

Jacinta, primeros pasos hacia la interacción entre personas y Servicios Web Semánticos^{*}

Mariano Rico, Pablo Castells

NETworked Semantics (NETS) Team

Departamento de Ingeniería Informática

Escuela Politécnica Superior de la Universidad Autónoma de Madrid

Ciudad Universitaria de Cantoblanco, 28049 Madrid

{Mariano.Rico, Pablo.Castells}@uam.es

Resumen

Conseguir que agentes inteligentes interactúen entre sí y con los Servicios Web es una labor que está requiriendo el trabajo de muchos investigadores en todo el mundo. Sin embargo, hay un detalle al que todavía no se le ha prestado suficiente atención: la interacción con un usuario humano. Aunque todos tenemos la idea de que la Web Semántica es una Web para agentes software, éstos obedecerán muchas veces las órdenes directas de usuarios humanos, por lo que tendrán que saber interactuar con ellos. Jacinta es el precursor de este tipo de agentes, especializado en comunicación con humanos y capaz, a su vez, de comunicarse con Servicios Web. Jacinta usa técnicas de la Web Semántica y la semántica añadida por los administradores humanos del sistema, para permitir una interacción mucho más avanzada entre un usuario humano y los Servicios Web.

1. Introducción

La Web se encuentra en constante evolución. Empezó siendo un almacén distribuido de información cuando las páginas web eran estáticas, y evolucionó hacia un almacén de funcionalidad gracias a las Aplicaciones Web. Más tarde, a la vista del éxito, y para aprovechar

que la funcionalidad ya estaba creada, se planteó la posibilidad de que esa funcionalidad disponible para humanos también estuviese disponible para los agentes software gracias a los Servicios Web. Los Servicios Web aparecieron en 2.000 de la mano de las especificaciones de SOAP[13], UDDI[14], y WSDL[16]. Hoy, mientras en el mundo de las empresas se habla de las arquitecturas orientadas a servicios (SOA) en un intento de reactivar el uso de los Servicios Web tras la crisis “puntocom”, en los laboratorios se ponen a punto técnicas para añadir semántica a los Servicios Web, con el objetivo de conseguir unos sistemas inteligentes que logren desarrollar al máximo el potencial de la Web.

La necesidad de dotar de semántica a la Web se hace tanto más evidente cuanto más información hay disponible, aunque también se hace palpable cuando la información está muy relacionada. Por ejemplo, Google tiene indexadas más de ocho mil millones de páginas web, pero si le preguntamos por “libros en los que se menciona alguna de las obras de García Márquez”, no podrá darnos una respuesta válida. Sin necesidad de una pregunta tan sofisticada, si preguntamos por “número actual de servidores web” tampoco nos da un resultado satisfactorio. Claramente, la indexación y búsqueda por palabra clave se queda corta.

No sólo hay que dotar de semántica a las páginas web, también los Servicios Web (descritos por un fichero WSDL) necesitan una

^{*}Este trabajo ha contado con el apoyo del Ministerio de Ciencia y Tecnología a través del proyecto Arcadia (TIC2002-1948).

explicación semántica, bien anotándolos en el propio WSDL, bien añadiendo esa información semántica fuera de él. La simplicidad de los registros UDDI hace que cualquier búsqueda usando un buscador UDDI (como los UDDI Browsers de Sourceforge.net[15] o WSDP[17]) sea aún menos efectiva que Google.

Las principales iniciativas públicas que pretenden dotar de semántica a los Servicios Web, a fin de crear los denominados Servicios Web Semánticos, son OWL-S[8] (de DARPA, EE.UU.) y WSMO[12] (del VI programa marco de la U.E.) En el caso de OWL-S no se ha llegado a implementaciones funcionales aunque la primera especificación tenga ya 3 años. Aunque WSMO tiene poco más de un año de vida y ha crecido rápidamente, su entorno de ejecución WSMX[7] todavía no tiene resuelto el aspecto de la interacción persona-agente.

