datos...

Nuestro objetivo es usar la información almacenada en esas fuentes de datos para realizar consultas complejas, y suministrar una interfaz uniforme para ellas. En particular, el usuario debería poder expresar qué quiere, y el sistema debería encontrar las fuentes relevantes y obtener los resultados, posiblemente, combinando datos de fuentes diferentes.

Hemos diseñado e implementado un Framework para construir herramientas de integración y consultas de fuentes de datos heterogéneas. Instancias de éste podrían ser; consultas sobre un cubo de datos de una base de datos multidimensional, construcción de un lenguaje gráfico de consultas, un Browser inteligente, un campus virtual, una aplicación para consultar datos bursátiles, etc.

Nuestro sistema define, sobre porciones de la Web, esquemas externo, conceptual y físico, implementando e-servicios, que pueden resolver tareas

muy diferentes. Utilizamos XML-QL para realizar las consultas y XML para integrar la información recuperada, con lo que nos abstraemos del lenguaje de consulta propio de cada fuente. Un mismo Servicio puede integrar diferentes tipos de fuentes de datos con lenguajes de consulta diferentes, por ejemplo, bases de datos relacionales, páginas web, servicios de noticias, e-servicios, etc. Los Wrappers traducen las consultas XML-QL al lenguaje específico de cada fuente de datos.

Nuestro Framework permite soportar nuevos tipos de interrelación entre los esquemas de los diferentes Servicios para resolver consultas complejas: *Delegación entre Servicios*, *Cooperación entre Servicios* e *Integración de Servicios.*

2 Arquitectura del Framework

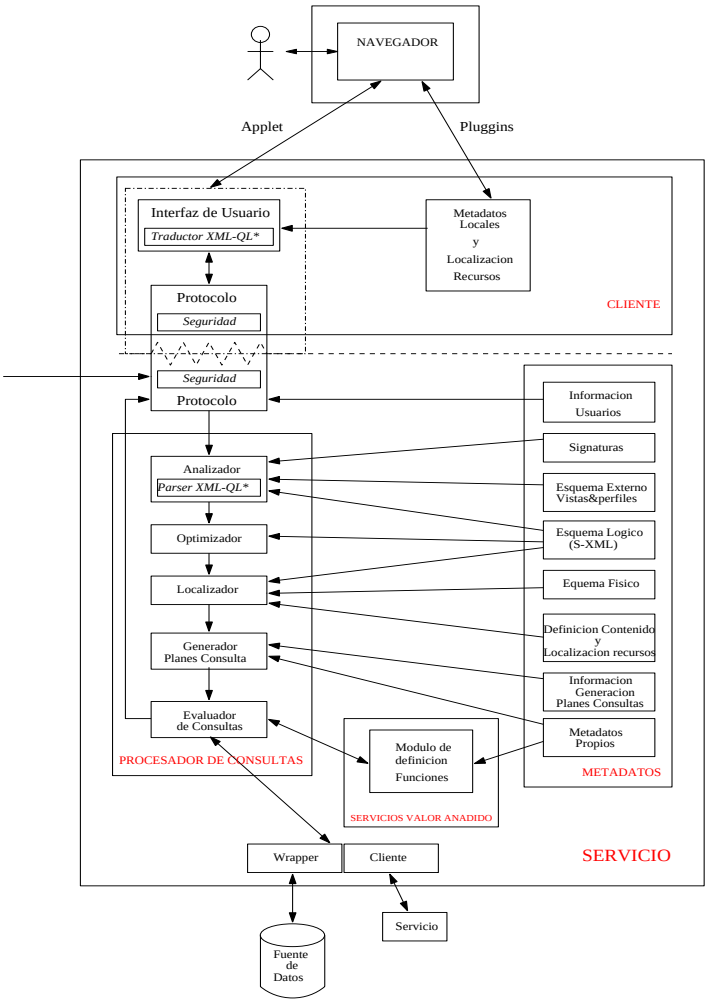

Figura 2.1 Arquitectura del Framework

2.1 Arquitectura

La figura 2.1 muestra los componentes del Framework y las relaciones entre ellos.
Las consultas llegan al Servicio en XML-QL extendido con las funciones especiales que implementa dicho servicio. Usaremos la notación XML-QL* para expresar que la consulta no está formulada en XML-QL puro. Utilizaremos los Wrappers para traducir nuestras consultas XML-QL al lenguaje concreto de cada fuente de datos.

