HERRAMIENTA PARA LA ESPECIFICACIÓN DEL ESPACIO DE INTERACCIÓN DE AGENTES DE LA PLATAFORMA CAPNET

TOOL FOR THE SPECIFICATION OF THE INTERACTION SPACE OF AGENTS OF THE CAPNET PLATFORM

Jaime Arturo Villaseñor-Marcial[1], Jesús Adolfo Rodelo-Moreno[1], Ernesto German[2]
[1]Facultad de Informática Mazatlán, Universidad Autónoma de Sinaloa. Mazatlán, Sinaloa, México
[2] Instituto Mexicano del Petróleo, México
E-mail: jesusrodelo.20@gmail.com

(*Enviado Diciembre 18, 2012; Aceptado Febrero 01, 2013*)

Resumen
La ingeniería del software orientado a agentes requiere el desarrollo de herramientas computacionales que ayuden a modelar las interacciones que se realizan entre agentes. En este artículo se describe el desarrollo de una herramienta que facilita la especificación de los atributos necesarios para configurar las capacidades de interacción de agentes que utilizan el lenguaje FIPA-ACL. Estas capacidades se agrupan en el concepto de espacio de interacción. La herramienta está diseñada para formar parte del ambiente de desarrollo de la plataforma CAPNET, permitiendo a los programadores de aplicaciones, la creación de agentes de una manera sistemática, y al mismo tiempo, asegurando la compatibilidad con la arquitectura de interacción de dicha plataforma.

Palabras Clave: *Agentes, Sistemas Multi-Agente, FIPA-ACL, Interacción, Ingeniería de Software Orientado a Agentes, CAPNET.*

Abstract
The engineering of the software oriented to agents requires the development of computational tools that help to model the interactions that are made between agents. This article describes the development of a tool that facilitates the specification of the necessary attributes to configure the interaction capabilities of agents that use the FIPA-ACL language. These capabilities are grouped in the concept of interaction space. The tool is designed to be part of the development environment of the CAPNET platform, allowing application programmers to create agents in a systematic way, and at the same time, ensuring compatibility with the platform's interaction architecture.

Keywords: *Agents, Multi-Agent Systems, FIPA-ACL, Agent Oriented Software Engineering, CAPNET.*

1 INTRODUCCIÓN

Los agentes inteligentes son entidades computacionales autónomas capaces de resolver problemas, que operan interactuando de una manera efectiva dentro de ambientes abiertos y dinámicos llamados Sistemas Multi-Agente (SMA) [1].

En este tipo de aplicaciones, las interacciones se realizan a través de un lenguaje de comunicación (ACL) que permite el intercambio de mensajes con un contenido semántico (este tipo de ACL surgieron a partir de la teoría de actos del habla de [2]) tomando en cuenta dos características principales de los agentes: autonomía e interoperabilidad entre agentes heterogéneos.

Para que el agente pueda interactuar en un entorno abierto y comunicarse con agentes heterogéneos, debe disponer de la infraestructura para la programación, ejecución y despliegue de SMA. La investigación en este campo ha establecido la necesidad de desarrollar lenguajes de programación y herramientas apropiados para la implementación de estos sistemas [3].

A pesar de un desarrollo impresionante en los últimos años, para que la tecnología de agentes se refleje del laboratorio a la práctica industrial, se requieren resolver varios problemas de ingeniería de software basado en agentes [4]. Estos problemas radican en el hecho de que (i) Las herramientas para facilitar el desarrollo de diferentes tipos de agentes prácticamente no existen (ii) Problemas de interoperabilidad de agentes con otro software distribuido y (iii) Problemas de reutilización de agentes.

En este contexto, sobresalen los esfuerzos de FIPA (*Foundation for Intelligent Physical Agents*), cuya finalidad es estandarizar el uso de la tecnología de agentes

definiendo los servicios para construir una infraestructura que permita la interoperabilidad entre agentes heterogéneos [5] y un lenguaje de comunicación de agentes llamado FIPA-ACL.

FIPA-ACL engloba habilidades de interacción inteligente en los agentes por medio de una serie de requerimientos. El contenido de un mensaje debe ser expresado en un lenguaje de contenido, referenciar parámetros de la comunicación para controlar conversaciones de manera explícita por medio de protocolos de interacción y ontologías del dominio de las aplicaciones. Su finalidad es establecer el contexto de las interacciones y el significado de los conceptos que están siendo utilizados en el contenido [6].