La experiencia alcanzada por nuestro grupo en la aplicación de técnicas de la Web Semántica [2][1], junto con la actual colaboración con el grupo de trabajo de WSMX, permitirá incorporar a WSMX las experiencias adquiridas con Jacinta[3][4][11].

La estructura de este artículo es la siguiente: en la sección 2 se muestra los objetivos de Jacinta, la sección 3 ofrece unos ejemplos significativos de escenarios de uso, en la sección 4 se muestra un proceso real de búsqueda de proveedores de servicios web y un análisis de los servicios web encontrados. Para terminar, en la sección 5 se presentan las conclusiones y el trabajo futuro.

2. Objetivos de Jacinta

El objetivo de Jacinta es conseguir que un usuario humano pueda utilizar de manera eficiente los servicios web a través de la interacción con servicios web semánticos (sws). Estos sws involucran típicamente a múltiples servicios web de diversos proveedores, y son el resultado de la anotación semántica de los servicios web utilizados mediante el uso de ontologías. Aunque la creación de sws no es el objetivo de Jacinta, al no existir todavía una implementación de sws nos vemos obligados a generar estos sws mediante el trabajo asistido

de los administradores de Jacinta, quedando lejos de la labor automática que realizarán las iniciativas mencionadas en la sección 1. Por tanto, las tareas que realiza Jacinta son las siguientes:

1. Selección del servicio. El sistema pone a disposición del usuario un conjunto de servicios (semánticos) que podrán ser usados por el usuario. El sistema se los mostrará de la manera más apropiada para que encuentre rápidamente el servicio deseado. Las soluciones consideradas tratan desde una simple lista cuando hay pocos sws hasta la creación de una clasificación jerárquica basada en “distancia semántica” mediante técnicas de clustering, algoritmos genéticos, o técnicas de Inteligencia Artificial. Sea cual sea la técnica utilizada, deberá permitir que se adapten a la “distancia semántica” de cada usuario puesto que no todos los usuarios relacionan de igual manera los servicios. Para lograr una atención personalizada, cada usuario del sistema debe identificarse.
2. Obtención de información. El sistema solicitará al usuario todos los datos necesarios para invocar los Servicios Web involucrados en el servicio seleccionado. La semántica añadida por los administradores del sistema será la que permita que si varios servicios web usan el mismo concepto (por ejemplo, “fecha”) de manera diferente (uno usa cadena con formato “dd/mm/aaaa”; otro usa tres enteros; otro tres cadenas, otro una estructura compleja, etc.), sean tratados como el mismo concepto “fecha” y se le solicite al usuario una sola vez.
3. Invocación del servicio. El sistema tratará la información proporcionada por el usuario para adaptarla al formato de cada uno de los servicios web utilizados por el sistema para la ejecución del servicio semántico seleccionado por el usuario. Esta transformación de los datos será realizada íntegramente por el sistema, idealmente reutilizando servicios semánticos del sistema que puede que invoquen a servicios web

auxiliares proporcionados por otras empresas.

4. Obtención de resultados. Tras la invocación de los servicios web el sistema recoge los resultados y los trata de manera adecuada para unificarlos y obtener unos resultados semánticos.
5. Presentación de resultados. El sistema mostrará al usuario estos resultados de una manera inteligente. Por ejemplo, permitiendo seleccionar qué datos del resultado le interesan y, si el número de resultados es excesivo, sugerir alternativas que resulten en un número más manejable de resultados.

De todas estas tareas, Jacinta se especializa en las que hay interacción con el usuario (puntos 1, 2 y 5), permitiendo así que cuando los sws de las iniciativas mencionadas sean una realidad sean ellos quienes traten en profundidad los puntos 3 y 4.

