Laboratorio 3
Páginas amarillas

Técnicas Avanzadas de Inteligencia Artificial

Dpt. Lenguajes y Sistemas Informáticos.
FISS. UPV-EHU

ÍNDICE

Ejemplo BookTrading (1)

- El **DF** (Directory Facilitator) proporciona un servicio de páginas amarillas por medio de los cuales un agente puede encontrar otros agentes que prestan los servicios que necesita para alcanzar sus metas.
- El escenario de este ejemplo incluye algunos agentes que venden libros y otros agentes que compran libros por parte de sus usuarios.
- Cada agente comprador recibe el título de un libro para comprar (el "libro objetivo") como un argumento en la línea de comandos y pide periódicamente a todos los agentes vendedores de libros que proporcionen una oferta. Tan pronto como se reciba una oferta el agente comprador acepta y emite una orden de compra. Si hay más de una oferta, el agente comprador acepta la mejor (la más barata). Después de haber comprado el libro objetivo, el agente comprador termina.

Ejemplo BookTrading (2)

- Cada agente vendedor tiene una GUI por medio del cual el usuario puede insertar nuevos títulos (y el precio asociado) en su catálogo local de libros para la venta. Los agentes vendedores esperar continuamente las peticiones de los agentes compradores. Cuando se le pide que proporcione una oferta por un libro tiene que comprobar si el libro en cuestión está en su catálogo y en este caso proporcionar una respuesta con el precio. De lo contrario se niegan. Cuando reciben una orden de compra lo sirven y eliminan el libro solicitado de su catálogo.
- https://github.com/bluezio/jade-booktrading
- java jade.Boot -gui -nomtp "Buyer1:BookBuyerAgent(El-Hobbit);Buyer2:BookBuyerAgent(El-Hobbit); Seller1:BookSellerAgent;Seller2:BookSellerAgent"

4.1. BookSellerAgent- setup()

```
// The catalogue of books for sale (maps the title of a book to its price)
  private Hashtable catalogue;

  // The GUI by means of which the user can add
  private BookSellerGui myGui;


// Put agent initializations here
  protected void setup() {
    // Create the catalogue
    catalogue = new Hashtable();
    // Create and show the GUI
    myGui = new BookSellerGui(this);
    myGui.show();

     …
```

La librería es una tabla hash con un único ejemplar por libro

Inicialización de variables

Visualización de la interfaz

4.1. BookSellerAgent- setup()

```
protected void setup() {
    …
// Register the book-selling service in the yellow pages
    DFAgentDescription dfd = new DFAgentDescription();
        dfd.setName(getAID());
            ServiceDescription sd = new ServiceDescription();
                sd.setType("book-selling");
                sd.setName("JADE-book-trading");
        dfd.addServices(sd);
    try {
      DFService.register(this, dfd);
    }
    catch (FIPAException fe) {
      fe.printStackTrace();
    }
     …
```

Registro en Páginas Amarillas

4.1. BookSellerAgent- setup()

```
protected void setup() {
   ...

    // Add the behaviour serving queries from buyer agents
   addBehaviour(new OfferRequestsServer());


   // Add the behaviour serving purchase orders from buyer
   addBehaviour(new PurchaseOrdersServer());

} // fin setup()
```

Añadir comportamientos

4.1. BookSellerAgent- takeDown()

```
// Put agent clean-up operations here

 protected void takeDown() {
   // Deregister from the yellow pages
   try {
     DFService.deregister(this);
   }
   catch (FIPAException fe) {
     fe.printStackTrace();
   }
    // Close the GUI
    myGui.dispose();
   // Printout a dismissal message
   System.out.println("Seller-agent "+getAID().getName()+"
    terminating.");
 }
```

