# Práctica 3: Interacción Multiagente. Cooperación Curso 2015/16

Protocolos de Interacción. Contract Net.

# Introducción

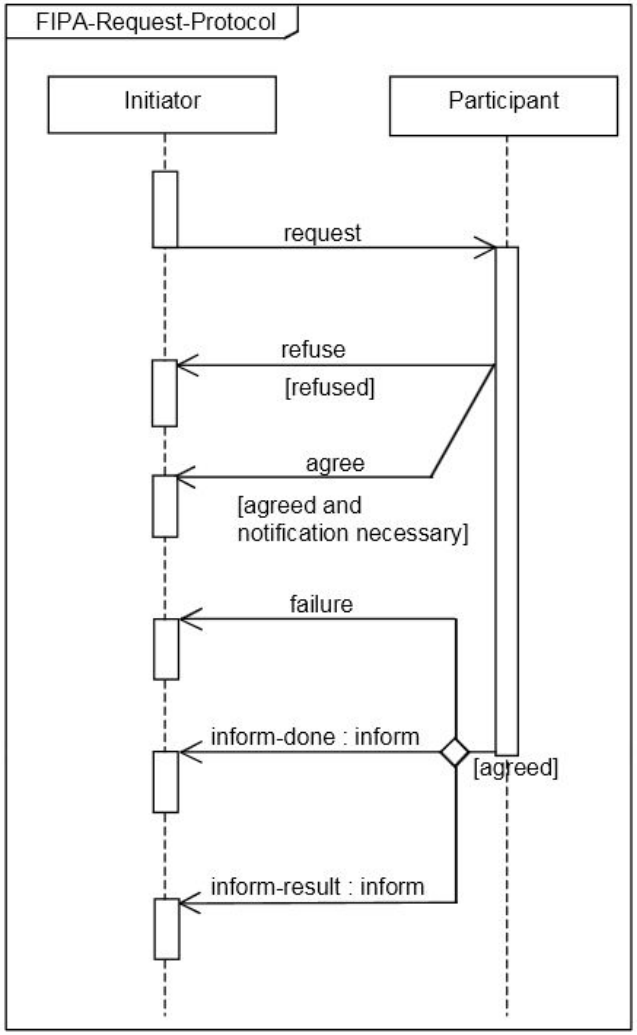
- Un protocolo de interacción es una conversación; una secuencia predeterminada de mensajes intercambiados entre varios agentes.
- Los agentes saben que mensajes pueden enviar a otro/s agente/s y los mensajes que pueden recibir.
- De esta forma un agente puede dialogar con diferentes agentes que utilicen distintos protocolos si aumentar excesivamente la complejidad de su implementación.
- En cualquier conversación hay un agente que la inicia (Initiator) y otro/s que responde/n (Responder o Participant).
- Los protocolos estándar definidos por FIPA son:

| | |
|---|---|
| *FIPA-Request* | *FIPA-Interacted-Contract-Net* |
| *FIPA-Query* | *FIPA-Auction-English* |
| *FIPA-Request-When* | *FIPA-Auction-Dutch* |
| *FIPA-ContractNet* | |