Esta plataforma consiste en un ambiente de ejecución que soporta el despliegue de sistemas multi-agente, un ambiente de desarrollo en forma de plantillas de agentes, herramientas de programación, galería de componentes y una serie de componentes para la integración con aplicaciones empresariales [7].

Actualmente, en la investigación asociada a CAPNET se desarrollan tres aspectos que abordan el problema de la interacción: 1) Definición de modelos de interacción, 2) Una Arquitectura de interacción que integra dichos modelos y 3) Un ambiente de desarrollo de SMA. Con los modelos de interacción se trata de sistematizar el uso correcto de los atributos del mensaje FIPA-ACL sobre los que se construyen la autonomía e interoperabilidad de estos agentes.

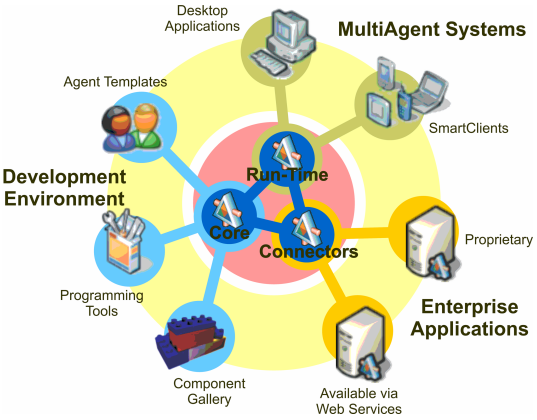

Figura 1 Arquitectura CAPNET.

CAPNET es una plataforma de agentes compatible con las especificaciones de FIPA, por tanto, sus agentes utilizan FIPA-ACL para realizar sus interacciones (Fig. 1).

La arquitectura de interacción es un componente del agente que integra un conjunto de capacidades que permiten la ejecución de interacciones dentro del espacio de interacción de un agente. Formalmente, se define por medio de una estructura que se describe como sigue:

AI = < S,Messaging,CLE,OE,CIE,SIE > donde:

- S: es el espacio de interacción del agente
- Messaging: Permite conectar a un agente con los canales de comunicación físicos disponibles.
- CLE es el motor de lenguajes de contenido
- OE es el motor de ontologías
- CIE es el motor de procesamiento de interacciones compuestas
- SIE es el motor de procesamiento de interacciones simples.

Para que un agente interactúe en tiempo de ejecución, resulta fundamental que sea capaz de conocer sus capacidades. El objetivo del espacio de interacción es mantener dichas capacidades accesibles al agente para realizar los modelos de interacción; este se define como:

S=< C, KB, L, O, P, IM > donde:

- C es el conjunto de los canales de comunicación compatibles con el Servicio de Transporte de Mensajes (MTS) de FIPA.
- KB es un componente que permite administrar el conocimiento que el agente tiene acerca de su dominio de aplicación.
- L es el conjunto de lenguajes de contenido que el agente puede utilizar para validar la expresión de contenido de los mensajes que maneja.
- O es el conjunto de ontologías de los dominios de aplicación que el agente puede utilizar para validar la semántica de la expresión de contenido de los mensajes que maneja.
- P es el conjunto de protocolos de interacción que el agente puede utilizar para manejar interacciones compuestas.
- IM es el conjunto de modelos de interacción que puede procesar un agente.

La especificación sistemática, fácil y consistente del espacio de interacción son metas que ayudarán a construir agentes que aprovechen los beneficios de tener esta arquitectura de interacción en CAPNET. Esta es la motivación que ha dado origen al desarrollo de la herramienta que a continuación se describe. Los usuarios directos de esta herramienta serán los programadores de agentes, ya que está diseñada para formar parte del ambiente de desarrollo de SMA de la plataforma CAPNET.

2 METODOLOGÍA

La metodología de investigación que ha seguido este trabajo está formada por varias fases cuyos antecedentes se han introducido previamente. La primera fase se basa en el modelo teórico de la arquitectura de interacción desarrollado en la plataforma CAPNET.