Además de realizar las tareas anteriores, Jacinta proporciona la siguiente funcionalidad:

- Independencia de dispositivo. Aunque siempre se asume un navegador web (PC, teléfono móvil, PDA, etc), este puede tener limitaciones de visualización como, por ejemplo, las inherentes al tamaño de la pantalla o el número máximo de colores que pueden mostrar. Jacinta tendrá en cuenta las características del dispositivo durante todo el proceso de interacción con el usuario.
- Proveedores externos de visualización. Jacinta permite que el aspecto estético del sistema con el que interactúa el usuario pueda ser proporcionado por empresas especializadas en diseño gráfico. Esto permite poder ofrecer un aspecto profesional (por ejemplo, empresas que desean una estética común conforme a su imagen corporativa), o adaptado a usuarios con limitaciones de interacción (por ejemplo, daltonismo o reducida agudeza visual), o por puro gusto estético (por ejemplo, ofrecido por empresas que incluyan publicidad, o por simple altruismo).

- Detección de modificaciones, disponibilidad y validez. Jacinta depende de los servicios web disponibles a través de la red. Estos servicios web pueden ser modificados por los proveedores de cada servicio web, quedando reflejados estos cambios en su fichero WSDL; puede que, aunque no haya cambios en el fichero WSDL, el servicio web no esté operativo; o puede que el servicio web responda con datos erróneos. Jacinta dispone de un sistema automático que comprueba periódicamente si los ficheros WSDL de los servicios web utilizados por el sistema han sido modificados. De ser así, los servicios de Jacinta que dependen de esos servicios web quedan inhabilitados temporalmente y se informa a los administradores para que rehagan las anotaciones semánticas de los servicios web modificados. Una vez realizadas las correcciones adecuadas se darán por válidos los servicios web y se volverán a habilitar automáticamente los servicios afectados. De la misma manera, el sistema comprueba periódicamente el estado de cada servicio web invocando alguna de las operaciones descritas en cada fichero WSDL para obtener una respuesta. Si no se obtiene respuesta el sistema inhabilita los servicios de Jacinta implicados e informa a los administradores. Una batería de pruebas definidas por los administradores comprobará de forma periódica que los resultados devueltos son los esperados.

La idea subyacente a Jacinta es que la funcionalidad se puede relacionar entre sí gracias a la semántica aportada por las ontologías, y estructurarla por capas. De esta forma podemos crear nueva funcionalidad reutilizando funcionalidad más sencilla de manera análoga a la reutilización de funciones (o librerías de funciones) en programación tradicional cuando queremos desarrollar nuevas funciones. Gracias a las ontologías y a su capacidad inherente de distribución, podemos asignar en Jacinta una funcionalidad concreta a cada uno de los conceptos distribuidos referidos en nuestra ontología. Cuando el corpus de funcionalidad sea suficientemente grande, conseguiremos que

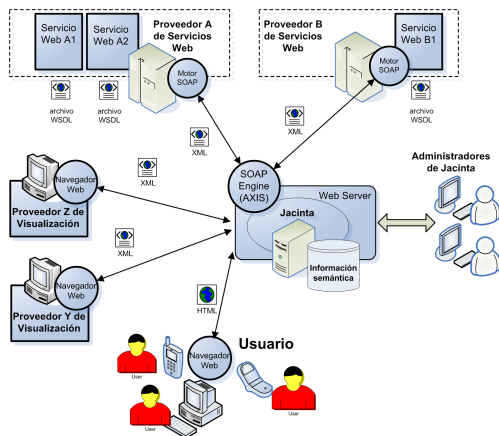


Figura 1: El sistema Jacinta. Participantes.

con una mínima “programación” por parte de los administradores de Jacinta queden a disposición de los usuarios funcionalidades realmente complejas de forma sencilla e intuitiva. Además, nuestro sistema dota a esta funcionalidad de un aspecto gráfico por defecto con el que un usuario humano puede interactuar, tanto en la adquisición de la información requerida al usuario como en la presentación de los resultados.

3. Escenarios de uso

Para centrar los objetivos y alcance del sistema propuesto, así como para poder mejorar el análisis y diseño del mismo, resultó útil plantear un conjunto de escenarios típicos en los que los diversos participantes interactúan con el sistema Jacinta. La Figura 1 muestra las partes implicadas en Jacinta. A continuación se muestra una selección de estos escenarios de uso.