# Introducción

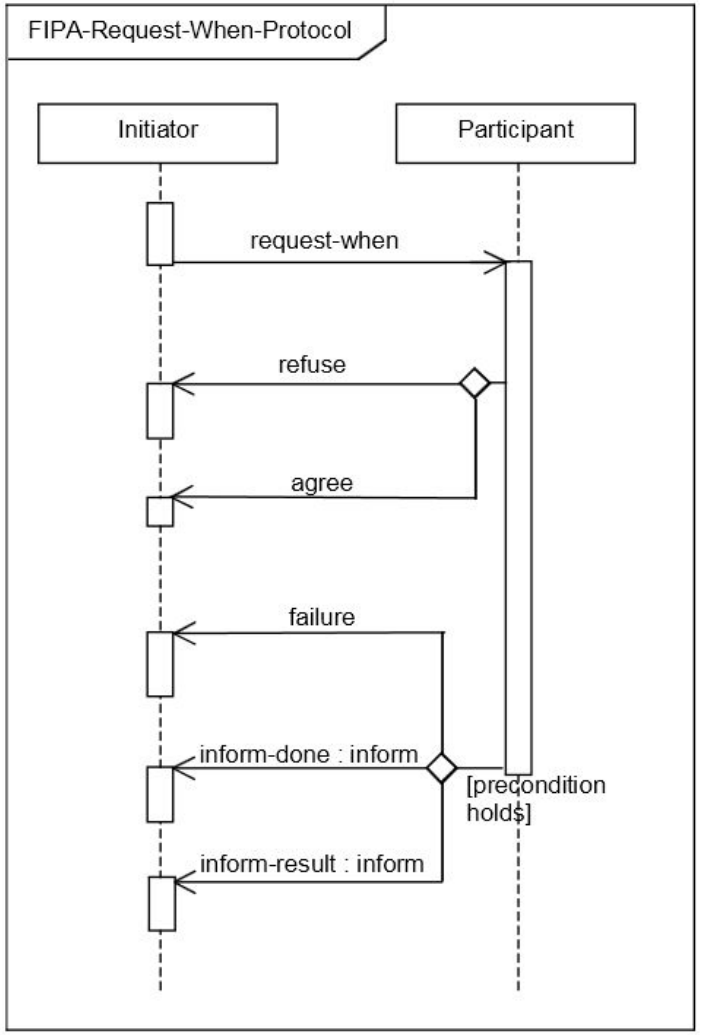
- Utilización de protocolos
  - Cuando un agente quiere utilizar un determinado protocolo en la conversación con otro agente, debe rellenar el parámetro **procotol** del mensaje.
  - Un protocolo finaliza:
    - Se llega al estado final del protocolo.
    - Se elimina el nombre del protocolo del parámetro **:protocol**
  - Si un agente recibe un mensaje que intenta seguir un protocolo que no entiende, tendrá que devolver un mensaje de tipo **refuse**, explicando el motivo por el que se rechaza la comunicación.
  - Si un agente, siguiendo un protocolo, recibe algún mensaje que no está contemplado, devuelve un mensaje del tipo **not-understood**.
    - No se debe responder a un mensaje **not-understood** con otro mensaje **not-understood**. Así evitamos crear bucles infinitos en la interacción.