Enseguida, para la segunda fase, se ha estudiado el enfoque de modelos de interacción para llevarlos a la práctica por medio de implementaciones en entornos computacionales, lenguajes, herramientas y desarrollo de

aplicaciones con agentes. Esto ha llevado a determinar los requerimientos de la interacción con el lenguaje FIPA-ACL que son importantes durante el diseño de las capacidades de interacción de los agentes.

La tercera fase de la metodología consistió en el desarrollo de técnicas y herramientas para conformar el espacio de interacción de los agentes en base a criterios y estructuras que podrán restringir, validar y regular la comunicación entre agentes.

Para fines de probar la utilidad de la arquitectura de interacción de CAPNET, se desarrolló la herramienta que aquí se presenta para permitir la especificación de las abstracciones que conforman el espacio de interacción y sus requerimientos tomando en cuenta su integración con el ambiente de desarrollo de CAPNET.

Finalmente, la metodología utilizada comprende una revisión del estado del arte de la tecnología de agentes y plataformas de agentes (JADE, FIPAOS, ZEUS son algunas de ellas), lo que ha permitido evaluar el funcionamiento de las plataformas de desarrollo de sistemas multiagentes más utilizadas en la actualidad en el aspecto de la disposición de herramientas para el diseño de especificaciones de agentes y la programación de interacciones en tiempo de ejecución.

3 RESULTADOS

En esta sección se explican los detalles de la herramienta que permite la especificación del espacio de interacción de los agentes de la plataforma CAPNET. Su codificación ha sido a través del lenguaje de programación C#, utilizando el entorno de desarrollo de Microsoft Visual Studio.NET 2003.

El funcionamiento general de la herramienta se analiza mediante tres aspectos principales: (i) el usuario captura los atributos del espacio de interacción, (ii) Al mismo tiempo, se valida y almacena cada uno de ellos en su objeto correspondiente, (iii) Se genera un archivo XML con las especificaciones.

El objetivo principal de la especificación guardada en el archivo XML, es que un agente de CAPNET cargue la información del espacio de interacción, una vez que va a empezar su ejecución. Finalmente, el agente básico de CAPNET está implementado para que el espacio de interacción creado desde dicho XML, se integre a la arquitectura de interacción, como puede apreciarse en la Figura 2.

A continuación se presentan los resultados obtenidos, divididos en cuatro secciones. En ellas se describen los componentes principales de la herramienta, sus características funcionales, la forma en que se da la integración de la herramienta con CAPNET y sus ventajas.

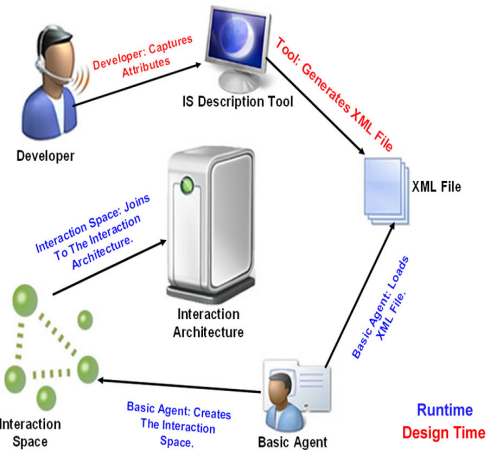

Figura 2 Diagrama funcional de la herramienta.

3.1 Descripción de los componentes

La herramienta permite al programador de agentes especificar, en ambiente gráfico, una serie de atributos para formar el espacio de interacción (ver Fig. 3) de acuerdo con los siguientes componentes:

1.1. Especificación de lenguajes de contenido (*Content Language*).
1.2. Captura de ontologías (*Ontology*).
1.3. Especificación de los protocolos de interacción (*Interaction Protocols*).
1.4. MTS o canales de comunicación (*Message Transport System*).
1.5. Base de conocimiento (*Knowledge Base*).
1.6. Modelos de interacción (*Interaction Models*).

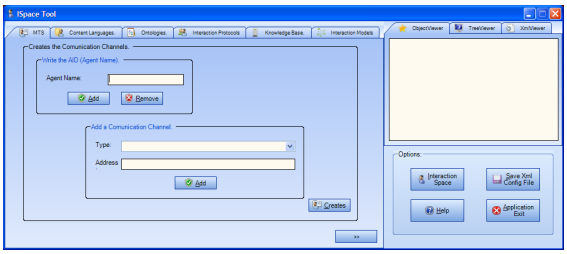

Figura 3 Pantalla principal de la herramienta.

