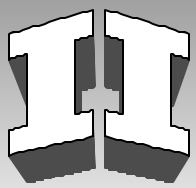

INFORMÁTICA

MODELO DE LOS DATOS INCORPORADOS EN LOS AGENTES FIPA QUE CONFORMAN UN SISTEMA DISTRIBUIDO PARA LA INVOCACIÓN AUTOMÁTICA DE SERVICIOS WEB

Resumen / *Abstract*

El trabajo está orientado a mostrar la concepción de las ontologías utilizadas en el desarrollo de una arquitectura de sistema multiagente para la contratación automática de servicios Web. Primeramente, se citan las características del estándar FIPA, que ha sido considerado para la implementación de sistema multiagente, y algunos estándares para la representación y ejecución de servicios Web. Se describe el funcionamiento general y los requerimientos de la arquitectura que se propone. Se muestra la estructura y utilización de los dos cuerpos ontológicos que han sido modelados. Estas ontologías brindan el modelo conceptual que es utilizado por los agentes de la arquitectura para la representación del conocimiento que posee cada agente de su entorno, y para la representación de los servicios Web que serán contratados por el usuario.

The present paper is oriented to show the ontologies development ideas used into the multiagent agent system architecture for Web services automatic invocation. First, points the characteristics of the FIPA standard, considered into the multiagent system implementation, and others standards take into account for the Web services representation and execution. Also, describes the general functionality and requirements of the proposed architecture. It shows the structure and functionality of two developed ontologies. These ontologies give the conceptual model that has been used for agents for knowledge representation about the agent environment, and Web services location and invocation.

Palabras clave / *Key words*

Agentes, sistemas multiagentes, ontologías, servicios Web

Agents, multiagent systems, ontologies, Web services

Alberto Caballero Martínez, *Ingeniero en Informática, Máster en Telemática, Instructor, Departamento de Matemática General, Facultad de Ingeniería Industrial, Instituto Superior Politécnico José Antonio Echeverría. Cujae, Ciudad de La Habana, Cuba*
e-mail:alberto@ind.cujae.edu.cu

Juan Botía Blaya, *Ingeniero en Informática, Doctor en Informática, Profesor Titular, Departamento de Ingeniería de la Información y la Cominicación Facultad de Informática, Universidad de Murcia, España*
e-mail:juanbot@um.cu

Antonio Gómez Skarmeta *Ingeniero en Informática, Doctor en Informática, Profesor Titular, Departamento de Ingeniería de la Información y las Comunicaciones, Facultad de Informática, Universidad de Murcia. España*
e-mail:skarmeta@dif.um.es

Recibido:Mayo del 2004
Aprobado:Julio del 2004

INTRODUCCIÓN

Actualmente, un gran número de problemas están siendo resueltos con la utilización de sistemas multiagentes (MAS). Los agentes software son considerados los elementos del nuevo modelo para la obtención de sistemas informáticos heterogéneos, escalables, abiertos y distribuidos, en los cuales se exhiban características inteligentes en aras de ofrecer la mejor solución al usuario final del sistema.[1-3]

Para exhibir las características relacionadas anteriormente, estos sistemas deben poseer organizaciones eficientes de sus datos.[4,5] Sus modelos de datos, además de representar el dominio de conocimiento concreto al que están aplicados, tienen que ser capaces de describir todas las interacciones que se dan entre las diferentes partes del sistema. Así, cada agente tendrá un modelo de datos, llamado ontología, que le permitirá representar su conocimiento del entorno y las interacciones que lleva a cabo con el resto de la comunidad de agentes del sistema.

La eficiencia de cada agente, y por consiguiente la del sistema, depende en gran medida de la adecuación de la estructura de las ontologías utilizadas para la descripción de las interacciones y del conocimiento.[3-5]

El presente trabajo tiene como objetivo fundamental mostrar la concepción de dos cuerpos ontológicos que han sido desarrollados para representar el conocimiento de los agentes en una arquitectura de sistema multiagente para la contratación automática de servicios de aprendizaje a través de servicios Web. Para ello se ha organizado de la siguiente manera: primeramente, se citan las características del estándar FIPA, particularmente el protocolo de negociación ContractNet (CNP), y algunos estándares para la representación y ejecución de servicios Web; se relacionan, mediante la descripción de la utilización, los requerimientos de la arquitectura; se muestra la estructura y utilización de los dos cuerpos ontológicos que han sido modelados, ofreciendo el modelo conceptual que es utilizado por los agentes.

