Práctica 3. Comunicación entre agentes

DISEÑO BASADO EN AGENTESCURSO 2024/2025 MANUEL JESÚS COBO MARTÍN

UNIVERSIDAD DE GRANADA

MANUEL JESÚS COBO MARTÍN
DEPARTAMENTO DE CIENCIAS DE LA COMUTACIÓN E INTELIGENCIA ARTIFICIAL |
UNIVERSIDAD DE GRANADA

1 Introducción a FIPA

El estándar FIPA se creó en 1996 como asociación internacional sin ánimo de lucro para desarrollar un conjunto de normas relativas a la tecnología de agentes de software. Los miembros iniciales, un conjunto de organizaciones académicas e industriales, redactaron una serie de estatutos que guiarían la producción de un conjunto de especificaciones estándar para las tecnologías de agentes software. En aquel momento, los agentes de software ya eran muy conocidos en la comunidad académica, pero hasta la fecha sólo habían recibido una atención limitada por parte de las empresas comerciales, más allá de una perspectiva exploratoria. El consorcio acordó elaborar unas normas que constituirían la base de una nueva industria al poder utilizarse en un gran número de aplicaciones.

El núcleo de la FIPA es el siguiente conjunto de principios:

1. Las tecnologías de agentes proporcionan un nuevo paradigma para resolver problemas antiguos y nuevos
2. Algunas tecnologías de agentes han alcanzado un grado de madurez considerable.
3. Para ser útiles, algunas tecnologías de agentes requieren estandarización.
4. Otros foros de normalización han demostrado que la normalización de tecnologías genéricas es posible y ofrece resultados eficaces.
5. La normalización de la mecánica interna de los propios agentes no es la principal preocupación, sino más bien la infraestructura y el lenguaje necesarios para una interoperabilidad abierta.

Un mensaje FIPA tiene la siguiente estructura:

- Performative. Tipo de acto comunicativo del mensaje.
- Sender. Identidad del emisor del mensaje.
- Receiver. Identidad de los destinatarios del mensaje.
- Reply-to. A qué agente dirigir los mensajes posteriores dentro de un hilo de conversación
- Content. Contenido del mensaje.
- Language. Idioma en el que se expresa el parámetro de contenido.
- Encoding. Codificación específica del contenido del mensaje.
- Ontology Referencia a una ontología para dar significado a los símbolos del contenido del mensaje.
- Protocol. Protocolo de interacción utilizado para estructurar una conversación.
- Conversation-id. Identidad única de un hilo de conversación.
- Reply-with. Expresión que utilizará el agente que responde para identificar el mensaje.
- In-reply-to. Referencia a una acción anterior de la que el mensaje es una respuesta.
- Reply-by. Hora/fecha en la que debe recibirse una respuesta.

La FIPA define la comunicación en términos de una función o acción, denominada acto comunicativo, realizada por el acto de comunicar. Esta norma ofrece una biblioteca de todos los actos comunicativos especificados por la FIPA.

- Accept proposal. La acción de aceptar una propuesta presentada previamente para realizar una acción
- Agree. Acción de comprometerse a realizar una acción, posiblemente en el futuro.
- Cancel. La acción de un agente de informar a otro agente de que el primer agente ya no tiene la intención de que el segundo agente realice alguna acción
- Call for Proposal (CFP). La acción de solicitar propuestas para realizar una acción determinada
- Confirm. El emisor informa al receptor de que una proposición dada es verdadera, cuando se sabe que el receptor no está seguro de la proposición.
- Disconfirm. El emisor informa al receptor de que una proposición dada es falsa, cuando se sabe que el receptor cree, o cree probable, que la proposición es verdadera.
- Failure. Acción de comunicar a otro agente que se ha intentado realizar una acción, pero que el intento ha fracasado.