Desregistro en Páginas Amarillas

Eliminación de la Interfaz

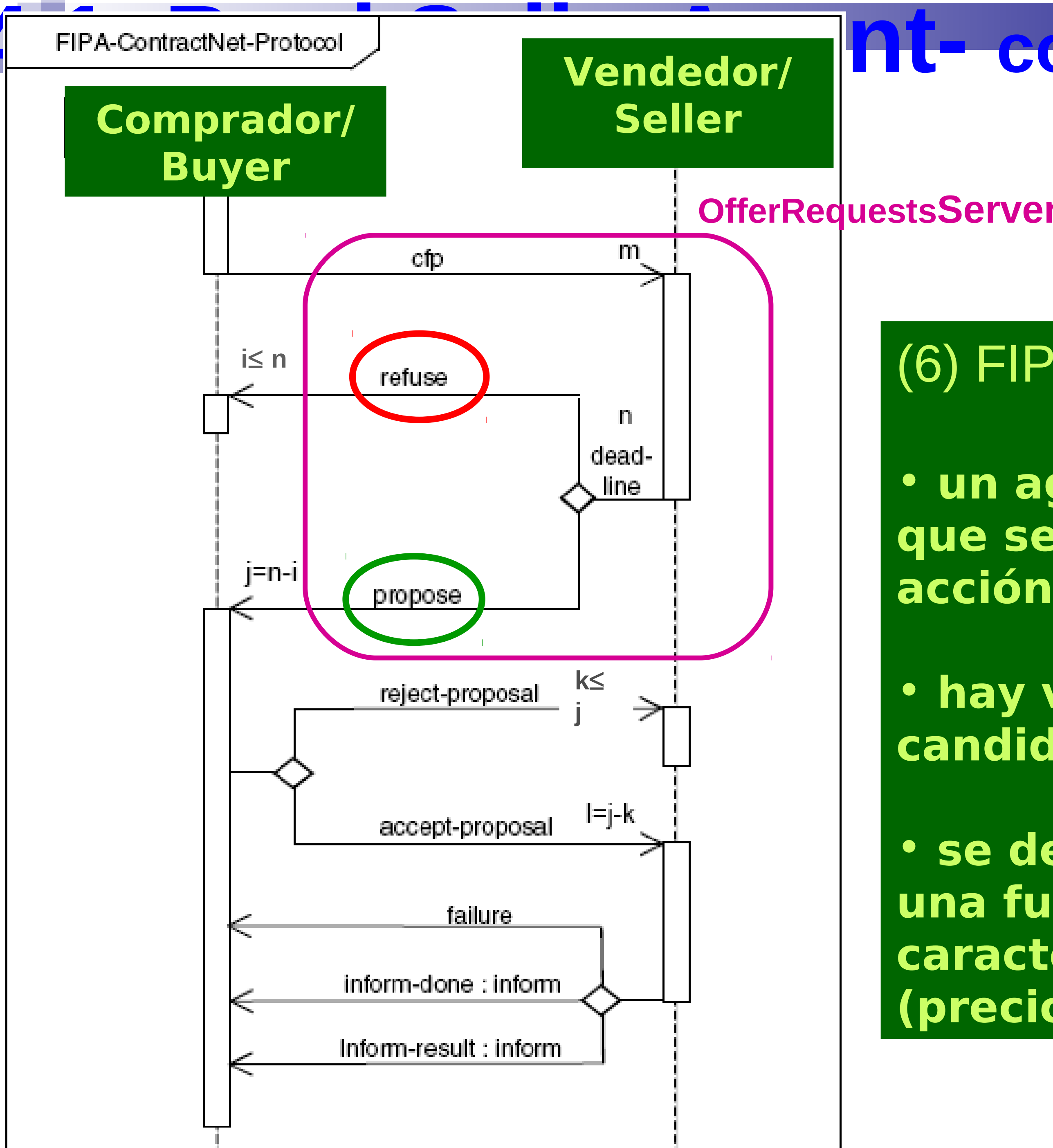
4.1. BookSellerAgent- Método de clase

Añade libro y precio a la librería

```
public void updateCatalogue(
                                    final String title, final int price) {

  addBehaviour(new OneShotBehaviour() {
    public void action() {
      catalogue.put(title, new Integer(price));
      System.out.println(title+" … = "+price);
    }
  } );
}
```

