Chapter 6

Implementation of Curriculum Design System

6.1 Introduction

Agents are considered one of the most important software disciplines that on the one hand many improve on current methods for conceptualizing, designing and implementing software systems, and on the other hand may be the solution in implementing complex real world problems in information systems. When considering on agent architectures those are the fundamental mechanisms underling the autonomous component that support effective behavior in real world, dynamic and open environments. In this implementation it is manly focus on a generic multi agent based system for designing curriculums.

The previous chapter presented the design of the system including the top level architecture with the main modules of the system. This chapter describes implementation of each modules and technologies that use to implement each module in the previous chapter.

6.2 Creating the curriculum design system

Curriculum design system mainly divided in to two sections which are user interaction module and Multi Agent System module. User interaction module consists of graphical user interface to interact with the user and MAS module takes the decisions about the curriculum. These modules were described broadly in the previous chapter.

Since this system is based on multi agent approach for the curriculum design, mainly this has been implemented in JADE environment. There are few agents which are involving to perform different kind of tasks which are integrated to achieve the goal of the system. Also ontology has been used to provide required knowledge for the agents.

6.3 Implementation of User Interaction Module

The main role of the UIM is present the graphical user interface to the user for interact with MAS module to create and review curriculums. Therefore this module has been implemented to as it can input required details about curriculum and the view the final output. Here user can enter data such as module code, module title, pre-requisites, number of credits, lab hours, lecture hours, aim and objective of a particular module, learning outcome for that module and etc.

Java swing technology has been used to create this module. This module is mainly connected with the request agent in the MAS module. The main class of UIM module is called AgentGui which extend JFrame and implements the ActionListener interface. Normally a GUI has already a built-in mechamism of handling event which is implemented via the *actionPerformed*() method of every component that registered with *ActionListener* object. Therefore UIM has been implements the *ActionListener* interface and then register all the interactive component of the UIM with this *ActionListener* via the method *addActionListener()*. An ActionEvent is generated whenever call to the AgentGui() is made. This event is generated by the source component, that invoke the actionPerformed() method. According to the code given in the above method, UIM response by processing the event. This module post the event to be handle by the request agent using the postGuiEvent(). The following code shows how to post the GUI event by AgentGui frame.

```
GuiEvent ge=new GuiEvent(this, 0);
ge.addparameter(“parameter”);
reqAgent.postGuiEvent();
```

Following coding segment shows the basic structure of the AgentGui frame class which is connected with the Request agent called reqAgent.

```
import jade.gui.GuiAgent;
import jade.gui.GuiEvent;
import ontology.*;


Public class AgentGui extends JFrame implements ActionListener,
CurriculumVocabulary{

private ReqAgent myAgent;

public AgentGui(ReqAgent a){
    myAgent=a;

     setTitle("Request Agent - " + myAgent.getLocalName());
//remaining frame generation code will goes here
}
....
public void actionPerformed(ActionEvent ae) {
....
               ge.addParameter(new String(sub));
               ge.addParameter(new Float(cred));
....
               myAgent.postGuiEvent(ge);
}
```

6.4 MAS Module

As describe in previous chapter this is the core module in this system. When user input the data through the UIM, Request Agent start the sending messages through the message space to the resource agents. According to this system design, java classes for *Request Agent* and *Resource Agents* have been created on the JADE platform.

6.4.1 Implementation of Request Agent to interact with user interaction module

Request Agent is the main agent who interacts with the user. Therefore we have used the special kind of abstract class in JADE called GuiAgent that extends the Agent class to create the Request Agent. This class has two specific methods: postGuiEvent() and onGuiEvent(). These two classes provide facilities to interact with the UIM and the Request Agent. To post an event to the Request Agent, the UIM module simply creates a GuiEvent object, add the required parameters and passes it in argument to the method postGuiEvent() as described in section 6.3.Then the posted event will be receive to the Request Agent through the onGuievent().when the agent receives an event posted by the agent, it test the type of event by calling the method getType() on the GuiEvent object and process the event by calling appropriate handler When the agent extends GuiAgent it automatically added the specific behavior called GuiHandlerBehaviour. This handles the incoming events from the UIM and dispatches them to the appropriate handlers following exactly same mechanism as in UIM. Next sections will describe the implantation of common feature of Agents.