3.1. Usuario con navegador de PC

En este escenario el dispositivo utilizado por el usuario no presenta limitaciones de visualización, esto es, el navegador dispone de una pantalla con suficiente tamaño como para mostrar mucha información simultáneamen-

te. El usuario utiliza el navegador web de su PC y teclea la URL del sistema Jacinta (<http://nets.ii.uam.es/jacinta>). El sistema pide al usuario que se identifique y, tras su identificación, muestra al usuario los servicios que puede proporcionarle clasificados por categorías, para que seleccione el servicio que desea utilizar. Supongamos que selecciona “Búsqueda de libros”. El sistema le indica que ese servicio lo ofrecen cinco empresas a través de sus respectivos Servicios Web. Puede que alguno de estos Servicios Web sea un “wrapper” de Portales/Aplicaciones Web. Al usuario se le muestra la información que tiene que facilitar diferenciando claramente la información obligatoria de la opcional. También se le muestra la cantidad de proveedores que podrán ser invocados conforme facilite la información; cuanto más información facilite, más servicios web serán susceptibles de ser invocados. En este ejemplo, deberá facilitar obligatoriamente el título del libro o su ISBN. Una vez que el usuario facilita la información, el sistema Jacinta la procesa y la pone en el formato adecuado para la invocación de los Servicios Web. Invoca los Servicios Web y recoge el resultado devuelto por estos. Procesa los resultados para presentarlos de la manera más adecuada al usuario, ofreciendo al usuario alternativas para su visualización o su poda si el número de datos es muy elevado. En este ejemplo, mostrará al usuario los tipos de datos recogidos (proveedor, título, autor, ISBN, precio) para que seleccione qué datos le interesan (por ejemplo, proveedor, y precio) y cuál será el criterio de ordenación (por ejemplo, el precio). El sistema mostrará los resultados conforme al criterio de ordenación indicado. De forma transparente al usuario, el sistema ha utilizado el servicio de cambio de moneda (uno de los sws que proporciona Jacinta) para convertir los precios, que a su vez utiliza el sws de identificación; así como el sws de información del usuario, que indica que el usuario suele visualizar los precios en euros.

3.2. Usuario con navegador de PDA

En este escenario el usuario utiliza el navegador web de su PDA, de forma que la cantidad

Tabla 1: Hora de salida de los vuelos de las compañías aéreas A y B.

A		B	
ida	vuelta	ida	vuelta
8:30	16:40	9:00	16:10
9:50	18:50	11:00	19:20
15:00			22:40

de información que puede visualizar de una sola vez es menor que en el caso anterior. El usuario, una vez identificado, selecciona el servicio de “Búsqueda de billetes de avión”. Este servicio utiliza los servicios web de varios proveedores. Se le pedirá al usuario que indique la ciudad de origen y de destino, así como el día de salida y de llegada. El sistema obtendrá de cada proveedor una lista de vuelos de salida y de llegada así como el precio de cada vuelo. Supongamos que el usuario desea ordenar los resultados por hora de salida. Debido a las limitaciones del navegador del usuario, queremos evitar mostrarle una lista extensa, por lo que esta información se le mostrará agrupada de forma “conceptual” de manera que el sistema le ofrece al usuario la posibilidad de mostrar el resultado por orden de “ida, hora de salida”, “regreso, hora de llegada”, o “compañía aérea”.

Por ejemplo, si las compañías aéreas A y B disponen de los horarios de vuelos para los días de salida y regreso mostrados en la Tabla 1, la lista de resultados, ordenados por “hora de salida”, será la siguiente:

ida	vuelta	compañía
8:30	16:40	A
8:30	18:50	A
9:00	16:10	B
9:00	19:20	B
9:00	22:40	B
9:50	16:40	A
9:50	18:50	A
11:00	16:10	B
11:00	19:20	B
11:00	22:40	B
15:00	16:40	A
15:00	18:50	A

dada su extensión y las limitaciones del navegador del usuario, se evita mostrarla ofreciendo al usuario los criterios de ordenación citados anteriormente. Si el usuario selecciona la ordenación por “ida, hora de salida” se le mostrará lo siguiente:

ida	compañía
8:30	A
9:00	B
9:50	A
11:00	B
15:00	A

Una vez seleccione el vuelo de salida se le ofrecerá seleccionar el vuelo de regreso. Este mecanismo permite una visualización más sencilla a costa de incrementar el número de ciclos de interacción. En este caso, la restricción de que tiene que ir y volver con la misma compañía queda oculta al usuario. En este caso de ordenación por “ida, hora de salida”, el paso siguiente será seleccionar el vuelo de regreso. Si selecciona un vuelo de ida de la compañía A se le dará a elegir entre los siguientes vuelos de regreso:

vuelta
16:40
18:50

Pero, si hubiese seleccionado un vuelo de ida de la compañía B se le daría a elegir entre los siguientes vuelos de regreso:

vuelta
16:10
19:20
22:40

3.3. Administrador y creación de servicios semánticos

Los administradores de Jacinta son los responsables de la creación y el mantenimiento de los servicios ofrecidos por Jacinta. Para crear un nuevo servicio el administrador debe conectarse al sistema via web e identificarse. Una vez que el sistema le ha identificado, seleccionará la herramienta de creación de servicios. Esta herramienta utiliza la tecnología Java Web

Start para proporcionar un interfaz avanzado realizado en Java. En esta herramienta se mostrarán los servicios disponibles, la ontología del sistema, las ontologías externas disponibles, y los modelos de interacción y visualización disponibles. Toda esta información será relacionada por el administrador gracias a la herramienta, creando nuevos elementos “servicio” en la ontología. El sistema ofrece a los administradores unos servicios básicos, cuya implementación requiere codificar en lenguaje Java, para la implementación de los servicios web utilizados por cada servicio.

4. Casos de prueba

Para poner a prueba el diseño inicial del sistema se creó un conjunto de casos de prueba. Para ello se necesitó un conjunto de empresas reales que proporcionasen el mismo servicio a través de Servicios Web. Se eligió inicialmente el servicio “Búsqueda de libros” y se buscaron proveedores de servicios web adecuados. Sin embargo, pronto encontramos que dependíamos de un conjunto de “subservicios” ajenos a estas empresas como, por ejemplo, un servicio de cambio de moneda o un sistema de identificación de usuarios. Nuestro usuario humano puede querer buscar libros a través de Jacinta, pero Jacinta debe conocer no sólo la forma de operar sobre los servicios web de las empresas que facilitan ese servicio, sino también saber convertir de la moneda del cliente (por ejemplo, euros) a la moneda de venta del proveedor (por ejemplo, dólares americanos).

Por tanto, comenzamos por un servicio más sencillo: el cambio de moneda. El estudio en detalle de este caso, aparentemente tan simple, nos llevó a descubrir bastantes detalles inesperados.

4.1. Buscando servicios web de cambio de moneda

El primer paso fue buscar proveedores de este servicio. Aunque se usaron buscadores UDDI como los mencionados en la sección 1, resultaron mucho más útiles los “agregadores” de servicios web. En concreto, se exploraron los siguientes agregadores de servicios web:

- www.salcentral.com
- www.xmethods.com
- www.webservicex.net
- www.remotemethods.com
- www.xignite.com

Los “agregadores” son portales web dedicados a agrupar referencias a servicios web, ofreciendo a los usuarios facilidades para localizar estos servicios web. Típicamente organizan los servicios web por categorías, proporcionan buscadores por palabra clave, o proporcionan clientes web para la invocación del servicio web.

Un aspecto que nos llamó la atención fue la poca cantidad de servicios web que agrupan estos portales (del orden de las centenas), aunque muchos de ellos dicen alimentarse de registros UDDI. Una deficiencia habitual es la gran cantidad de referencias a otros agregadores. Esto genera una cantidad considerable de información redundante.