CONSIDERACIÓN DE ALGUNOS ESTÁNDARES

Se ha revisado la literatura relacionada con el tema para establecer la tecnología a utilizar en el desarrollo de la arquitectura deseada, pues de ella depende la estructura del modelo de datos representados en cada cuerpo ontológico. En este sentido, se ha definido un modelo de conocimiento para cada agente basado en las características de negociación que exhiben los agentes FIPA. También, basado en estándares para la representación en invocación de servicios Web, se ha definido un modelo que considera las necesidades de la actual arquitectura.

Estándar FIPA

La especificación del estándar FIPA establece la forma de implementación de una comunidad de agentes que, entre otras cosas, posean capacidades de cooperación.[6] Define un conjunto de protocolos de negociación que regulan las interacciones entre los agentes: Contract Net Protocol (CNP),[7] Dutch Auction (DA),[8] English Auction (EA),[9] entre otros.

En cada uno de estos protocolos de negociación existe un agente que inicia el protocolo, aquel que requiere la ejecución de un servicio. Este agente lanza una solicitud de propuesta a los agentes que puedan brindar el servicio (en el cuerpo del mensaje CFP), y estos a su vez hacen una propuesta al agente iniciador, mediante el mensaje PROPOSE. A partir de ese momento se desarrolla un proceso de negociación con las particularidades de cada protocolo.

En el caso del *CNP*, una vez que el iniciador recibe las propuestas del resto de los agentes (en el cuerpo de los mensajes *PROPOSE*) las evalúa y decide a quien contratar, entonces lanza un mensaje del tipo *ACCEPT_PROPOSAL* a aquellos agentes a los cuales se contrata, estos ejecutan la acción que han propuesto y según el resultado emiten los mensajes correspondientes *INFORM_DONE, INFORM_RESULT o FAILURE*.[7]

Los datos contenidos en cada mensaje dependen de la aplicación para la cual se han diseñado los agentes. En este caso de contratación automática de servicios Web, el contenido de estos mensajes debe estar en correspondencia con el modelo de datos definido para representar el conocimiento que tiene cada agente de los demás, sus prestaciones y de los servicios que representan.

Algunos estándares para la representación de servicios Web

Se reportan varios estándares para la descripción de servicios, entre los más representativos podrían citarse: WSDL,[10] DAML-S,[4,11] UDDI T-Models.[12] Se han analizado cada uno de ellos con el objetivo de definir una variante de solución para la integración de la representación de los servicios Web en el directorio de agentes de FIPA.

Realizar la contratación de servicios Web mediante una arquitectura multiagente, hace de los servicios Web un entorno activo. Frente a esta nueva solución, la contratación de un servicio va más allá de la simple búsqueda del nombre de un servicio en el directorio de servicios (brindado por alguno de los estándares anteriores) y su posterior invocación. Los procesos de coordinación entre los agentes, para tener en cuenta las preferencias del usuario y el conocimiento previo del sistema en el momento de efectuar la búsqueda y contratación del servicio adecuado, resulta más complejo.

En estos casos se hace necesario incorporar la descripción de los servicio en la estructura del directorio de agentes de FIPA. De manera que este nuevo directorio de agentes que, entre otras, brinde las siguientes funciones:

- Búsquedas de servicios según las características actuales del proveedor
- Búsquedas de servicios según el tipo de servicio.
- Búsquedas de servicios según el desempeño del proveedor en momentos anteriores.
- Descripción de servicios (páginas amarillas).
- Descripción de agentes (páginas blancas), mostrando las capacidades de negociación que indiquen la correcta elección del servicio a partir de los resultados de solicitudes previas.

