

UNIVERSITY OF
CALGARY

A Project Report for SENG 609.22

**Agent Based Software Engineering
Course Instructor: Dr. Behrouz H. Far**

**MULTI-AGENT SYSTEM FOR TUTORING AND LEARNING
(MASTL)**

December 2002

DOCUMENT NOTES

1. This is the design document for the Multi-Agent System for Tutoring and Learning System (MASTL). The document is written by following the industry style for software development projects.
2. The publications listed in the section 8 are referenced throughout this document and will not be mentioned for a specific section.
3. Since this is a course project with very limited time constrain, the document only shows designs for major components.

TABLE OF CONTENTS

1. INTRODUCTION.....	4
2. SYSTEM SPECIFICATIONS.....	4
2.1 ASSUMPTIONS.....	4
3. SYSTEM ARCHITECTURE	5
4. DETAIL DESIGN.....	6
4.1. PROFESSOR AGENT	6
4.1.1 <i>User Interface design</i>	7
4.1.2 <i>Use cases</i>	7
4.1.3 <i>Internal architecture for the Professor Agent</i>	9
4.2. TA AGENT	10
4.2.1 <i>User Interface design</i>	10
4.2.2 <i>Use cases</i>	10
4.2.3 <i>Internal architecture for TA Agent</i>	11
4.3. OFFICE AGENT	11
4.3.1 <i>Use cases</i>	11
4.3.2 <i>Internal Architecture of Office Agent</i>	13
4.4. LIBRARY AGENT	13
4.4.1 <i>Use cases</i>	13
4.4.2 <i>Internal architecture of Library Agent</i>	15
4.5. STUDENT AGENT.....	15
4.5.1 <i>Use cases</i>	15
4.5.2 <i>Internal architecture of Student Agent</i>	17
4.6. FACILITATOR AGENT	17
4.6.1 <i>Use case</i>	17
4.6.2 <i>Internal architecture of Facilitator Agent</i>	19
5 DATA SPECIFICATION	19
6. COMMUNICATION PROTOCOL.....	23
6.1. XML OVER HTTP	23
6.2. INTER-AGENT MESSAGING.....	24
7 FUTURE WORKS AND CONCLUSIONS.....	26
8 REFERENCES.....	27

1. Introduction

The Multi-Agent System for Tutoring and Learning (MASTL) is a multi-agent system designed to provide tutoring and learning in an academic milieu. Essentially, the system provides the following services: on-line lectures to students, student registration for courses, e-tutorials, e-library services and students' comprehension evaluation. This system deals with many students wishing to use the above services as well as e-professors with various backgrounds wishing to teach the students. Professors may sometimes use the help of e-teaching assistants (TA) to teach courses.

The system should allow students to register online, select courses to take, withdraw from courses, add courses, assign professors to students based on their needs, assign TAs to work with professors and allow students to use e-library if they are users.

2. System Specifications

Based on the high level requirements described in the last section, the following is a list of major functionality that MASTL will implement:

- MASTL shall allow students to register online for courses provided they meet the stated pre-requisites and conditions.
- MASTL shall be updated every four weeks to accommodate new knowledge
- MASTL is user friendly and presents learning material in textual and graphical forms
- MASTL shall allow student to use e-library services
- MAST shall allow students to ask questions to e-professors and e-TAs.
- MASTL shall evaluate students' comprehension level in order to choose a curriculum for them.

2.1 Assumptions

There are few assumptions based on the business requirements for the system design and implementation:

- The system will be running on networked and distributed environment: i.e. the system will be designed as multi-tier web-based application as shown in Figure 1.
- User interface will be implemented using web presentation technologies such as HTML, Java Script, XML, ...etc.
- New knowledge can be added to the different knowledge bases .i.e to student, professor, library, office, etc. Therefore agent will implement learning, knowledge-ability features.
- Java and J2EE technologies will be used for implementation.

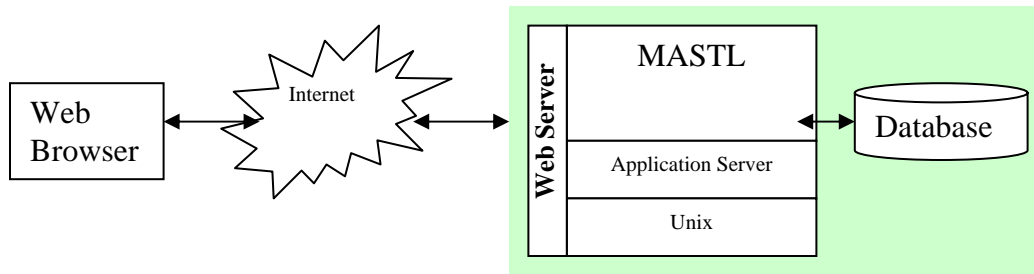


Figure 1. MASTL running environment

3. System Architecture

As shown in figure 2, the MASTL system is composed of six agents:

- Facilitator agent,
- Student agent,
- Library agent,
- Office agent,
- TA agent, and
- Professor agent

All these agents are linked up in a networked environment. Individual agents make requests through the facilitator to express their needs and intentions. The facilitator then uses its knowledge base to find the most appropriate agent that can serve the request. For the student agent, the facilitator determines the most suitable agent to deal with student and links up the student with that agent. In this way, while the student is conversing with that agent, the facilitator can continue with other activities.