El cliente es el punto de entrada de un Servicio. Los eventos del interfaz de usuario (UI) son traducidos a consultas (posiblemente preconstruidas) de forma especifica para cada servicio. En los metadatos locales se puede almacenar información relativa al usuario que sea relevante a la hora de realizar una consulta (configuraciones propias del usuario, *favoritos,* vistas que el usuario tiene de la Web, etc.) La idea fundamental es realizar consultas lo más específicas posibles (aprovechando para ello incluso la información que está implícita en el comportamiento o las preferencias del usuario) con la finalidad de evitar trabajo inútil al Servicio.

El Procesador de Consultas (QP) es el que se encarga de evaluar las consultas que le llegan de un Cliente. El analizador de consultas realiza un análisis sintáctico, mediante un parser XML, y un análisis semántico, usando el esquema externo, de la consulta. En el caso de que un Servicio integre otros servicios este componente es el encargado de hacer una descomposición de la consulta identificando las funciones que son locales al Servicio, y que por tanto están definidas en el componente de VAS y cuales pertenecen a otros servicios.

El optimizador de consultas recibe una consulta y procede a optimizarla. Una vez que tenemos la consulta optimizada hay que plantear el hecho de qué servicio va a resolverla. Esto lo resuelve el Localizador, fragmentando la consulta en subconsultas en función de las diversas fuentes de datos que se tienen. Además, es el encargado de localizar físicamente la fuente de datos (o servicio) con el que hay que comunicarse. También localiza los servicios que ejecutan las funciones que el Analizador no haya identificado como propias del Servicio.

El generador de planes de consulta optimiza la consulta basándose en información adicional de las diversas fuentes de datos, gestionando toda la información estadística, como tiempos de respuesta, fiabilidad del sistema, etc.
Finalmente, el Evaluador de Consultas es el que evalúa la consulta, llevando todo el control del proceso de evaluación.

2.2 Metadatos

En los metadatos incluimos toda aquella información relevante para el procesamiento de las consultas, la configuración del Servicio, la definición de los perfiles de usuario, así como las estadísticas, las características de la máquina, los recursos que utiliza, el estado de la misma, las ontologías, etc.

Los metadatos que pueden existir se clasifican en las siguientes categorías:

Información sobre los Usuarios (esquema RDF): Almacena la información sobre los usuarios relativa a la seguridad, como por ejemplo identificaciones, privilegios relativos al Servicio (los privilegios relativos a los datos se almacenan en el Esquema Externo), etc. También almacena los parámetros de configuración relativos a la delegación de consultas y/o la cooperación entre servicios en el caso de que ésta se implemente.

Signaturas (esquema RDF): Almacena las signaturas de las funciones que el servicio implementa. Puede contener anotaciones semánticas de la función, además de la signatura. Es, por lo tanto, fundamental tanto para la cooperación como para la integración de servicios, ya que una instancia que integra otros servicios debe conocer las funciones que éstos implementan.

Esquema Externo (consultas XML-QL y esquemas XML): El esquema externo es la forma en la que se define el modo en que los usuarios ven el sistema, es decir, no son más que descripciones de las vistas que los distintos usuarios tienen de los datos que el sistema suministra. La representación de estos Esquemas Externos se hará mediante vistas XML y dependerá del servicio concreto.

Esquema Lógico (esquema XML): Es una representación lógica de los datos del sistema que está expresada mediante un esquema XML. Este esquema lógico presenta las entidades y las relaciones entre ellas y las restricciones de integridad semántica que son propias del Servicio. El esquema lógico puede integrar semánticamente las fuentes de datos y/o a otros Servicios.

Esquema Físico (esquema RDF): Es la representación de la información física que se requiere. Entre la información que se debe almacenar en este esquema deberá estar la referente a los servicios y fuentes de datos con los que interactúa así como las direcciones físicas de las máquinas en que se encuentra. Otra información que deberá tener es la referente a la fragmentación de los datos, la replicación de los mismos, etc.

