Diseño cooperativo basado en SOA para Sistemas Multi-Agentes

Mauricio Paletta

Departamento de Ciencia y Tecnología
Universidad Nacional Experimental de Guayana
Puerto Ordaz, Venezuela
e-mail: mpaletta@uneg.edu.ve

RESUMEN

En los últimos años se ha venido observando un creciente interés en el desarrollo de aplicaciones prácticas basadas en agentes que interactúan con entornos abiertos y distribuidos. Agentes inteligentes, capaces de interactuar de una forma realista con humanos y otros agentes, requieren de una arquitectura sofisticada que permita la integración de múltiples elementos. En este sentido, se debe diseñar una arquitectura para agentes inteligentes más flexible que haga énfasis en el proceso de aprendizaje, con el fin de satisfacer estos requerimientos. Por otro lado, el concepto de Arquitectura Orientada a Servicios (SOA – de sus siglas en inglés Service Oriented Architecture) es ampliamente usado hoy en día para el diseño de modelos para sistemas de Internet, siendo la tecnología de Web Services una de las más importantes. En este artículo se busca relacionar las tecnologías de agentes inteligentes y SOA para presentar una arquitectura abierta y extensible que permite satisfacer las necesidades requeridas. La forma en la cual esta arquitectura logra satisfacer estos objetivos es mediante la colaboración y cooperación con otros agentes que habitan en el entorno. También se presentan en este artículo, un conjunto de lenguajes basados en el estándar XML que han sido diseñados para lograr una apropiada representación de todos los elementos de información requeridos por la arquitectura presentada. La implementación y posterior evaluación de la arquitectura en escenarios simples, mostró que un agente inteligente era capaz de interactuar con un entorno cualquiera y aprender la forma en la cual este entorno está estructurado.

Palabras claves: Agente inteligente, aprendizaje, arquitectura, servicio, SOA, Web Service, XML.

ABSTRACT

A COOPERATIVE DESIGN FOR MULTI-AGENT SYSTEMS BASED ON SOA

A growing interest in the practical applications of the intelligent agent paradigm in open and distributed environments has been observed in recent years. Intelligent agents, capable of interacting realistically with humans and others agents, require a sophisticated architecture for the integration of different elements. A more flexible architecture for intelligent agents emphasizing on the learning process should be designed to fulfil these requirements. On the other hand, Service Oriented Architecture (SOA) is extensively used at present for the design of development models of Internet systems, being the Web Services technology one of the most important. In this paper, an attempt to relate the intelligent agents and SOA technologies is sought, in order to present an open and flexible architecture that has been designed to accomplish the requested necessities. This architecture intends to accomplish determined scopes and/or objectives through the collaboration and cooperation with other agents that inhabit the environment. This paper also presents a set of languages, based on XML standards that have been designed to get a more appropriate representation of all the required information elements. The implementation and evaluation of this architecture in simple scenarios showed the capacity of an intelligent agent to interact with any environment and to learn the way in which this environment is conformed.

Keywords: Architecture, intelligent agent, learning, service, SOA, Web Service, XML.

Artículo recibido el 15 de Noviembre de 2006 y aceptado en su forma final el 13 de Diciembre de 2007

I. INTRODUCCIÓN

Los sistemas de agentes inteligentes han llegado a ser el tema de investigación de una gran cantidad de trabajos en los últimos años. Estos sistemas son capaces de observar el entorno, mantener una representación interna del mundo que los rodea, tomar decisiones, ejecutar tareas, aprender, entre otros. Tal como lo expresa Muller [10] en la frase "los agentes son sistemas de software o hardware autónomos o semi-autónomos que ejecutan tareas en un entorno complejo que cambia dinámicamente".

Sin embargo, para poder satisfacer todas estas funcionalidades, es necesario contar con arquitecturas especializadas para diseñar este tipo de agentes inteligentes. Aún y cuando se han definido una gran cantidad de arquitecturas para agentes inteligentes, es importante enfatizar que en la actualidad, no se tiene una arquitectura que pueda satisfacer en su totalidad, todas las funcionalidades que, en general, los usuarios esperan tener de un agente inteligente. En este sentido, los usuarios de este tipo de sistemas tienen uno de dos posibles caminos: 1) definir su propia arquitectura, o 2) usar una arquitectura existente. Ambas opciones requieren esfuerzos de investigación adicional y pueda dar lugar a limitaciones o cambios en la línea de investigación original en la cual se quiere utilizar el agente.

En este sentido, ¿será posible combinar las habilidades de algunos de los agentes desarrollados hasta la fecha, con el objeto de integrar todas estas habilidades para configurar un agente inteligente más completo que cada uno de ellos por separado? De ser así, ¿cómo se puede lograr esta integración? Un planteamiento que da respuesta a estas interrogantes es desarrollado en este artículo, tomando en consideración la idea de asociación entre un agente y un servicio, mediante la utilización del concepto de SOA para definir una arquitectura para agentes inteligentes.

