Modelado de un Sistema Multi-Agente mediante la aplicación de la metodología INGENIAS con el Ingenias Development Kit

Juan A. Botía
MASTER TITA, Convocatoria 2007/2008
Ingeniería de Agentes Software y Físicos
Departamento de Ingeniería de la Información y las Comunicaciones
Universidad de Murcia

1 Introducción

En la práctica final de la asignatura vamos a desarrollar un software que simule, de una forma un tanto *naive*, un sistema de pedidos de pizzas por Internet, que actúe en régimen competitivo (i.e. las pizzerias compiten por vender más pizzas que el resto). Por supuesto, el sistema será distribuido y cada pizzería será una entidad independiente, representada por un agente. También cabe la posibilidad de que podamos tener intermediarios. Intermediarios que actuarán como pizzerías pero que en realidad no lo son. Simplemente se encargan de gestionar un directorio de pizzerias propio. Pizzerias que, esta vez si, están representadas por agentes.

Con respecto al usuario, este podrá obtener una lista de todas las pizzerias, obtener una lista de todas las pizzas dentro de una pizzería, preguntar por el precio de una pizza concreta y realizar un pedido de una pizza concreta a una pizzería.

El planteamiento de la práctica será el siguiente. En esta misma memoria que estás leyendo, se incluirá, explicado, un modelado inicial del SMA que cubre algunos de los requisitos planteados en el escenario anterior. Se incluirán algunos diagramas INGENIAS, que habremos producido con el IDK. Con todo esto será suficiente para generar un modelo de SMA que especifique suficientemente el sistema que queremos construir.

A partir de ahí, la labor del alumno será: (1) entender y asimilar el resto de esta memoria y (2) **expandir el modelo de SMA** que se incluye aquí, en una siguiente práctica, permitir el resto de funcionalidad que se exige a través de la GUI. En todo caso, este **primero documento** de la práctica (lo cual quiere decir que aun hay otro más por venir), es un guión de cómo debería ser el modelo INGENIAS de la pizzería **con una funcionalidad básica**. Por tanto, en esta primera sesión de la práctica no se pide al alumno que entregue ningún material para evaluar. Esta práctica no puntúa. Sin embargo, es necesario comprenderla para realizar las ampliaciones al modelo que sí serán puntuables.

Como herramientas para esta práctica usaremos únicamente el IDK como editor de modelos INGENIAS. Esta herramienta está disponible en

ingenias.sourceforge.net.

La metodología INGENIAS se compone, sobre todo, de un lenguaje de modelado. Los modelos producidos con este lenguage se estructuran en cinco modelos (i.e. view points) diferentes, que aparecen en la figura 1. El Organization viewpoint tiene en cuenta la estructura del sistema multi-agente, sus roles, las relaciones sociales y los flujos de trabajo que se dan dentro del mismo. El Agent viewpoint tiene en cuenta que los agentes realizan tareas y persiguen objetivos. Por tanto, se incluyen los agentes, sus tareas, los objetivos que persiguen y su estado mental (i.e. lo que conocen inicialmente). El Goals and Tasks viewpoint identifica objetivos y tareas a perseguir y realizar por los agentes, y los descompone. En el Interaction viewpoint

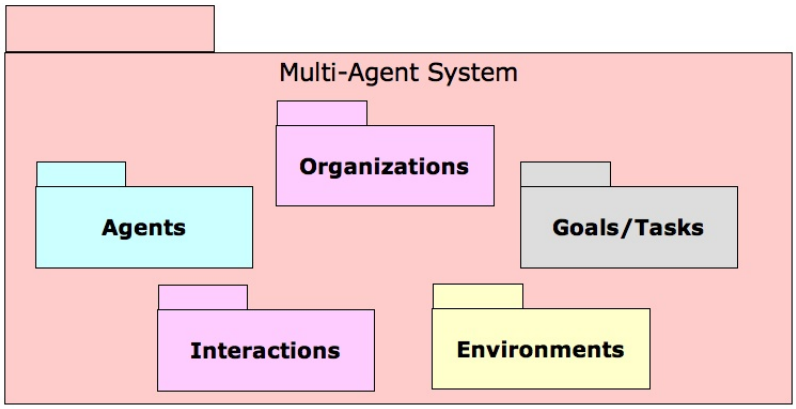

Figure 1: Los cinco puntos de vista de modelado de INGENIAS

se identifican las interacciones entre agentes y en el Environment viewpoint aquellas entidades con las que interaccionan los agentes y su relación con ellas.

