Una Propuesta de Metodología de Desarrollo de Sistemas Multi-Agente

Tulio José Marchetti
tjm@cs.uns.edu.ar

Alejandro Javier García
ajg@cs.uns.edu.ar

Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA)*
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Avda. Alem 1253 - (B8000CPB) Bahía Blanca
Tel: ++54 291 4595135 - Fax: ++54 291 4595136

En los últimos años han aparecido diferentes aproximaciones que tratan de presentar una metodología apropiada para el desarrollo de sistemas multi-agente. Dentro de nuestra linea de investigación sobre Metodologías y Plataformas de Desarrollo de Sistemas Multi-agente, el presente trabajo extiende a [3] donde se presenta una comparación más profunda de algunas metodologías.

Las metodologías que se estudiaron en [3] son algunas de las mas citadas en la literatura. GAIA [10] es una metodología para el diseño de sistemas basados en agentes cuyo objetivo es ayudar al analista a ir sistemáticamente desde unos requisitos iniciales a un diseño que, según los autores, esté lo suficientemente detallado como para ser implementado directamente. MaSE (Multi-agent systems Software Engineering) [8, 9] parte del paradigma orientado a objetos y asume que un agente es sólo una especialización de un objeto. La especialización consiste en que los agentes se coordinan unos con otros vía conversaciones y actúan proactivamente para alcanzar metas individuales y del sistema. y PASSI [2] la cual considera que un agente es *"una instancia de una clase agente, que es la implementación de software de una entidad autónoma capaz de lograr sus objetivos a través de sus decisiones autónomas, sus acciones y sus relaciones sociales"*

Este trabajo presenta una propuesta de Metodología de Desarrollo de Sistemas Multi-agente. Esta propuesta cuenta con la influencia de las tres estudiadas anteriormente y agregar elementos que a nuestro parecer son necesarios para cubrir todos los puntos de una metodología de desarrollo de sistemas multi-agente.

Nuestra propuesta

Consideramos que un sistema multi-agente (MAS) es un sistema formado por un conjunto de agentes, posiblemente heterogéneo, los cuales interactúan para conseguir algún objetivo o realizar alguna tarea. Por lo tanto estos agentes tendrán cierta habilidad social. Por esto, desde nuestro punto de vista, una metodología de desarrollo de sistemas multi-agente debería contemplar una variedad de

Financiado parcialmente por SeCyT Universidad Nacional del Sur (Subsidio: 24/ZN09) y por la Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 Nro 13096)

*Miembro del Instituto de Investigación en Ciencia y Tecnología Informática (IICyTI)

etapas. Las mismas, al igual que en los casos analizados, deben permitir una secuencia de pasos para llegar desde la descripción de alto nivel a la implementación del sistema. A continuación incluimos una breve descripción de cada una de ellas.

(1). Descripción del dominio e identificación de metas

En esta etapa del desarrollo se agruparon dos tareas que se encuentran muy relacionadas. Estas tareas son una combinación extraída de las metodologías PASSI y MaSE. La primer tarea consiste en la descripción del dominio del sistema. Esto nos permite ver mediante diagramas de casos de uso [4, 7] una descripción funcional del sistema. Los requerimientos son expresados en términos de diagramas de casos de uso usando los métodos clásicos orientados a objetos o mediante la aplicación informal de métodos basados en escenarios tales com GBRAM [1] o ScenIC [6].

Posteriormente a la descripción del dominio, se deben identificar las metas del sistema. Para esto, partiendo del modelo, podemos hacer uso del modelo de jerarquía de metas propuesto en MaSE. Cada nivel de la jerarquía contiene metas que son parecidas en alcance y todas las sub-metas están relacionadas funcionalmente con su "padre".

(2). Identificar los agentes y sus roles

En esta etapa se observa al sistema a través de la descomposición de sus funcionalidades y por lo tanto se comienzan a agrupar funcionalidades formando un rol de agente. Se puede pensar a un agente como un caso de uso o un paquete de casos de uso. Por esto, al igual que lo hace la metodología PASSI, partiendo del diagrama de casos de uso de la fase previa usamos paquetes para ilustrar las funcionalidades asignadas a cada agente.