En este mismo orden de ideas, Martin y compañía [9] explican la estructura y elementos para la construcción de sistemas basado en agentes usando OAA [17] de sus siglas en inglés (Open Agent Architecture), en el cual, varios agentes se ayudan entre sí bajo la premisa de requerir y/o proveer servicios basado en sus capacidades. En este planteamiento, cada agente que participa en el sistema, define y publica un grupo de capacidades mediante el lenguaje ICL de sus siglas en inglés (Interagent Communication Language), también definido en OAA. Un facilitador mantiene una base de conocimiento que registra las capacidades de todos los agentes que forman parte del sistema y usa este conocimiento para ayudar a los solicitantes y suplidores de servicios a establecer el contacto necesario. En [7] los autores consideran una arquitectura abierta para agentes inteligentes. Otro factor importante presentado por estos autores es que ellos se refieren a la incorporación de nuevos agentes a partir del momento en que son desarrollados nuevos métodos apropiados al área en el cual se enfoca su arquitectura.

En este sentido, además de ser una arquitectura abierta, la característica de extensibilidad también se observa en este trabajo. De igual manera, De Antonio y compañía [4] muestran una arquitectura para el desarrollo de agentes inteligentes basado en un conjunto de agentes de software cooperativos. La idea de usar agentes para estructurar la arquitectura del agente inteligente es explotada en este estudio. Debido a la presencia de múltiples agentes que cooperan entre sí, el tema de la comunicación entre agentes toma gran importancia. En este sentido, Soriano y compañía [14] consideran un plan de interacción entre agentes que se comunican entre sí usando una semántica común, logrando un intercambio efectivo de conocimiento en entornos heterogéneos. Estos autores enfatizan la necesidad que tienen los agentes de considerar el conocimiento que se requiere para conceptualizar sus acciones de comunicación en un entorno respectivo.

Por otro lado, una relación conceptual entre las teorías de Web Services y agentes de software, específicamente agentes basados en la arquitectura BDI (de sus siglas en inglés Beliefs, Desires and Intentions), es presentada en Dickinson y otros [5]. Los autores identifican aquí la importancia de que cada agente contenga meta-data para describir sus servicios disponibles. En este artículo se presenta la descripción de una arquitectura

abierta y extensible para agentes inteligentes y que está basada en el concepto de SOA. Se describen algunos de los elementos que conforman esta arquitectura así como también una consideración especial en la asociación entre un agente y un servicio. Estos resultados son parte de la investigación que se está llevando a cabo en la Universidad Nacional Experimental de Guayana en colaboración de la Universidad Politécnica de Madrid.

En las siguientes secciones del artículo se muestra una relación conceptual entre un sistema de agentes inteligentes y un sistema Web Service (sección 1); el diseño de la arquitectura (sección 2); los lenguajes basados en XML usados para describir todos los elementos de información requeridos por la arquitectura (sección 3); una breve comparación entre la arquitectura presentada y SOA (sección 4); la forma en la cual fue implementada y evaluada esta arquitectura (sección 5) y finalmente, las conclusiones y trabajos futuros.

II. DESARROLLO

1. Agentes Inteligentes y Web Services

Los Web Services [20] son una de las tecnologías más importantes basadas en el concepto de SOA que están siendo masivamente utilizadas en la actualidad para el diseño de modelos de sistemas en Internet. Según la W3C [21], "un Web Service es un sistema de software diseñado para soportar interacción conjunta máquina-a-máquina sobre una red", tal como ocurre en los sistemas multi-agente que cooperan entre sí para satisfacer sus objetivos. En este sentido, los Web Services y los agentes de software inteligentes comparten una motivación en la búsqueda de una mayor flexibilidad y adaptabilidad para los sistemas de información; razón por la cual es natural considerar una relación conceptual entre estas dos tecnologías, basado en los siguientes temas comunes (ver [5]):

- No hay una distinción conceptual: No hay una diferencia conceptual entre un Web Service y un agente porque ambos son componentes activos en una arquitectura acoplada.
- Integración bi-direccional: Los agentes y los Web Services son semejantes porque entre ellos ocurre un proceso de comunicación para inter-operar entre sí.
- Los agentes invocan Web Services: Los Web Services son invocados por agentes como componentes de comportamiento, pero la autonomía y la intención son sólo representados a nivel del agente.

Si la relación entre los agentes y los Web Services se extiende de manera tal de considerar un agente como un servicio (en lugar de tener agentes y Web Services comunicándose entre sí para lograr los objetivos del agente), es posible definir un sistema multi-agente que puede ser visto como un único agente inteligente con un conjunto amplio de habilidades. En este sentido, es posible hablar de una Arquitectura Basada en Agentes AOA de sus siglas en inglés (Agent-Oriented Architecture) de la misma forma como lo establece la definición de la Arquitectura Basada en Servicios (SOA).

En la próxima sección se presenta una arquitectura abierta y extensible para agentes inteligentes basada en esta idea, mostrando la estructura interna usada para integrar todos los elementos requeridos por este diseño.

2. Diseño de una Arquitectura abierta y extensible

La arquitectura que se presenta en este artículo está basada en un conjunto de agentes que cooperan entre sí para satisfacer los objetivos del agente inteligente. Cada uno de estos agentes está especializado en un servicio particular asociado con alguna habilidad: deliberar, reaccionar, socializar, interactuar con el ambiente, aprender, entre otros. Estos agentes están divididos en dos grupos: un Agente de Control (AC) y varios Agentes de Servicio (AS).