6.4.2 Implementation of Agent classes.

Creating a JADE agent is as simple as defining a class that extends the jade.core. Agent class and implementing the setup() method as exemplified in the code below. The skeleton of a typical *Resource Agent* class is shown below. JADE *Agents* are defined as subclasses of the predefined class Agent and their initial code must be placed in a method called setup. Agent actions are normally specified through Behaviour classes. More exactly, the actions are described in the "action" method of these Behaviours. The setup method only serves to create instances of these behaviours and linking them to the Agent object.

```
    import jade.core.Agent;
    import jade.core.behaviours.*;

    public class CreditAgent extends Agent
    {
        protected void setup()
        {
            addBehaviour( new myBehaviour( this ) );
        }

        class myBehaviour extends SimpleBehaviour
        {
            public myBehaviour(Agent a) {
                super(a);
            }

            public void action()
            {
               //...this is where the real programming
goes
            }

            private boolean finished = false;

            public boolean done() {
                return finished;
            }
```

6.4.3 Agent Communication

Communication among agent has been done by using jade.lang.acl.ACLMessage class. ACL is stand for Agent Communication Language. There are several attributes in an ACL message. Those are performative(message type), address , content, conversation Id, Language, ontology and so on .There types of messages are INFORM, REQUEST, REPLY, CONFIRM, PROPOSE and so on. First three types of messages are most commonly used in this system. Also to keep the common understanding of the structure of information among agents it has been used the ontology which is describes later in this chapter. Following code shows the Credit Agent inform available credit details to the Request Agent.

```
ACLMessage reply = request.createReply();
reply.setPerformative(ACLMessage.INFORM);
Result result = new Result((Action)content, obj);
getContentManager().fillContent(reply, result);
send(reply);
```

6.4.4 Add behaviors to the agents

The actual mission that agent has to perform is typically carried out within behaviors. Behaviors represent a task that an agent can carryout and is implemented as an object of a class that extends jade.core.behaviours.Behaviour. To make an agent execute the task implemented by behavior object, behavior has been added to the agents by means of addBehaviour() method of agent class. Behaviour can be added at any time when the agent startup or from within other behaviours. Each class extending behavior must implement two abstract methods called action() and done(). The action() method defined the operations to be perform when the behavior is in execution. The done() method returns a Boolean value to indicate whether or not a behavior has completed and is to be removed from the pool of behaviours an agent is executing. Flow chart of agent excecution will show in Figure 6.1. There is various kinds of behaviours such as OneShotBehaviour, CyclicBehaviour, TickerBehaviour, SimpleBehaviour, CompositeBehaviour and so on. Appropriate behaviours has been used for each agent.

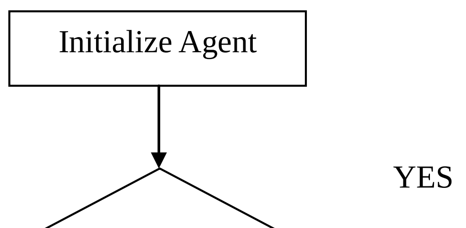


Figure 6.1: Agent Execution flow chart with behaviours

6.5 Implementation of Curriculum Ontology

According to the system design given in previous chapter, ontology consists of three parts. Ontology has been developed based on Protégé Ontology development tool. Even though we created ontology using protégé, JADE does not provide direct access to protégé ontology as a shared knowledge for the agents. Therefore curriculum ontology has been converted into JADE using Ontology Bean generator. Before covert this to JADE, ontology has been modified as supportable for three interfaces called Concept, Agent Action and Predicate based on the content reference model.

The actions which are suppose to do by agents have been included in the AgentAction subclass and agent identifications are included in the AID subclass in the Concept class. Remaining classes are included as subclass of concept class. Also relations were defines in the predicate class. Finally the Curriculum Ontology has been converted into JADE using Ontology Bean generator.

