Chapter 5

Implementation
In the previous chapter, agents of Multi-agent System for Resource Allocation and Monitoring (MASRAM) are designed in detail along with their actions to be performed. All the three agents: Fund Seeker Agent (FSA), Fund Allocator and Monitor Agent (FAMA) and Facilitator Agent (FCA) are implemented using multi-agent development tool, Java Agent Development Framework (JADE) and Oracle database as backend support in web-based environment.

FCA interacts with users (fund seekers, fund allocators, experts and reviewers) and other agents (FSA and FAMA) operating in multi-agent environment. Graphical User Interface (GUI) is developed using Java Server Pages (JSP) to establish the interaction between users and FCA. FCA, after capturing request from users, passes the request either to FSA or to FAMA. FSA and FAMA perform tasks depending upon the request either to fetch data from database (Oracle) or to store data in database. FSA and FAMA use database to manage proposals, allocations and progress reports. Since FCA interacts with users and FSA & FAMA interact with database, therefore, the complete system is implemented using three-layered approach: Presentation, Agent and Database. FCA uses presentation layer to interact with users. FSA and FAMA use database layer to interact with database. Actions of FCA, FSA and FAMA are implemented using business objects developed in Java in agent layer. Figure 5.1 shows the three-layered software architecture of MASRAM in web-based environment.

5.1. Presentation Layer

Presentation layer consists of the Graphical User Interfaces provided to the users. These interfaces are implemented using JSP. These interfaces reside on the server side. Users call these interfaces from their client machines using any Internet Browser. The user wise interfaces implemented are listed in Table 5.1. The first main screen that appears to user is shown in figure 5.2.
Chapter 5. Implementation

Facilitator Agent
Agent Layer (JADE)

Figure 5.1: Web-based Architecture

5.1.1. Interfaces for Fund Seeker User

Fund seeker user is provided with three main interfaces having further links. Three interfaces are:

> Seeker Profile
> Finding Sources/Proposal Submission
> Projects due for Progress Updation

The links include viewing project details, status of the proposals submitted, updating the information on funds utilized and tasks completion. Following section provides the details of the interfaces provided.
5.1.1.1. Seeker Profile

This interface is used to create Login ID for the fund seeker. The system asks fund seeker user to provide values for particulars like Name, address, phones, email, profile, experience, password and re-type password as shown in figure 5.3.
On submitting the information, MASRAM validates the information given and then passes to FSA for storing and creating Login ID. Fund seeker user uses this Login ID to know the appropriate sources of funding, proposal submission, allocation status, uploading progress of the project and uploading utilization of funds. The same interface is used to create Login IDs for the fund allocators, experts and reviewers. User selects the type from drop down list.

5.1.1.2. Finding Sources/Proposal Submission

This interface helps in providing list of sources from where funds can be availed and submitting project proposals under particular fund category. It also shows the status of the proposals earlier submitted and list of the projects for which progress status can be updated. Figure 5.4 shows the functionality available in the interface for fund seeker.

Fund seeker can also view the complete proposal submitted by clicking on the title of the projects and can update information of tasks completed. Fund seeker can view status of funds utilized by clicking on the Project ID. On clicking the desired hyperlink, the input is handled at server side using another JSP script that interprets the input and calls FCA, a gateway to agent environment.

5.1.1.3. Projects due for Progress Updation

This link provides the lists of the projects due for submitting project progress as per schedule given by FAMA. Depending upon settings in fund category, the schedule is prepared by FAMA. The list of such projects as appears to user is shown in figure 5.5. On clicking particular project, FSU submits the progress. Fund seeker enters the actual starting and ending dates. The interface provides list of all the activities along with the expected date of completion.
Chapter 5. Implementation

Figure 5.3: Login Creation Screen

Figure 5.4: Interface for Fund Seeker User
5.1.2. Interfaces for Fund Allocator User

Once the fund allocator user logs in successfully, the main screen displays the fund category wise allocation status, list of the fund seekers, projects’ status for monitoring purpose, links for setting up parameters and reports as shown in figure 5.6. The interface sends the login information to JSP script resided at server and the script passes information to FAMA through FCA. FAMA provides the complete information to display through FCA.