nt- comunicación



(6) FIPA-contract-net

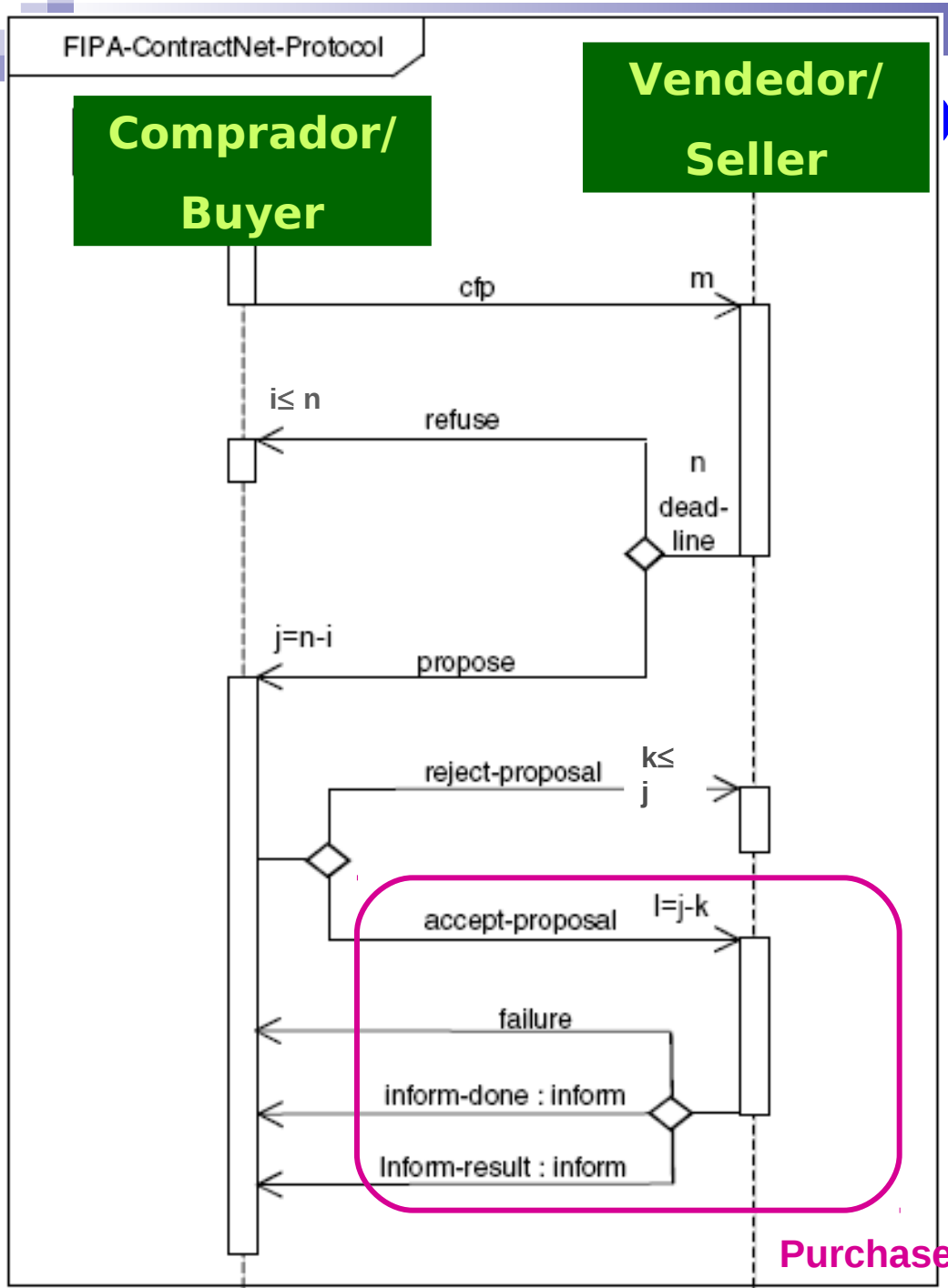
- **un agente desea que se realice una acción**
- **hay varios candidatos**
- **se desea minimizar una función que caracteriza la tarea (precio)**

