Travaux Pratiques

FI BDCC S4 : Systèmes Multi Agents et Intelligence Artificielle Distribuée

TP : Acheter un livre (Mohamed HAMMANE)

• Introduction :

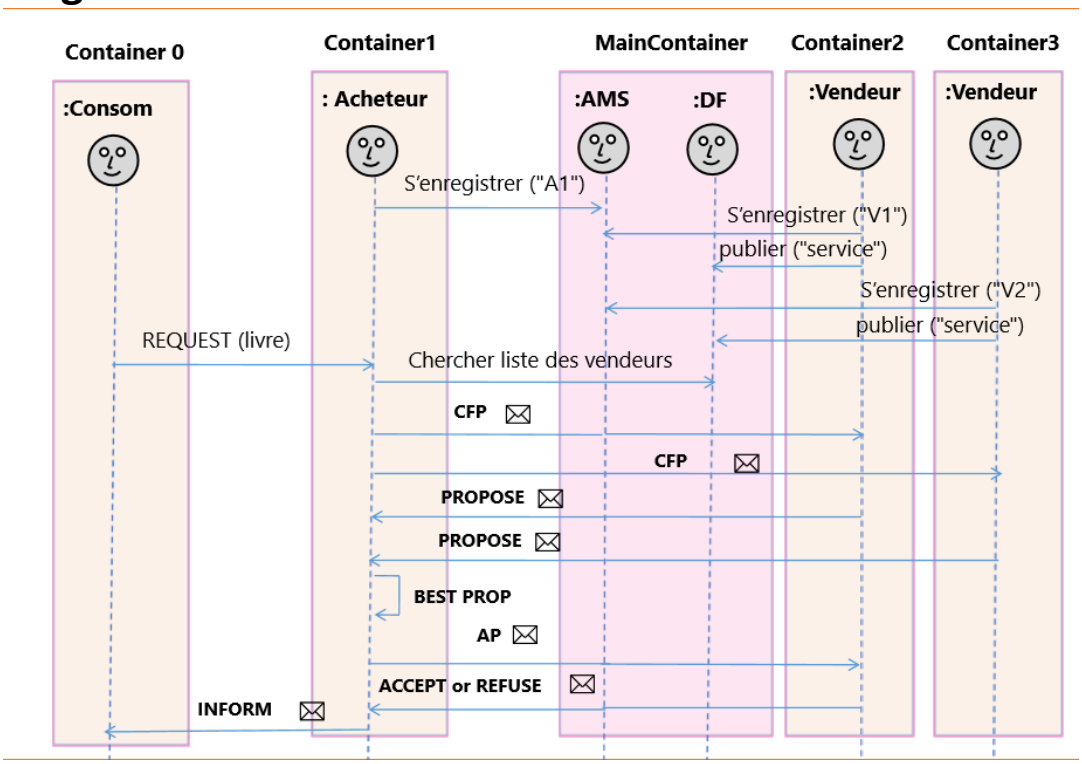
Un Agent est une entité : qui agit d'une façon autonome, pour atteindre les objectifs pour lesquels il a été conçu, peut communiquer avec d'autres agents, doté de capacités semblables aux êtres vivants un agent peut être un processus, un robot, un être humain, etc…

Un système multi-agent (SMA) est un système distribué : composé d'un ensemble d'agents distribués, situés dans un certain environnement et Interagissant selon certaines organisations. Un SMA permet de résoudre des problèmes complexes en exploitant l'intelligence collective des agents qui les compos.

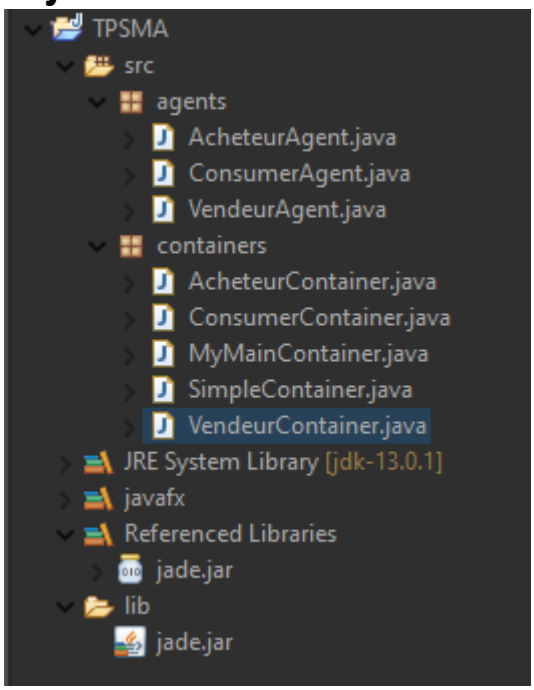
• Enoncé :