PROCESO DE COORDINACIÓN ENTRE AGENTES PARA LA CONTRATACIÓN DE SERVICIOS

Se ha diseñado e implementado una comunidad de agentes con capacidades de cooperación que negocian con el fin de establecer relaciones de contratos de servicios entre ellos. Se ha tomado como caso de aplicación la contratación de servicios de aprendizaje, tomando en cuenta algunos elementos desarrollados en METALA.[13]

De esta forma, se utiliza la especificación FIPA para implementar diferentes agentes que desarrollan funciones de aprendizaje. De este modo se distinguen dos tipos de agentes:

1. Agente contratista: agente que representa los intereses de los usuarios finales del sistema. A partir de una necesidad de servicio, negocia una solicitud con el resto de los agentes.

2. Agente contratado: agente que, al recibir solicitudes de los agentes contratistas, hacen propuestas a partir de los servicios que brindan.

Un esquema general de la arquitectura que se tiene en cuenta se muestra en la figura 1.

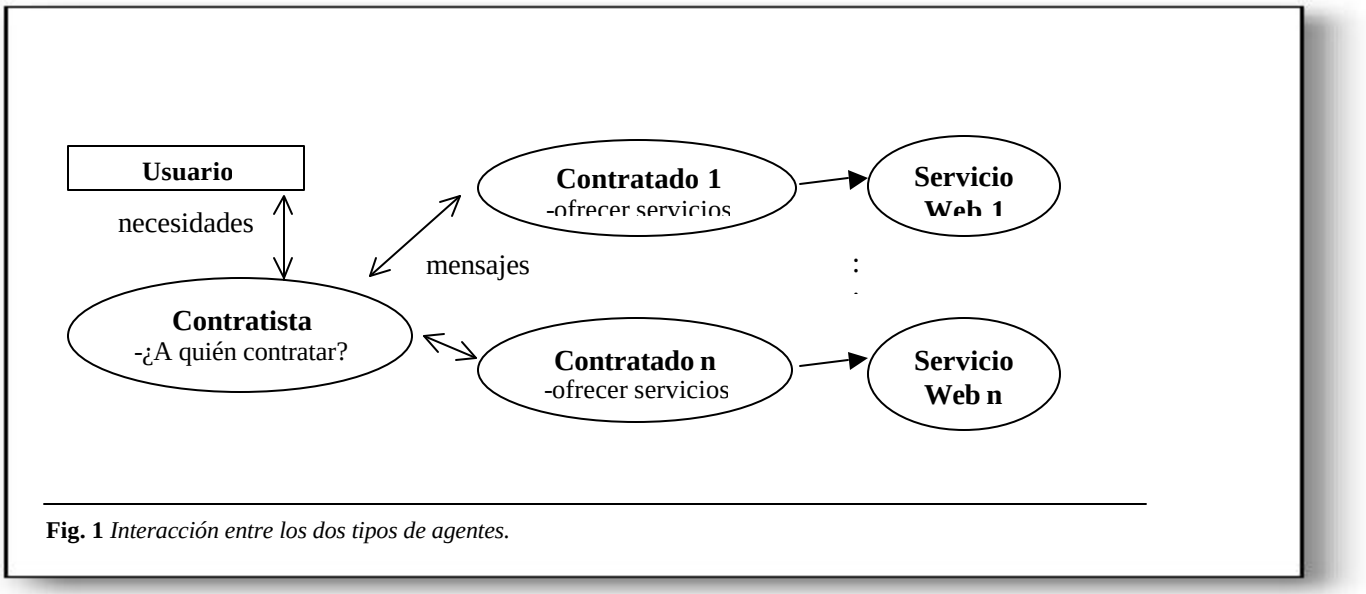

Fig. 1 *Interacción entre los dos tipos de agentes.*

La negociación (coordinación) entre los agentes se desarrolla mediante el intercambio de mensajes según el protocolo CNP estandarizado por FIPA. De esta manera, se advierten las siguientes necesidades en cuanto a la representación del conocimiento de cada agente:

• El agente contratista debe ser capaz de representar las necesidades del usuario y lanzar una solicitud de servicio al resto de los agentes. (De manera general, el agente contratista no tiene conocimiento acerca de la ubicación, desempeño, responsable, proveedor de los servicios Web (él solo necesita un servicio y lanza una solicitud al resto de los agentes.)

• Los contratados deben analizar las solicitudes y realizar ofertas mediante un mecanismo donde se tengan en cuenta las necesidades de los usuarios y las características de los servicios brindados.