5.1.2.1. Parameters

This part of the interface allows the fund allocator user to set the parameters. Parameter includes Nature of Projects, Designation Master, Project Objective Master, Outcome Master, Budget Item Master, Update Funding Category, Allocation Parameters and Funding Schedule. As an example, input screen for ‘Update Funding Category’ link is shown in figure 5.7. It shows the current fund categories along with minimum qualifying criteria to qualify for availing funds and availability of the funds. The interface provides link to change existing information as well as to enter new funding category.
5.1.2. Reports

Report section of the main screen helps the fund allocator user to view the reports related with fund seekers and allocation of funds. Table 5.2 shows the list of available reports to fund allocator.
Chapter 5. Implementation

<table>
<thead>
<tr>
<th>Report</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seeker Wise Funding</td>
<td>This report shows the seeker wise amount of funds given to different projects.</td>
</tr>
<tr>
<td>Category Wise Funding</td>
<td>This report shows the category wise allocation of funds to different fund seekers</td>
</tr>
<tr>
<td>Project Detail</td>
<td>This report shows the complete project proposal.</td>
</tr>
<tr>
<td>Progress Due Projects</td>
<td>This report shows the projects whose project status is due.</td>
</tr>
<tr>
<td>Allocation Report</td>
<td>This report shows the batch wise allocation of funds within fund category. There may be more than one batch under one fund category. One batch corresponds to one lot of allocation at a time.</td>
</tr>
<tr>
<td>Qualifying Criteria Report</td>
<td>This report shows the project wise status on qualifying criteria. The comparison of criteria set by allocator with the contents available in proposal is displayed.</td>
</tr>
<tr>
<td>Evaluation Criteria Report</td>
<td>This report shows the project wise criteria wise weight assigned by FAMA.</td>
</tr>
<tr>
<td>Feedback (Pre-allocation)</td>
<td>This report shows the feedback provided by experts after having seen the presentation given by fund seeker before fund allocation.</td>
</tr>
<tr>
<td>Weight Calculation Sheet</td>
<td>This sheet shows intermediate values given to proposal by FAMA.</td>
</tr>
<tr>
<td>Project Monitoring</td>
<td>This report shows the feedback provided by experts after having seen presentation on progress of projects and utilization of funds.</td>
</tr>
</tbody>
</table>

Table 5.2: Description of Reports Available to the Fund Allocator

5.1.2.3. Funds Allocation Status

This section of the main interface shows the allocations made by FAMA and status of the proposals.

5.1.2.4. List of Fund Seekers

This part of the main interface shows the list of fund seekers who have applied to the logged-in fund allocator.

5.1.2.5. Project Progress Status

This section of the main interface shows the status of progress of the proposals. The information like completed tasks, total tasks and budget utilized is available under this head.
5.1.3. Interfaces for Reviewers

Reviewers of the allocation are provided with third category of the interfaces. Steps to create logins remain same as those of fund seeker and fund allocator. The aim of this is to provide access to allocation made by FAMA and to get the confirmation of the allocation. The reviewers can change the allocation after viewing proposals, qualifying criteria and evaluated weights. Figure 5.8 shows the user interface available to reviewer. Once reviewer sanctions the grant, final allocation is done.

![Figure 5.8: Interface for Reviewer User](image)

5.1.4. Interfaces for Experts

Experts are provided with fourth category of interfaces. Steps to create logins remain same as those of fund seeker and fund allocator. The aim of this is to provide access to projects whose presentations are due and feedback is being awaited.

5.2. Agent Layer

Agent layer comprises three agents implemented using JADE. In addition to JSP scripts used to handle requests submitted by the users and information provided by agents for users. The details of components of Agent Layer are given below.