Los roles de los agentes pueden verse como descripciones de la funcionalidad esperada de cada agente. Un rol está definido por cuatro atributos: responsabilidades, permisos, actividades y protocolos. Las responsabilidades determinan la funcionalidad y como tal, son posiblemente los atributos claves asociados con un rol. Para poder implementar responsabilidades, un rol tiene un conjunto de permisos. Los permisos son los "derechos" asociados con cada rol. Los permisos de un rol por lo tanto identifican los recursos que están disponibles para ese rol para poder implementar sus responsabilidades. En el tipo de sistemas que típicamente se modela, los permisos tienden a ser recursos de información.

Las actividades de un rol son computaciones asociadas con el rol que pueden ser llevadas a cabo por el agente interactuando con otros agentes. Finalmente, un rol también esta definido con un número de protocolos, que definen la forma en la que puede interactuar con otros roles.

(3). Asociar cada meta con un rol de un agente

Igual que en las metodologías estudiadas, cuando se mapean metas a roles, hay generalmente un mapeo uno a uno entre roles y clases de agentes. Sin embargo, un diseñador puede combinar múltiples roles en una única clase de agente o mapear un rol a múltiples clases de agentes.

Como los agentes heredan las vías de comunicación entre los roles, cualquier camino entre dos roles se convierte en una conversación entre sus respectivas clases. Como tal es deseable, siempre que sea posible, combinar dos roles que comparten un alto volumen de tráfico de mensajes. Cuando se

determinan que roles combinar, tamaño y frecuencia de comunicaciones son importantes, no sólo el número de vías de comunicación.

(4). Identificar las interacciones entre agentes

Al igual que las metodologías evaluadas, se debe identificar todas las formas en que los agentes interactuarán. Para esto, se confecciona un modelo en el que se ven todas las comunicaciones que se realizan entre los agentes.

En este modelo se ven todas las interacciones que existen entre los agentes dentro del sistema. Esto permite controlar si las comunicaciones que se realizarán están completas en participantes, canales de comunicación y en mensajes.

(5). Seleccionar un protocolo para cada interacción

Una vez identificadas las interacciones entre agentes es necesario seleccionar un protocolo para cada una. Una opción es utilizar los ya conocidos Protocolos de Interacción FIPA. Estos, son patrones que se usan para llevar por unos cauces concretos una conversación. Son como conversaciones guiadas, en que cada agente sabe qué mensaje enviar y cuales puede recibir.

La otra posibilidad es definir los protocolos de comunicación propios para el sistema que está en desarrollo. Para definir un protocolo se utilizará el siguiente formato o molde con los campos más importantes.

Protocolo : ***Nombre***
Propósito: *Descripción*
Iniciador: *Agente A*
Receptor: *Agente* B_1*, ..., Agente* B_n
Entrada: *Datos de entrada*
Salida: *Datos de salida*
Procesamiento adicional: *Descripción*

(6). Seleccionar el lenguaje de comunicación

En este punto ya están determinados todos los mensajes que formarán parte de las comunicaciones, así como también los protocolos que utilizarán los agentes para sus comunicaciones. Sólo queda determinar el tipo de lenguaje de comunicación que se utilizará en el sistema.

A diferencia de las metodologías estudiadas, presentamos dos opciones: utilizar un lenguaje de comunicación entre agentes estandarizado como KQML o FIPA ACL o definir una estructura de mensaje propia del sistema en desarrollo. Cualquiera de estas opciones presenta ventajas y desventajas.

La posibilidad de definir una estructura de mensaje particular a cada sistema presenta la ventaja de poder definir de forma muy simple el punto de la comunicación. Pero también presenta desventajas a la hora de incluir un nuevo agente al sistema, ya que este no tiene conocimiento del lenguaje de comunicación que se está utilizando.

(7). Clasificar los agentes según su tipo

A diferencia de las otras metodologías, consideramos que a esta altura del desarrollo, es necesario determinar el tipo de agente que es cada uno de los que componen el sistema en desarrollo. Esto facilitará la selección de la arquitectura correspondiente, tarea que se realiza en el próximo paso.