6.5.1 Defining an application specific ontology

An application specific ontology describes the elements that can be used as content of agent messages. The curriculum ontology is composed of two parts, curriculum vocabulary that describe the terminology of concepts used by agents in their space of communication, and the taxonomy of the relationships between these concepts, and that describe their semantic and structure. This ontology has been implemented to the agents by extending the class ontology in JADE and by adding a set of element schemas describing the structure of concepts, actions and predicates that are allow composing the content of the agent messages. Now we will look at how agents do the operation using ontology.
Using the ontology bean generator it has been created CurriculumVocabulary interface, CurriculumOntology class and all the classes mentioned in the ontology (see the Appendix B). In the CurriculumVocabulary interface defines the vocabulary for the agent communication. For instance, below code fragment shows common vocabulary for the concept of credit balancing.

```
public interface CurriculumVocabulary
```

```
    {

...
        public static final String CREDIT_BALANCE = "creditBalance";
        public static final String CREDIT_BALANCE_CURRICULUM =
"curriculum";
        public static final String CREDIT_BALANCE_LEVEL = "level";
        public static final String CREDIT_BALANCE_SUBCODE =
"subcode";
        public static final String CREDIT_BALANCE_LECTURE =
"lecture";
        public static final String CREDIT_BALANCE_LAB = "lab";
        public static final String CREDIT_BALANCE_CREDIT = "credit";

...
    }
```

In the CurriculumOntology it has been defines the schemas for the particular object. The lines of codes that specify the schema of concept *CreditBalance* is given below.

```
public class CurriculumOntology extends Ontology implements
CurriculumVocabulary {

// ----------> The name identifying this ontology

// ----------> The singleton instance of this ontology

// ----------> Method to access the singleton ontology object

// Private constructor
private CurriculumOntology() {

super(ONTOLOGY_NAME, BasicOntology.getInstance());

try {

// ------- Add Concepts

// ------- Add AgentActions

// Agent Action - CreditBalance
 AgentActionSchema as;

add( as = new AgentActionSchema(CREDIT_BALANCE),
CreditBalance.class);
as.add(CREDIT_BALANCE_CURRICULUM, (PrimitiveSchema) getSchema
(BasicOntology.STRING), ObjectSchema.MANDATORY);
as.add(CREDIT_BALANCE_LEVEL, (PrimitiveSchema)
getSchema(BasicOntology.STRING), ObjectSchema.MANDATORY);
as.add(CREDIT_BALANCE_SUBCODE, (PrimitiveSchema)
getSchema(BasicOntology.STRING), ObjectSchema.MANDATORY);
as.add(CREDIT_BALANCE_LECTURE, (PrimitiveSchema)
getSchema(BasicOntology.FLOAT), ObjectSchema.MANDATORY);
as.add(CREDIT_BALANCE_LAB, (PrimitiveSchema)
getSchema(BasicOntology.FLOAT), ObjectSchema.MANDATORY);
as.add(CREDIT_BALANCE_CREDIT, (PrimitiveSchema)
getSchema(BasicOntology.FLOAT), ObjectSchema.MANDATORY);

...
}
```

After creating ontology classes for the system it should set the content of a message using ontology. For that first ontology has been registered with agent's content manager, the curriculum ontology and the codec language that is use to coding and decoding the content of message. The way codec and ontology has been registered shown below. The entire agents have been registered with same language and same ontology to share the common knowledge.

```
...
Private Codec codec=new SLCodec();
Private Ontology ontology=CurriculumOntology.getInstance();
//Register Language and Ontology
getContentManager().registerLanguage(codec);
getContentManager().registerOntology(ontology);
...
```

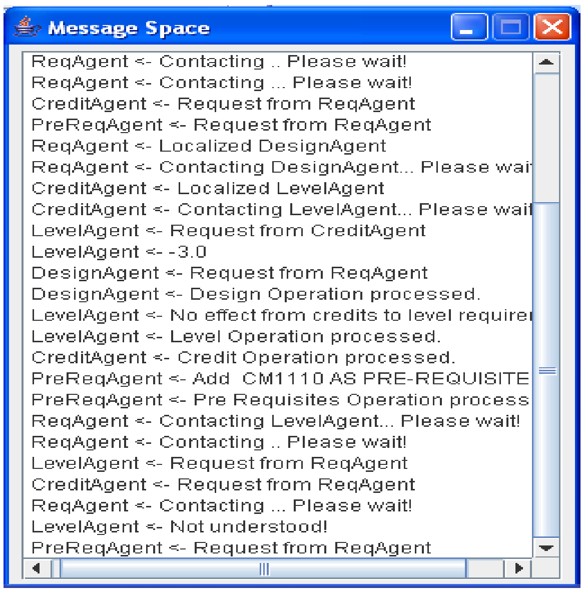
When composing the message, first set the attributes of java objects, and then specify within the message instance, the language and ontology. It has been obtain ContentManager object by calling the method getContentManager() of the Agent class. Then use the fillContent() method to pass the two arguments message and content.