El resultado final fue que únicamente encontramos tres proveedores de este servicio en los agregadores (CurrencyExchangeRate en XMethods, CurrencyConvertor en Webservicex, y CurrencyExchange en Xignite). Gracias a Google encontramos uno más (al que denominamos Cloanto, de currencySystem.com), por lo que finalmente dispusimos de cuatro proveedores de este servicio, cada uno con características completamente distintas:

- Identificación. De estos cuatro proveedores, dos requieren identificación del usuario. Además, las identificaciones son distintas. En Xignite se debe rellenar un formulario con tres datos (Email, Nombre empresa, Name), aunque en realidad lo importante es la IP de la máquina desde la que se hace la petición. Tras rellenar el formulario, la IP queda registrada en Xignite y ya se pueden invocar servicios (desde la IP registrada). En Cloanto hay que rellenar un formulario con email y nombre, el sistema envía un email con

un número de licencia que habrá que suministrar en cada una de las invocaciones de los servicios.

- Número de invocaciones. Hay que tener en cuenta que los proveedores suelen tener limitado el número de invocaciones. En el caso de Xignite, a 10 si el usuario no se ha registrado, y a 50 si se ha registrado.

Analizando el fichero WSDL proporcionado por estos cuatro proveedores, encontramos que los tipos de datos necesarios para la invocación del servicio son completamente distintos. Los más sencillos son los de Webservicex y Salcentral, que no define tipos complejos. Webservicex define 3 tipos complejos, y Cloanto define unos 90.

Los argumentos de la operación que invocaremos también son muy variables. En el caso de Xignite sólo se pasan dos cadenas, una para identificar la moneda origen y otra para identificar la moneda destino (aceptando más de 170 monedas diferentes, pero sin indicar sus códigos). En el caso de Cloanto, además de las monedas origen y destino (que no tenían porqué ser las mismas que en el caso de Xignite pues consideran del orden de 100 monedas) se podía indicar la cantidad a convertir (1 unidad de moneda origen en el caso de Xignite), e incluso el redondeo a aplicar. En el caso de Webservicex y XMethods, las monedas origen y destino estaban limitadas a cierto conjunto concreto.

El resultado de la invocación también es muy variable. Xignite devuelve un tipo complejo del que se puede extraer un valor con 15 decimales, Cloanto devuelve un tipo string con dos decimales, Webservicex devuelven un tipo double con 4 decimales, y XMethods devuelve un float con 4 decimales.

Toda esta variabilidad debe ser incluida en la ontología que describa el concepto "cambio de moneda". Este concepto usará otros como "Moneda" (con identificadores de moneda ISO 4217) o "País" (con código ISO 3166 alpha-2 (dos letras) y alpha-3 (tres letras), y nombres en distintos idiomas) con referencias a elementos de ontologías estándar. Por tanto, los conceptos exclusivos de nuestra ontología deberán

incluir datos acerca de:

- Condiciones del servicio. Donde se definen parámetros como número máximo de peticiones por unidades de tiempo (que puede ser desde 0, para servicios web inoperativos; hasta ilimitado, para servicios web que no ponen restricciones de uso)
- Modelos de identificación. Útil para la inicialización de servicios. Definen elementos como "Login Requerido", Identificadores de identidad (email, IP, ID, etc.), conexión, o sesión.
- Operadores sobre datos numéricos. aquí se definen conceptos como "número decimal", precisión, y diversas funciones de redondeo.

Como ejemplo ilustrativo, para poder utilizar el servicio web de Cloanto tuvimos que crear un modelo de identificación de usuarios que generase un identificador de usuario, lo enviase por email a Cloanto, quedase a la espera del mail de respuesta con el id de usuario (mientras, al usuario se le informa de que ese servicio no es utilizable), y extrajese del mail de respuesta el id. Sólo después de ese proceso quedará el servicio de Cloanto disponible para los usuarios de Jacinta. Por si fuese poco, el id caduca cada cierto tiempo de inactividad, o cada cierto número de usos. Todo este proceso de reactualización de identificador es gestionado automáticamente por Jacinta.