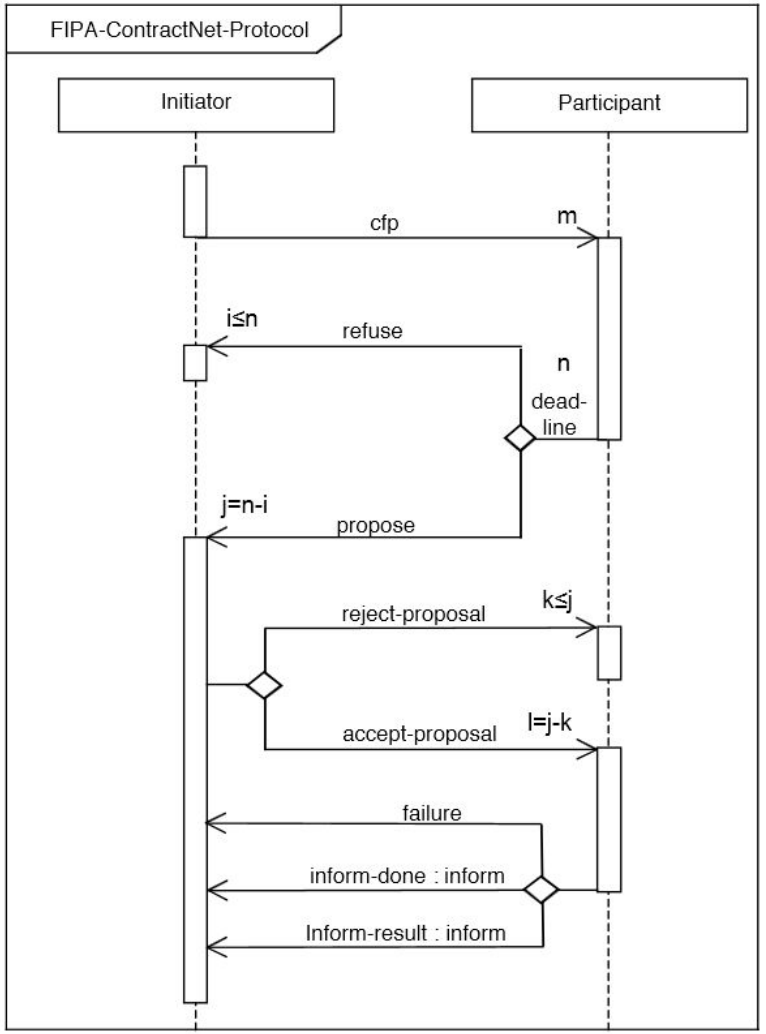
# Protocolo FIPA-Request



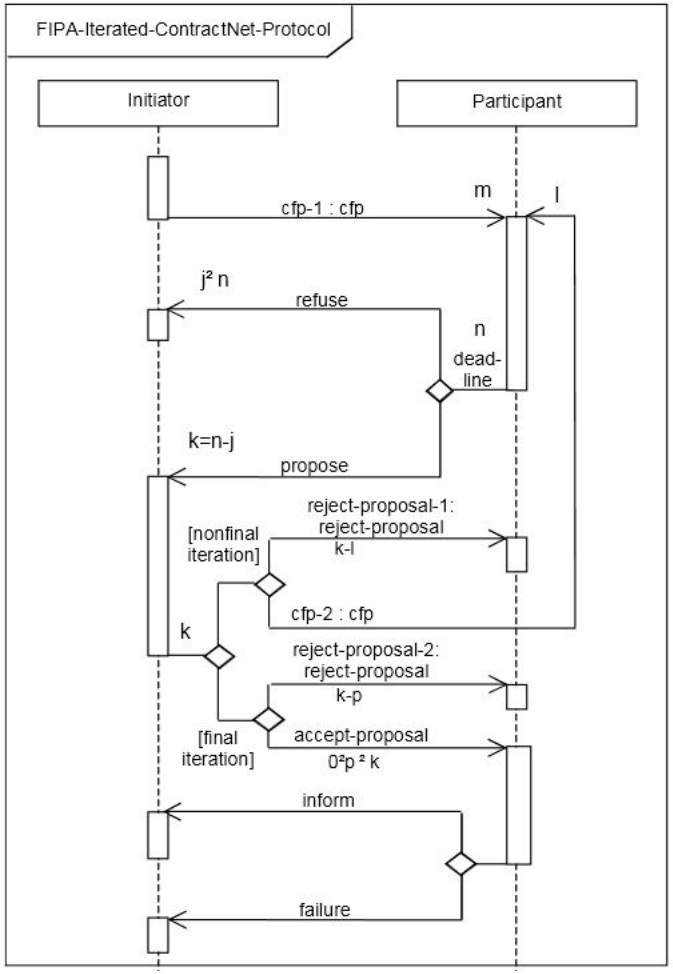

- Utilización:
  - Un agente le pide (**request**) a otro que realice una operación.
  - El participante puede aceptar (**agree**), rechazar (**refuse**) o no entender la petición (**not-understood**).
  - Si acepta la petición se compromete a realizarla y puede pasar:
    - Que no la realice => **failure**
    - Si acaba informa al Initiator con un mensaje **inform**.
    - Si el resultado contiene información responde con **inform-ref**