1.1 Especificación de lenguajes de contenido (Content Language).

El contenido de los mensajes debe ser representado de tal forma que sea entendido por el agente emisor y el receptor. Establecer el lenguaje para el contenido del mensaje es la finalidad de utilizar lenguajes de contenido (CL) comunes. La plataforma de agentes CAPNET dispone de algunos CLs, que cumplen la especificación de un CL genérico. Los CLs están disponibles en una Biblioteca de Enlace Dinámico (DLL). La idea es que otros CLs pueden estar disponibles para ser utilizados en las interacciones.

De acuerdo con lo anterior, la herramienta ofrece al programador la posibilidad de extraer desde cualquier DLL las clases que contengan algún CL. De esta forma, se puede cargar la biblioteca de CAPNET (Agent.dll) o seleccionar cualquier otra. Al seleccionar una DLL, la herramienta despliega el tipo (*Type*) del CL en una lista y además el programador debe indicar el nombre (*Name*) único asociado a ese tipo (Fig. 4).

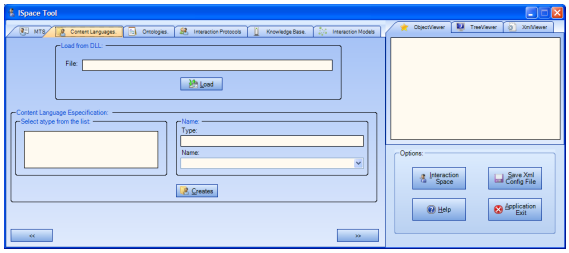

Figura 4 Especificación de un CL.

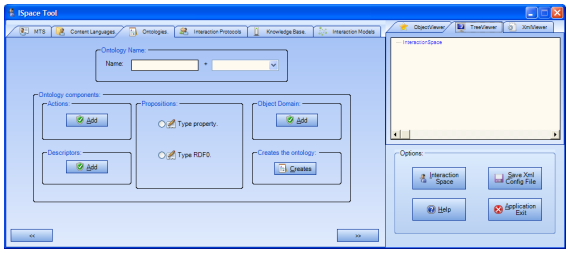

Figura 5 Especificación de ontologías.

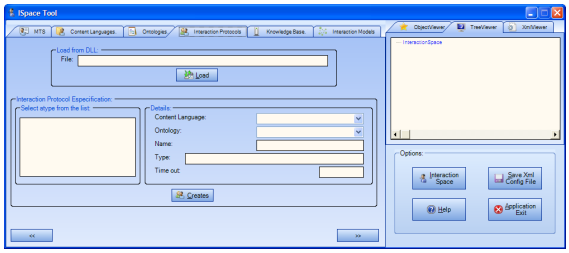

Figura 6 Especificación de Pis.

Es importante señalar que en la herramienta se puede capturar cualquier cantidad de lenguajes de contenido.

1.2. Captura de ontologías (Ontology).

La integración de los agentes inteligentes en un entorno abierto y complejo plantea la necesidad de asegurar la semántica sea común entre los agentes que interactúan, esta necesidad es la que se trata de cubrir con la especificación de ontologías.

La Ontología tiene cinco atributos. Contiene un nombre para la ontología (*Ontology* = Onto1 en Fig. 5) y un nombre de un lenguaje de contenido existente en el que las entidades están representadas (CAPNETRDF0 en Fig. 5). Luego, está formada por el conjunto de acciones (*Actions*), con sus respectivos argumentos. Los descriptores de objetos (*object descriptors*) son referencias a objetos de un dominio. Enseguida, contiene el conjunto de proposiciones, que pueden ser de dos tipos (propiedades de la forma atributo-tipo-valor y proposiciones lógicas).

Finalmente, la ontología contiene la lista de objetos de dominio (*Object Domain*) que representan entidades concretas del dominio.

Una vez capturados los atributos, el programador podrá verlos en el Visor de Ontología (*Ontology Viewer*) dónde aparecen en forma de un diagrama de árbol con sus nodos desplegables.

Antes de especificar una ontología se necesita haber capturado al menos un lenguaje de contenido.