Definición de Contenido y Localización de Recursos (esquemas RDF): Almacena la información relativa al servicio que proporciona el propio servicio. Es fundamental a la hora de la cooperación de servicios (uso, delegación, etc.), ya que a través de una consulta sobre él un implementador de una instancia puede conocer cuales son los contenidos y funciones que ofrece el servicio, es decir, que es lo que éste hace. En este componente se almacenan también las referencias a otros servicios del mismo campo semántico a los que puede enviar/reenviar la consulta para intentar resolverla en el caso de que se implemente la cooperación entre servicios y/o la delegación de consultas/funciones.

Información para la Generación de Planes de Consultas (esquema RDF): Estos datos serán los referentes a tiempos de respuestas, estadísticas, número de referencias, TMR (Tiempos Medios de Respuesta), etc., es decir, toda aquella información principalmente estadística que puede usarse para optimizar los accesos a las fuentes de datos.

Metadatos Propios: Aquí se almacena todo lo que el implementador del servicio considere necesario para el funcionamiento del mismo y/o para la

implementación y ejecución de sus funciones. La información que se almacena y los valores de estos parámetros dependerán del servicio concreto. Esta información específica del Servicio podría aprovecharse en todas las fases del proceso de optimización de la consulta tanto en la optimización lógica como en la optimización semántica como en la optimización física.

El esquema lógico es el que define la semántica del servicio, por tanto, es fundamental a la hora de implementar cualquiera de los modelos de interrelación entre servicios que se definen en el apartado siguiente. En los metadatos se almacena toda la información necesaria para implementar dichas interrelaciones. El implementador de un servicio tiene que dotarlo de la semántica que desee construyendo este esquema. La integración de servicios se realiza a nivel de esquemas, integrando en el esquema lógico del servicio los esquemas lógicos de los servicios que integra.

3 Modelos de Interrelación de Servicios

Los Servicios se pueden combinar entre sí para resolver un problema concreto. Según se combinen los servicios establecemos la siguiente clasificación:

El Servicio recibe una consulta que no es capaz de resolver y decide enviarla a otro Servicio para que éste intente resolverla. Llamamos a este proceso *Delegación de Consultas entre Servicios*.

El Servicio recibe una consulta de la que puede resolver solo una parte y decide enviar la parte que no puede resolver a otro Servicio para que éste intente resolverla. Llamamos a éste proceso *Cooperación entre Servicios*.

Varios servicios se integran semánticamente para resolver un problema concreto integrando sus esquemas lógicos. Llamamos a este proceso *Integración de Servicios*.

Nuestras consultas llevan un identificador de Consulta (qid) que es utilizado para controlar los procesos de Delegación, Cooperación e Integración de Servicios.

3.1 Cooperación y Delegación entre Servicios

Si durante el funcionamiento normal de un servicio, este recibe una consulta que no puede resolver total o parcialmente, nuestro modelo soporta la posibilidad de que el servicio envíe la consulta o partes de ella a otros servicios que sí puedan resolverla. Opcionalmente el implementador puede almacenar en los metadatos todas las referencias a servicios de su mismo campo semántico, y además, puede priorizarlos, o en el peor de los casos, a un servicio de localización de recursos/servicios.

Si un Servicio recibe una consulta que no puede resolver y decide enviar la consulta completa a otro Servicio de su mismo campo semántico hablamos de *Delegación de Consultas*. El proceso de Delegación se puede activar por distintos motivos:

El Servicio puede resolver la consulta, pero está muy cargado y le resulta imposible hacerlo.

El Servicio no posee información suficiente sobre dónde están almacenados los datos necesarios para resolver la consulta.

El Servicio puede resolver sólo un porcentaje pequeño de la consulta y decide no gastar recursos en intentar resolverla.

Si un Servicio recibe una consulta de la cual sólo puede resolver un porcentaje determinado y decide resolver lo que él pueda de la consulta y enviar el resto a otro Servicio hablamos de *Cooperación de Servicios*.