• Una vez recibidas todas las ofertas, el agente contratista las analiza. Para ello tiene en cuenta los parámetros propios de cada propuesta, así como la calidad del cumplimiento de contrataciones anteriores a ese agente. Por tanto, el agente contratista debe incorporar modelos para representar el desempeño anterior de cada uno de los otros agentes en función de sus necesidades, y organizados según los servicios que se han contratado hasta el momento, sus resultados, la efectividad de los contratos anteriores, entre otros.

De esta forma se han implementado varios agentes contratista y contratados que siguen con las especificaciones FIPA y con las consideraciones hechas anteriormente que se han tenido en cuenta en el diseño de la ontología que los agentes utilizan en el proceso de negociación.

Por otra parte, los agentes contratados también necesitan invocar el(los) servicio(s) Web relacionado(s), o sea, conocer ¿dónde está?, ¿cómo invocarlo?, ¿qué parámetros utilizar? etc. Por tal motivo, los agentes contratados que han sido implementados según las consideraciones anteriores, utilizan la ontología ServicesOntology donde se describen los conceptos relacionados con la ubicación e invocación de los servicios brindados, así como la forma de enlazarse con la tecnología servidora de servicios (JBoss) mediante WSDL.

ONTOLOGÍAS IMPLEMENTADAS

A continuación se ofrecen las dos ontologías desarrolladas, MLMetalaOntology para la descripción del proceso de negociación de la contratación de los servicios y ServicesOntology de la representación del conocimiento relativo a los servicios Web.

La implementación de estas ontologías se realizó en Protege 2000.[14] Utilizando el paquete BeanGenerator,[15] se generaron las clases Java correspondientes. Para inicializar (y almacenar) el conocimiento de los agentes (tanto contratista como contratados) se utilizaron ficheros texto que se leen / actualizan utilizando la clase Properties de Java.

ML metalaontology

Como se ha tratado anteriormente, el dominio de conocimiento utilizado se refiere a servicios de aprendizajes. Para ello, en la ontología MLMetalaOntology se describen los conceptos y las relaciones entre los conceptos que se utilizan en el proceso de negociación entre los agentes contratistas y contratados. En esta ontología se describen, fundamentalmente, los contenidos de los mensajes que se utilizan en cada uno de los protocolos de negociación.

Interacciones FIPA según la ontología MLMetalaOntology

Al describir las interacciones en el proceso de negociación entre los agentes se hace referencia a las clases de la ontología que se utilizan para ello. También se destacan las clases que componen el contenido de cada mensaje intercambiado en el CNP estandarizado por FIPA. El diagrama de clases que muestra la estructura de esta ontología se muestra en la figura 2.

