References


[38] “George Rzevski, 08 New Directions in Searching for Artificial Intelligence.pdf”.
Appendix A

Essentials of Multi Agent technology

A.1 Introduction

This appendix consists of the features in multi-agent systems and also describes the difference between complex system and others.

<table>
<thead>
<tr>
<th>CLASSES/Features</th>
<th>RANDOM SYSTEMS</th>
<th>COMPLEX SYSTEMS</th>
<th>STABLE SYSTEMS</th>
<th>ALGORITHMIC SYSTEMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictability</td>
<td>Total uncertainty</td>
<td>Considerable uncertainty</td>
<td>No uncertainty</td>
<td>No uncertainty</td>
</tr>
<tr>
<td>Behavior</td>
<td>Random</td>
<td>Emergent</td>
<td>Planned</td>
<td>Deterministic</td>
</tr>
<tr>
<td>Norms of behavior</td>
<td>Total freedom of behavior</td>
<td>Some external guidance is essential</td>
<td>Governed by laws and regulations</td>
<td>Follows instructions</td>
</tr>
<tr>
<td>Degree of organization</td>
<td>None</td>
<td>Self-organization</td>
<td>Organized</td>
<td>Rigidly structured</td>
</tr>
<tr>
<td>Degree of control</td>
<td>None</td>
<td>Self-control by self-organization</td>
<td>Centralized control</td>
<td>No need for control</td>
</tr>
<tr>
<td>Irreversible changes</td>
<td>Random changes</td>
<td>Co-evolves with environment</td>
<td>Small temporary deviations possible</td>
<td>None</td>
</tr>
<tr>
<td>Operating point</td>
<td>None</td>
<td>Operates far from equilibrium</td>
<td>Operates at an equilibrium</td>
<td>Operates according to the specification</td>
</tr>
</tbody>
</table>

Table A.1: A Multi Agent System Features

Source: George Rzevski, a New Direction of Research into Artificial Intelligence
Appendix B

Design of MAS for SCM

B.1 Introduction

As said in design chapter, this appendix consist of various design diagrams to align with implementation stage.

Figure B.1: Class Diagram for SCM
Figure B.2: Sequence Diagram for Human Agent Interaction
Figure B.3: State Transition Diagram
Appendix C

Implementation of the MAS Solution for SCM

C.1 Introduction

During this appendix have included important code segments, figures and data have used in implementation stage.

Figure C.1: Login Screen of MASSCM

Figure C.2: Dashboard of MASSCM
**Figure C.3:** Ontology Update form

**Figure C.4:** Transport Schedule Data

Source: Link Natural Products (PVT) LTD
Figure C.5: Transport Schedule Data (Confirmed by Manufacturers)

<table>
<thead>
<tr>
<th>tr_auto_id</th>
<th>tr_aid</th>
<th>tr_start</th>
<th>tr_end</th>
<th>tr_time</th>
<th>tr_qty</th>
<th>read_status</th>
<th>manu_aid</th>
<th>tr_date_holiday</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T1</td>
<td>2015-04-01</td>
<td>2015-04-01</td>
<td>09:00:00</td>
<td>2000</td>
<td>1</td>
<td>M3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>T2</td>
<td>2015-04-01</td>
<td>2015-04-02</td>
<td>12:00:00</td>
<td>300</td>
<td>1</td>
<td>M5</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>T3</td>
<td>2015-04-02</td>
<td>2015-04-02</td>
<td>08:30:00</td>
<td>4000</td>
<td>1</td>
<td>M2</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>T1</td>
<td>2015-04-02</td>
<td>2015-04-02</td>
<td>06:30:00</td>
<td>200</td>
<td>1</td>
<td>M3</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>T3</td>
<td>2015-04-01</td>
<td>2015-04-01</td>
<td>14:30:00</td>
<td>250</td>
<td>1</td>
<td>M2</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>T2</td>
<td>2015-04-02</td>
<td>2015-04-02</td>
<td>23:00:00</td>
<td>500</td>
<td>1</td>
<td>M5</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>T3</td>
<td>2015-04-01</td>
<td>2015-04-01</td>
<td>12:00:00</td>
<td>700</td>
<td>1</td>
<td>M2</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>T1</td>
<td>2015-04-02</td>
<td>2015-04-08</td>
<td>06:30:00</td>
<td>500</td>
<td>1</td>
<td>M3</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>T3</td>
<td>2015-04-04</td>
<td>2015-04-07</td>
<td>12:00:00</td>
<td>300</td>
<td>1</td>
<td>M2</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>T2</td>
<td>2015-04-02</td>
<td>2015-04-02</td>
<td>06:30:00</td>
<td>2000</td>
<td>1</td>
<td>M5</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure C.6: Transport Schedule of Manufacturers Agents