5.2.1. Scripts (Handling Input at Server Side) and Black Board Bean (Board)

A script is used to capture the input received from the user and send the same to FCA. Depending upon the task, agent passes information either to FSA or FAMA via board. The board (also Black Board Bean) is a program written in Java to set and get values of Java variables.objects. It contains all the variables.objects used in the
system meant for agent's communication with two public methods for each variable/object: `setXXXX` and `getXXXX`, where `XXXX` is variable/object name. FCA reads the value from the board and passes the captured data to the target agent. On receiving the message from target agent, it writes on board from where script sends message to the user. `JadeGateway.initO` method in the board sets the path of FCA. It initializes the FCA agent. After sending message, it waits for response. `setReceiverO` method sets the name of agent to whom message is to be sent. For demonstration, codes for handling input at server side and black board bean are shown in table 5.3 and 5.4 respectively.

```java
<!—import statements — >

JadeGateway.init("Gateway.agent.MyGateWayAgent",null);
BlackBoardBean board = new BlackBoardBean();
String lgn = request.getParameter("txtlogin_id");
String pwd = request.getParameter("txtlogin_pwd");
objectclasses loginobject = new objectclasses(lgn,pwd);
... 
  board.setReceiver("FSA");
  board.setMessage(loginobject);
...
JadeGateway.execute(board);
if (board.getMessage().equals("1"))
  response.sendRedirect("seekermain.jsp?nm="+
                  board.getUser_name());
else
  response.sendRedirect("errorpage.jsp?e=Invalid+
                    login_or_password");
}

Table 5.3: Data Capturing at Server Side

// imports
...
public String message = new String("";
public String receiver = new String("";
public String user_name = new String("";
public List type_names = new ArrayList();
public List sourcenames;
... 
public List getSource_names(){
    return sourcenames;
}
public void setReceiver(String str)
    {receiver = str; ...}

Table 5.4: Black Board Listing

5.2.2. Facilitator Agent

Facilitator Agent interprets the message sent by scripts and processes it. Depending upon the user, it calls either FSA or FAMA. The message received from the user through board is processed using appropriate ontology. The partial code for calling agent FSA or FAMA is given in table 5.5.
private void opr_login()
{
    Login lg = new Login();
    ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
    try {
        msg.addReceiver(new AID(board.getReceiver(), AID.ISLOCALNAME));
        msg.setLanguage(codec.getName());
        msg.setOntology(ontology.getName());
        objectclasses ol = board.getMessage(l);
        lg.setLogin_id(ol>Login_id.toString());
        lg.setLogin_pwd(ol>Login_pwd.toString());
        lg.setOper('1');
        action.setAction(new AID(board.getReceiver()));
        this.getContentManager().fillContent(msg,action);
        send(msg); }
    catch(Exception e) {
        board.errmsg = board.err_msg+ e;
        releaseCommand(board);________}________________________

Table 5.5: Partial Code to Call Agent

On receiving the reply from either FSA or FAMA, FCA interprets message
using appropriate ontology, prepares contents for user and writes on the board. FCA
performs 46 different tasks including requesting agents and receiving reply. Table 5.6
shows the partial code of one such task for preparing reply for users of MASRAM.

addBehaviour(new TickerBehaviour(this,1000) {
    public void onTick() { i++; 
        ACLMessage msg = receive();
        if ((msg!=null) && (board!=null)) { 
            try {
                board.err_code = board.opr;
                board.err_msg = "manish " + board.opr;
                switch(board.opr) 
                { 
                    case 1:
                        board.setUser_name(msg.getContentObject().toString());
                        if (msg.getContentObject().toString() != null)
                        board.setMessage("1");
                    else
                        board.setMessage("0");
                        releaseCommand(board);
                        break;
                }
            }
        }
    }
}

Table 5.6: Partial Code for Preparing Reply by FCA

5.2.3. Fund Seeker Agent

FSA receives message from FCA, interprets it and performs action accordingly
using different ontologies. It senses the environment on regular interval. On getting
message, it processes the same. All the responsibilities have been implemented here. The partial code to interpret ontology and operation is given in table 5.7.

![Protected void setup()
{getContentManager().registerLanguage(codec);
getContentManager().registerOntology(ontology);
addBehaviour(new TickerBehaviour(this,5000))
{ public void onTick()
{ msg = receive();
reporting rep = new reporting();
....
   if (msg!=null) { try {
   String content= "" + myAgent.getLocalName() +
    " sent: " + " MESSAGE";
    ce = getContentManager().extractContent(msg);
al = ((Action)ce).getAction();
   if (al instanceof Login) {
lg = (Login)al; op = Integer.parseInt(lg.getOper());
   } else if (al instanceof proposal) {
pl = (proposal)al; op = 31; }
   switch(op)
   { case 1:
   login_action(al);break;}
   default: ___________
....
Table 5.7: Partial Code to Process Message by FSA

While processing the request, it uses appropriate method of the class objectclasses.java, a business logic object. The partial code to call method of object of objectclass.java class is shown in table 5.8.

private void login_action(Concept al)
{ try {
   lg = (Login)al; System.out.println("Received message: ");
   replyPerson;
   reply = msg.createReply();
   String replyPerson;
   System.out.println("2");
   System.out.println("1");
   lg.setOper("1");
   reply.setPerformativ(ACLMessage.INFORM);
   objectclasses objlg = new objectclasses
   (lg.getLogin_id(),lg.getLogin_pwd());
   String ret = objlg.validateLogin();
   replyPerson = ret;
   reply.setObject(new Object());
   send(reply);
   send(reply);
} catch(Exception e) { send(reply); } }

Table 5.8: Partial Code of Processing Request by FSA

96
Chapter 5. Implementation

5.2.4. Fund Allocator and Monitor Agent

FAMA receives message from FCA, interprets it, processes it and sends the replies to FCA. It calls the methods of business logic object allocatorRoles.java. This is a class library for FAMA. Interpreting message, processing message, invoking method and sending reply are done in the same way as in case of FSA.

5.2.5. Defining Ontology

Ontology contains messages and Communication Acts (CA) such as Inform and Request. Agents interact online with each other using Ontology. Table 5.9 shows the partial list of Ontology used in MASRAM. In defining ontologies for MASRAM, simple data items like project_id and complex data items like list of projects are declared with their respective data types required to pass data/information from one agent to another.