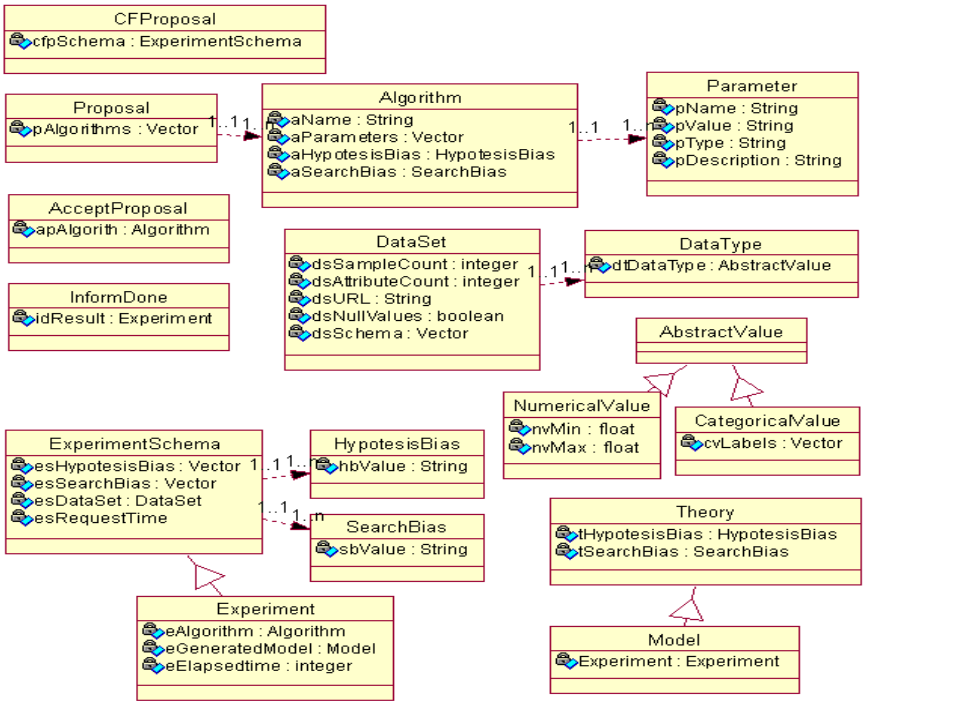

Fig 2. *Representación de las clases de la ontología MLMetalaOntology.*

En este caso, donde los agentes necesitan o brindan servicios de aprendizaje, las solicitudes que hace el agente contratista se basa en el esquema del experimento deseado (*ExperimentSchema*), el cual se incluye en el cuerpo del mensaje CFP. Este esquema esta compuesto por el conjunto de sesgos de representación (*HypotesisBias*) y de búsqueda (*SearchBias*) deseados, por el conjunto de datos a analizar (*DataSet*) y el tiempo de procesamiento que se requiere como máximo (*RequestTime*).

Los valores de los sesgos de representación y búsqueda constituyen, listas cuyos elementos representan los valores deseados de estos dos parámetros.

El conjunto de datos que se desean analizar es una colección de datos numéricos (*NumericValue*) o de etiquetas (*CategoricalValue*) caracterizadas por la cantidad de ejemplos (*SampleCount*), la cantidad de atributos de cada ejemplo (*AttributeCount*), la posibilidad de existencia o no de valores nulos (*NullValues*), entre otros.

En función del contenido de la solicitud recibida, cada agente contratado analiza si alguno(s) de los servicios de aprendizaje brindados (*Algorithms*) coincide con la solicitud. En caso afirmativo, responde, mediante el mensaje PROPOSE, al contratista haciéndole conocer cuál es el algoritmo de aprendizaje que se ofrece, sus parámetros (*Parameters*) y los sesgos de representación y búsqueda que se satisfacen con ese algoritmo de aprendizaje. De cada parámetro se especifica su nombre, valor, tipo y descripción.

El contratista analiza todas las propuestas recibidas y decide cuál(es) contratar. Envía la aceptación, mediante el mensaje *ACCEPT_PROPOSAL*, y el(los) agente(s) contratado(s) comienza(n) la ejecución del servicio de aprendizaje. Una vez finalizada la ejecución del servicio de aprendizaje, cada agente contratado involucrado, extiende el esquema del experimento añadiendo campos tales como algoritmo que se ha empleado, el tiempo real en que se ha realizado el servicio(*ElapsedTime*), el modelo resultante de la ejecución del servicio (*GeneratedModel*), entre otros. Los resultados del experimento realizado (*Experiment*) son informados al agente Contratista. En este caso se envían mensajes del tipo *INFORM_DONE*, *INFORM_RESULT* o, si hubo errores, *FAILURE*.

Este modelo de conocimiento, que se comparte entre contratista y contratados para los servicios de aprendizaje, ha quedado representado en la ontología MLMetalaOntology que utilizan los agentes de los dos tipos para hacer posible el proceso de negociación.

ServicesOntology

También se implementó la ontología ServicesOntolgy para permitir la representación de los servicios Web por parte de cada uno de los agentes contratados relacionados. Cada uno de los agentes contratados puede incorporar la descripción de los servicios Web utilizando los propios ficheros wsdl. La semántica y funcionalidad de cada una de las clases de la ontología se defi-

nen a continuación. En cada momento que se haga referencia a la semántica de un atributo, este aparecerá entre paréntesis indicando, como prefijo, la clase de la ontología a la cual pertenece. El diagrama de la ontología ServicesOntology se muestra en la figura 3.

En esta ontología, la interfaz de un servicio Web está dada por una instancia de la clase *ServiceInterface*, de igual manera cada una de los cuatro elementos que la forman vienen dadas por las clases *SIDatatype* (para los Data types), *SIMessages* (para los *Messages*), *SIPortTypes* (*Port types*) y *SIBindings* (para los *Bindings*).