En la figura 1a se puede observar la arquitectura básica presentada. En este caso la estructura está conformada únicamente por dos elementos: el AC y un área de Memoria Compartida (MC) entre todos los agentes. Esta área de memoria preferiblemente está localizada en un espacio, ubicada en el mismo sitio donde el AC reside, el cual todos los agentes conocen y deben ser capaces de acceder a el.

Tal como puede ser observado en la figura 1a, la MC está formada por documentos que repre-

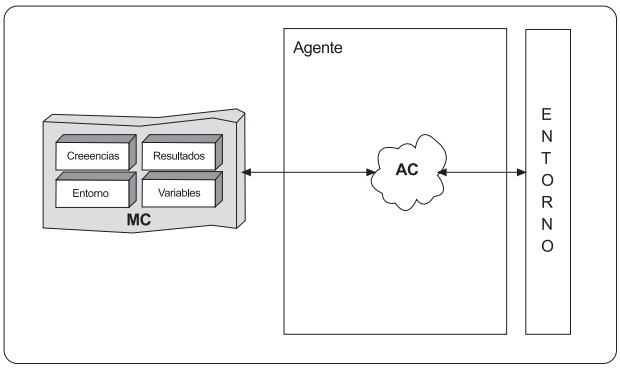

Figura 1a. Arquitectura básica

sentan y almacenan los siguientes elementos de información: la descripción del entorno, las creencias del agente, las conclusiones y resultados obtenidos por los agentes y las variables y estados internos que se requieran. Estos documentos, al igual que toda la información representada y manejada por esta arquitectura, están basados en el estándar XML (ver [18]).

Esta información representa lo mínimo necesario requerido por todos los agentes, tanto el AC como los AS, para mostrar sus habilidades y lograr los objetivos específicos para los cuales ellos han sido diseñados. Los agentes comparten el conocimiento sobre el entorno con el cual ellos están interactuando, así como también las creencias generales del agente inteligente. De las conclusiones y resultados obtenidas de otros agentes, un agente específico puede ser capaz de: ejecutar funciones particulares como por ejemplo el aprendizaje; reforzar una respuesta, en caso de que se compartan habilidades con otros agentes y, requerir la colaboración de otros agentes como estímulo para generar su propia respuesta. Por último, las variables y estados internos representan un conjunto de cualquier otra información que debe ser compartida entre los agentes.

Este diseño sigue los principios básicos de un modelo de arquitectura de pizarrón [15], en el cual los agentes comparten la información disponible sobre el estado del sistema y el entorno. En este sentido, cada agente obtiene del pizarrón la información que le es útil y luego escribe en el mismo pizarrón los resultados obtenidos. Los AS pueden ser agregados / removidos del sistema de agentes, originando un incremento / decremento de las habilidades del agente general. Cada uno de estos agentes maneja su propia base de conocimiento y/o memoria particular necesarios para cumplir con sus objetivos específicos. Ellos no tienen interacción directa con el entorno a menos que estén relacionados con servicios especializados asociados con las funciones típicas de interfaz con el entorno: manejo del sonido, procesamiento de imágenes, procesamiento de lenguaje natural, entre otros. Este trato particular con el entorno es importante para los sistemas multi-agente ya que debe haber una continua interacción con, no sólo el entorno donde cada agente reside, sino también con el resto de los agentes que residen en ese mismo entorno. Un mismo servicio puede ser dado por varios agentes a la vez, lo que permite reforzar la habilidad del agente general asociado con este servicio. El AC debe resolver cualquier posible inconsistencia. La figura 1b muestra el cambio de la arquitectura básica luego de la incorporación de dos AS, uno de ellos con la habilidad de interactuar con el entorno.

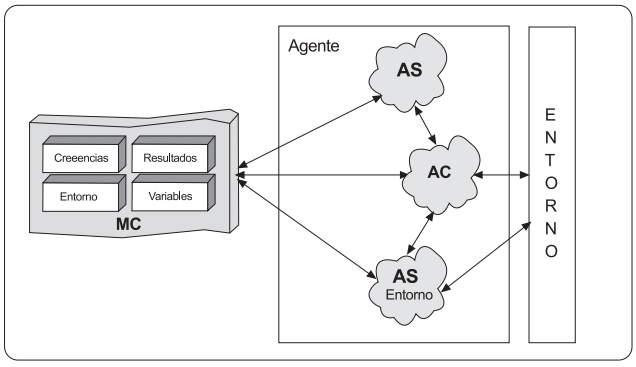

Figura 1b . Arquitectura con la incorporación de dos AS, uno de ellos relacionado con el entorno.

La manera en la cual un AS se incorpora en esta estructura sigue la misma idea considerada en los Web Services para la publicación de los servicios. Así como existe un documento WSDL de sus siglas en inglés (Web Service Description Language) que describe el contenido de un Web Service, hay un correspondiente documento XML que describe las habilidades del AS. Las características de este documento se presentan más adelante en este artículo.

La función principal de los AS es cubrir la habilidad que les ha sido asignada, además de mantener comunicación continua de sus resultados con el AC. Los AS pueden ser elaborados siguiendo arquitecturas ya conocidas de agentes, especializadas en la habilidad específica que deben soportar.