```
Package Gateway.raontology;
import jade.content.onto.*;
import jade.content.schema.*;
import Gateway.raontology.*;
import jade.util.leap.List;
public class RAOntology extends Ontology {
    public static final String ONTOLOGY_NAME = "ra-ontology";
    public static final String LOGIN = "Login_id";
    public static final String PWD = "Login_pwd";

    private RAOntology() {
        super(ONTOiLOGY_NAME, BasicOntology.getInstance());
        try {
            add(new ConceptSchema(LOGIN), Login.class);
            add(new ConceptSchema(PROPOSAL_OBJECT), proposal.class);
            ConceptSchema cs = (ConceptSchema) getSchema(LOGIN);
            cs.add(LOGIN, (PrimitiveSchema) getSchema(BasicOntology.STRING));
            cs.add(PWD, (PrimitiveSchema) getSchema(BasicOntology.STRING), ObjectSchema.OPTIONAL);
            cs.add(OPER, (PrimitiveSchema) getSchema(BasicOntology.STRING), ObjectSchema.OPTIONAL);
            cs.add(PROPOSAL_ERRORS, (AggregateSchema) getSchema(BasicOntology.SEQUENCE), 0, ObjectSchema.UNLIMITED);
            cs.add(PROPOSAL_DATA, (AggregateSchema) getSchema(BasicOntology.SEQUENCE), 0, ObjectSchema.UNLIMITED);
            cs.add(REP_DATA, (AggregateSchema) getSchema(BasicOntology.SEQUENCE), 0, ObjectSchema.UNLIMITED);
            cs.add(FUNDS_UPDATE, (AggregateSchema) getSchema(BasicOntology.SEQUENCE), 0, ObjectSchema.UNLIMITED);
            AgentActionSchema as;
            add(as = new AgentActionSchema(PROPOSAL), proposal.class);
        }
    }
}
```

Table 5.9: Partial Code of Ontology
Chapter 5. Implementation

5.2.6. Defining Business Logic Objects

Two business logic objects are defined: objectclasses.java and allocatorRoles.java. These classes implement the functions to be performed by the fund seeker, fund allocator, expert and reviewer. Two agents: FSA and FAMA have direct access to these classes. These classes technically contain Java functions. One such function is shown in Table 5.10.