4.1. BookSellerAgent

Behaviour OfferRequestsServer

```
public void action() {                    //CyclicBehaviour
  MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.CFP);
  ACLMessage msg = myAgent.receive(mt);
    if (msg != null) {                    // CFP Message received. Process it
      String title = msg.getContent();
      ACLMessage reply = msg.createReply();
      Integer price = (Integer) catalogue.get(title);
       if (price != null) {
         // The requested book is available for sale. Reply with the price
         reply.setPerformative(ACLMessage.PROPOSE);
         reply.setContent(String.valueOf(price.intValue()));      }
       else {      // The requested book is NOT available for sale.
         reply.setPerformative(ACLMessage.REFUSE);
         reply.setContent("not-available");          }
       myAgent.send(reply);
    }
      else {
           block(); …
```

ookSellerAgent



(6) FIPA-contract-net

- **un agente desea que se realice una acción**
- **hay varios candidatos**
- **se desea minimizar una función que caracteriza la tarea (precio)**

PurchaseOrdersServer

4.1. BookSellerAgent

PurchaseOrdersServer

```
public void action() {                    //CyclicBehaviour
    MessageTemplate mt = …(ACLMessage.ACCEPT_PROPOSAL);
          ACLMessage msg = myAgent.receive(mt);
      if (msg != null) {    // ACCEPT_PROPOSAL Message received.

       String title = msg.getContent();
       ACLMessage reply = msg.createReply();
       Integer price = (Integer) catalogue.remove(title);
       if (price != null) {
        reply.setPerformative(ACLMessage.INFORM);
        System.out.println(title+" … "+msg.getSender().getName());
       }
       else {  // The requested book has been sold to another buyer.
        reply.setPerformative(ACLMessage.FAILURE);
        reply.setContent("not-available");
       }
       myAgent.send(reply);
      }
       else {
            block(); …
```

4.2. BookBuyerAgent- setup()

```
// The title of the book to buy
  private String targetBookTitle;
  // The list of known seller agents
  private AID[] sellerAgents;

  // Put agent initializations here
     protected void setup() {
       // Printout a welcome message
       System.out.println("Hallo! …"+getAID().getName()+…);

       // Get the title of the book to buy as a start-up argument
       Object[] args = getArguments();
       if (args != null && args.length > 0) {  // Hay título
         targetBookTitle = (String) args[0];
         System.out.println("Target book is "+targetBookTitle);
```

4.2. BookBuyerAgent- setup()

```
// hay título
// Add a TickerBehaviour: a request to seller agents every minute
    addBehaviour(new TickerBehaviour(this, 60000) {
      protected void onTick() {
         System.out.println("Trying to buy "+targetBookTitle);
        // Update the list of seller agents
      DFAgentDescription template = new DFAgentDescription();
                ServiceDescription sd = new ServiceDescription();
                sd.setType("book-selling");
      template.addServices(sd);
         try {
            DFAgentDescription[] result =
                                                DFService.search(myAgent, template);
            System.out.println("Found the following seller agents:");
          sellerAgents = new AID[result. length];
          for (int i = 0; i < result.length; ++i) {
                sellerAgents[i] = result[i].getName();
                System.out.println(sellerAgents[i].getName());
          …
```