Basado en una arquitectura del tipo BDI de sus siglas en inglés (Beliefs, Desires and Intentions) [13], una estructura para los AS tiene ciertas similitudes con la del AC con algunas diferencias particulares. Para poder incorporar el AS al sistema multi-agente, el AS debe implementar, como mínimo, el mecanismo de comunicación con el AC.

La comunicación entre un AS y el AC se lleva a cabo mediante un flujo de información en formato XML, de la misma forma como es usado SOAP de sus siglas en inglés (Simple Object Access Protocol) en los sistemas de Web Service. Un canal de comunicación TCP/IP cliente/servidor es abierto entre el AS y el AC. El AC está constantemente escuchando el canal lo que permite al AS conectarse en cualquier momento.

Lo primero que debe hacer el AS una vez que se conecta con el AC, es enviar el documento XML correspondiente a la descripción de sus habilidades. Luego de esto, el AC modifica su base de conocimiento con la lista de habilidades actualmente soportada por el agente inteligente.

Por otro lado, el AC es la pieza principal de la arquitectura. Su principal función es la de configurar una estructura que agrupe todas las habilidades que, no sólo ofrece el mismo AC sino todos los AS con los cuáles él interactúa. Esto permite que el agente sea visto desde afuera, por el resto de los agentes que lo rodea, como un agente inteligente equipado con todas estas habilidades. Adicionalmente, el AC es el elemento clave que permita hacer que esta arquitectura sea extensible. Sus funciones se pueden resumir como sigue:

- Tener una interacción básica con el entorno.
- Establecer los objetivos finales del agente y preparar el plan para satisfacer estos objetivos.
- Comunicarse con los AS.
- Verificar que los AS ejecuten sus tareas.
- Sintetizar la información recibida por los AS.
- Manejar el proceso de aprendizaje con los AS que corresponden con esta habilidad.

Para poder cumplir con estas funciones, el AC está formado por (ver figura 2):

- Una base de conocimiento responsable de contener: los objetivos, los planes, las habilidades actuales provistas por los AS con los cuales hay interacción y, el conocimiento aprendido.
- Los siguientes módulos funcionales:

  Módulo de comunicación: es responsable de manejar, no solo el canal de conexión con los AS sino también el protocolo de comunicación.

  Módulo de planificación: contiene los algoritmos necesarios que permiten al agente ejecutar sus planes de acción para la satisfacción de objetivos.

  Módulo de aprendizaje: contiene el conjunto de algoritmos básicos de aprendizaje; integra además, el aprendizaje con los AS asociados con esta habilidad.

  Módulo de interacción: maneja la interacción básica del agente con el entorno.

  Módulo central: sincroniza la ejecución del resto de los módulos y permite la interacción de la estructura con la MC.

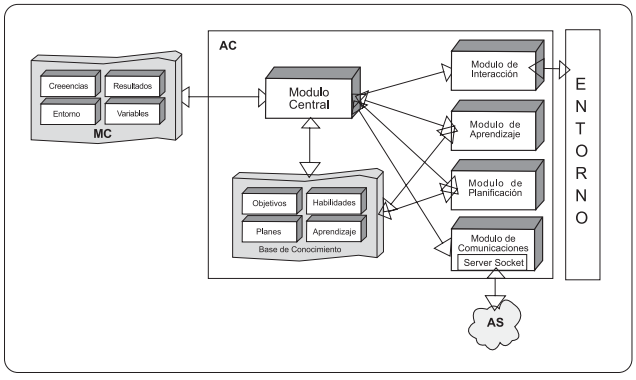

Figura 2. Estructura interna del AC.

La figura 3 muestra una relación conceptual entre estos elementos haciendo uso de un diagrama de clases UML. De este diagrama se puede extraer la siguiente información: 1- el AC está compuesto por la base de conocimiento y los módulos de aprendizaje, comunicación, interacción, planificación y central; 2- los módulos de aprendizaje, comunicación, interacción y planificación dependen

del módulo central, particularmente por el uso de la MC; 3- el módulo central usa la MC; 4- el módulo de comunicación es el enlace para la comunicación con los AS; 5- los módulos de aprendizaje, planificación y central usan la base de conocimiento; 6- el módulo de interacción es el canal de interacción con el entorno; y, 7- el AS y el AC son tipos particulares de agentes.

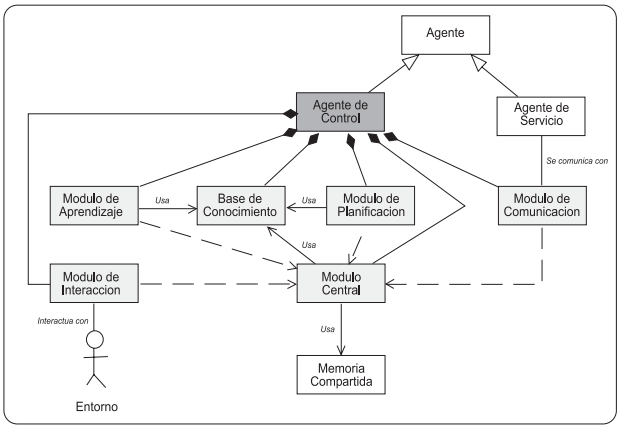

Figura 3. Diagrama conceptual de la estructura interna del AC.