```java
public ResultSet getFundSources(Connection c) {
    ResultSet result;
    Statement Select;
    try{
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        c = DriverManager.getConnection(conn,ora_user,ora_pwd);
        Select = c.createStatement();
        sql = " select a.allocator_id,a.allocator_name, a.contact_person,";
        sql += " a.allocator_address,a.allocator_phone,";
        sql += " e.nature_of_project.nature_of_project_id pid ",;";
        sql += " e.nature_of_project.nature_of_project_desc pdesc,form_name from fund_category e, allocator_mas a";
        sql += " where a.allocator_id = e.allocator_id";
        result = Select.executeQuery(sql);
    }catch(Exception e) {
        System.err.println("Error at database site "+e);
        e.printStackTrace();
    return null;
    }
    return result;
}
```

Table 5.10: Partial code of Business Logic Objects

5.3. Database Layer

Database layer comprises Database Objects. Oracle 9i is used to support agents at backend. Object classes implemented in agent layer interact with the database layer.

5.3.1. Defining Database Schema Objects

Based on the requirement, 42 User Defined Types are defined in Oracle. These types were used to create tables. There are 70 tables including nested tables and 21 views meant to support agents. The complete schema listing is attached at Appendix ‘D’. Database objects also include one oracle package comprising procedures and functions. This package is called in java business logic classes. Table 5.11 shows the prototypes of procedures and functions defined in oracle package. As an instance, implementation of procedure evaluate_national_development is given in Table 5.12.
function allocate_fund_main(sss varchar2) return varchar2;
function match_criteria(all_id number, pid number) return number;
procedure process_insert(p_item varchar2, p_item_proposal varchar2, p_item_required varchar2, p_weight number, p_remarks varchar2, p_project_id number, p_status char, p_item_type char);
procedure update_status_criteria(pid number);
procedure evaluate_core_area(pid number, all_id number);
procedure evaluate_hr_development(pid number, all_id number);
procedure evaluate_national_development(pid number, all_id number);
procedure evaluate_eco_benefit(pid number, all_id number);
procedure evaluate_infra_availability(pid number, all_id number);
procedure evaluate_mang_cap(pid number, all_id number);
procedure evaluate_staff_exp(pid number, all_id number);
procedure evaluate_project_completion(pid number, all_id number, p_of_suc number);
procedure evaluate_risk_analysis(pid number, all_id number);
procedure allocate_funds(all_id number, cate_id number);
procedure allocate_fund_percentile(all_id number, cate number);
function final_allocation(startps number, endps number, per number, avail_amt number, all_id number, cate number) return number;
function countwords(flag number, pid number, alloc_id number, dmain_id number, ddetail_id number) return number;
procedure calculate_weight(all_id number, cate number);
function get_sum(dm number, dd number) return number;
procedure getqualwt(all_id number);
procedure check_feedback(pid number, flagstat in number);
procedure takeaction;
procedure generate_schedule(status in number, cate in number);

<table>
<thead>
<tr>
<th>Table 5.11: List of procedure and functions in Oracle Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>procedure evaluate_national_development(pid number, all_id number) is</td>
</tr>
<tr>
<td>opr number;</td>
</tr>
<tr>
<td>dmain number;</td>
</tr>
<tr>
<td>ddetail number;</td>
</tr>
<tr>
<td>cnt number;</td>
</tr>
<tr>
<td>tcnt number;</td>
</tr>
<tr>
<td>p number;</td>
</tr>
<tr>
<td>begin</td>
</tr>
<tr>
<td>cnt:=0;</td>
</tr>
<tr>
<td>opr:=2;</td>
</tr>
<tr>
<td>dmain:=1;</td>
</tr>
<tr>
<td>ddetail:=3;</td>
</tr>
<tr>
<td>cnt:=countwords(opr, pid, all_id, dmain, ddetail);</td>
</tr>
<tr>
<td>if cnt = 0 then</td>
</tr>
<tr>
<td>cnt:=1;</td>
</tr>
<tr>
<td>end if;</td>
</tr>
<tr>
<td>begin</td>
</tr>
<tr>
<td>select pid into p from project_proposals p where project_id = pid and</td>
</tr>
<tr>
<td>p.project_status.status_id = 6;</td>