2 El modelo inicial

En esta sección introducimos el modelo inicial del que se parte. La metodología INGENIAS es bastante flexible y nos permite comenzar el modelado por cualquier tipo de modelo, dependiendo de cuál sea el tópico más importante en nuestro caso, aunque esto no es obligatorio. También es posible seguir algún proceso de desarrollo, como RUP.

En este caso, lo más importante para nosotros van a ser las interacciones y, por tanto, vamos a comenzar el modelo desde aquí. De esta forma, comenzaremos generando un modelo de interacciones en el que incluiremos modelos de alto nivel de las mismas. A partir de aquí realizaremos modelos más específicos de las interacciones mediante la definición de protocolos. Al definir los protocolos detectaremos las tareas básicas que han de realizar los agentes, sus entradas y salidas. Con esta información generaremos un modelo de tareas y objetivos. Con las entradas y salidas de esas tareas generaremos una ontología. Tras esto, definiremos los agentes participantes en el modelo de agente. Después solamente nos restará definir una configuración de despliegue con la que lanzar el sistema. La secuencia de acciones del modelado viene representada esquemáticamente en la figura 2.

2.1 Modelo de interacciones

Una interacción es un proceso comunicativo entre entidades. En un SMA, pueden existir interacciones entre agentes y entre agentes y otros objetos del SMA. En el lenguaje de modelado INGENIAS solamente se tiene previsto el modelado explícito de interacciones entre agentes. El modelado entre agentes y otros objetos se realiza a través del modelo del entorno, de tal forma que los objetos se ven como aplicaciones.

Para modelar interacciones entre agentes, el proceso es siempre el mismo:

1. Crear un modelo de interacciones
2. Incluir interacciones dentro del modelo
3. Para cada interacción
   (a) Incluir roles iniciador y participante (posible indicar aridad en el participante)
   (b) Incluir el objetivo correspondiente para cada interacción
   (c) Incluir icono para especificación y