On souhaite créer une application multi-agents. Ce SMA contient des agents qui demandent l'achat d'un livre et d'autres agents qui vendent les livres, l'acheteur devrait choisir l'agent qui offre le meilleur prix à travers un processus de communication entre les agents.

• Diagramme d'interactions :

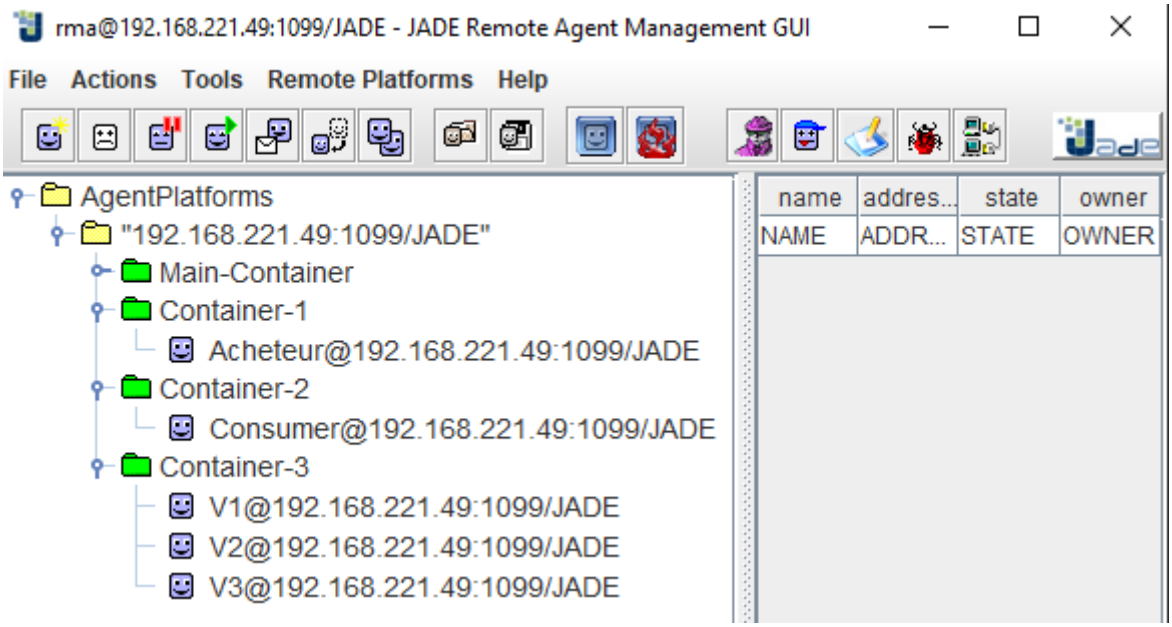


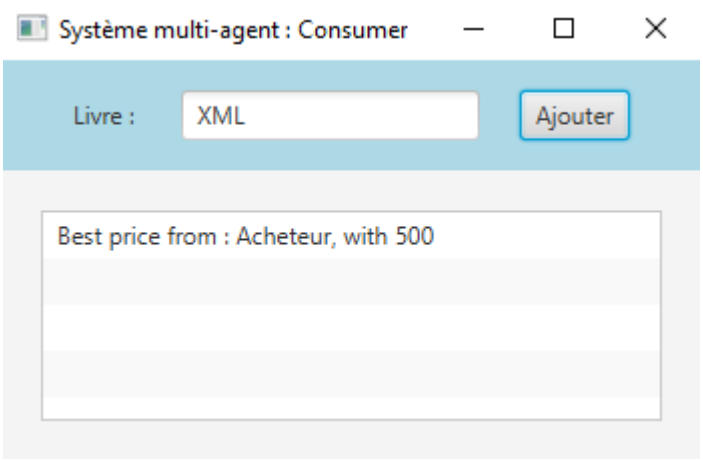
- **Structure de projet :**