Por otro lado, el diagrama de colaboración UML ilustrado en la figura 4 permite describir con mayor detalle la relación entre los elementos que conforman la estructura del AC. Los enlaces etiquetados no representan un orden específico y se interpretan de la siguiente manera: 1- recepción del estímulo desde el entorno; 2- modificación de la MC; 3- aprendizaje; 4- modificación del conocimiento; 5- comunicación con los AS que tienen la habilidad de aprender; 6- recepción de la respuesta desde los AS; 7- obtención del conocimiento asociado a los planes y objetivos; 8- modificación de los resultados; 9- reporte de resultados; 10- preparación para actuar con el entorno; 11- actuación.

Como un complemento del diseño de la arquitectura antes presentado, la siguiente sección describe la manera en la cual son representados los diferentes elementos de información requeridos.

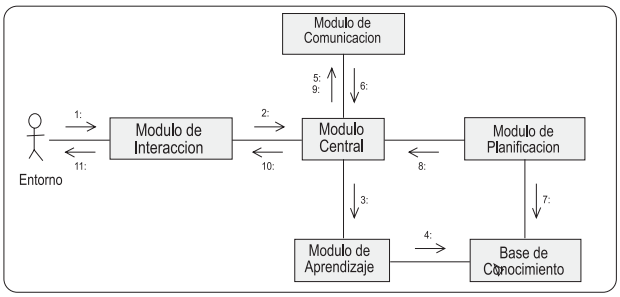

Figura 4. Diagrama de colaboración entre los elementos del AC.

3. Lenguajes basados en XML

Con el objeto de soportar la característica de arquitectura abierta, XML, debido a su simplicidad y flexibilidad, fue utilizado como base para la representación de todos los elementos de información requeridos por el agente inteligente. Estos elementos se pueden resumir básicamente en: descripción del ambiente, descripción del AS, comunicación entre el AS y el AC y, el conocimiento. En este sentido fueron diseñados una serie de lenguajes XML para la representación y manejo de estos elementos de información, los cuales serán brevemente descritos en las siguientes sub-secciones.

3.1. Un lenguaje para la descripción de los AS

Con el objeto de identificar y describir las habilidades o servicios de un AS, se ha definido el lenguaje IA-SADL de sus siglas en inglés (Service Agent Description Language for Intelligent Agents), tomando como base las etiquetas o elementos descriptivos definidos por DAML [16] de sus siglas en inglés (DARPA Agent Markup Language) y AML [6] de sus siglas en inglés (Agent Markup Language).

El archivo IA-SADL de un agente contiene tres niveles de información: 1- información básica relativa a la implementación, tales como la versión y autor; 2- el conjunto de habilidades que el agente soporta; y 3- el conjunto de servicios o comandos al cual el agente responde. En la figura 5 se puede observar un ejemplo.

```
<IA-SADL:Agent name="un nombre">
  <IA-SADL:Bas ic>
    <IA-SADL:Version> número de versión </IA  -SADL:Version>
    <IA-SADL:Author> nombre del autor </IA  -SADL:Author>
    <IA-SADL:Author_url> url del autor </IA  -SADL:Author_url>
    <IA-SADL:Author_email> email del autor </IA  -SADL:Author_email>
  </IA-SADL:Basic>
  <IA-SADL:Abilities quantity="2">
    <IA-SADL:Ability -1> habilidad 1 </IA -SADL:Ability -1>
    <IA-SADL:Ability -2> habilidad 2 </IA -SADL:Ability -2>
  </IA-SADL:Abilities>
  <IA-SADL:Services quantity="1">
    <IA-SADL:Service -1 name="Nombre del servicio">
      <IA-SADL:Parameters quantity="1" >
        <IA-SADL:Parameter -1 name="um nombre" type="um tipo"/>
      </IA-SADL:Parameters>
      <IA-SADL:Result type="un tipo" value="un valor"/>
    </IA-SADL:Service -1>
  </IA-SADL:Services>
</IA-SADL:Agent>
```

Figura 5. Ejemplo de un documento escrito en IA-SADL

Una vez que el AC recibe de un AS esta información, puede dar respuesta a las siguientes interrogantes:

- ¿Cómo identificar el AS?
- ¿Cómo saber que este agente está disponible para atender una habilidad específica?
- ¿Cómo enviar al AS las solicitudes?
- ¿Cómo es la respuesta que se recibe de cada solicitud?

3.2. Un lenguaje para la comunicación entre agentes

Para satisfacer las necesidades de comunicación entre el AC y los AS, se ha definido el lenguaje IA-VACL de sus siglas en inglés (Virtual Agent Communication Language for Intelligent Agents). Para la definición de este lenguaje fueron considerados el estándar definido en la versión XML de FIPA-ACL de sus siglas en inglés (Foundation for Intelligent Physical Agents-Agent Communication Language) y algunas etiquetas descritas en los trabajos de Choi y compañía [2] y de Makatchev y compañía [8].