- Inform. El emisor informa al receptor de que una proposición dada es verdadera
- Inform If. Una macroacción para que el agente de la acción informe al destinatario de si una proposición es cierta o no
- Inform Ref. Una macroacción que permite al emisor informar al receptor de un objeto que el emisor cree que corresponde a un descriptor específico, por ejemplo, un nombre.
- No Understood. El emisor del acto (por ejemplo, i) informa al receptor (por ejemplo, j) de que ha percibido que j ha realizado alguna acción, pero que i no ha entendido lo que j acaba de hacer. Un caso muy habitual es que i comunique a j que no ha entendido el mensaje que j acaba de enviarle.
- Propagate. El emisor pretende que el receptor trate el mensaje incrustado como enviado directamente al receptor, y quiere que el receptor identifique a los agentes indicados por el descriptor dado y les envíe el mensaje de propagación recibido.
- Propose. Acción de presentar una propuesta para realizar una determinada acción, dadas ciertas condiciones previas.
- Proxy. El emisor desea que el receptor seleccione agentes objetivo indicados por una descripción dada y les envíe un mensaje incrustado.
- Query If. Acción de preguntar a otro agente si una proposición dada es verdadera o no.
- Query Ref. Acción de pedir a otro agente el objeto al que se refiere una expresión referencial
- Refuse. La acción de negarse a realizar una acción determinada y explicar el motivo de la negativa.
- Reject Proposal. Acción de rechazar una propuesta para realizar alguna acción durante una negociación.
- Request. El emisor solicita al receptor que realice alguna acción. Una clase importante de usos del acto de petición es solicitar al receptor que realice otro acto comunicativo
- Request When. El emisor desea que el receptor realice alguna acción cuando una proposición determinada se convierte en verdadera
- Request Whenever. El emisor quiere que el receptor realice alguna acción tan pronto como alguna proposición se convierta en verdadera y, a partir de entonces, cada vez que la proposición vuelva a ser verdadera.
- Subscribe. El acto de solicitar a una intención persistente que notifique al remitente el valor de una referencia y que vuelva a notificar cada vez que cambie el objeto identificado por la referencia.

2 Comunicación básica entre agentes

En una comunicación intervienen dos o más agentes, el agente que envía el mensaje, y el que lo recibe. Es decir, hay un emisor y un receptor del mensaje.

De este modo, un agente puede enviar un mensaje haciendo uso del método `send` que se hereda al extender la clase `Agent` de `JADE`. Para enviar un mensaje, podemos usar el siguiente código. El `AID` es el identificador del receptor y viene dado en los comandos de inicio.

```
ACLMessage msg = new ACLMessage();
msg.addReceiver(new AID("mjcobo-receiver", AID.ISLOCALNAME));
msg.setContent("Hola agente");
send(msg);
```

Para recibir un mensaje, podemos usar el siguiente código.

```
ACLMessage msg = blockingReceive();
```

Realmente, para recibir mensajes existe el método no bloqueante `receive()`, pero para estas prácticas usaremos el método bloqueante, ya que no podremos avanzar hasta que recibamos un mensaje.

3 Comunicación entre agentes con performativas

En el ejemplo hemos realizado una conversación muy básica. De hecho, el constructor por defecto del `ACLMessage` está obsoleto. Un mensaje se debe crear o bien en replica a otro mensaje, o indicando la performativa en un mensaje nuevo.

Es posible que, tras recibir un mensaje, el agente receptor quiera enviar un mensaje de vuelta al agente emisor. El mensaje debe ir en la misma secuencia de mensajes, y no como un mensaje nuevo. Esto hace que se pueda seguir la traza de toda la conversación. Para ello, se puede hacer uso del método `createReply()` del mensaje.

Cuando establecemos una comunicación entre unos agentes, los agentes pueden estar en estados diferentes, y por lo tanto, pueden estar a la espera de recibir un tipo de mensaje en particular. Esto podemos controlarlo comprobando los diferentes campos del mensaje recibido. De modo que, si se recibe un mensaje incorrecto o no esperado, se puede terminar la ejecución del agente, o informar del error.

Para modelar los estados del agente, se puede emplear una estructura del tipo switch-case.