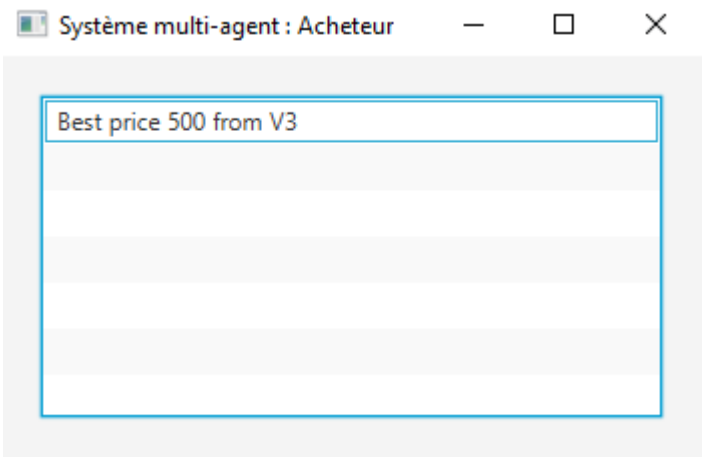


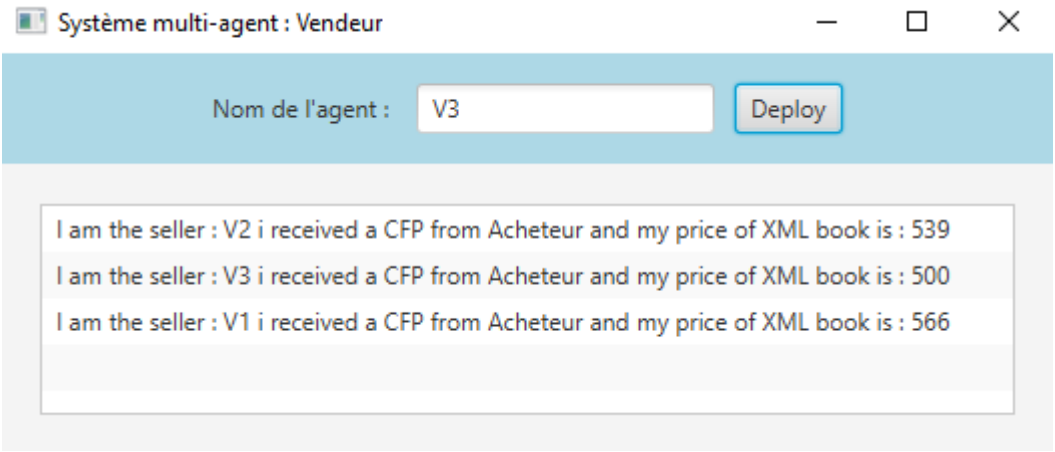
- **Exécution :**

On va déployer 3 agents vendeurs puis on va acheter un livre XML, chaque vendeur va afficher un message avec le prix qui propose pour le livre, l'acheteur va afficher un comme quoi le meilleur prix de quel agent et finalement le consommateur va afficher le prix du livre ainsi le nom de l'acheteur.