El lenguaje IA-VACL permite representar las necesidades del AC para enviar un requerimiento de servicio, así como también, las forma en la cual los AS retornan su respuesta. Esto se logra mediante un intercambio de mensajes, cada uno de los cuales contentivos de la siguiente información: 1- La identificación del agente que hace la solicitud; 2- La identificación del servicio o proceso requerido; 3- Los parámetros asociados al servicio, si se trata de un requerimiento; y 4- El resultado correspondiente, si se trata de la respuesta a una solicitud. La figura 6 presenta un ejemplo.

```
<IA-VACL:Message sender="id de un agente">
  <IA-VACL:Service>
    <IA-VACL:Name> nombre del servicio </IA  -VACL:Name>
    <IA-VACL:Parameters quantity="2">
      <IA-VACL:Parameter -1> par1 </IA-VACL:Parameter -1>
      <IA-VACL:Parameter -2> par2 </IA-VACL:Parameter -2>
    </IA-VACL:Parameters>
    <IA-VACL:Result> el resultado </IA  -VACL:Result>
  </IA-VACL:Service>
</IA-VACL:Message>
```

Figura 6. Ejemplo de un mensaje escrito en IA-VACL

3.3. Un lenguaje para la descripción del entorno

Se define el lenguaje IA-VWDL de sus siglas en ingles (Virtual World Description Language for Intelligent Agents), basado en los lenguajes ELMS [11] de sus siglas en inglés (Environment Description Language for Multi-Agent Simulation) y EDL [1] de sus siglas en inglés (Environment Description Language), que utilizan XML para describir el entorno, también conocido como WD [12] de sus siglas en inglés (World Description Knowledge Base), como una forma de representar el mundo virtual. En este sentido, IA-VWDL es una adaptación de WD basado en las premisas de ELMS y EDL.

En IA-VWDL, el lugar o área (mundo virtual) donde está localizado el agente, es un "mapa"; los objetos observados en el mapa son "miembros"; entre los miembros de un mapa se establecen "relaciones". Cada miembro puede estar asociado con un nombre, tipo, ubicación (coordenadas), color, entre otros. Las relaciones tienen un operador y uno o más operandos que pueden ser miembros del mapa o el mapa mismo. Cada archivo IA-VWDL es un mapa, de manera tal que, cada vez que los agentes cambian de entorno, hay un mapa diferente correspondiente. El nombre del mapa y el nombre del archivo IA-VWDL coinciden y, este nombre a su vez, identifica un mundo virtual particular observado por el agente inteligente.

3.4. Representación del conocimiento

Además de la información asociada al entorno, se requieren también documentos XML para representar los otros niveles de información almacenados en la MC: creencias, resultados y variables. En este sentido, se definen los lenguajes IA-BDL de sus siglas en inglés (Beliefs Description Language for Intelligent Agents) e IA-TEL de sus siglas en inglés (Temporally Elements Language for Intelligent Agents).

Para la definición de IA-BDL fue considerado el trabajo de Clark y compañía [3] en el cual, los autores plantean una representación de las creencias del agente haciendo uso de axiomas de una antología basada en RDF (ver [19]).

4. Comparando la Arquitectura presentada con SOA

Tal como se puede apreciar en el diagrama conceptual de la figura 7a, una aplicación final usa una interfaz abierta UDDI (de sus siglas en inglés Universal Description, Discovery, and Integration), para obtener el documento WSDL que permite integrar un Web Sercice a la aplicación. Esto sigue el principio de SOA asociado con, no sólo la construcción y uso de una aplicación final basado en uno o más servicios, Web Services en este caso, también hace que la comunicación con estos servicios sea transparente para el usuario final de la aplicación.

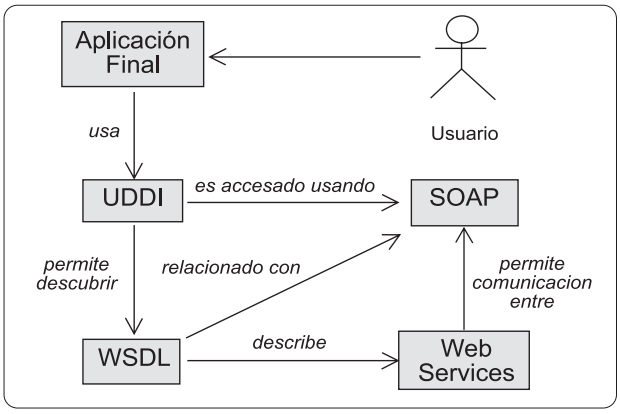

Figura 7a. Relación entre los elementos de un sistema Web Services.

Por otro lado, el agente inteligente usa el AC (figura 7b) para integrar todas las habilidades o servicios soportados por los AS. El documento IA-SADL es utilizado para descubrir y describir al AS mientras que los mensajes IA-VACL son usados para la comunicación entre los AS y el AC, de la misma forma como SOAP es usado en los sistemas Web Services.

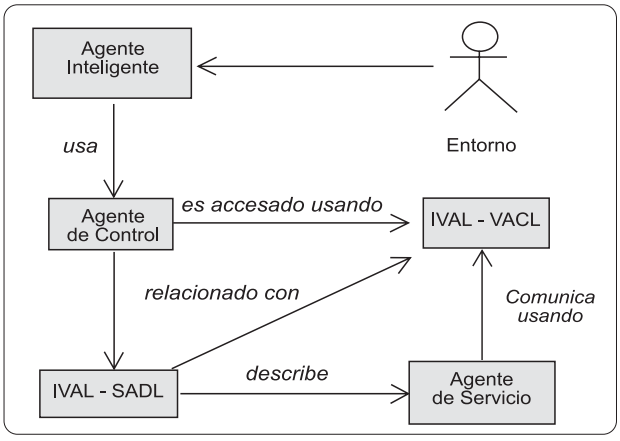

Figura 7b. relación entre los elementos de la arquitectura presentada.