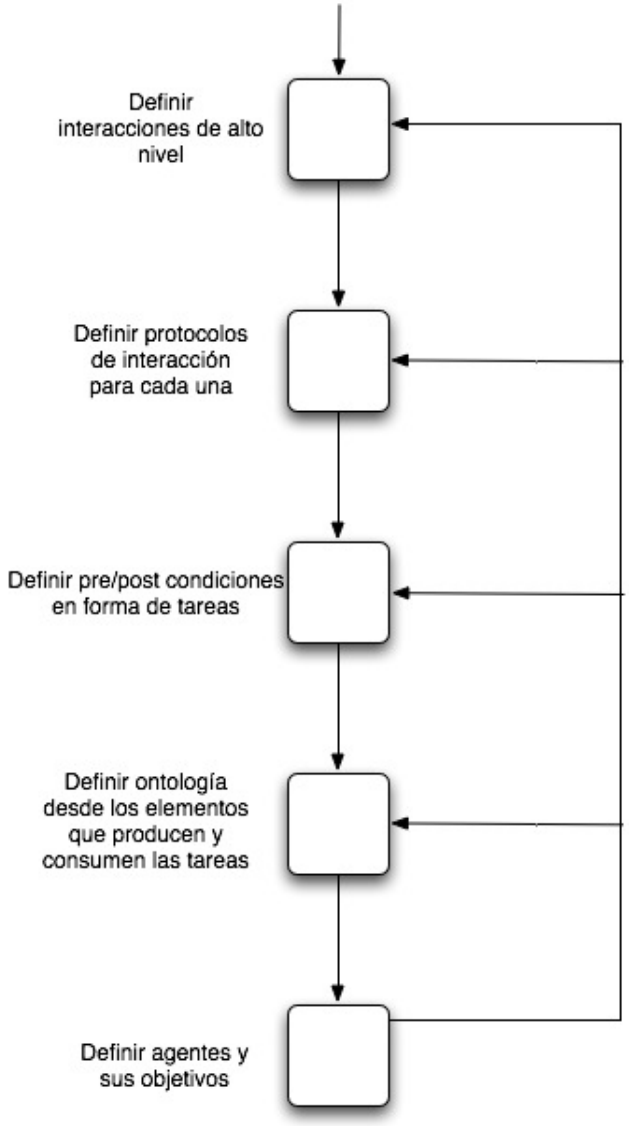

Figure 2: Pasos en el modelado INGENIAS para esta práctica, centrada en las interacciones

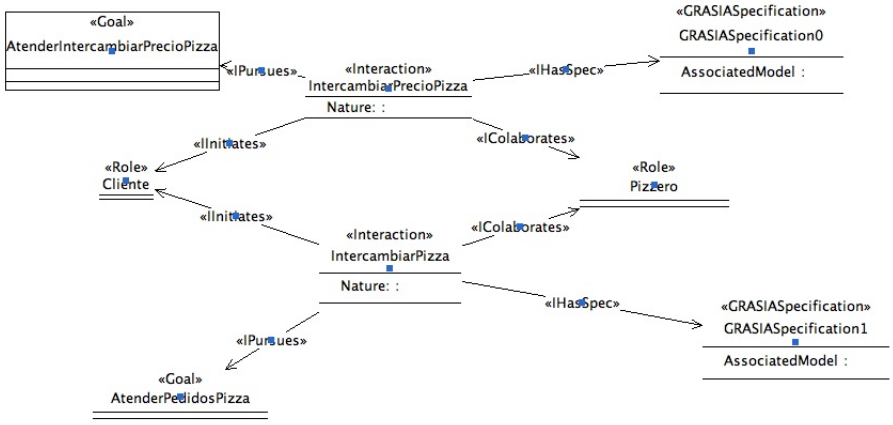

Figure 3: Diagrama general de las interacciones del sistema

i. Crear un nuevo modelo de interacción
ii. Definir el protocolo de interacción en este nuevo modelo
iii. Asociar el icono de la especificación a este nuevo modelo

en donde los pasos 3.(a), 3.(b) y 3.(c) se pueden aplicar en cualquier orden.

Inicialmente, vamos a tener dos interacciones diferentes. Por un lado, vamos a diseñar cómo un agente se hace con el precio de una pizza `IntercambiarPrecioPizza`, a partir de su nombre. Por el otro, se ha de pedir una pizza concreta a un agente determinado, lo especificaremos mediante la interacción `IntercambiarPizza`.

Por tanto, vamos a tener dos interacciones sencillas. Una de ellas se va a corresponder con el protocolo `fipa-query` y la otra con el `fipa-request`, respectivamente. Son interacciones en las que solo participan dos agentes por tanto, para ambas tendremos un iniciador y un participante (paso 3.(a)). Para ambas, iniciador y participante serán el mismo. Los denominaremos `Cliente` y `Pizzero`. Posteriormente, se debe indicar, para cada una de las interacciones, que tipo de diagrama vamos a usar para la especificación concreta de la interacción. Podemos utilizar varios tipos de diagramas entre los que tenemos AUML, que son diagramas de secuencia UML adaptados para agentes, y diagramas Grasiá, un tipo de diagrama más sencillo en cuanto a su notación y perteneciente a la metodología INGENIAS. Usaremos este último (paso 3.(c)). Lo mejor para especificar cada protocolo es definir un diagrama de interacción por separado para cada uno (ver sección 2.1.1). Por tanto, crearemos un nuevo diagrama de interacción por protocolo (paso 3.(c).ii). Así mismo, definiremos un objetivo que cada interacción persigue y con eso será suficiente por ahora (paso 3.(b)). Denominaremos a los objetivos con `AtenderIntercambiarPrecioPizza` y `AtenderIntercambiarPizza`. Una vez hemos indicado en el modelo de interacción del párrafo anterior que ambas interacciones tenían especificaciones Grasiá, hemos de definir ahora las interacciones de manera detallada. Para ello, como hemos dicho, creamos dos diagramas nuevos para estas dos nuevas interacciones (paso 3.(c).i). Sus nombres comenzarán con el prefijo `protocolo` y así los distinguiremos del resto. Definiremos ambos diagramas y cuando esto esté hecho, nos iremos al diagrama de interacciones original y, pinchando en el icono de cada especificación de interacción, asociaremos a la misma su protocolo correspondiente (paso 3.(c).iii).