- **Code source :**
  - AcheteurAgent :

```
public class AcheteurAgent extends GuiAgent {
   public transient AcheteurContainer gui ;
   AID[] sellerAgents;
   @Override
   protected void setup() {
      if(getArguments().length > 0) {
         gui = (AcheteurContainer)getArguments()[0];
         gui.setAcheteurAgent(this);
      }
      ParallelBehaviour behaviour = new ParallelBehaviour();
      behaviour.addSubBehaviour(new TickerBehaviour(this, 5000) {
         protected void onTick() {
            DFAgentDescription template = new DFAgentDescription();
            ServiceDescription sd = new ServiceDescription();
            sd.setType("transaction");
            sd.setName("Vente livre");
            template.addServices(sd);
            try {
               DFAgentDescription[] result = DFService.search(myAgent, template);
               sellerAgents = new AID[result.length];
               for (int i = 0; i < result.length; ++i) sellerAgents[i] = result[i].getName();
            }
            catch (Exception fe) {
               fe.printStackTrace();
            }
```

```
        }
      }
      );
      behaviour.addSubBehaviour(new CyclicBehaviour() {
         private List<ACLMessage> listMsg = new ArrayList<ACLMessage>();
         @Override
         public void action() {
            MessageTemplate template=
                  MessageTemplate.or(
                     MessageTemplate.MatchPerformative(ACLMessage.REQUEST),
                     MessageTemplate.or(
                        MessageTemplate.MatchPerformative(ACLMessage.PROPOSE),
                        MessageTemplate.or(
                           MessageTemplate.MatchPerformative(ACLMessage.AGREE),
                           MessageTemplate.MatchPerformative(ACLMessage.REFUSE)
                        )
                     )
                  );
            ACLMessage aclMessage = receive(template);
            if(aclMessage != null) {
               switch (aclMessage.getPerformative()) {
                  case ACLMessage.REQUEST: {
                     ACLMessage msg = new ACLMessage(ACLMessage.CFP);
                     msg.setContent(aclMessage.getContent());
                     for(AID aid : sellerAgents) msg.addReceiver(aid);
                     send(msg);
                  } ;
                  break;
                  case ACLMessage.PROPOSE: {
                     listMsg.add(aclMessage);
                     if(listMsg.size() == sellerAgents.length) {
                        ACLMessage best = listMsg.get(0);
                        for(ACLMessage m : listMsg) {
                           if(Double.valueOf(m.getContent()) < Double.valueOf(best.getContent())) best
= m ;

                        }
                        ACLMessage replyAccept = best.createReply();
                        replyAccept.setContent(best.getContent());
                        replyAccept.setPerformative(ACLMessage.ACCEPT_PROPOSAL);
                        send(replyAccept);
                     }
                  };
                  break;
                  case ACLMessage.AGREE: {
                     ACLMessage msg = new ACLMessage(ACLMessage.CONFIRM);
                     msg.setContent(aclMessage.getContent());
                     msg.addReceiver(new AID("Consumer",AID.ISLOCALNAME));
                     send(msg);
                     gui.logMsg("Best price "+aclMessage.getContent()+" from
"+aclMessage.getSender().getName().split("@")[0]);
                  };
                  break;
                  case ACLMessage.REFUSE:  ;
                  break;
               }
            }
            else block();
         }
      });
      addBehaviour(behaviour);
```

```
    }
    @Override
    public void onGuiEvent(GuiEvent params) {}
}
```

- ConsumerAgent :

```
public class ConsumerAgent extends GuiAgent {
    private transient ConsumerContainer gui ;
    @Override
    protected void setup() {
        if(getArguments().length > 0) {
            gui = (ConsumerContainer)getArguments()[0];
            gui.setConsumerAgent(this);
        }
        ParallelBehaviour behaviour = new ParallelBehaviour();
        behaviour.addSubBehaviour(new CyclicBehaviour() {
            @Override
            public void action() {
                ACLMessage aclMessage = receive();
                if(aclMessage != null) {
                    if(aclMessage.getPerformative() == ACLMessage.CONFIRM) gui.logMsg("Best price from :
"+aclMessage.getSender().getName().split("@")[0]+", with "+aclMessage.getContent());
                }
                else block();
            }
        });
        addBehaviour(behaviour);
    }
    @Override
    public void onGuiEvent(GuiEvent params) {
        if(params.getType() == 1) {
            ACLMessage message = new ACLMessage(ACLMessage.REQUEST);
            message.setContent((String)params.getParameter(0));
            message.addReceiver(new AID("Acheteur",AID.ISLOCALNAME));
            send(message);
        }
    }

}
```

