Implementación de protocolos de interacción FIPA compatibles para agentes con la herramienta CAP-AgentTool

Ernesto Germán Soto [1], Rigoberto Velásquez Elenes [2], Omar Olivas Zazueta [2]

[1] Instituto Mexicano del Petróleo, Eje Central Lázaro Cárdenas 152, Col. San Bartolo Atepehuacan, 07730, México, D.F.
egerman@imp.mx

[2]Universidad Autónoma de Sinaloa, Angel Flores y Riva Palacio S/N, Colonia Centro, Culiacán Sinaloa.
verxx@info.uasnet.mx, oaoz@hotmail.com

Resumen. Los protocolos de interacción constituyen un mecanismo que ayuda a la implementación de conversaciones y patrones de mensajes comunicativos entre agentes. En este trabajo se presenta la forma en que estos protocolos pueden ser incorporados en el tiempo de diseño de los agentes que se crean con la herramienta CAP-AgentTool y se muestra el papel que juega el programador en las diferentes etapas de comunicación que conforman a un protocolo. Además, se explica la implementación de tres protocolos especificados por FIPA: Query, Request y Contract-Net. Por último, se incluye una pequeña aplicación con varios agentes que intercambian mensajes por medio del protocolo Contract-Net. La infraestructura utilizada para el despliegue de estos agentes es la plataforma de agentes CAP.

1 Introducción

Las conversaciones que se llevan a cabo entre los agentes caen en determinados patrones regularmente. En tales casos, se espera que ocurra cierta secuencia de mensajes y en algún momento de la conversación, se espera que sigan otros mensajes. Estos patrones típicos de intercambio de mensajes son llamados protocolos de interacción (PI). En principio, un agente es visto como aquella entidad de software que permite la colaboración entre aplicaciones por medio de un lenguaje de agentes común y que utiliza los servicios de una plataforma compatible con FIPA.

Un diseñador de aplicaciones de agentes puede decidir hacer a los agentes lo suficientemente conscientes del significado de los mensajes de tal forma que el proceso de planeación cause que los PI emerjan espontáneamente de las decisiones del agente. Esto sin embargo, implica una gran carga de capacidades y complejidad en la implementación del agente. Una forma alternativa y muy pragmática es pre-definir los PI para que agentes implementados en forma sencilla puedan involucrarse en conversaciones que tengan sentido con otros agentes, simplemente apegándose al protocolo conocido. Tal es el caso de las aplicaciones de comercio electrónico, mercados electrónicos y de subastas que están siendo desarrolladas con agentes y mecanismos de negociación específicos para su interacción [1], [2].

A. Gelbukh, M. Hernández Cruz (Eds.) Avances en la ciencia de la computación en México, CORE-2003, pp. 107–117, 2003.

El apegarse al uso de protocolos trae ventajas a los desarrolladores de agentes, pues pueden enfocarse a implementar sólo aquellos actos comunicativos involucrados los protocolos en lugar de considerar todo el espectro de actos. La forma de especificar el protocolo en una conversación es poniendo su nombre en el parámetro *protocol* del mensaje. FIPA es una organización internacional que trabaja en la especificación y estandarización de agentes y ha establecido un conjunto de protocolos estándar través de sus especificaciones en la biblioteca de protocolos de interacción [3]. ventaja que esto trae es que se logra cierto grado de interoperabilidad entre aplicaciones distribuidas con agentes heterogéneos.

Por otra parte, los agentes componentes que se generan con la herramienta CAP-AgentTool tienen la posibilidad de incorporar una serie de PI que les permiten comunicarse de manera predefinida con otros agentes siguiendo la secuencia de mensajes establecida en cada uno. Los protocolos que aquí se presentan son: Fipa-Query, Fipa-Request y Fipa-ContractNet y están basados en la especificación que FIPA establece para cada uno. Además de seguir la secuencia de mensajes predefinida en cada protocolo, la implementación que se propone en este trabajo considera el uso del lenguaje de comunicación de agentes FIPA-ACL. La sintaxis y semántica del contenido de mensajes están basadas en la representación de objetos de contenido definidos en lenguaje de contenido FIPA-RDF0. La ventaja de utilizar esta herramienta radica que los programadores pueden implementar los PI mucho más fácil, rápida y consistentemente, sin tener que preocuparse por estudiar y entender las especificaciones respectivas.