2.1.1 Definición inicial de los protocolos de interacción

En el primer protocolo, vamos a incluir primero los dos roles que hemos indicado anteriormente iban a participar en la interacción. Para ello, en la ventana inferior izquierda aparece un panel de texto con el que podemos hacer búsquedas sobre el árbol de elementos que forman parte de nuestro modelo. Buscamos con `Cliente` y el árbol se despliega a la altura de los roles que tenemos definidos. Pinchamos en cliente y con el botón derecho del ratón indicamos que se añada al diagrama actual. Hacemos lo mismo con el rol `Pizzero`.

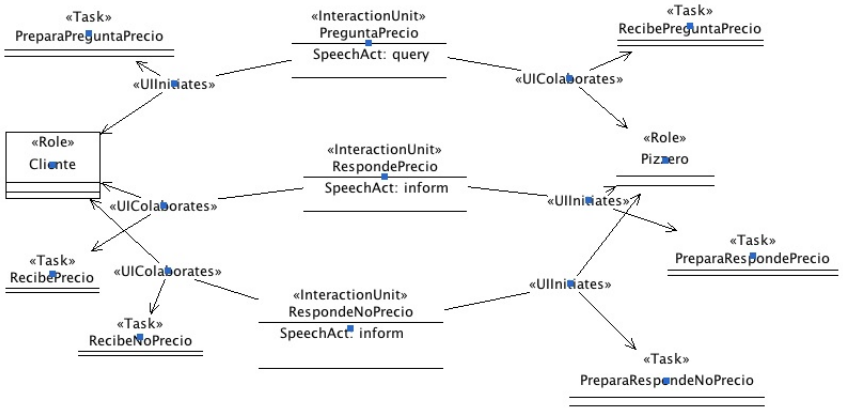

Figure 4: Protocolo de interacción (sin orden específico para las unidades de interacción) para pedir el precio de una pizza

Esta operación la haremos típicamente cada vez que en un modelo concreto necesitemos un elemento (e.g. un objetivo, tarea, agente, rol, frame fact, etc) que hemos definido previamente. Vemos como aparecen en el protocolo de la figura 4.

Ahora hay que incluir tres unidades de interacción. El protocolo va a consistir en una petición del precio de pizza y una respuesta que puede ser el precio o bien que la pizza no se conoce. Por tanto, incluiremos tres unidades de interacción, una por cada posible ocurrencia de envío de mensaje. Serán `PreguntaPrecio`, `RespondePrecio` y `RespondeNoPrecio`. Para la primera unidad de interacción, el iniciador (relación `IInitiates`) será el `Cliente` y `Pizzero` el participante (relación `ICollaborates`). Para las otras dos unidades de interacción, al contrario. Los objetos de tipo `Task` que aparecen en la figura 4 vienen explicados en la sección 2.1.2.

Abandonamos ese diagrama de interacción de especificación del protocolo para preguntar el precio de una pizza y creamos uno nuevo, con el mismo prefijo, para definir el protocolo de interacción para pedir una pizza. Este protocolo tendrá los mismos roles. Dado que también estamos realizando una petición, el diagrama de interacción será muy similar de tal forma que tendremos otras tres unidades de interacción.

2.1.2 Definición inicial del modelo de tareas

Con esta forma de modelar el sistema multi-agente, el elemento central del modelo es el conjunto de interacciones. Y a partir de este surge el resto de elementos. Ahora vamos a ver cómo surgen las tareas. Para ello, debemos fijarnos en cada una de las unidades de interacción de los protocolos que acabamos de definir, tomemos el de la figura 4 como ejemplo.