Los parámetros de configuración relativos a la Delegación/Cooperación se almacenan en los metadatos tales como:

-Un umbral para activar el proceso. Si este umbral no se supera (por ejemplo, porque la porción de la consulta que no se puede resolver es muy elevada y eso nos indica que i) posiblemente no vaya dirigida a ese Servicio y le ha llegado por error o ii) que no merece la pena gastar recursos en intentar resolverla), entonces los componentes del Procesador de Consultas ignoran la consulta y se envía un mensaje de error al cliente, ya que la consulta no se puede resolver. En caso de superar el umbral, el Servicio decide si enviar la consulta completa o parte de la consulta a otro Servicio.

-El número de referencias a servicios del campo semántico que se adjuntan a la consulta

-El número de veces que se puede activar el proceso de Delegación/Cooperación, que no es más que un contador. Con esto evitamos que la consulta o subconsulta entre en un ciclo, siendo enviada de un Servicio a otro infinitamente.

En el caso de que el Servicio decida enviar a otro Servicio una consulta (completa o parcialmente) que es incapaz de resolver por si mismo deberá aportar a la consulta toda la información semántica referente a dicho servicio para el caso en el que haya que enviar la consulta a un servicio de localización de recursos.

El proceso de Delegación/Cooperación termina en el momento en que:

-Un Servicio detecta que la consulta o subconsulta fue enviada por él mismo. Esto puede ocurrir porque cuando un servicio recibe una consulta o una subconsulta de otro servio comienza a evaluarla como una consulta normal, por lo que, si lo soporta, puede activar de nuevo el proceso de cooperación/delegación. Mediante el qid el Servicio detecta que la consulta fue enviada por él y decide si enviar la consulta a otro servicio (si es que existe) o si considera que la consulta no se puede resolver.

-El parámetro que controla el número de veces que puede activarse el proceso llega a cero.

Una vez que se activa el proceso de Delegación/Cooperación, el proceso de evaluación de la consulta o subconsulta es el siguiente:

Caso 1: Delegación

La consulta llega al Evaluador de Consultas y éste procederá a enviar la consulta que él no puede resolver al servicio que ocupe el primer lugar en la lista de posibles servicios que podrían resolver la consulta. Si el primer servicio de la lista no resuelve la consulta, la enviará al siguiente de la lista y así sucesivamente. Si ninguno de ellos resuelve la consulta, entonces el Evaluador de Consultas enviará al cliente un mensaje de error, ya que es imposible resolver la consulta. En el caso de que alguno de los servicios resuelva la consulta, se enviará el resultado al cliente.

Caso 2: Cooperación

La consulta llega al Evaluador de Consultas y éste envía la subconsulta que no puede resolver al primer servicio de la lista de posibles servicios que podrían resolver la subconsulta, encolando el resto de la consulta para evitar operaciones costosas hasta que no estemos seguros de poder resolver la consulta completa. El evaluador tendrá una cola de consultas pendientes de ser evaluadas. La subconsulta se envía a todos los servicios de la lista hasta que uno de ellos la resuelva. Entonces se repite el proceso con la siguiente subconsulta etiquetada como no resoluble. Si ninguno de ellos la resuelve, se desencola el resto de la consulta y se desecha, ya que la consulta no se puede resolver. En el caso de que todas estas subconsultas se resuelvan por otros servicios, el evaluador desencola la consulta correspondiente y continua el proceso de evaluación.

Podría imaginarse otro nivel de cooperación/delegación en el que los Servicios fueran capaces de consultar la semántica de otros servicios localizados e interpretarla (signaturas, etc.) pero eso va mucho más allá de nuestros propósitos actuales, aunque lo consideramos como un posible trabajo futuro.

3.2 Integración de Servicios

El implementador que vaya a realizar una integración de servicios para resolver un problema concreto, debe conocer la semántica de dichos servicios, que viene dada en sus esquemas lógicos, e integrar manualmente dichos esquemas (la integración se produce a nivel de esquemas) dotando al servicio de la semántica que él decida, y las funciones que implementan cada uno de ellos. (no consideramos actualmente el problema de la integración dinámica de servicios). En todo momento asumimos que los servicios se integran al implementar una de las distintas agregaciones de servicios posibles.