6.5.2 Defining knowledge for the curriculum ontology

Agents need some knowledge to do their specific task. For this purpose it has been created a Database and some XML files to keep the required knowledge about curriculums. This XML file contains classes, attributes, proprieties and their instances. Also different rules related to the design of curriculum have been included in that file. XML parser has been used to read the knowledgebase. And Database consists of all the data related to deferent curriculums.

6.6 Massage Space

In JADE message space is not visible to the users. But the communication among agent should be visible to the user to show the agent negotiation to the user. Therefore it has been developed as separate window as shown in Figure 6.2.

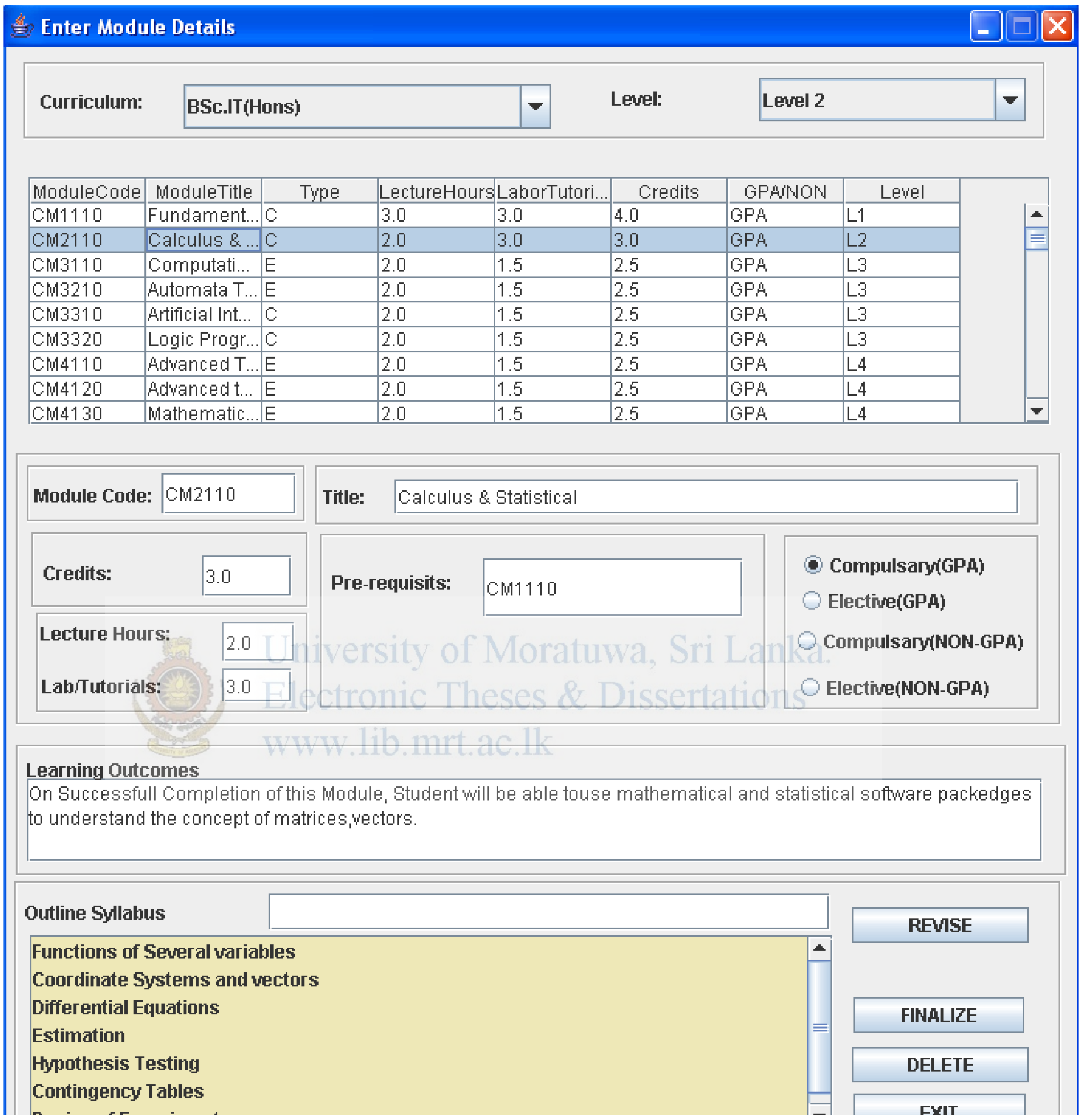


```
ReqAgent <- Contacting .. Please wait!
ReqAgent <- Contacting ... Please wait!
CreditAgent <- Request from ReqAgent
PreReqAgent <- Request from ReqAgent
ReqAgent <- Localized DesignAgent
ReqAgent <- Contacting DesignAgent... Please wai
CreditAgent <- Localized LevelAgent
CreditAgent <- Contacting LevelAgent... Please wai
LevelAgent <- Request from CreditAgent
LevelAgent <- -3.0
DesignAgent <- Request from ReqAgent
DesignAgent <- Design Operation processed.
LevelAgent <- No effect from credits to level require
LevelAgent <- Level Operation processed.
CreditAgent <- Credit Operation processed.
PreReqAgent <- Add  CM1110 AS PRE-REQUISITE
PreReqAgent <- Pre Requisites Operation process
ReqAgent <- Contacting LevelAgent... Please wait!
ReqAgent <- Contacting .. Please wait!
LevelAgent <- Request from ReqAgent
CreditAgent <- Request from ReqAgent
ReqAgent <- Contacting ... Please wait!
LevelAgent <- Not understood!
PreReqAgent <- Request from ReqAgent
```