En Services Ontology, los tipos de datos de la interfaz no son mas que estructuras de datos simples (clase *SISimpleDataType*) o compuestas (clase *SIComplexDataType*). Por cada tipo de dato simple se tiene su nombre (*SISimpleDataType*: sdName), su tipo (*SISimpleDataType*: sdType), y valores que indican la multiplicidad (*SISimpleDataType*: sdMinOcurrs y SISimpleDataType: sdMaxOcurrs). De igual manera, un tipo de dato compuesto puede verse como un conjunto de tipos de datos simples (*SIComplexDataType*: *cdElements*) identificados por un nombre (*SIComplexDataType*: *cdName*).

Cada uno de los mensajes que aparecen en la parte Messages son instancias de la clase (*SIMessage*). Por cada uno de los mensajes se recoge su nombre (*SIMessage*: *mName*) y un conjunto de partes (*SIComplexDataType*: *cdParts*) que no son más que datos identificados por un nombre (*MessagePart*: *mpName*) y su tipo (*MessagePart*: *mpType*), o sea, simples o complejos.

De igual forma, los *PortTypes* son instancias de la clase *SIPortType*, cada una de ellas identificada por un nombre (*SIPortType*: *ptName*) y que agrupa un conjunto de operaciones definidas (*SIPortType*: *ptOperations*). Cada una de estas operaciones es una invocación que puede hacerse al servicio utilizando su nombre y los mensajes de entrada y salida.

Cada operación del PortType se describe según la clase *PTOperation*. Cada una de estas operaciones es una descripción abstracta, solo considerando el enlace se conocerá la operación en cuestión. En este nivel, por cada operación se relaciona su nombre (*PTOperation: ptoName*), y la entrada (*PTOperation*: *ptoInput*), salida (*PTOperation*: *ptoOutput*) y falla asociado (*PTOperation*: *ptoFault*). Cada uno de estos tres parámetros constituye una instancia de la clase *PTOInOut* donde se tiene el nombre de la entrada / salida / falla (*PTOInOut*: *ioName*) y el mensaje asociado (*PTOInOut*: *ioMessage*).

Por cada PortType pueden existir varios enlaces asociados a él.

Cada enlace especificado en la interface está compuesto por el nombre (*SIBinding*: *bName*), el PortType asociado (*SIBinding*: *bType*), el estilo (*SIBinding*: *bStyle*) indicando si las operaciones contenidas (*SIBinding*: *bOperations*) serán orientadas a RPC o a documentos (cada operación puede redefinir su estilo). También se indica el mecanismo de trasporte utilizado dirigidos a un enlace (*Binding*) concreto (*SIBinding*: *Transport*).

Cada Enlace puede tener varias operación que en la ontología ServicesOntology se reflejan como instancias de la clase *BOperation*.

La clase *BOperation* está formada por atributos tales como el nombre de la operación (*BOperation*: *boName*); de manera opcional, el estilo (*BOperation*: *boStyle*); y las entradas y salidas de la operación dados por los atributos *BOperation*: *boInput*,