Una diferencia importante entre estas dos arquitecturas está asociada con el inicio del proceso de comunicación. En los sistemas Web Servives, la comunicación es iniciada por la aplicación mientras el Web Service espera por la conexión. En la arquitectura presentada, la comunicación se inicia en un AS con el objeto de que éste sea integrado a la estructura del agente inteligente. En este caso, el AC espera por la conexión.

Por otro lado, una similitud importante entre ambas arquitecturas, es la manera en la cual todos los servicios o habilidades se integran en un único elemento de interacción con el exterior.

5. Implementación y Evaluación

Para implementar los conceptos relacionados con la arquitectura presentada en este artículo, se desarrolló una biblioteca de clases en **C#**. Esta biblioteca contiene las clases para la definición de cada uno de los módulos que componen el AC, la MC y cada uno de los documentos XML requeridos. La comunicación entre el módulo central y el resto de los módulos se lleva a cabo mediante el intercambio de mensajes propio de la programación orientada a objetos. Con el propósito de evaluar la arquitectura presentada, se desarrolló un agente que interactúa y aprende sobre diferentes entornos, primero sin tener ningún AS y luego, con la incorporación de dos AS, uno de ellos con la habilidad de interactuar con el entorno y el otro con la habilidad de aprender. La integración de estos dos AS demostró que las habilidades del agente inteligente crecen a medida que se incorpora un nuevo AS.

Para la simulación se implementó un agente que interactuó con un entorno formado por un conjunto de objetos. El agente tenía que seguir un plan simple que incluía seguir una ruta a través del entorno y construir un mapa del mismo. Una vez que el agente completó el plan y construyó el mapa, fue consultado sobre los objetos observados para verificar la manera en la cual se realizó el aprendizaje del entorno. Con el objeto de probar la capacidad del agente para mantener actualizado el mapa y el conocimiento aprendido, la simulación fue complicada agregando más objetos en el entorno o cambiando la posición de los objetos existentes.

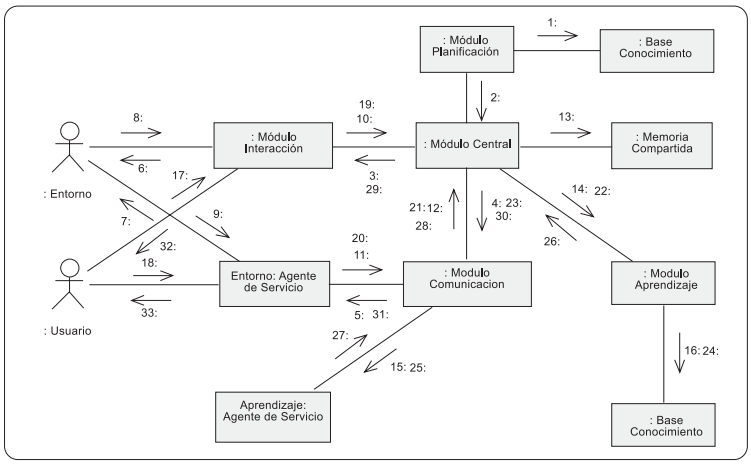

Figura 8. Diagrama de colaboración que ilustra el proceso de evaluación.

La figura 8 muestra un diagrama de colaboración UML que sintetiza el proceso descrito anteriormente: 1- leer el plan; 2- reportar la respuesta; 3- informar la necesidad de observar el entorno; 4,5 – comunicar el requerimiento (AS con habilidad de interacción con el entorno); 6,7- observar el entorno; 8,9- recibir el estímulo; 10,11,12 – informar las observaciones recibidas; 13- construir el mapa; 14- aprender el mapa; 15- comunicar el requerimiento (AS con habilidad para aprender); 16- modificar la base de conocimiento; 17,18 – consultar sobre el entorno; 19, 20,21 – informar sobre lo consultado; 22,23,25 – buscar experiencia; 24 – revisar en la base de conocimiento; 26,27,28 – responder sobre la experiencia; 29,30,31,32,33 – actuar.

III. CONCLUSIONES

Este artículo presenta la estructura y elementos necesarios que definen una arquitectura abierta y extensible para agentes inteligentes, basado en la cooperación de otros agentes y haciendo énfasis en el proceso de aprendizaje.

Se ilustra la forma en la cual los conceptos de SOA y OAA fueron usados para definir las especificaciones de esta arquitectura para agentes inteligentes y la manera en la cual la asociación entre agente y servicio facilitó la comprensión que permitió relacionar estos dos conceptos.

Gracias a que la comunicación entre los AS y el AC se hace mediante el protocolo TCP/IP, los AS y el AC pueden estar localizados en sitios físicos diferentes y la cooperación entre ellos se puede seguir dando. En este sentido, este trabajo puede ser usado como base para proponer otras arquitecturas para el desarrollo de sistemas distribuidos.