Después de explicar el funcionamiento de la implementación, se muestra la forma de utilizar los protocolos en una aplicación de demostración con varios agentes guiendo el protocolo Contract-Net. La finalidad es indicar las partes del protocolo que deben ser implementadas en los agentes y de acuerdo a la aplicación particular que los utilice. El despliegue de esta aplicación se realiza sobre la plataforma de agentes CAP.

2 Infraestructura utilizada

2.1 Component Agent Platform (CAP)

La plataforma CAP está basada en el modelo de referencia de FIPA (*Foundation for Intelligents Physical Agents*). Los servicios básicos y obligatorios tienen la característica de que se han implementado sobre un servidor DCOM a través de los componentes AMS (*Agent Management System*), DF (*Directory Facilitator*) y ACC (*Agent Communication Channel*).

El lenguaje de comunicación entre agentes de CAP es FIPA-ACL debido a que FIPA impone que la comunicación entre agentes debe ser de alto nivel refiriéndose la manipulación e intercambio de conocimiento entre los agentes. En un ambiente distribuido, un agente suele cumplir con sus objetivos a través de la interacción con otros agentes y ejecutando acciones, lográndolo precisamente a través del uso de su lenguaje de comunicación entre agentes [4], [5].

2.2 Agentes

Un agente es el actor fundamental en una plataforma que combina una o más capacidades de servicios en un modelo de ejecución integrado y unificado que puede incluir acceso a software externo, usuarios humanos y facilidades de comunicación. Definido por FIPA, un agente es todo aquel proceso computacional que implementa la funcionalidad autónoma y de comunicación de una aplicación.

En la teoría de agentes colaborativos heterogéneos un agente es todo aquel proceso computacional que se comunica a través de un lenguaje de comunicación de agentes [6].

2.3 Herramienta CAP-AgentTool

CAP-AgentTool es una herramienta que tiene por finalidad principal ayudar a los programadores en el proceso de creación de controles agente que corran sobre la plataforma de agentes CAP. A través de ésta es posible ir configurando paso a paso los aspectos que permiten la creación de nuevas clases de agentes como controles ActiveX [7]. En particular, uno de los aspectos que se puede configurar es la incorporación de algunos protocolos de interacción que ya han sido pre-definidos y cuya secuencia de mensajes es incorporada y controlada al momento de diseñar el agente [8].

Solución

3.1 Lenguaje de comunicación de agentes

El lenguaje de comunicación usado por los agentes creados por CAP-AgentTool es FIPA-ACL, dado que están diseñados para ejecutarse en una plataforma CAP. Este es un lenguaje de comunicación entre agentes que está definido a través de la teoría de actos del habla o *speech acts*. Por consiguiente, FIPA ofrece una semántica para diferentes tipos de actos comunicativos [9] y una forma de construirlos [10]. Todos los agentes creados con CAP-AgentTool tienen un mecanismo interno para enviar y recibir mensajes, por medio de una plataforma CAP, hacia otros agentes. De esta manera, también la implementación de los PI en agentes que aquí se presenta utiliza este lenguaje.

3.2 Selección de protocolos en CAP-AgentTool

En la herramienta para crear controles de agente CAP-AgentTool se permite al programador que especifique los protocolos que el agente va a usar. En [8] se presenta el trabajo que da soporte a esta herramienta para programar agentes. Algunas extensiones fueron realizadas a CAP-AgentTool para permitir que los agentes utilicen los PI, y a continuación se explican. En general, con este trabajo se trata de dotar al agente con cierto grado de flexibilidad con respecto a sus capacidades internas. Al indicarle los protocolos que el agente va a usar, la herramienta está diseñada para incrustar el código fuente necesario para controlar la implementación y ejecución del protocolo.

Con esto se trata de hacer la implementación generalizada de la funcionalidad de protocolos lo cual conlleva a que los detalles de la implementación específicos de aplicación que quiera utilizar los protocolos en los agentes estén abiertos a las necesidades de cada una.

La implementación de los protocolos que aquí se da se concentra en especificar y controlar la secuencia de mensajes que cada protocolo tiene definidos, pero en cada uno de ellos existen aspectos que deben ser dejados abiertos. En este sentido, agentes que los incorporen a su funcionalidad interna están obligados a ofrecer la cionalidad específica de la aplicación que los utilice o del dominio al que corresponde el agente. En la figura 1 se aprecia la interfaz que existe en la herramienta CAP-AgentTool para especificar los protocolos que un nuevo tipo de agente va a incorporar a su funcionalidad.