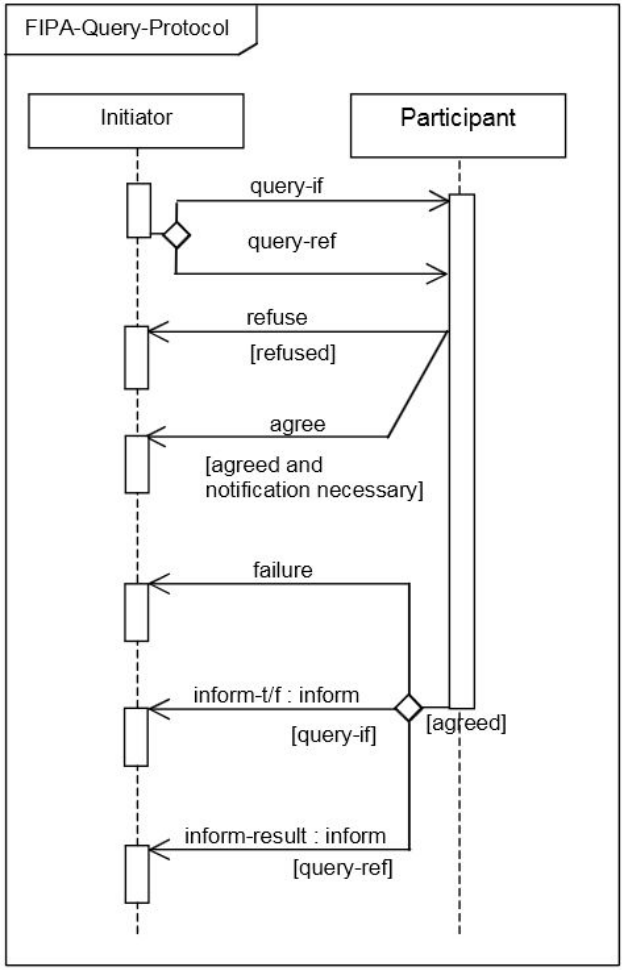
# Más protocolos



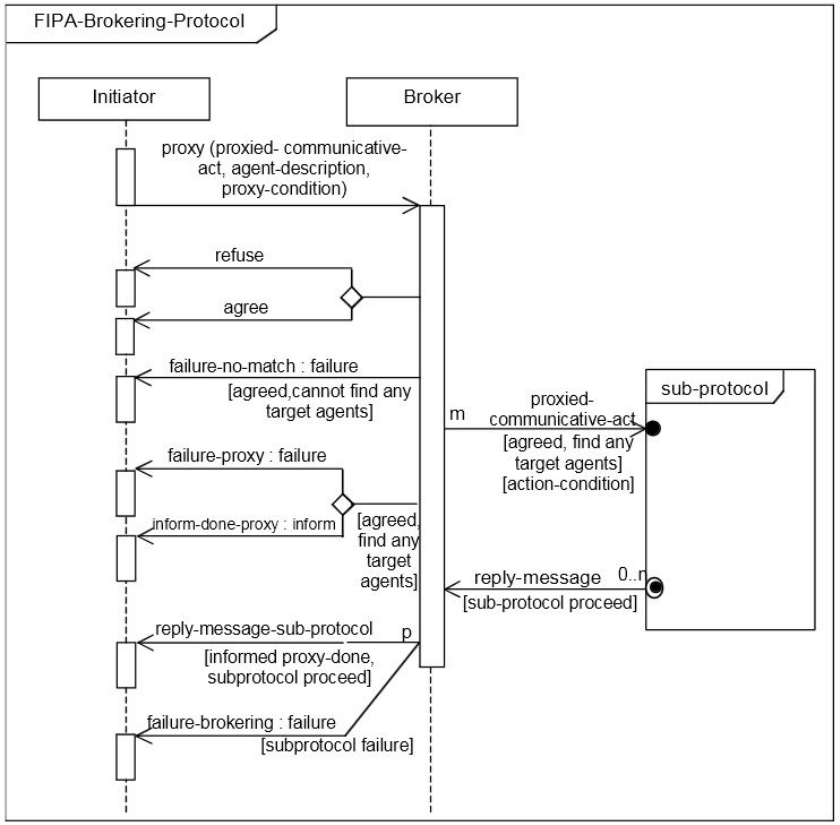



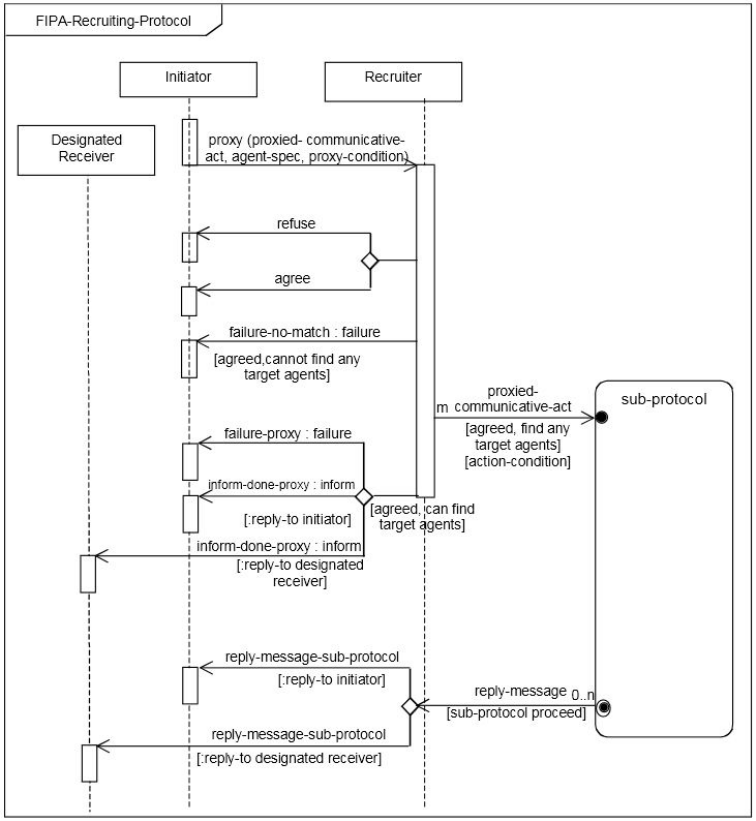
# Más protocolos



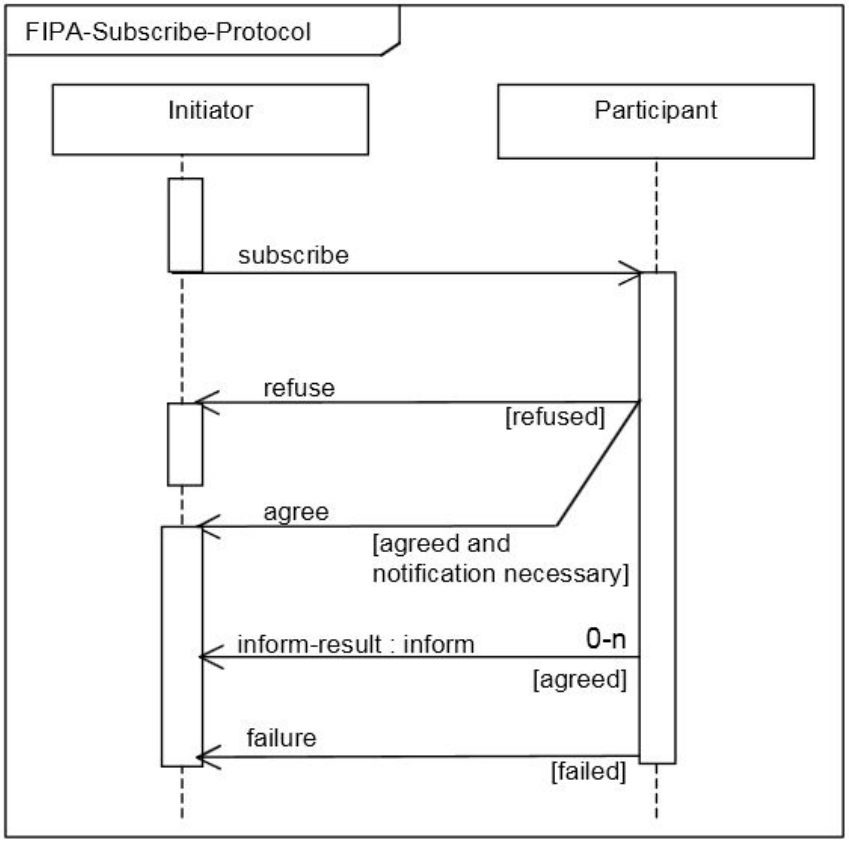



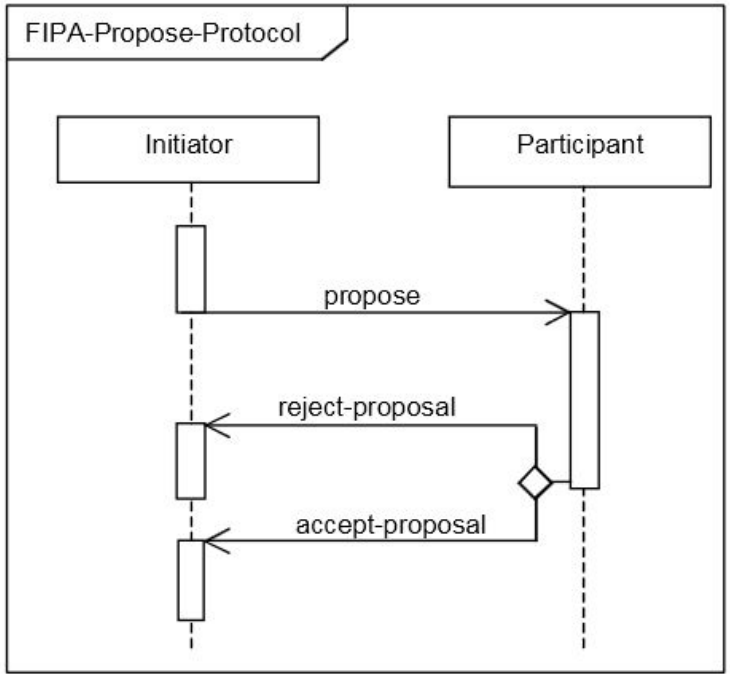
# Más protocolos



# Más protocolos



# Plantillas para mensajes

- Las plantillas permiten:
  - Comprobar el tipo de performativa.
  - Gestionar distintas conversaciones con el mismo protocolo.
  - Obtener el mensaje de la cola, comprobar que es el que esperamos y manejarlo.
    - Si no fuera el mensaje esperado habrá que volverlo a dejar en la cola para que otros comportamientos lo puedan consultar y/o manejar.

```
MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.CFP);
ACLMessage msg = myAgent.receive(mt);
if (msg != null) {
  // Hemos recibido un CFP. Hay que procesarlo
} else {
  block();
}
```

# Plantillas de mensajes

- Se puede especificar cualquier fórmula lógica bien escrita para especificar un test sobre el buzón de entrada para seleccionar mensajes:

```
MessageTemplate mt =
   MessageTemplate.and(MessageTemplate.MathConversatioId("123"),
   MessageTemplate.MatchInReplyTo(req.getReplyWith()));
```

- O incluso definir nuevos test como clases java:

```
public class MyMatchExpression implements MessageTemplate.MatchExpression {
 List senders;
 MyMatchExpression(List l) {
   senders = l;
 }

  public boolean match(ACLMessage msg) {
    AID sender = msg.getSender();
    String name = sender.getName();
    Iterator it = senders.iterator();
    boolean out = false;
    while (it.hasNext() && !out) {
       String tmp =((AID) it.next()).getName();
       if (tmp.equalsIgnoreCase(name)) {
          out = true;
       }
    }
  }
}
```

# Conversaciones concurrentes

- Un agente puede estar atendiendo a varios protocolos de interacción o conversaciones de forma concurrente.
- Los mensajes que se intercambian pueden incorporar información relativa al protocolo o instancia del protocolo al que están siendo dirigidos.
- Hay 3 formas de incluir esta información en un mensaje FIPA-ACL:
  - **conversation-id:** identifica la conversación unívocamente.
  - **reply-with**: identificador que la respuesta debe incluir.
  - **in-reply-to**: identificador de la respuesta a un mensaje que incluía el reply-with

# Ejercicio

- Programar un sistema multiagente con las siguientes características:
  - Hay dos tipos de agentes **Iniciador** (sólo uno) y **Continuador** (varios)
  - Los agentes Continuadores se registran en el DF.
  - El agente Iniciador detecta cíclicamente cada segundo los agentes Continuadores que hay registrados, elige uno al azar y con ese establece una conversación del protocolo *FIPA-Request* en el que le solicita una tarea que le llevará un tiempo al azar de entre 1 y 3 segundos (simulada mediante Thread.sleep(x)). Con un 80% de probabilidad el agente Continuador le contestará con un mensaje `inform-done` y con un 20% con un `failure`.