Nuestro modelo soporta varios tipos de integración:

Si los Servicios que se integran no son homogéneos, es decir, realizan tareas diferentes, hablamos de *Integración Vertical de Servicios* (VI).

Si los Servicios que se integran son homogéneos, es decir, realizan la misma tarea, hablamos de *Integración Horizontal de Servicios* (HI).

Si los Servicios que se integran lo hacen tanto Vertical como Horizontalmente, entonces hablamos de *Integración Híbrida de Servicios* (HII).

La nomenclatura utilizada para cada uno de los tipos de Integración de Servicios corresponde a cómo se integran los esquemas de cada uno de los Servicios. Para describirlos utilizaremos el siguiente ejemplo.

Supongamos un servicio de una agencia de viajes que realiza reservas de vacaciones. El servicio busca información sobre vuelos, hoteles, restaurantes y autobuses y realiza las reservas oportunas.

El Servicio necesita realizar una secuencia de procesos, buscar un vuelo, un hotel, restaurantes y autobuses para los traslados durante las vacaciones, para realizar la tarea dentro de una cadena de valor añadido. Para ello, el Servicio integra un Servicio de búsqueda de vuelos que le proporciona información sobre los vuelos disponibles, un Servicio de búsqueda de hoteles y otros tipos de alojamiento que reservará las noches de alojamiento una vez fijado el vuelo, un Servicio de búsqueda de

restaurantes dónde comer en esos días y un Servicio de reserva de autobuses y de reservas de rutas turísticas para las excursiones. El Servicio de la Agencia de Viajes debe pasar información a los distintos Servicios que integra para que todas las reservas se hagan en las mismas fechas. A esto es lo que llamamos Integración Vertical de Servicios.

El Servicio de búsqueda de vuelos, por ejemplo, puede integrar varios Servicios de búsqueda de vuelos que realicen búsquedas para compañías aéreas concretas. A esto es lo que llamamos Integración Horizontal de Servicios.

Una Integración Híbrida de Servicios será aquella en la que hay tanto Integración Horizontal como Integración Vertical. En nuestro ejemplo, el Servicio de reservas de la agencia de viajes integrara los Servicios de búsqueda de Hoteles, Vuelos, Restaurantes y autobuses Verticalmente, mientras a su vez, el Servicio de reserva de vuelos integra Horizontalmente otros servicios de reserva de vuelos. La figura 3.1 muestra un ejemplo de Integración Híbrida.

Es importante destacar que los servicios conservan en todo momento su autonomía, pudiendo, aunque estén integrados con otros servicios, recibir consultas desde su IU, delegar consultas (o porciones de ellas) o cooperar con otros servicios, o participar en múltiples integraciones, lo que puede producir integraciones cíclicas.

La integración cíclica no es un problema trivial, y se hace necesario establecer un mecanismo que la detecte y que permita acotar el proceso de evaluación para evitar caer en ciclos infinitos. De nuevo mediante el qid el servicio detecta que la consulta fue enviada por él mismo. Es el implementador del servicio el que debe tomar cualquier tipo de decisión que considere oportuna para el buen funcionamiento del mismo.

Podrían existir otras formas de Integración de Servicios que dependen de cómo se organicen las cadenas de valor añadido en la empresa y de cómo se establezcan las relaciones con el consumidor ó con otras empresas (proveedores, socios, filiales, consumidores, etc...), pero su estudio va más allá del ámbito de este articulo aunque, en principio, pensamos que son combinaciones de los descritos anteriormente.