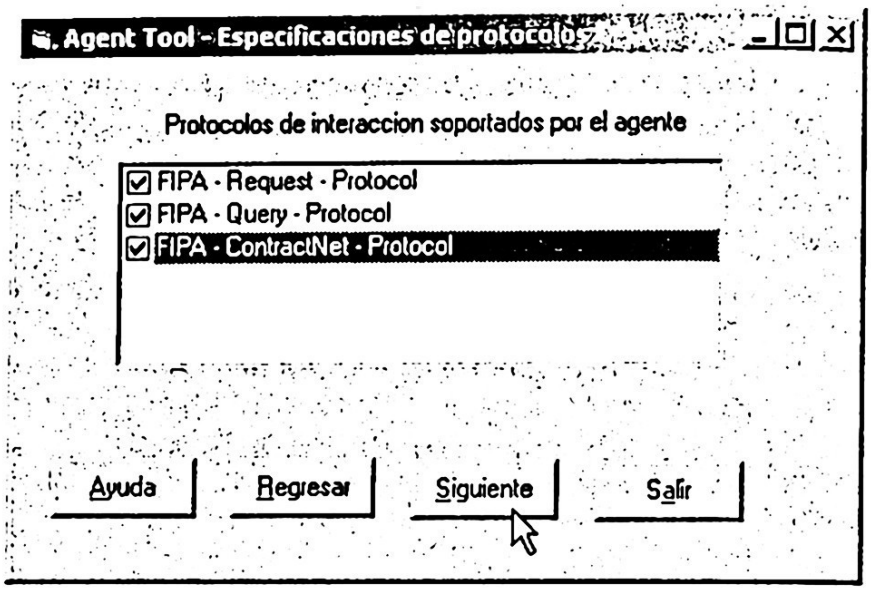

Fig. 1. Selección de protocolos de interacción en CAP-AgentTool

Cuando se procede a la etapa de generación del control de agente en la herramienta, se verifica si el programador seleccionó PI de la lista y si fue así, entonces, incrusta en el código fuente del control de agente las instrucciones necesarias para permitir seguimiento al protocolo. Como parte de este código está la sección en la que se verifican los diferentes mensajes que pueden pertenecer a una conversación y algunos métodos internos que se requieren para tomar las decisiones que son necesarias en gunos puntos de la interacción y que quedan abiertos a la implementación específica de la aplicación. Para cada uno de los protocolos considerados, se crearon plantillas genéricas en archivos de texto, los cuales contienen el código necesario mencionado anteriormente para el funcionamiento de los protocolos. Estos archivos se van a usar dependiendo del protocolo que se añada en tiempo de diseño de cada clase de agente.

3.3 Implementación del protocolo FIPA-Request

El protocolo de interacción *FIPA-Request* [11] permite a un agente que solicite a otro la realización de una acción y el agente receptor realiza la acción o responde que no puede realizarla. La representación de este protocolo se da en la figura 2.