5. Conclusiones y trabajo futuro

Aunque algunos trabajos [6][10] de otros autores abordaban problemas similares, sus resultados han sido poco claros. El hecho de haber implementado nuestros propios servicios web semánticos nos han permitido descubrir muchos detalles inesperados de los servicios web. Entre ellos destacamos la existencia de menos servicios web de los esperados así como detalles de la comunicación que obligan a la identificación del usuario. También nos han permitido comprobar que hay mucho trabajo por hacer no sólo en el área de la interacción persona-agente aplicada a los servicios web semánticos

en particular, sino aplicada a la Web Semántica en general.

En cuanto a los detalles de implementación, en su día consideramos las ventajas de usar Java Server Faces (JSF)[5] como front-end de usuario, pero las pocas implementaciones existentes entonces nos hicieron desecharlo. Hoy existen algunas implementaciones comerciales y de uso público que nos obligan de nuevo a plantearnos este cambio. También estamos en una situación parecida con el lenguaje usado por nuestra ontología: al empezar nuestro proyecto no había un número elevado de ontologías en OWL[9], pero desde entonces ha crecido mucho más que RDFS y los razonadores han mejorado. Por tanto, también nos planteamos cambiar el lenguaje de nuestra ontología (RDFS) por OWL.

En cualquier caso, tenemos por delante la creación de más sws, lo que supone ampliar los modelos utilizados, refinar la ontología, y mejorar las herramientas que permiten a los administradores de Jacinta la introducción de toda esta información.

Referencias

- [1] Castells P., et al. *Semantic Web Technologies for a Digital Newspaper Archive*. 1st European Semantic Web Symposium (ESWS 2004). May 2004. Springer Verlag LNCS, Vol. 3053.
- [2] Castells P., et al. *Semantic Web Technologies for Economic and Financial Information Management*. 1st European Semantic Web Symposium (ESWS 2004). May 2004. Springer Verlag LNCS, Vol 3053.
- [3] Gómez J.M, Rico M., García-Sánchez F., Martínez-Béjar R., Bussler, C. *GO-DO: Goal-driven orchestration for Semantic Web Services*. 1st Workshop on Web Services Modeling Ontology Implementations (WIW 2004). September 2004.
- [4] Gómez J.M, Rico M., García-Sánchez F., Acuña C.J. *The Semantic Web Services Tetrahedron: Achieving Integration with Semantic Web Services*. To be published in procs of International Conference on Web Engineering (ICWE), Sydney, Australia, August 2005.
- [5] JSF: Java Server Faces. See <http://java.sun.com/j2ee/javaserverfaces/>
- [6] Kassoff M., Kato D., Mohsin W. *Creating GUIs for Web Services*. IEEE Internet Computing Press, September-October 2003.
- [7] Oren E., Zaremba M., Moran M., *Overview and Scope of WSMX*. WSMX Working Draft, 2004. Available from <http://www.wsmo.org/TR/d13/d13.0/v0.2/>
- [8] OWL-S: Semantic Markup for Web Services. <http://www.w3.org/TR/owl-guide/>
- [9] OWL: Web Ontology Language. <http://www.daml.org/owls>
- [10] Petrie P., Genesereth M., Bjornsson H., *Adding AI to Web Services*. LNCS Volume 2926 / 2004: pp.322 - 338, March 2003.
- [11] Rico M. and Castells P., *Jacinta: a Mediator Agent for Human-Agent Interaction in Semantic Web Services*, In Procs. International Semantic Web Conference (ISWC) 2004. Selected Posters, Hiroshima, Japan, November 2004.
- [12] Roman D., Lausen H., Keller U., *Web Service Modeling Ontology Standard*. WSMO Working Draft v02, 2004. Available from <http://www.wsmo.org/2004/d2/v02/>
- [13] SOAP: Simple Object Access Protocol. <http://www.w3.org/TR/soap/>
- [14] UDDI: Universal Description, Discovery, and Integration. <http://www.uddi.org/>
- [15] UDDI Browser. <https://sourceforge.net/projects/uddibrowser/>
- [16] WSDL: Web Services Description Language. <http://www.w3.org/TR/wsdl>
- [17] WSDP: Web Services Developer Pack. <http://java.sun.com/webservices/jwsdp/>