1.3. Especificación de los protocolos de interacción (Interaction Protocols).

Los protocolos de interacción (PI), definen una secuencia de mensajes que representan un diálogo completo entre dos agentes.

Para especificar un PI, la herramienta debe contar con un lenguaje de contenido (CL) y una ontología ya especificada. Lo siguiente es realizar la captura de los tres atributos restantes.

El primero es el Tipo (*Type*), el cual es cargado a través de alguna DLL que implementa las clases para cada PI disponible. Enseguida, el usuario debe establecer el nombre (*Name*) y el tiempo de espera entre mensajes (*Default Time Out*).

Parra finalizar, se selecciona el nombre (*Name*) de los Lenguajes de Contenido y de las Ontologías existentes.

1.4. MTS (Message Transport System).

El MTS es el encargado de controlar el intercambio físico de mensajes ACL, esto, por medio de los canales de comunicación que maneja CAPNET en su arquitectura (.Net Remoting Manager, HTTP, XML -SOAP, IIOP, SMTP).

Para el Espacio de Interacción (IS), es necesario especificar la dirección (Address) y el tipo de protocolo de comunicación (Type) para que el agente disponga de la información acerca de los canales de comunicación en tiempo de ejecución.

La herramienta permite la captura de los atributos del MTS, especificando las direcciones, y seleccionando el tipo desde una lista de protocolos de comunicación disponibles en CAPNET.

1.5. Base de conocimiento (Knowledge Base).

Una base de conocimiento es una representación abstracta de un tema en un área incluyendo los conceptos de principal interés en dicha área y las relaciones entre las entidades. El Espacio de Interacción, en su modelo teórico indica que se debe tener una sola base de conocimiento (KB).

Para que la herramienta permite capturar la KB es necesario contar con: Objetos de Dominio (Object Domain), Proposiciones (para denotar propiedades y hechos lógicos) y acciones (Actions) para crearla.

1.6. Modelos de interacción (Interaction Models).

Los Modelos de Interacción permiten sistematizar la validación de interacciones que permitan asegurar el uso adecuado de cada uno de los actos comunicativos componen el lenguaje FIPA-ACL. Para cada acto comunicativo, es posible agregar un modelo de interacción. En la herramienta se permite incorporar estos modelos a partir de su búsqueda y selección desde DLLs que contienen las clases que los implementan.

3.2 Características funcionales

Al capturar cada atributo, la herramienta dispone de un objeto contenedor para cada uno de ellos, implementado por medio de clases. Al terminar de capturar cada uno de los atributos en la herramienta, se crea el objeto InteractionSpace (Espacio de Interacción).

El InteractionSpace es el encargado de recopilar y acomodar en su lugar cada atributo especificado por el programador. Finalmente, todo lo almacenado en este objeto contenedor es extraído por la propia herramienta para generar un archivo de especificación en XML (*eXtensible Markup Language*). Con el archivo, el agente carga en tiempo de ejecución los atributos especificados para acceder a ellos a través de la arquitectura de interacción y del proceso de validación de interacciones.

3.3 Integración con CAPNET

El resultado de la herramienta es un archivo de especificación en XML (*eXtensible Markup Language*). Este archivo es el medio integrador de la herramienta al ambiente de desarrollo de CAPNET. Permite que el agente básico de la plataforma (*BasicAgent*) al ejecutarse, cargue el archivo de especificación y lo procese para crear el Espacio de Interacción. Esto le permitirá al agente básico iniciar con una serie de atributos y un conocimiento inicial, aumentando así sus capacidades de autonomía e interoperabilidad.

3.4 Ventajas

Esta herramienta está diseñada para facilitar la tarea de programación de agentes en la plataforma CAPNET en el aspecto de la construcción de las capacidades que forman parte de su espacio de interacción. Una de las ventajas principales que ofrece es que con ella, se evita que los programadores escriban una cantidad considerable de código.

A mayor complejidad del espacio de interacción (dado por el número de ontologías, de CLs, una base de conocimientos con múltiples acciones, proposiciones, y objetos complejos, etc.) el número de líneas de código que se requieren se incrementa sustancialmente.

Con la herramienta, todas esas líneas de código son ahorradas, impactando directamente en la calidad del software de agentes (es menos propenso a errores), es más fácil lograr la interoperabilidad de los agentes (se fomenta la construcción sistemática y basado en una metodología de ingeniería para los componentes que se usan en las interacciones), y se brinda mayor velocidad y facilidad de programación (al utilizar un entorno visual, gráfico y amigable con el usuario-programador de agentes).