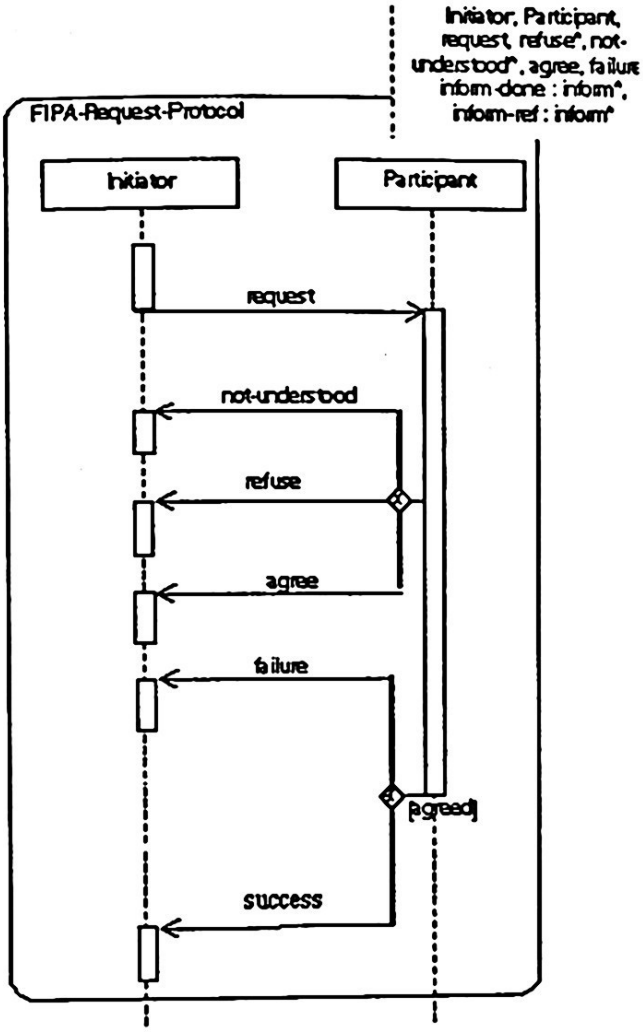

Fig. 2. Diagrama del protocolo Fipa-Request

El agente que incluye este protocolo verifica el contenido del mensaje *request* cada vez que recibe un mensaje de este tipo. Si determina que el mensaje trae consigo en el campo *protocol* la cadena "FIPA-Request-Protocol" entonces el agente entiende que este mensaje debe ser tratado de forma consistente con el protocolo *request*. Para lograr esto, debe utilizar los objetos de contenido que existen de acuerdo con la sintaxis de cada acto comunicativo. En el caso específico de *request*, el contenido del mensaje es una expresión de acción que denota la acción solicitada. Para manipular esta expresión de acción se puede utilizar la clase *Accion* que permite utilizar las funciones de crear y leer los elementos que forman parte de una acción, según se especifica en el lenguaje de contenido FIPA-RDF0 [12], el cual está disponible para estos agentes como parte de las herramientas de programación de CAP-AgentTool. Si la estructura del mensaje o el contenido del mensaje no están bien formados (su sintaxis o semánti-

ca no es correcta), el agente contesta con un mensaje del tipo *not-understood* para indicarle al agente que solicita la acción que no pudo entender el mensaje.

Por el contrario, si el agente puede procesar el mensaje y la acción expresada en el contenido, entonces procede a buscar dicha acción en la base de acciones que cada agente posee. Si la acción solicitada es encontrada, entonces el agente debe decidir si ejecuta la acción o se rehúsa. Para decidir cuándo se realiza una acción o no, se creó una función llamada "Decidir_Accion" recibiendo como parámetro una variable de tipo vector para indicar las condiciones de la realización de la acción expresadas como hechos o proposiciones del dominio, de acuerdo al lenguaje de contenido. En el listado de código siguiente se muestra la implementación del método "ProtocoloRequest" que los agentes poseen para controlar el seguimiento del protocolo de interacción *FIPA-Request*.

```
Public Sub ProtocoloRequest()
  'preguntar si utiliza el protocolo FIPA-Request-Protocol
  Dim accionr As requestContent
  Set accionr = New requestContent
  If parser.protocol = "FIPA-Request-Protocol" Then
  'ver si entiende el mensaje
      If (accionr.deparse(parser.content) = True) Then
      'buscar la accion
            If (buscarAccion(accionr.accion) = True) Then
                  'llamar al procedimiento que decide
                  '       si se realiza la accion o no
                Dim Requisitos As Collection
                Set Requisitos = New Collection
                  'agregar los parametros necesarios para tomar
                  '     la decisión de realizar o no una acción
                If (Decidir_Accion(Requisitos) = True) Then
                     'si la encontró invocar la accion
                      InvocarAccion accionr.accion
                Else
                   ' se reusa hacer la accion y manda un refuse
                  ExecRefuse 1, "FIPA-Request-Protocol"
                End If
            Else
               ExecNotUnderstood 1, "FIPA Request-Protocol"
            End If
       Else
            'si no es que no entiende el mensaje llamar
            '   al notunderstood
         ExecNotUnderstood 2, "FIPA-Request-Protocol"
       End If
     End If
End Sub
```

Si la función "Decidir_Accion" se evalúa a verdadero, entonces el agente invoca la acción por medio de su función interna "InvocarAccion". Dentro de esta función el agente determina si la acción invocada fue realizada con éxito. En este caso, envía un mensaje *success* al agente que solicitó la acción, de lo contrario envía un mensaje *failure*.