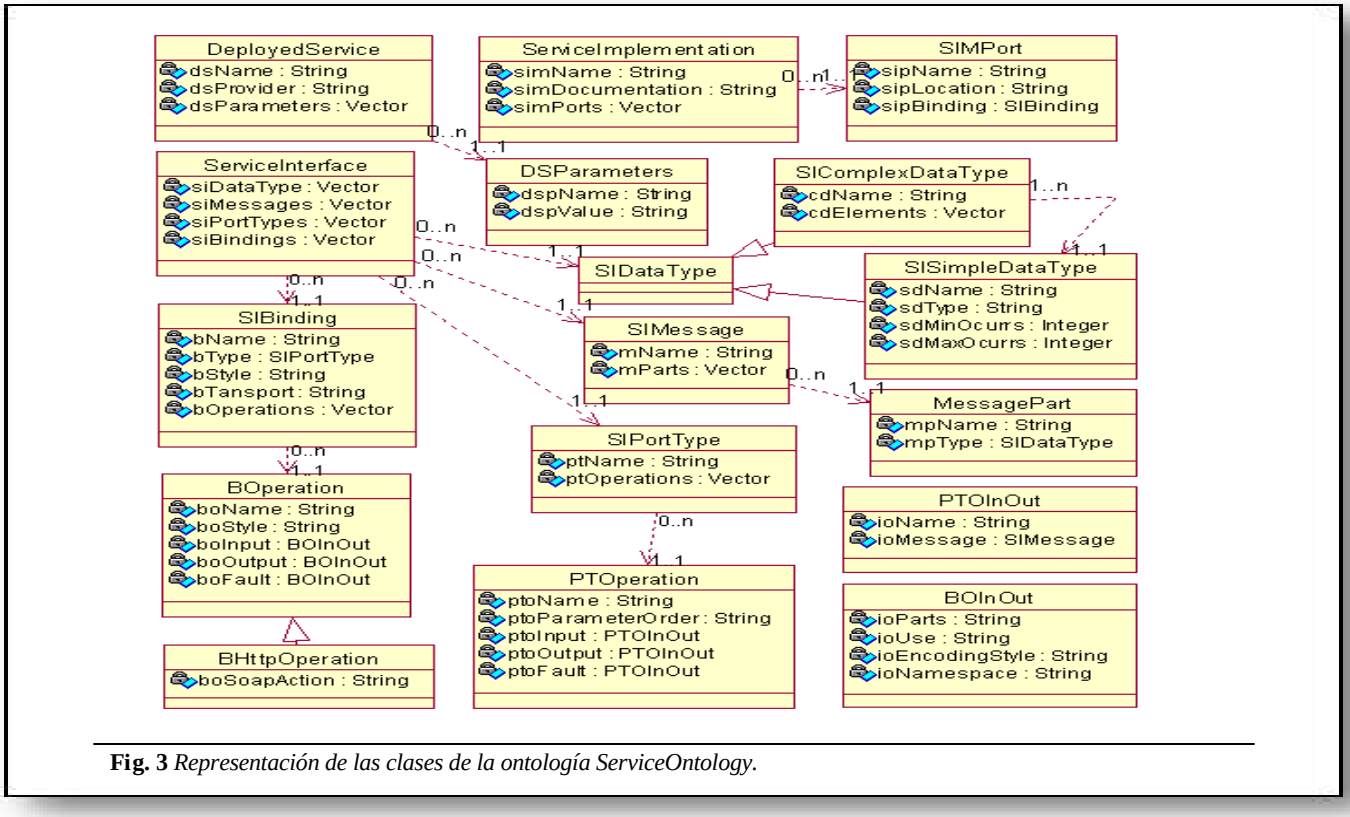

Fig. 3 *Representación de las clases de la ontología ServiceOntology.*

BOperation: *boOutput* y *BOperation*: *boFault*. Estos atributos son instancias de la clase *BOInOut*.

En la clase *BOInOut* se especifican qué partes aparecen en otra porción del documento (*BOInOut*: *ioParts*), las reglas de codificación de estas partes (*BOInOut*: *ioUse*), el estilo de codificación (*BOInOut*: *ioEncodingStyle*) y el namespace asociado (*BOInOut*: *ioNamespace*).

Para enlaces HTTP, que se representan mediante la clase *BHttpOperation*, debe incluirse el parámetro (*BHttpOperation*: *boSoapAction*) que no debe aparecer en el resto de los enlaces y que indica la intención de la petición HTTP correspondiente, su valor es un URI (*Unified Resource Idetifier*).

Luego, la implementación de un servicio se incorpora mediante las clases *ServiceImplementation y SIMPort*. La implementación está dada por su nombre (*Service Implementation: simName*), documentación (*ServiceImplementation: simDocumentation*) y un conjunto de puertos (*ServiceImplementation: simPorts*), cada uno de los cuales relaciona su dirección URL *(SIMPort: sipLocation*), su nombre (SIMPort: sipName) y enlace asociado (*SIMPort: sipBinding*). Este enlace debe ser un elemento del conjunto de enlaces definido en la interface del servicio.