4 DISCUSION

A pesar de la apertura inherente a los SMA que utilizan FIPA-ACL para comunicarse, es importante reconocer la necesidad de trabajar en la definición, especificación e implementación de los elementos de control y las abstracciones de software que permitan, en tiempo de diseño y ejecución, incorporar y forzar restricciones en las interacciones, como parte de la infraestructura.

La herramienta que aquí se presentó se centra en esta problemática, y permite que la plataforma CAPNET ofrezca un ambiente de experimentación más completo que lo que se tiene en las infraestructuras de este tipo en la actualidad.

La discusión se centra en la exigencia de lograr que estas abstracciones sean lo suficientemente flexibles para soportar agentes heterogéneos y, al mismo tiempo, que sean efectivas para minimizar el hueco entre el diseño, desarrollo y ejecución del sistema [8].

Para completar el ciclo de desarrollo tecnológico, es fundamental desarrollar herramientas para permitir la manipulación de tales abstracciones a través de las diferentes etapas del proceso de ingeniería de software, particularmente en tiempo de ejecución.

La herramienta para ayudar a construir el espacio de interacción de los agentes permite sistematizar la creación de agentes y SMA incorporando capacidades de software que podrán ser utilizadas por los agentes para restringir, validar y regular la comunicación entre agentes construidos por diferentes programadores. Al mismo tiempo, otras herramientas de software serán necesarias para visualizar y depurar las interacciones que se realizan en tiempo de ejecución, una vez que los agentes estén en funcionamiento en un SMA.

5 CONCLUSIONES

En este artículo se presentó una herramienta de software que permite la especificación de los diferentes componentes que conforman el espacio de interacción de agentes. Se mostraron los detalles de la implementación y la funcionalidad de la herramienta en el ámbito de su integración con la plataforma de agentes CAPNET.

Así mismo, se han mencionado algunas de las ventajas que conlleva la utilización de esta herramienta, considerando los beneficios para la ingeniería de software orientado a agentes.

Finalmente, se hizo una discusión breve sobre algunos de los temas centrales en lo que se refiere al trabajo de investigación para el desarrollo de herramientas de ingeniería de software orientado a agentes.

6 AGRADECIMIENTOS

A la Universidad Autónoma de Sinaloa, al Dr. Leonid Sheremetov (Instituto Mexicano del Petróleo), al MC Noé Sierra Romero (CINVESTAV - Instituto Politécnico Nacional) y al MC Aníbal Zaldívar Colado, por su apoyo y facilidades para realizar la estancia para el desarrollo de este trabajo.

7 REFERENCIAS

[1] Luck, M.; McBurney, P.; Shehory, O.; Willmott, S. Agent Technology Roadmap, A Roadmap for Agent Based Computing. 2006.
[2] Searle, J. R. Speech Acts, Cambridge, UK: Cambridge University Press, 1969.
[3] Bordini, R.; Braubach, L.; Dastani, N.; Falla, A.; Gomez-Sanz, J.; Leite, J.; O`Hare, G.; Pokahr, A.; Ricci, A. A survey of Programming Languages and Platforms for Multi-Agents Systems. Informática, 30 (2006), pp. 33–44.
[4] Germán E. Desarrollo de una herramienta para la creación de agentes sobre la plataforma de agentes componentes. Tesis de maestría, Centro de Investigación en Computación, Instituto Politécnico Nacional, 2002.
[5] FIPA: The Foundation for Intelligent Physical Agents. URL:http://www.fipa.org/ (02.06.2006).
[6] FIPA spec. SC00061G: FIPA ACL Message Structure Specification. URL:http://www.fipa.org/specs/fipa00061/. (08.07.2003).
[7] Contreras, M.; Germán, E.; Chi, M.; Sheremetov, L. Design and implementation of a FIPA compliant agent platform in .NET. Journal of Object Technology, 3, 9 2004).
[8] Serrano, J. M.; Ossowski, S. On the Impact of Agent Communication Languages on the Implementation of Agent Systems. Lecture Notes in Computer Science Springer Verlag, 3191(2004), pp. 92-106.