Cabe destacar la forma en la cual XML, por su simplicidad y flexibilidad, fue utilizado como estándar para la definición de un conjunto de lenguajes que permiten representar y almacenar todos los elementos de información requeridos por la arquitectura presentada.

A partir de la evaluación realizada se puede concluir que un agente inteligente implementado sin la incorporación de ningún AS, tiene las funcionalidades mínimas requeridas para tomar la información del entorno, realizar un proceso básico de aprendizaje y actuar según lo aprendido.

La incorporación de diferentes AS permite obtener un agente mucho más capacitado, de acuerdo a las habilidades de estos AS. Una vez que la biblioteca de clases y la correcta integración entre el AC y los AS se hayan evaluado sobre escenarios simples, el próximo paso es integrar esta biblioteca en sistemas y entornos virtuales más completos, a fin de producir aplicaciones reales relativas a agentes inteligentes. Una de las áreas que se ha considerado es el aprendizaje automatizado, como por ejemplo el relacionado con estrategias militares, fraudes financiero bancarios, entre otros.

IV. REFERENCIAS

1. Chella A., Cossentino M., Tomasino G., *An environment description language for multirobot simulations.* Proc. of ISR (2001), Seoul, Korea.

2. Choi J., Kim Y., Kang S., *An Agent Communication Language Using XML and RDF*. Session A5, DBPIA, (2001), pp. 61-64.

3. Clark K. L., McCabe F. G., *Ontology schema for an agent belief store.* Journal Special Issue Article; International Journal of Human Computer Systems; ISSN 1071-5819. (2006).

4. De Antonio A., Ramírez J., Méndez G., *An Agent-Based Architecture for Virtual Environments for Training*. In Developing Future Interactive Systems**,** Chapter VIII, Idea Group Publishing, ISBN 1591404118. (2005).

5. Dickinson I., Wooldridge M., *Agents are not (just) web services: investigating BDI agents and web services.* HP Labs 2005 Technical Reports. Proc. SOCABE 2005.

6. Garbe W.; AML - *Agent Markup Language – Syntax Description - versión 1.1*. (2002). [En línea]. Disponible: http://www.bingooo.com/images/BINGOOO_sintax_1_1-engl.pdf Diciembre 2006.

7. Gurer D., Lakshminarayan V., Sastry A., *An Intelligent-Agent-Based Architecture for the Management of Heterogeneous Networks.* Proc. Ninth Annual IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, Delaware,USA. (1998).

8. Makatchev M., Tso S. K., *Human-Robot Interface Using Agents Communicating in an XML-Based Markup Language*. Proc. IEEE International Workshop on Robot and Human Interactive Communication, Osaka, Japan. (2000).

9. Martin D. L., Cheyer A. J., Moran B. D., *The Open Agent Architecture: A Framework for Building Distributed Software Systems.* Applied Artificial Intelligence, Vol. 13, Números 1-2, 21-128. (1999).

10. Muller J. P., *The Design of Intelligent Agents: a Layered Approach.* LNAI, 1177, Springer, Berlin. (1996).

11. Okuyama F. Y., Bordini R. H., da Rocha Costa A. C., *ELMS: an environment description language for multi-agent simulations.* Proc. of the First International Workshop on Environments for Multiagent Systems (E4MAS), held with AAMAS-04, number 3374 in Lecture Notes in Artificial Intelligence, 91-108. Berlin: Springer- Verlag. (2005).

12. Panayiotopoulos T., Katsirelos G., Vosinakis S., Kousidou S., *An Intelligent Agent Framework in VRML Worlds.* Kluwe Academic Publishers, Netherlands. (1999).

13. Rao A. S., Georgeff M. P., *Modeling Rational Agents within a BDI-Architecture.* Proc. Second International Conference on Principles of Knowledge Representation and Reasoning, San Mateo, CA, USA. (1991).

14. Soriano J, López G., Alonso F., *Policy Infrastructure as an Extension to the FIPA Abstract Architecture for Open Agent Platform Design.* Net Object Days (International Workshop on Agents and Software Engineering), Erfurt, Germany. (2002).

15. Vranes S., Stanojevic M., *Integrating Multiple Paradigms within the Blackboard Architecture.* IEEE Transactions on Software Engineering, Vol. 21, Number. 3, 244-262. (1995).

16. The DARPA Agent Markup Language Homepage. [En línea]. Disponible: http:/ www.daml.org/ . Diciembre 2006.

17. The Open Agent Architecture ; A framework for integrating a community of heterogeneous software agents in a distributed environment. [En línea]. Disponible: http:// ai.sri.com/oaa/. Diciembre 2006.

18. W3C; *Extensible Markup Language (XML).* (2003). [En línea]. Disponible: http:// www.w3.org.XML/. Diciembre 2006.

19. W3C; *Resource Description Framework (RDF)*. (2004). [En línea]. Disponible: http:// www.w3.org.RDF/. Diciembre 2006.

20. W3C; *Web Services Activity;* [En línea]. Disponible: http:// www.w3.org./2002/ws/. Diciembre 2006.

21. W3C; *Web Services Architecture;* W3C Working Group Note 11, February 2004; [En línea]. Disponible: http:// www.w3.org/TR/2004/NOTE-ws-arch-20040211/. Febrero 2004.