<table>
<thead>
<tr>
<th>tr_manu_auto_id</th>
<th>tr_manu_aid</th>
<th>tr_manu_date</th>
<th>tr_manu_confirm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>M1</td>
<td>2015-04-01</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>M2</td>
<td>2015-04-06</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>M3</td>
<td>2015-04-02</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>M4</td>
<td>2015-04-02</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>M5</td>
<td>2015-04-01</td>
<td>0</td>
</tr>
</tbody>
</table>

```java
public class DataAccess {
    Statement stmt = null;
    // Select SQL query.
    public ResultSet SelectSql(String query) {
        ResultSet rs = null;
        Statement stmt = null;
        try {
            DBCon useDBCon = new DBCon(); // Creating instance for a DBCon class.
            Connection newCon = useDBCon.connect(); // Accessing variable which used in DBCon class.
            stmt = newCon.createStatement();
            rs = stmt.executeQuery(query);
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

Figure C.7: Code for MYSQL Data Base Connection
Figure C.8: Message Space Agent Code

```java
private ACLMessage msg;

public ACLMessage getMessage() {
    return msg;
}

public myReceiver(Agent a, int millis, MessageTemplate mt) {
    super(a);
    timeOut = millis;
    template = mt;
}

public void onStart() {
    wakeupTime = (timeOut < 0 ? Long.MAX_VALUE : System.currentTimeMillis() + timeOut);
}

public boolean done() {
    return finished;
}

public void action() {
    if (template == null) {
        msg = myAgent.receive();
    } else {
        msg = myAgent.receive(template);
    }
}
```

// T - Transport
myAgent.getContainerController().createNewAgent("T1","pki.TransportAgent",null).start();
myAgent.getContainerController().createNewAgent("T2","pki.TransportAgent",null).start();
myAgent.getContainerController().createNewAgent("T3","pki.TransportAgent",null).start();

// W - Warehouse
myAgent.getContainerController().createNewAgent("W1","pki.WarehouseAgent",null).start();
myAgent.getContainerController().createNewAgent("W2","pki.WarehouseAgent",null).start();
myAgent.getContainerController().createNewAgent("W3","pki.WarehouseAgent",null).start();

// H - Human Agent
myAgent.getContainerController().createNewAgent("H1","pki.HumanAgent",null).start();
```

Figure C.9: Agent Initialization Code
Figure C.10: Common Domain Ontology Access (XML Access)

```java
try {
    // Access XML content
    Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();
    myDef = (AgentConfiguration) jaxbUnmarshaller.unmarshal(ontoXmlFile);
}
```

Figure C.11: Sending Multicast Messages to Manufacturer Agents

```java
for (int i = 0; i < agents.length; i++) {
    msg.addReceiver(new AID("M" + i, AID.ISLOCALNAME), agents[i].getName());
    per.addSubBehaviour(new myReceiver(this, 2000, template) {
        public void handle(ACLMessage msg) {
            System.out.println("Got quote Rs:" + offer + " from " + msg.getSender().getLocalName());
            if (offer < bestPrice) {
                bestPrice = offer;
                bestOffer = msg;
            }
        }
    });
}
```

Figure C.12: Agent Performatives Code

```java
seq.addSubBehaviour(new myReceiver(this, 2000,
    MessageTemplate.and(
        MessageTemplate.MatchConversationId(msg.getConversationId()),
        MessageTemplate.or(
            MessageTemplate.MatchPerformative(ACLMessage.AGREE),
            MessageTemplate.MatchPerformative(ACLMessage.REJECT)))) {
    private static final long serialVersionUID = 1L;
```

88
Figure C.13: Ontology Update and Send Reply to Manufacturers

```java
seq.addSubBehaviour(new DelayBehaviour(this, rnd.nextInt(3000)) { 
    public void handleElapsedTimeout() { 
        if (bestOffer == null) { 
            System.out.println("Got no quotes"); 
        } else { 
            // Insert query .............. 
            
            String query = "INSERT INTO agents_manufacture" + "(mf_best_price, mf_aid, mf_conid) VALUES " + "+" + bestPrice + ",", + bestOffer.getSender().getLocalName() + 
            try { 
                ds.insertSql(query); 
            } catch (SQLException ex) { 
                ex.printStackTrace(); 
            } 
            
            System.out.println("\nBest Price Rs:" + bestPrice + " from " + bestOffer.getSender().getLocalName()); 
            ACLMessage reply = bestOffer.createReply(); 
            
            // Ontology should goes here ............... 
            
            if (bestPrice <= 50) { 
                reply.setPerformative(ACLMessage.REQUEST); 
                reply.setContent("" + rnd.nextInt(80)); 
                System.out.print("ORDER at Rs:" + reply.setContent() + "); 
                send(reply); 
            } 
        } 
    } 
});
```