- VendeurAgent :

```
public class VendeurAgent extends GuiAgent{
    public transient VendeurContainer gui ;
    @Override
    protected void setup() {
        if(getArguments().length > 0) {
            gui = (VendeurContainer)getArguments()[0];
            gui.setVendeurAgent(this);
        }
        ParallelBehaviour behaviour = new ParallelBehaviour();
        behaviour.addSubBehaviour(new CyclicBehaviour() {
            @Override
            public void action() {
                ACLMessage aclMessage = receive();
                if(aclMessage != null) {
                    switch (aclMessage.getPerformative()) {
                        case ACLMessage.CFP: {
                            String book = aclMessage.getContent();
                            ACLMessage msg = aclMessage.createReply();
                            var price = String.valueOf(500+(int)(100.0 * Math.random()));
```

```
                    msg.setContent(price);
                    msg.setPerformative(ACLMessage.PROPOSE);
                    send(msg);
                    gui.logMsg("I am the seller : "+getName().split("@")[0]+" i received a CFP from
"+aclMessage.getSender().getName().split("@")[0]+" and my price of "+book+" book is : "+price);
                 }
                 break;
                 case ACLMessage.ACCEPT_PROPOSAL: {
                    ACLMessage msg = aclMessage.createReply();
                    msg.setContent(aclMessage.getContent());
                    msg.setPerformative(ACLMessage.AGREE);
                    send(msg);
                 }
                 break;
                 default : System.out.println("walo") ;
              }
           }
           else block();
        }
     });
     behaviour.addSubBehaviour(new OneShotBehaviour() {
        @Override
        public void action() {
           DFAgentDescription dfd = new DFAgentDescription();
           dfd.setName(getAID());
           ServiceDescription sd = new ServiceDescription();
           sd.setType("transaction");
           sd.setName("Vente livre");
           dfd.addServices(sd);
           try {
              DFService.register(myAgent, dfd);
           } catch (FIPAException e) {
              e.printStackTrace();
           }
        }
     });
     addBehaviour(behaviour);
  }

  @Override
  protected void takeDown() {
     try {
        DFService.deregister(this);
     } catch (FIPAException e) {
        e.printStackTrace();
     }
  }
  @Override
  public void onGuiEvent(GuiEvent params) {}
}
```

- AcheteurContainer :

```
public class AcheteurContainer extends Application {
   @SuppressWarnings("unused")
   private AcheteurAgent acheteurAgent ;

   ListView<String> list;
   public static void main(String[] args) {
      launch(args);
   }
```

```
    @Override
    public void start(Stage stage) throws Exception {
        startContainer();

        list = new ListView<String>();

        VBox centerBox = new VBox();
        centerBox.setPadding(new Insets(20));
        centerBox.getChildren().add(list);

        BorderPane root = new BorderPane();
        root.setCenter(centerBox);

        Scene scene = new Scene(root,350,200) ;
        stage.setTitle("Syst�me multi-agent : Acheteur");
        stage.setScene(scene);
        stage.show();
    }


    public void startContainer() throws Exception  {
        Runtime runtime = Runtime.instance();
        ProfileImpl profileImpl = new ProfileImpl();
        profileImpl.setParameter(ProfileImpl.MAIN_HOST, "localhost");
        AgentContainer agentContainer = runtime.createAgentContainer(profileImpl);
        AgentController agentController = agentContainer.createNewAgent("Acheteur",
"agents.AcheteurAgent", new Object[] {this});
        agentController.start();
    }

    public void setAcheteurAgent(AcheteurAgent acheteurAgent) {
        this.acheteurAgent = acheteurAgent;
    }

    public void logMsg(String msg) {
        Platform.runLater(() -> {
            list.getItems().add(msg);
        });
    }
}
```