3.4 Implementación del protocolo FIPA-Query

En el protocolo de interacción *FIPA-Query* [13], el agente receptor es solicitado a realizar un tipo de acto *inform*. El acto *inform* es una consulta, y hay dos tipos de actos *Query: query-if y query-ref* y cualquiera de estos actos pueden ser usados para iniciar este protocolo. En cualquier caso, un *inform* es usado en respuesta, aunque el contenido del *inform* dado en respuesta a un *query-ref* debería ser una expresión de referencia.

En este protocolo, al agente destinatario se le pide que efectúe una acción de carácter informativo. El remitente puede contestar con un *not-understood*, un *failure*, un *refuse* o un *inform*.

Según las especificaciones de FIPA el Protocolo de interacción *FIPA-Query* funciona de la siguiente manera:

1.- El agente iniciador envía un *query-if* o un *query-ref* al agente participante que lleva la proposición a comprobar.
2.- El agente participante debe contestar con *not-understood*, *refuse*, *failure* o *inform*.
3.- Responde con *not-understood* cuando no entiende el mensaje.
4.- Responde con *refuse* cuando se rehúsa a comprobar el hecho o proposición.
5.- Responde con *failure* cuando el agente no tiene información sobre el hecho o proposición.
6.- Responde con *inform* cuando encontró la proposición sin ningún problema.

Cuando se crea un agente y se selecciona que va a utilizar el protocolo *FIPA-Query* entonces se le añade una porción de código fuente para hacer la invocación del método "ProtocoloQueryIf" que es el que se encarga de verificar el seguimiento del protocolo. En el funcionamiento de este método se realizan las siguientes cosas:

Cada vez que le llega un mensaje de tipo *query* al agente, éste invoca el método que le da seguimiento al protocolo. Internamente en el método, se verifica que el campo *protocol* del mensaje recibido lleve la cadena "FIPA-Query-Protocol". Si es así, se procede a obtener la proposición que denota el hecho que se está solicitando para comprobar en la base de conocimientos del agente; esto se hace través de un objeto de contenido *QueryIfContent*, para manipular el contenido de los actos *query*.

Si el mensaje es correcto, entonces se manda buscar el hecho solicitado con los mecanismos internos que cada agente posee para consultar su base de conocimientos. Si encontró la proposición solicitada, debe decidir si se genera la respuesta. Para esto, el programador del agente tiene que implementar la función de decisión llamada "Decidir_Proposicion" que está abierta para que se establezcan las condiciones bajo las cuales el agente va a realizar la acción de informar o se va a rehusar a hacerlo. Si ésta función devuelve verdadero significa que decide generar el *inform* y genera el mensaje de respuesta; si devuelve falso significa que decide no generar el *inform* por lo tanto invoca al método "ExecRefuse" indicando que se rehúsa a generar el mensaje de respuesta.

Si al buscar el hecho solicitado el agente no lo encuentra, entonces genera un mensaje ACL de tipo *failure* que indica que el agente no tiene información sobre la proposición dada y por lo tanto se considera un fallo.

En caso de que la estructura del mensaje recibido originalmente este mal construido se genera un mensaje de tipo *not-understood* y termina la conversación.

3.5 Implementación del protocolo FIPA-Contract-Net

En el protocolo de interacción *FIPA-ContractNet* [14], un agente toma el rol de ciador de la interacción en busca de que una tarea sea realizada por uno o más de grupo de otros agentes y trata optimizar una función que caracteriza a la acción. Esta característica es comúnmente expresada como el precio, en alguna forma específica del dominio, pero podría ser también el tiempo más corto en el que se completa la ción o una aceptable distribución de tareas. Los agentes que utilizan este protocolo pueden ocupar dos roles: iniciador y participante según sea el caso.