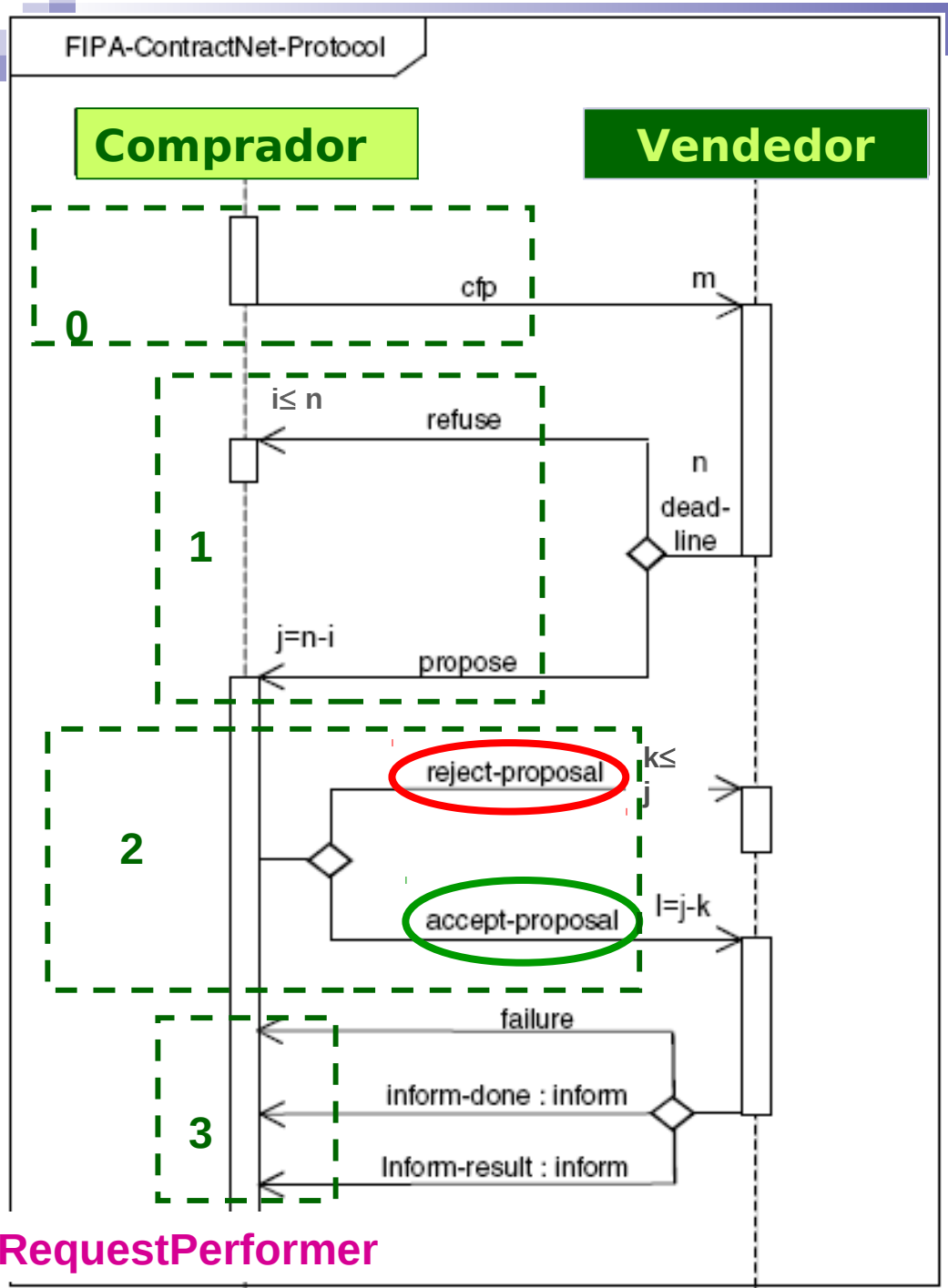
Busca vendedores en las páginas amarillas cada minuto