Figure C.14: Message Template for to Receive Messages from Transport Agent

```java
addBehaviour(new MessageReceivingBehaviour()); 
// This code need to modify. ........................................

MessageTemplate mt1 = 
    MessageTemplate.and(
        MessageTemplate.MatchPerformative(ACLMessage.INFORM), 
        MessageTemplate.MatchSender( new AID( "T1", 
            AID.ISLOCALNAME)))); 

MessageTemplate mt2 = 
    MessageTemplate.and(
        MessageTemplate.MatchPerformative(ACLMessage.INFORM), 
        MessageTemplate.MatchSender( new AID( "T2", 
            AID.ISLOCALNAME)))); 

MessageTemplate mt3 = 
    MessageTemplate.and(
        MessageTemplate.MatchPerformative(ACLMessage.INFORM), 
        MessageTemplate.MatchSender( new AID( "T3", 
            AID.ISLOCALNAME)))); 
```
protected void onTick() {

    ACLMessage receiveFromT1 = receive(mt1);
    if (receiveFromT1 != null) {
        try {
            String maid = "";
            String traid = "";
            Date agDate1 = null;
            try {
                System.out.println(receiveFromT1.getContent() + receiveFromT1.getSender());
                String sql1 = "SELECT DISTINCT manu_aid, tr_aid, confirm_date FROM agents_delivery;
                rs = da.SelectSql(sql1);
                while (rs.next()) {
                    maid = rs.getString("manu_aid");
                    traid = rs.getString("tr_aid");
                    agDate1 = rs.getTimestamp("confirm_date");
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }

            System.out.println("Transport agent " + traid + " has agreed with manufacturer agent " + maid);
            // Arrange ware date and trans date are equals

            String consql = "SELECT ware_aid FROM agents_delivery_ware WHERE ware_aid = " + myid;
            rs = da.SelectSql(consql);
            boolean val1 = rs.next();
            if (val1 == false) {
                System.out.println("No confirmation among manufacturer, transport and warehouse agent.");
            }

            String genCID() {
                if (cidBase == null) {
                    cidBase = getLocalName() + hashCode()
                        + System.currentTimeMillis() % 10000 + "_";
                }
                return cidBase + (cidCnt++);
            }

            ACLMessage newMsg(int perf, String content, AID dest) {
                ACLMessage msg = newMsg(perf);
                if (dest != null) {
                    msg.addReceiver(dest);
                }
                msg.setContent(content);
                return msg;
            }

            //Figure C.15: Warehouse Agent Ontology Management

            //Figure C.16: Warehouse Agent Utility Methods

        }
    }
}

Figure C.17: Warehouse Agent Ontology Management

www.lib.mrt.ac.lk
Figure C.17: Transport Agent Different Behaviors

```java
@Override
protected void setup() { // Setup start
    super.setup(); // To change body of generated methods, choose Tools | Template
    addBehaviour(new TickerBehaviour(this, 20000));
    addBehaviour(new delayBehaviourOnTrans (this, 30000));

    SequentialBehaviour seq = new SequentialBehaviour();
    addBehaviour(seq);

    ParallelBehaviour par = new ParallelBehaviour(ParallelBehaviour.WHEN_ALL);
    seq.addSubBehaviour(par);

    // This behavior is created to send agreed dates from transport agent and now
    seq.addSubBehaviour(new DelayBehaviour(this, 90000));
}
```

---

Figure C.18: Human Agent Interference

```java
// Create data base connection to obtain holidays...........

String tr_date="";
String sdate="";
try {
    for (int i = 0; i < 3; i++) {
        String dateUpdate = "INSERT INTO agents_human_holiday (head,holi_dates) VALUES " + "(" + myAgent.getLocalName() + "," + holidayDateTime + ");";
        da.executeUpdate();
    }
}
```
Figure C.19: Retailor and Warehouse Agents Negotiation

Figure C.20: Customer and Retailer Negotiation