El iniciador del protocolo solicita propuestas de otros agentes por medio del envío de un mensaje *call for proposals* o *cfp* en el que especifica la acción y algunas condiciones para su ejecución. Los agentes que reciben el *cfp* son vistos como participantes potenciales y son capaces de generar propuestas para realizar la acción a través mensajes *propose*. Las propuestas del participante incluyen sus precondiciones para realización de la acción, las cuales pueden ser el precio o el tiempo en el que la tarea será realizada, etc. Alternativamente, el participante se puede rehusar a generar y enviar propuesta. Una vez que el tiempo límite especificado ha pasado, el iniciador evalúa las propuestas recibidas y selecciona a los agentes para que realicen la acción. Uno, varios o ningún agente puede ser seleccionado. A los agentes de las propuestas seleccionadas se les envía un mensaje *accept-proposal* y los demás recibirán un *reject-proposal.* Las propuestas están ligadas al participante, así que, una vez que el iniciador acepta la propuesta, el participante adquiere un compromiso de realizar la acción. Una vez que el participante ha realizado la acción, este envía un mensaje terminación al iniciador.

Para la implementación de este protocolo se creó un procedimiento llamado "*ProtocoloContractNet*". Cuando el agente recibe un mensaje *cfp*, verifica si la estructura del contenido del mensaje que recibió es correcta. Si es así, significa que el mensaje esta bien construido (sintáctica y semánticamente) y por lo tanto llama a la función "*buscarAccion*", la cual busca el nombre de la acción a realizar, si no encontró el acto envía un mensaje *NotUnderstood,* indicando que el acto no es soportado. De lo contrario, va a invocar la función "*Decidir_Accion*", con sus parámetros correspondientes, donde se implementan las especificaciones necesarias para que el programador establezca las reglas para decidir si se realiza o no la acción, dependiendo del conjunto de hechos que fueron recibidos como parámetro. Si la función se evalúa a verdadero, significa que la acción puede ser realizada, por lo tanto invoca la función "*elabora_propuesta*". En este método, el programador debe elaborar y enviar un mensaje *propose* con las condiciones para formar la propuesta respectiva (puede aplicar algoritmos diversos para representar la función de decisión). De lo contrario si la acción no puede ser realizada, entonces envía un mensaje *Refuse*, indicando su desinterés por realizar la acción solicitada.

Cuando el agente iniciador de la conversación recibe un mensaje *propose,* este verifica si el contenido de la propuesta recibida esta bien estructurado. Si está correcto si no ha transcurrido el tiempo de espera de las propuestas entonces almacena las propuestas. En caso de que se reciban *propose* fuera de tiempo llama al procedimiento "*ExecRejectProposal*" en el cual envía un mensaje *reject-proposal* indicando el motivo por el cual fue rechazada, entre otras cosas.

Si un agente participante de la conversación recibe un mensaje *accept-proposal,* significa que el agente iniciador aceptó su propuesta. Esto implica automáticamente que deberá realizar la acción solicitada por medio del mecanismo interno de los agen-

tes (su método InvocarAccion). El agente participante puede generar y enviar un mensaje *failure* cuando la acción no se realizó correctamente o se interrumpió por alguna circunstancia. De otra manera, un mensaje *success* se envía para informar que la acción se realizó de manera satisfactoria junto con los resultados obtenidos. Cuando se recibe un *failure*, *not-understood*, *refuse*, *reject-proposal* o *success* se termina la negociación entre los agentes implicados (iniciador -participante).

Existen unos procedimientos adicionales para lograr el funcionamiento del Protocolo *FIPA-ContractNet*, como es el caso del método "*IniciarProceso*" para empezar la secuencia de mensajes. Se invoca desde una aplicación, indicándole al agente iniciador, los nombres de los agentes a los que enviará el mensaje *cfp*, el tiempo que deberá estar esperando propuestas, la acción solicitada y las condiciones que se establecen para realización de dicha acción.

Una vez transcurrido el tiempo establecido para la interacción, se procede a analizar propuestas en donde se invoca a la función "*AceptaPropuesta*" para que el programador analice las condiciones para decidir si una propuesta es aceptada. Si está de acuerdo en la propuesta que recibió entonces invoca al método "*ExecAcceptProposal*", indicando que ha aceptado la propuesta, de lo contrario invoca al método "*ExecRejectProposal*", diciendo así que no desea que realice la acción, y debe indicar el motivo por el cual la propuesta no fue aceptada.

Para el caso de la implementación que aquí se presenta (para los tres protocolos), se sustituyó el acto comunicativo *inform-done* especificado en algunos protocolos por el acto *success* que, aunque no es parte de la biblioteca de actos comunicativos de FIPA, se propone su uso para expresar la necesidad de informar los resultados que se obtienen al ejecutar acciones por los agentes (esto es más consistente con el lenguaje de contenido FIPA-RDF0 utilizado para expresar los diferentes objetos de contenido de los mensajes).