En la comunicación de agentes, el enviar un mensaje de un agente a otro se considera una acción de primer orden (i.e. tiene precondiciones y postcondiciones). En el contexto de la planificación convencional dentro de la Inteligencia Artificial, una acción dentro de un plan viene especificada por una precondición y una postcondición. Por tanto, una unidad de interacción (i.e. un mensaje a intercambiar entre agentes) también vendrá especificado igualmente.

Por un lado, para que una interacción se inicie, el objetivo que cumple debe estar en el estado mental del agente como pendiente de cumplirse. Por el otro, para que una interacción comience mediante el envío de la primera unidad de interacción (e.g. `PreguntaPrecio`), esta interacción debe tener disponible en el estado mental del agente aquellos hechos que necesita para empezar. Estos hechos son las precondiciones. Por tanto, para que una interacción entre dos o más agentes se inicie, (1) el agente iniciador debe querer hacerlo y (2) se deben dar las condiciones necesarias para que se haga.

A parte de las precondiciones, podemos especificar tareas que se han de ejecutar tras esas precondiciones (en el lado del agente que envía el mensaje) y antes de que el mensaje se haya enviado concretamente. Estas

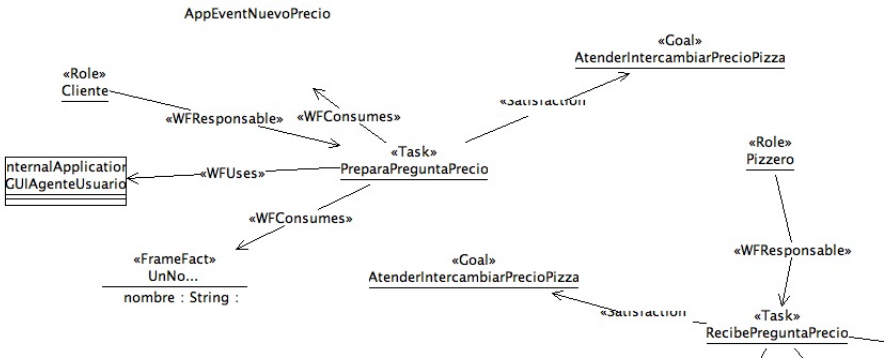

Figure 5: Especificación de una tarea con el lenguaje de modelado de INGENIAS

tareas pueden servirnos para generar el cuerpo del mensaje, por ejemplo. La tarea correspondiente a la precondición de una unidad de interacción determinada se especifica transformando la relación `UIInitiates` entre rol y unidad de interacción de binaria en una relación ternaria, incluyendo la tarea. Fijémonos, por tanto, en la tarea `PreparaPreguntaPrecio`, que se ejecutará en el agente con rol `Cliente`, inmediatamente antes de ejecutar la unidad de interacción `PreguntaPrecio`. Otras son `PreparaRespondePrecio` y `PreparaRespondeNoPrecio`. El hecho de que para cada relación `UInitiates` aparezca en el modelo una asociación a una tarea no quiere decir que toda unidad de interacción necesite, antes de enviarse, que se ejecute una tarea. En nuestro caso, para este ejemplo concreto, sí va a ser así. Por tanto, asociamos un tarea a cada relación de este tipo.

Con respecto a las postcondiciones, se van a corresponder con los efectos de posibles tareas que podemos ejecutar en el lado del agente receptor, cuando el mensaje de la unidad de interacción correspondiente se ha recibido. Para indicar la tarea a ejecutar en el lado del agente receptor cuando se recibe un mensaje, se transforma la relación `UICollaborates` que relaciona el rol y la unidad de interacción de binaria a ternaria, como hacemos para las precondiciones.

Así, unidad de interacción a unidad de interacción, mirando a las pre y postcondiciones de cada una, generaremos las tareas necesarias al menos para la comunicación. Las creamos en el diagrama correspondiente al protocolo, por otro lado definimos un diagrama de tareas en donde especificamos cada tarea de manera más específica y finalmente definimos el código de cada una en un diagrama de tareas a parte.