Una vez descrito el servicio este debe ser desplegado en un entorno de ejecución local, para ello debe ser posible especificar, entre otros, el nombre del servicio (*DeployedService: dsName*), el nombre del proveedor (*DeployedService*: *dsProvider*) y los parámetros de configuración (*DeployedService: dsParameters*) que no son más que pares de valores del tipo nombre del parámetro (*DSParameters: dspName*) - valor del parámetro (*DSParameters*: *dspValue*). La descripción del despliegue de los servicios Web se incorpora en las clases *DeployedService y DSParameters* de la ontología ServicesOntology.

CONCLUSIONES

- Se identificaron las potencialidades de la especificación FIPA en la construcción de un sistema de agentes con capacidades de negociación para la contratación de servicios. De las cuales se precisaron las características más importantes que fueron capturadas en la ontología MLMetalaOntology.
- Se revisaron varios estándares para la representación de servicios Web para definir el conocimiento que debe tenerse en cuenta para su contratación automática.
- Se identificaron las necesidades de una arquitectura de un sistema de agentes que desarrollan procesos de coordinación para la contratación de servicios Web según las necesidades del usuario.
- Se diseñó e implementó la ontología MLMetalaOntolgoy para la descripción de los contenidos de los mensajes para cada uno de los protocolos de negociación en el proceso de contratación de un servicio

Se diseñó e implemento la ontología ServicesOntology para la descripción de la funcionalidad y ubicación de los servicios Web según el lenguaje WSDL en el directorio de agente s FIPA.

RECOMENDACIONES

- Refinar la ontología MLMetalaOntology, a partir de las características de los modelos de agentes considerados y los protocolos de negociación.
- Refinar la ontología ServicesOntology, teniendo en cuenta la descripción que en el MAS se tiene de los servicios Web.

REFERENCIAS

1. **LYELL, M.:** *Interoperability, Standards and Software Agent Systems*, Proceedings of 23rd Army Science Conference, Florida, www.asc2002.com/oral_summaries/O/OO-02.PDF, 2003.
2. **SHEN, W.; Y. LI; H. GHENNIWA AND C. WANG:** *Adaptive Negotiation for Agent-Based Grid Computing. Proceedings of Agentcities*: Challenges in Open Agent Environments, 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems, Bologna, Italy. http://www.agentcities.org/Challenge02/Proc/Papers/ch02_25_shen.pdf, 2002.
3. **KO, I. AND R. NECHES:** "Composing Web Services for Large-Scale Tasks", *IEEE Internet Computing*, September-October, 2003.
4. **PAOLUCI, M. AND K. SYCARA:** "Autonomous Semantic Web Services". *IEEE Internet Computing*, (September-October), 2003.
5. **DAS, S.; K. SHUSTER AND C. WU:** *Ontologies for agent-Based Information Retrieval and Sequence Mining. Proceedings of the Workshop on Ontologies in Agent Systems*, 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems, Bologna, Italy. http://www.cra.com/publications/Abstracts/2002 Bologna_Ontologies. html, 2002.
6. FIPA Abstract Architecture Specification. 2000. http://www.fipa.org/
7. FIPA Contract Net Protocol Specification. 2002. http://www.fipa.org/
8. FIPA Dutch Auction Protocol Specification. 2002. http://www.fipa.org/
9. FIPA English Auction Protocol Specification. 2002. http://www.fipa.org/
10. **CHRISTENSEN, E.; F. CUBERA; G. MEREDITH, AND S. WEERAWARANA:** *Web Services Description Language (WSDL) 1.1* . http://www.w3.org/TR/2001/NOTE-wsdl-20010315, 2001.
11. DAML Services Coalition. DAML-S: Semantic Markup for Web Services. http://www.daml.org/services, 2002.
12. *Universal Description, Discovery and Integration of Business for the web-Specifications*, http://www.uddi.org/specification,html
13. **BOTÍA, J.:** *METALA: Guía de uso, versión 1.0., Tech. Report*, Departamento de Ingeniería de la Información y las Comunicaciones, Universidad de Murcia, http://ants.dif.um.es/staff/juanbot/metala/manual.pdf, 2002.
14. Protégé 2000. http://www.protege.stanford.edu/
15. BeanGenerator http://www.swi.psy.uva.nl/usr/aart/beangenerator/