4 Aplicación de demostración

Con la finalidad de mostrar la manera en que una aplicación puede utilizar agentes y los PI que aquí se han descrito se presenta esta sección. La idea es que se presenta una aplicación programada en el lenguaje Visual Basic 6.0 que contiene un grupo de siete agentes, cuya clase fue creada con la herramienta CAP-AgentTool, y que incorpora en su funcionalidad el uso del protocolo de interacción FIPA -ContractNet. Los nombres de los agentes utilizados en la aplicación son: AgenteCN1, AgenteCN2, así hasta AgenteCN7. El agente AgenteCN1 es el encargado de iniciar la conversación planteada por el protocolo ContractNet, por medio de una solicitud de propuestas al resto de los agentes.

Lo más importante que hay que resaltar es que los protocolos llegan a ciertos puntos o estados en los que se requiere tomar decisiones que son específicas del dominio de las aplicaciones que los utilizan. En este sentido, la implementación de estas partes específicas referentes al protocolo en cuestión requiere varias consideraciones.

Es importante tomar en cuenta que se considera que los agentes pueden decidir la realización de las acciones necesarias para lograr sus objetivos. Esto lleva a que se necesite implementar mecanismos que permitan al programador del protocolo tomar las decisiones apropiadas. En el caso particular de ContractNet, se han incluido varios métodos que ayudan a tomar tales decisiones. Por ejemplo, cuando a un agente parti-

cipante se le pide que envíe una propuesta para realizar una acción, este agente debe decidir si puede o quiere realizar la acción solicitada por medio de la implementación de su método respectivo, tal y como se explicó en la sección 3.5. De igual manera, específico de la aplicación determinar la forma de generar una propuesta, analizarlas, y aceptarlas si es que el agente así lo ha determinado. En la implementación de este ejemplo, estas funciones de decisión se han resuelto por medio de la asignación costos numéricos asignados arbitrariamente a las acciones solicitadas y los límites aceptación de propuestas para cada ejecución. En la figura 3 se muestra la interfaz la aplicación y el despliegue de los diferentes mensajes intercambiados entre los agentes.