</tr>
<tr>
<td>exception</td>
</tr>
<tr>
<td>when no_data_found then</td>
</tr>
<tr>
<td>return;</td>
</tr>
<tr>
<td>end;</td>
</tr>
<tr>
<td>insert into dm_evaluation(project_id, dmain_id, ddetail_id, evaluation)</td>
</tr>
<tr>
<td>values(pid, dmain, ddetail, cnt);</td>
</tr>
<tr>
<td>end evaluate_national_development;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5.12: Implementation of Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>procedure evaluate_national_development(pid number, all_id number) is</td>
</tr>
<tr>
<td>opr number;</td>
</tr>
<tr>
<td>dmain number;</td>
</tr>
<tr>
<td>ddetail number;</td>
</tr>
<tr>
<td>cnt number;</td>
</tr>
<tr>
<td>tcnt number;</td>
</tr>
<tr>
<td>p number;</td>
</tr>
<tr>
<td>begin</td>
</tr>
<tr>
<td>cnt:=0;</td>
</tr>
<tr>
<td>opr:=2;</td>
</tr>
<tr>
<td>dmain:=1;</td>
</tr>
<tr>
<td>ddetail:=3;</td>
</tr>
<tr>
<td>cnt:=countwords(opr, pid, all_id, dmain, ddetail);</td>
</tr>
<tr>
<td>if cnt = 0 then</td>
</tr>
<tr>
<td>cnt:=1;</td>
</tr>
<tr>
<td>end if;</td>
</tr>
<tr>
<td>begin</td>
</tr>
<tr>
<td>select pid into p from project_proposals p where project_id = pid and</td>
</tr>
<tr>
<td>p.project_status.status_id = 6;</td>
</tr>
<tr>
<td>exception</td>
</tr>
<tr>
<td>when no_data_found then</td>
</tr>
<tr>
<td>return;</td>
</tr>
<tr>
<td>end;</td>
</tr>
<tr>
<td>insert into dm_evaluation(project_id, dmain_id, ddetail_id, evaluation)</td>
</tr>
<tr>
<td>values(pid, dmain, ddetail, cnt);</td>
</tr>
<tr>
<td>end evaluate_national_development;</td>
</tr>
</tbody>
</table>
5.4. Web-based Implementation

As discussed above, all the layers of web-based systems are implemented using development tools and deployed for verification purpose.

5.4.1. Tools Used

To develop the web-based multi-agent system for resource allocation and monitoring, following tools have been used.

i. Java Agent Development Framework (JADE)

JADE, a development framework for multi-agent systems and fully implemented in Java Language is used to implement agents of MASRAM [81].

ii. Java 2 Enterprise Edition (J2EE)

Java 2 Enterprise Edition has the many features like Applet and Servlets to develop applications from simple desktop to complex enterprise level integrated web-based with database connectivity. In implementing the agents, its main features like JDBC (Java Database Connectivity), classes, beans and JSP are used to develop MASRAM. JSP is used to provide interfaces to the users, classes are used for performing tasks/actions and beans are used to communicate information from one agent to other.

iii. Oracle 9i

Oracle 9i, an Object Relational Database Management System (ORDBMS) is used at the backend to support agents.

iv. Web Service Integration Gateway (WSIG)

WSIG provides support for invocation of JADE agent services from web service clients [82]. This is used to develop browser-based interfaces for the users of MASRAM.

v. Tomcat Apache

A web server used to host JSP programs [83]. It processes JSP programs and gives output in HTML format which is sent to the client. It is used to execute dynamic pages.
The system developed using above tools can be implemented in any platform like Windows or Linux. In our research, Windows 2003 Server edition is used to run the application. Details of the above mentioned tools have been given in Appendix ‘E’.

5.4.2. Deploying and Operating Agents of the System

All the three agents are deployed on the server side. Two servers are proposed to deploy multi-agent based MASRAM: Database server and Agent server. Agent Server hosts all the three agents. While Database Server hosts Oracle Database. Users interact from client machines through Graphical User Interface (GUI) as shown in figure 5.9. Internet connectivity is required to connect client and server. For testing purpose, Local Area Network (LAN) is used. The users of MASRAM can access server through different client machines or through the same machine. Single physical machine can also work virtually as two machines: Client and Server to verify the system. The implemented system runs successfully.

Figure 5.9: Agent Deployment