En este punto, con la información de los pasos anteriores, es posible realizar esta clasificación. Algunos tipos de agentes, según [5], son los siguientes:

- Colaborativos
- De interface
- Móviles
- De información/internet
- Reactivos
- Híbridos

(8). Seleccionar una arquitectura para los agentes

En contraste con las metodologías analizadas, consideramos que esta etapa debe ser independiente y no parte de una. Basado en la etapa anterior, seleccionamos una arquitectura para cada tipo de agente encontrado. Cada arquitectura provee un enfoque distinto de implementación y sobre la estructura interna y operación de los agentes. Por ejemplo,

- basada en lógica
- Reactiva
- BDI
- Por capas

(9). Producir el código

Como último paso de la metodología propuesta, queda la etapa de producción de código. En este punto tenemos toda la información necesaria para poder generar los agentes y ensamblarlos formando el sistema deseado.

Para cada una de las arquitecturas seleccionadas anteriormente se debe decidir como implementar el agente. Para esto hay varias opciones:

- Utilizar una plataforma de desarrollo de sistemas multi-agente
- Utilizar el mismo lenguaje de programación para todas las arquitecturas de agentes
- Utilizar diferentes lenguajes de programación dependiendo de la arquitectura del agente

Conclusiones y trabajo futuro

En este trabajo se han mencionado tres metodologías que fueron estudiadas y se puede observar todo nuestro análisis en [3]. Muchos de los pasos de la propuesta tienen su base en uno de las metodologías estudiadas. El principal objetivo fue estructurar estos pasos y combinarlos con los que no forman parte de ninguna propuesta pero que a nuestro parecer sí deben pertenecer a la metodología. A continuación resumimos los cambios y novedades respecto de las metodologías estudiadas.

- La etapa de descripción del dominio e identificación de metas es una combinación de tares de las metodologías PASSI y MaSE.
- Se introduce la etapa de seleccionar el lenguaje de comunicación.
- Adicionamos la etapa de clasificación de los tipos de agentes.
- Basados en lo anterior agregamos una etapa de selección de arquitectura para los agentes.

Como trabajo futuro queda pendiente el desarrollo de una plataforma que permita implementar esta metodología de una manera automática facilitando el desarrollo de los sistemas Multi-agente. También el refinamiento de la propuesta y la búsqueda de nuevas metodologías para seguir evaluando y ampliando nuestra investigación.

Referencias

[1] A.I. Antón and C. Potts. The use of goals to surface requirements for evolving systems. In *International Conference on Software Engineering (ICSE 98)*. Kyoto, Japón, 1998.

[2] Cossentino and Potts. A case tool supported methodology for the design of multi-agent systems. *The 2002 International Conference on Software Engineering Research and Practice*, 2002.

[3] Tulio Marchetti and Alejandro García. Metodologías de desarrollo de sistemas multi-agente: un análisis comparativo. *X Congreso Argentino de Ciencias de la Computación. La Matanza, Argentina*, pages 1755–1766, 2004.

[4] Bertrand Meyer. *Construcción de software orientado a objetos*. Prentice Hall.

[5] Hyacinth S. Nwana. *Software Agent: An Overview*. www.idi.ntnu.no/ conradi/dif8914/p7-sw-agent.ppt, 2000.

[6] C. Potts. Scenic: A strategy for inquiry-driven requirements determination. In *IEEE Fourth International Symposium on Requirements Engineering (RE 99)*. Limerick, Irlanda, 1999.

[7] Joseph Schmulerr. *Tech TourSelf UML in 24 Hours*. Sams.

[8] Mark F. Wood Scott A. DeLoach and Clint H. Sparkman. Multiagent systems engineering. *The International Journal of Software Engineering and Knowledge Engineering*, 11(3), 2001.

[9] M. F. Wood. Multiagent systems engineering: A methodology for analysis and design of multiagent systems. Master's thesis, Air Force Institute of Technology, 2000.

[10] Michael Wooldridge, Nicholas R. Jennings, and David Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.