4.2. BookBuyerAgent- Setup

```
// Add a TickerBehaviour: a request to seller agents every minute
   addBehaviour(new TickerBehaviour(this, 60000) {
     protected void onTick() {
       …

       // Perform the request
       myAgent.addBehaviour(new RequestPerformer());
     }
   } );                          // Fin addBehaviour TickerBehabiour
  }
  else { // No hay título
   // Make the agent terminate
   System.out.println("No target book title specified");
   doDelete();
  }
}
```

okBuyerAgent



(6) FIPA-contract-net

- **un agente desea que se realice una acción**
- **hay varios candidatos**
- **se desea minimizar una función que caracteriza la tarea (precio)**

4.2. BookBuyerAgent

Behaviour RequestPerformer

```
// Behaviour {
    private AID bestSeller; // The agent who provides the best offer
    private int bestPrice;  // The best offered price
    private int repliesCnt = 0; // The counter of replies from seller agents
    private MessageTemplate mt; // The template to receive replies
    private int step = 0;

    public void action() {
      switch (step) {
      case 0:
        // Envía cfp a todos los vendedores
      case 1:
        // Recibe todos los proposals/refusals de los vendedores
      case 2:
        // Envía la orden de compra al mejor vendedor
      case 3:
        // Recibe contestación de la orden de compra
```

BookBuyerAgent
Behaviour RequestPerformer

```
case 0:
    // Send the cfp to all sellers
    ACLMessage cfp = new ACLMessage(ACLMessage.CFP);
    for (int i = 0; i < sellerAgents.length; ++i) {
      cfp.addReceiver(sellerAgents[i]);
     }
     cfp.setContent(targetBookTitle);
     cfp.setConversationId("book-trade");
     cfp.setReplyWith("cfp"+System.currentTimeMillis()); // Unique value
     myAgent.send(cfp);

     // Prepare the template to get proposals
     mt = MessageTemplate.and(MessageTemplate.
                                          MatchConversationId("book-trade"),
                        MessageTemplate.MatchInReplyTo(cfp.getReplyWith()));
        step = 1;
```

4.2. BookBuyerAgent

Behaviour RequestPerformer

```
case 1:
// Receive all proposals/refusals from seller agents
    ACLMessage reply = myAgent.receive(mt);
    if (reply != null) {
     // Reply received
      if (reply.getPerformative() == ACLMessage.PROPOSE) {
       // This is an offer
       int price = Integer.parseInt(reply.getContent());
       if (bestSeller == null || price < bestPrice) {
        // This is the best offer at present
         bestPrice = price;
         bestSeller = reply.getSender();
        }
      }
      repliesCnt++;
      if (repliesCnt >= sellerAgents.length) {
        // We received all replies
        step = 2;
      }
    }
```

4.2. BookBuyerAgent

Behaviour RequestPerformer

case 2:

```
// Send the purchase order to the seller that provided the best offer
        ACLMessage order = new ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
       order.addReceiver(bestSeller);
       order.setContent(targetBookTitle);
       order.setConversationId("book-trade");
       order.setReplyWith("order"+System.currentTimeMillis());
       myAgent.send(order);

        // Prepare the template to get the purchase order reply
        mt = MessageTemplate.and(MessageTemplate.
                                                    MatchConversationId("book-trade"),
                                 MessageTemplate.MatchInReplyTo(order.getReplyWith()));
          step = 3;
          break;
```

¿Qué pasa con los que NO son bestSeller?

4.2. BookBuyerAgent

Behaviour RequestPerformer

```
case 3:
// Receive the purchase order reply
      reply = myAgent.receive(mt);
      if (reply != null) {
          // Purchase order reply received
          if (reply.getPerformative() == ACLMessage.INFORM) {
              // Purchase successful. We can terminate
              System.out.println(targetBookTitle+" successfully purchased
                                                  from agent "+reply.getSender().getName());
              System.out.println("Price = "+bestPrice);
              myAgent.doDelete();
            }
            else {
              System.out.println("Attempt failed: requested book already sold.");
            }

            step = 4;
```