Figure 6.2: The Message Space

6.7 Processing a module in a curriculum

This system is designed and implemented to initialize and start the ReqAgent with GUI as first step. The main GUI in this system is shown below in Figure 6.3. After displaying the user interface select the curriculum from the list given in the combo box. Then it displays the entire modules in that particular curriculum or summary of it as user required. Then user can select existing module or enter a new module to the system. When click on “REVISE” button request agent send message to other through the message space. Then resource agents will start on their works. Credit Agent checks the credit details with respect to lecture hours, lab hours and Level Agents details. Then display the changes suggestions in message space. Level agent checks the module code and the pre-requisites with respect to the level. Prerequisite agent check whether it needs to add pre-requisite for that module and check whether there is an equivalent modules in different curriculum. Module Agent mainly checks the learning outcome of the module and check whether changes are needed. Design agent finalizes the formatting of the design. All agents will display their specific task and communicate with other agent using the message space. Finally request agent the knowledgebase with finalized data and display the final result in the user interface. Message space is also making as visible to the user.



Curriculum: BSc.IT(Hons)
Level: Level 2

ModuleCode	ModuleTitle	Type	LectureHours	LaborTutori...	Credits	GPA/NON	Level
CM1110	Fundament...	C	3.0	3.0	4.0	GPA	L1
CM2110	Calculus & ...	C	2.0	3.0	3.0	GPA	L2
CM3110	Computati...	E	2.0	1.5	2.5	GPA	L3
CM3210	Automata T...	E	2.0	1.5	2.5	GPA	L3
CM3310	Artificial Int...	C	2.0	1.5	2.5	GPA	L3
CM3320	Logic Progr...	C	2.0	1.5	2.5	GPA	L3
CM4110	Advanced T...	E	2.0	1.5	2.5	GPA	L4
CM4120	Advanced t...	E	2.0	1.5	2.5	GPA	L4
CM4130	Mathematic...	E	2.0	1.5	2.5	GPA	L4

Module Code: CM2110
Title: Calculus & Statistical
Credits: 3.0
Pre-requisits: CM1110
Lecture Hours: 2.0
Lab/Tutorials: 3.0

[✓ Compulsary(GPA)] Elective(GPA) Compulsary(NON-GPA) Elective(NON-GPA)

Learning Outcomes
On Successfull Completion of this Module, Student will be able touse mathematical and statistical software packedges to understand the concept of matrices,vectors.

Outline Syllabus
Functions of Several variables
Coordinate Systems and vectors
Differential Equations
Estimation
Hypothesis Testing
Contingency Tables

Figure 6.3: Main GUI in Curriculum Design System

6.8 Summary

This system has been implemented by using JADE environment Protégé and XML. This section describes the implementation of UIM and the MAS module. There are some agents in MAS module to perform different kind of tasks which are integrated to achieve the goal of the system. Those agents are created in JADE environment. Agents are communicating with each other and they have behaviours which are stands for the role of them. Protégé has been used to develop content ontology for the agents and XML has been used for represent the knowledgebase other remarks details. Next challenge is to evaluate the system.