Imaginemos el siguiente comportamiento. El agente emisor envía un saludo al agente receptor, y este segundo le contesta agradeciendo dicho saludo. Por un lado, tendremos al agente emisor. Este tendrá varios estados internos: 1) emitir el mensaje y 2) recibir mensaje de respuesta. Nótese que en el estado 2 se comprueba que el mensaje sea continuación del mensaje iniciado preveniente mediante el uso del `conversation-id`. Así el código podría ser el siguiente:

```
switch (this.step) {
    case 0 -> {

        ACLMessage msg = new ACLMessage();
        msg.addReceiver(new AID("mjcobo-receiver", AID.ISLOCALNAME));
        msg.setContent("Hola agente. ¿Cómo estás?");
        msg.setConversationId("greeting-conversation");
        myAgent.send(msg);
        this.step = 1;
    }

    case 1 -> {

        ACLMessage msg = myAgent.blockingReceive();

        if (msg.getConversationId().equals("greeting-conversation")) {

            System.out.println("Content of the replay message: " +
  msg.getContent());
            this.finishConversation = true;
            System.out.println("End of conversation");

        } else {
```

```
            System.out.println("Error in the coversation protocol");
            myAgent.doDelete();

        }
    }

    default -> {

        System.out.println("Error in the coversation protocol");
        myAgent.doDelete();
    }
}
```

El código del agente receptor sería el siguiente:

```
ACLMessage msg = myAgent.blockingReceive();

System.out.println("Received this message: " + msg.getContent());

ACLMessage replay = msg.createReply();

replay.setContent("Good, nice to meet you!");
myAgent.send(replay);
```

4 Comunicación mediante un protocolo previamente establecido

La conversación entre los agentes tiene que estar perfectamente protocolizada, para que así, se puedan entender entre ellos de forma autónoma.

Imaginemos el siguiente ejemplo. El agente solicita mediante un `REQUEST` que otro agente le salude. El segundo agente le responde con un `AGREE`, indicando que está dispuesto a saludarle. El primer agente entonces envía un saludo mediante un `INFORM` y el segundo agente le responde con otro `INFORM`. Una vez llegados a este punto, la conversación termina.

Así, el código del agente que inicia la conversación sería el siguiente:

```
switch (this.step) {
    case 0 -> {

        ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
        msg.addReceiver(new AID("mjcobo-receiver", AID.ISLOCALNAME));
        msg.setConversationId(CONVERSATION_ID);
        myAgent.send(msg);
        this.step = 1;
```

```
 }

 case 1 -> {

     ACLMessage msg = myAgent.blockingReceive();

     if (msg.getConversationId().equals(CONVERSATION_ID) &&
msg.getPerformative() == ACLMessage.AGREE) {

         ACLMessage replay = msg.createReply(ACLMessage.INFORM);
         replay.setContent("I'm saying hello using a protocolized
conversation!");

         this.myAgent.send(replay);

         this.step = 2;

     } else {

         System.out.println("Error in the coversation protocol - step
" + 1);
         myAgent.doDelete();
     }
 }

 case 2 -> {

     ACLMessage msg = myAgent.blockingReceive();

     if (msg.getConversationId().equals(CONVERSATION_ID) &&
msg.getPerformative() == ACLMessage.INFORM) {

         System.out.println("Agent receive: " + msg.getContent());

         this.finish = true;

     } else {

         System.out.println("Error in the coversation protocol - step
" + 2);
```

```
            myAgent.doDelete();

        }
    }

    default -> {
        System.out.println("Error in the coversation protocol - step " +
  2);
        myAgent.doDelete();

    }
}
```

Por otro lado, el código del agente receptor sería el siguiente:

```
switch (this.step) {

    case 0 -> {

        ACLMessage msg = myAgent.blockingReceive();
              System.out.println(msg);

        if (msg.getPerformative() == ACLMessage.REQUEST) {

            ACLMessage replay = msg.createReply(ACLMessage.AGREE);

            this.myAgent.send(replay);

            this.step = 1;

        } else {

            System.out.println("Error in the coversation protocol - step
  " + 1);
            myAgent.doDelete();
        }
    }

    case 1 -> {

        ACLMessage msg = myAgent.blockingReceive();
        System.out.println(msg);
```

```
        if (msg.getPerformative() == ACLMessage.INFORM) {

            ACLMessage replay = msg.createReply(ACLMessage.INFORM);
            replay.setContent("Hello! Let's work!");

            this.myAgent.send(replay);

            this.finish = true;

        } else {

            System.out.println("Error in the coversation protocol - step
   " + 2);
            myAgent.doDelete();
        }
    }
}
```

5 Ejercicios

Este es un tutorial que no hay que entregar, pero se recomienda realizarlo en clase para comprender como funciona la comunicación entre agentes en JADE.

Por lo tanto, se recomienda hacer un conjunto de agentes con una comunicación básica (emisor / receptor), así como agentes que se comuniquen mediante un protocolo específico.