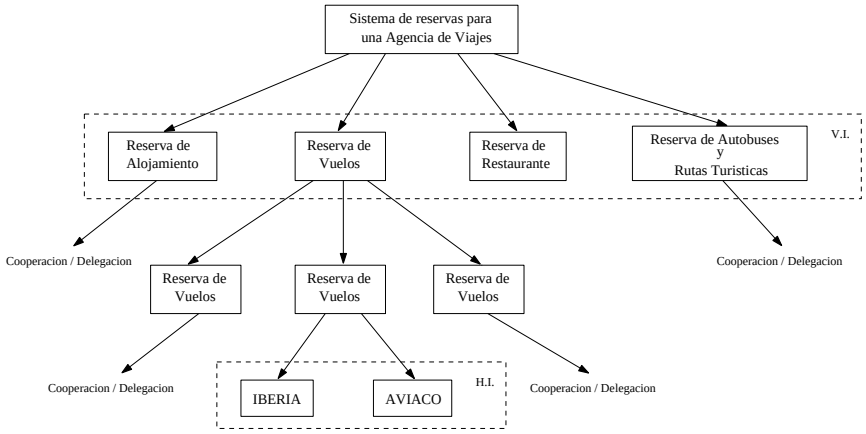

Figura 3.1: Integración híbrida de servicios

4 Conclusiones y Trabajos Futuros

Hemos diseñado la arquitectura de un Framework para consultar e integrar fuentes de datos heterogéneas, que permite la construcción de servicios electrónicos de datos de una manera fácil y sencilla, solucionando los problemas que actualmente nos encontramos al acceder a este tipo de fuentes de datos disponibles en la web, como la integración semántica, la agregación, la realización de consultas complejas o el aportar valor añadido a los datos recuperados. La arquitectura establece una serie de contenedores de metadatos que almacenan información de todo tipo necesaria para el procesamiento de las consultas, como los esquemas externo, lógico y físico de los datos, a partir de los cuales definimos un nuevo modelo de interrelación entre servicios que nos ofrece la posibilidad de ver la Web como un conjunto de pequeños servicios que cooperan entre sí mas que como un gran servicio o un conjunto de grandes servicios, cambiando la concepción de la Web que se tiene hasta el momento. La búsqueda de nuevas formas de interrelación entre servicios y de consultar la semántica de los mismos es un campo en el que continuar trabajando. Por otro lado, además de aplicar el Framework desarrollado a dominios de aplicación diferentes queda mucho trabajo por hacer en el ámbito de la optimización y evaluación de consultas sobre fuentes de datos distribuidas e integradas en uno o varios de estos servicios y en el de delegación, cooperación y localización de servicios.

5 Referencias

[1] ABITEBOUL, Serge. BUNEMAN, Peter. SUCIU, Dan. Data on the Web From Relations to Semistructured Data and XML. San Francisco, California: Morgan Kaufmann Publishers. 2000.

[2] Extensible Markup Language (XML). Version 1.0 (Second Edition). http://www.w3.org/TR/2000/REC-xml-20001006

[3] FAYAD, Mohamed E. SCHIMIDT, Douglas. Building Application Frameworks. Object-Oriented Foundations on Framework Design. 1999.

[4] FAYAD, Mohamed E. and SCHIMIDT, Douglas. Object Oriented Application Frameworks. *Communication of the ACM*. October 1997.

[5] GRAVANO, Luis. PAPAKONSTANTINOU, Yannis. Mediating and Metasearching on the internet**.** *Bulletin of the technical Committee on Data Engineering. IEEE Computer Society*. June. 1998.

[6] HARTEY, Dan and EDWARDS, Jeri. Object Frameworks: An Overview. *The Essential Distributed Objects Survival Guide*, Wiley, 1996. Chapter 12 **.**

[7] LEVY, Along. RAJARAMAN, Anand. ORDILLE, Joann. Querying heterogeneous Information Sources Using Source Descriptions. *Proceedings of the 22nd VLDB Conference*. Bombay. India. 1996

[8] SAHUGET, Arnaud. AZAVANT. Building light-weight wrappers for legacy Web data-sources using W4F. *International Conference on Very Large Databases (VLDB).* 1999.

[9] SAHUGET, Arnaud. AZAVANT, Fabien. Taming the Web with "minute-made" wrappers. U*npublished* (1999).

[10] SAHUGET, Arnaud. AZAVANT**.** Web Ecology: Recycling HTML pages as XML documents using W4F. *WebDB'99.* 1999

[11] XML Query Requirements. W3C Working Draft 15 August 2000. http://www.w3.org/TR/2000/WD-xmlquery-req-20000815

[12] XML Schema Definition Language: W3C XML Schema Working Group and Schema Specifications. http://www.w3.org/XML/Schema