- ConsumerContainer :

```
public class ConsumerContainer extends Application {

    protected ConsumerAgent consumerAgent;
    ListView<String> list;
    public static void main(String[] args) throws Exception {
        launch(args);
    }
    public void startContainer() throws Exception  {
        Runtime runtime = Runtime.instance();
        ProfileImpl profileImpl = new ProfileImpl();
        profileImpl.setParameter(ProfileImpl.MAIN_HOST, "localhost");
        AgentContainer agentContainer = runtime.createAgentContainer(profileImpl);
        AgentController agentController = agentContainer.createNewAgent("Consumer",
"agents.ConsumerAgent", new Object[] {this});
        agentController.start();
    }
```

```
    @Override
    public void start(Stage stage) throws Exception {
        startContainer();

        Label lblLvr = new Label("Livre : ");
        TextField livreField = new TextField();
        Button btnLvr = new Button("Ajouter");
        list = new ListView<String>();
        btnLvr.setOnAction(click -> {
            if(livreField.getText().trim() != "") {
                var event = new GuiEvent(this, 1);
                event.addParameter(livreField.getText());
                consumerAgent.onGuiEvent(event);
            }
        });

        VBox centerBox = new VBox();
        centerBox.setPadding(new Insets(20));
        centerBox.getChildren().add(list);

        HBox topBox = new HBox();
        topBox.setStyle("-fx-background-color:lightblue;");
        topBox.setAlignment(Pos.CENTER);
        topBox.setPadding(new Insets(15));
        topBox.setSpacing(20);
        topBox.getChildren().addAll(lblLvr,livreField,btnLvr);

        BorderPane root = new BorderPane();
        root.setTop(topBox);
        root.setCenter(centerBox);

        Scene scene = new Scene(root,350,200) ;
        stage.setTitle("Syst\uFFFDme multi-agent : Consumer");
        stage.setScene(scene);
        stage.show();
    }
    public void setConsumerAgent(ConsumerAgent consumerAgent) {
        this.consumerAgent = consumerAgent;
    }
    public void logMsg(String msg) {
        Platform.runLater(() -> {
            list.getItems().add(msg);
        });
    }

}
```

- MainContainer :

```
public class MyMainContainer {
    public static void main(String[] args) {
        try{
            Runtime runtime=Runtime.instance();
            Properties properties=new ExtendedProperties();
            properties.setProperty(Profile.GUI,"true");
            ProfileImpl profileImpl=new ProfileImpl(properties);
            AgentContainer mainContainer=runtime.createMainContainer(profileImpl);
            mainContainer.start();
        }
        catch(Exception e){
```

```
            e.printStackTrace();
        }
    }
}
```

- VendeurContainer :

```
public class VendeurContainer extends Application {
    private VendeurAgent vendeurAgent ;

    ListView<String> list;
    AgentContainer agentContainer;
    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage stage) throws Exception {
        startContainer();
        list = new ListView<String>();

        Label lblAgent = new Label("Nom de l'agent : ");
        TextField agentField = new TextField();
        Button btnAgent = new Button("Deploy");
        list = new ListView<String>();
        btnAgent.setOnAction(click -> {
            if(agentField.getText().trim() != "") {
                try {
                    AgentController agentController = agentContainer.createNewAgent(agentField.getText(),
"agents.VendeurAgent", new Object[] {this});
                    agentController.start();
                } catch (Exception e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        });

        VBox centerBox = new VBox();
        centerBox.setPadding(new Insets(20));
        centerBox.getChildren().add(list);

        HBox topBox = new HBox();
        topBox.setStyle("-fx-background-color:lightblue;");
        topBox.setAlignment(Pos.CENTER);
        topBox.setPadding(new Insets(15));
        topBox.setSpacing(10);
        topBox.getChildren().addAll(lblAgent,agentField,btnAgent);

        BorderPane root = new BorderPane();
        root.setCenter(centerBox);
        root.setTop(topBox);

        Scene scene = new Scene(root,400,200) ;
        stage.setTitle("Système multi-agent : Vendeur");
        stage.setScene(scene);
        stage.show();
    }

    public void startContainer() throws Exception  {
```

```
        Runtime runtime = Runtime.instance();
        ProfileImpl profileImpl = new ProfileImpl();
        profileImpl.setParameter(ProfileImpl.MAIN_HOST, "localhost");
        agentContainer = runtime.createAgentContainer(profileImpl);
    }

    public void setVendeurAgent(VendeurAgent vendeurAgent) {
        this.vendeurAgent = vendeurAgent;
    }

    public void logMsg(String msg) {
        Platform.runLater(() -> {
            list.getItems().add(msg);
        });
    }
}
```

- ## Conclusion :

On peut utiliser SMA dans plusieurs domaines (Simulation de phénomènes complexes, Résolution de problèmes, Conception de programmes …), mais est ce qu'on peut dire qu'un agent intelligent peut remplacer l'intelligence humaine.

Code source GitHub : https://github.com/Medomane/SMA.

Table de matière