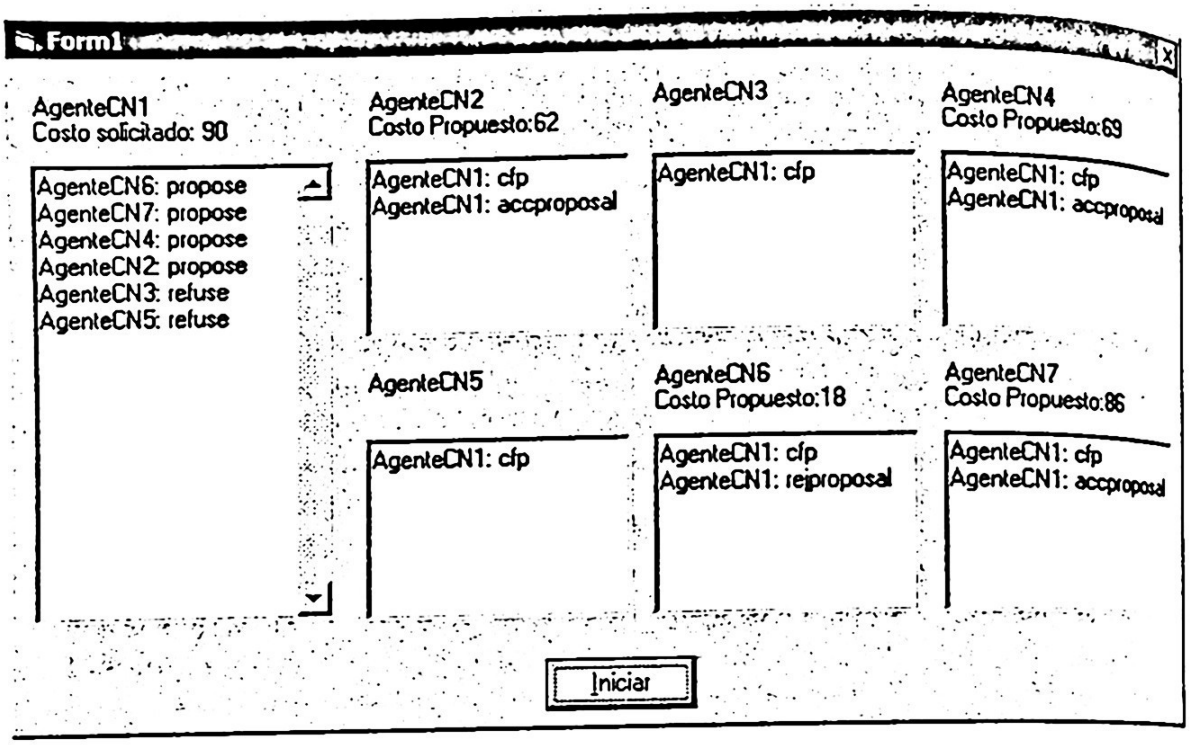

Fig. 3. Prueba del protocolo *FIPA-Contract-Net*

5 Conclusiones

Se estudió con detalle las características de la especificación de FIPA con respecto la implementación de tres protocolos de interacción que son parte de su biblioteca protocolos de interacción. Estos son los protocolos *FIPA-request*, *FIPA-query* y *FIPA-ContractNet*.

Se revisó la definición general de cada uno de los protocolos y a partir de esto logró desarrollar un mecanismo de implementación que se enfoca en las secuencias generales establecidas para dar seguimiento a los protocolos en sus distintas fases de interacción, dejando los detalles específicos de las aplicaciones abiertos en cada caso, por medio de funciones generales. La implementación presentada fue desarrollada con la herramienta de creación de agentes CAP-AgentTool, en donde se añadió la parte necesaria para permitir a los programadores que especifiquen si sus agentes van a lizar los protocolos implementados. Por medio de esta parte se pueden generar

agentes con la funcionalidad básica necesaria para implementar cada uno de los protocolos. Se explica el proceso que deben seguir los programadores de agentes para lograr una implementación específica de los protocolos seleccionados durante la creación del agente. Los agentes creados con esta herramienta tienen la característica de que son desplegados a través de una plataforma de agentes CAP.

Por último, se explican los detalles de la implementación de los protocolos a través de agentes de ejemplo y aplicaciones de prueba de la funcionalidad. En cada paso de la implementación se hace énfasis en el trabajo que debe realizar el programador y los detalles que debe tomar en cuenta para lograr un funcionamiento adecuado de los mismos.

Referencias

1. R. H. Gutman, A.G. Moukas, y P. Maes. Agent-mediated electronic comerse: A survey. The Knowledge Engineering Review, 13(2):147-159, 1998.
2. J. Yamamoto y K. Sycara. A stable and efficient buyer coalition formation scheme for e - marketplaces. In proceedings of the 5th International Conference on Autonomous Agents, 2001.
3. FIPA Interaction Protocol Library Specification http://www.fipa.org/specs/fipa00025/XC00025E.html
4. Contreras, M., 2001. Desarrollo de una infraestructura MultiAgentes para organizaciones virtuales. CIC-IPN, México D.F. 2001.
5. Sheremetov, L. & Contreras M., 2001. Component Agent Platform. CEEMAS-2001
6. Genesereth, M. R. & Ketchpel, S. P., 1994. Software Agents, *Communications of the ACM* 37 (7), 48-53.
7. Appleman Dan, Desarrollo de componentes COM/ActiveX con Visual Basic 6, Editorial Prentice Hall, Madrid España, año 2000
8. Germán, E. 2002. Desarrollo de una herramienta para la creación de agentes sobre la plataforma de agentes componentes. CIC-IPN, México D.F. 2002
9. *FIPA Communicative Act Library Specification*, http://www.fipa.org/fipa00037/
10. *FIPA ACL Message Structure Specification*, http://www.fipa.org/specs/fipa00061/
11. FIPA Request Interaction Protocol Specification http://fipa.org/specs/fipa00026/
12. *FIPA RDF Content Language Specification*. http://www.fipa.org/specs/fipa00011/
13. FIPA Query Interaction Protocol Specification. http://fipa.org/specs/fipa00027/
14. FIPA Contract Net Interaction Protocol Specification. http://fipa.org/specs/fipa00029/