Para definir una tarea determinada, necesitamos especificar para ella las entradas a la tarea (i.e. las precondiciones), las salidas o los hechos que la tarea produce, los objetivos que persigue, el rol responsable de la tarea (i.e. el que la ejecuta). Haremos eso para cada una de las tareas detectadas a partir de las interacciones. En la figura 5 vemos un ejemplo de la especificación de una tarea.

2.2 Definición inicial de la ontología

Es aconsejable, por claridad, el tener un modelo aparte en donde aparezcan todos los frame facts que se van a utilizar como entrada o salida de las tareas. Estos formarán la ontología básica de nuestro sistema. Los elementos de la ontología (i.e. los *frame facts*, f.f.) pueden verse como tokens de información a intercambiarse por los agentes. Por regla general, nada que no vaya a salir al exterior de la base de creencias de un agente necesita ser modelado. Es decir, solamente modelaremos como f.f. aquellos que deba ir en el cuerpo de los mensajes intercambiados por los agentes. Por tanto, todos los f.f. de la ontología surgen de forma natural a partir de la especificación de los protocolos de interacción, más específicamente a partir de la especificación de cada unidad de interacción.

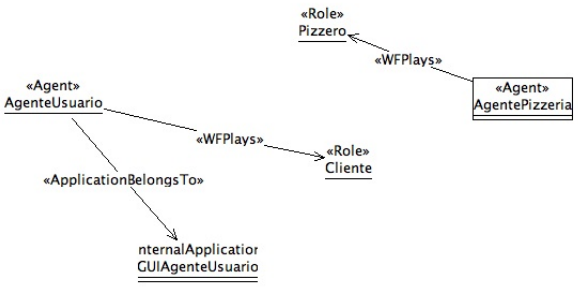

Figure 6: Especificación de los agentes con el lenguaje de modelado de INGENIAS

2.3 Modelo de agentes

Una vez que tenemos los roles, tenemos también determinados los agentes. Ahora es necesario un modelo de agentes, que relaciones agentes con roles. Un agente de usuario `UserAgent` jugará el rol de `Cliente` y un `AgentePizzeria` jugará el rol de `Pizzero`. En este modelo también indicaremos los objetivos que sigue cada agente de tal forma que al intentar cumplirlos, se intentará lanzar las interacciones con las que hemos comenzado el diseño. Vemos así, la especificación del modelo de agentes en la figura 6.

Obsérvese que este modelo es muy simple pero cada agente puede jugar más de un rol, y varios tipos de agentes pueden compartir roles.

2.4 Interacción de los agentes con el resto del mundo

Los agentes software suelen interaccionar con otros elementos del entorno. Es decir, un SMA no es un sistema cerrado en el que los agentes solamente interaccionan con otros agentes. Tendrán conexiones con el exterior. Estas conexiones con el exterior al final derivan en interacciones agente-objeto. Sin embargo, estas interacciones no se consideran comunicación al estilo de la que realizan los agentes. Se consideran interacciones más simples, basadas en el uso de APIs. Por ejemplo, un agente puede estar conectado con una aplicación mediante servicios Web, interfaces del tipo Corba, Java-RMI, etc.

En INGENIAS, todo lo que tiene que ver con software de aplicaciones se considera también en el modelado.

2.5 Depliegue del sistema multi-agente

El despliegue del sistema multi-agente es la parte del modelo en donde podemos especificar qué tipo de agentes queremos lanzar, cuántos de cada tipo y con qué valores iniciales, dentro de un límite. De esta forma, podemos tener varias configuraciones de despliegue y usar en cada momento la que más nos interese, dependiendo de las pruebas que necesitemos si estamos desarrollando o dependiendo de las configuraciones de SMA según el entorno y condiciones si hemos terminado el desarrollo.

3 Conclusiones

En esta primera práctica de modelado hemos generado una versión inicial del SMA que vamos a crear en la asignatura de Ing. Agentes Software y Físicos dentro del Master TITA.

Esta práctica no puntúa por lo que no es necesario entregar nada para evaluar, de hecho tras la práctica se os va a proporcionar un modelo totalmente desarrollado en formato HTML para que lo reproducais vosotros mismos. De esta forma, en la siguiente práctica, vuestro modelo debería ser exactamente igual al que se os presenta para poder seguir avanzando.