After coordination done by the Facilitator, the Professor Agent

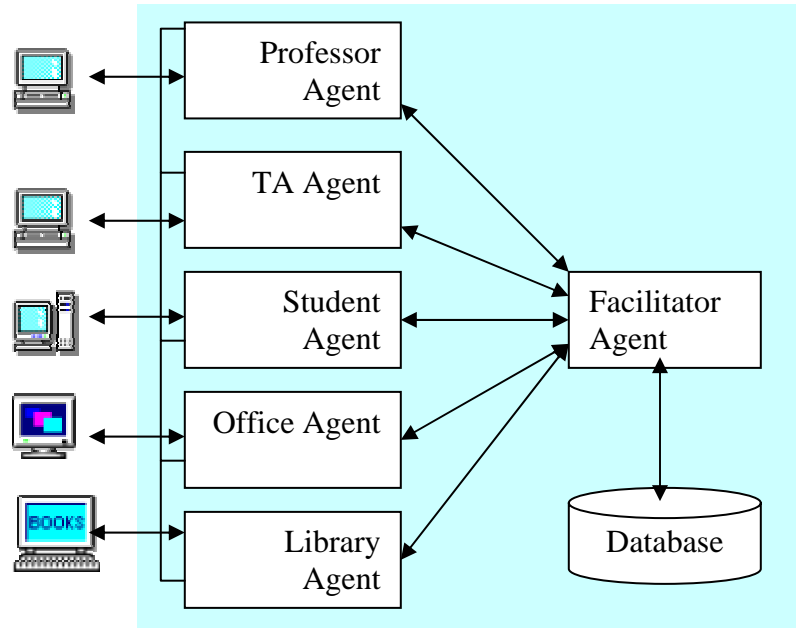


Figure 2. The overall architecture of MASTL System

4. Detail design

This section contains more detailed descriptions about each agent's design.

4.1. Professor Agent

The Professor Agent in the MASTL system plays the major roles of offering courses and online tutoring. In order to provide related services, the agent has to implement basic requirements as depicted in the use cases of section 4.1.2. In addition, the agent has to communicate with other Agents through the coordination of the Facilitator Agent. Therefore, there is a proxy component (such as a Class) within the Professor Agent (*same or similar component exist in each and every agent in the MASTL system*) that implements knowledge-level communication functionality and interacts with the Facilitator Agent. The proxy component can both send requests to the other agent and respond to requests from other agents.

The agent is implemented using the MVC (*Model-View-Control*) pattern [Figure 3]. The MVC design pattern is one of the fundamental ways to architect a program that involves both user interfacing, a "view", and internal data processing, a "model". The MVC pattern hinges on a clean separation of objects into one of three categories:

- **models** for maintaining data,

- **views** for displaying all or a portion of the data, and
- **controllers** for handling events that affect the model or view(s).

The MVC provides a technique with which to completely decouple the view and model, enabling easy interchange of either. That is, the way a model presents itself to the user can be changed without modifying the view itself or the specific model used to process the user inputs can be changed without affecting the user.

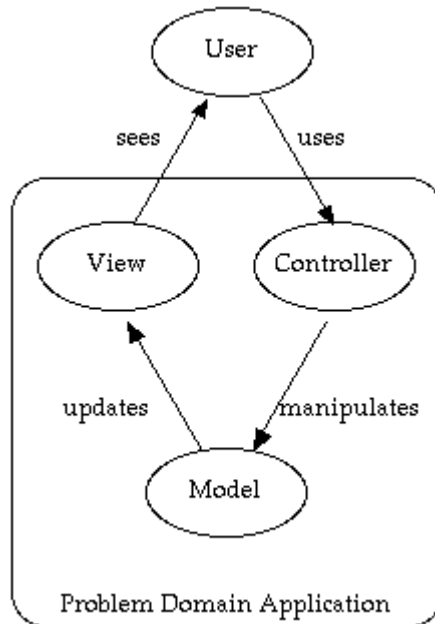


Figure 3. MVC pattern

4.1.1 User Interface design

A web front end will be implemented in HTML. The rendering component of the agent will render both static and dynamic contents through web presentation techniques such as Java Servlet, JSP, Java Script etc. Since we will use HTTP as communication protocol between the Professor Agent and Web Browser, state information has to be reserved whenever it is necessary.

4.1.2 Use cases

The following is a list of use cases that depicts some major use case scenarios. This is not meant to be an exhaustive list.

- Create Course outline
- Design Assignment (communicate with TA through Facilitator)
- Design Lab (communicate with TA through Facilitator)
- Design Quiz and exam

- Create Slides
- Student evaluation: marking scheme (communicate with Registrar through Facilitator)
- Scheduling: room and time allocations (communicate with office through Facilitator)
- Offer courses

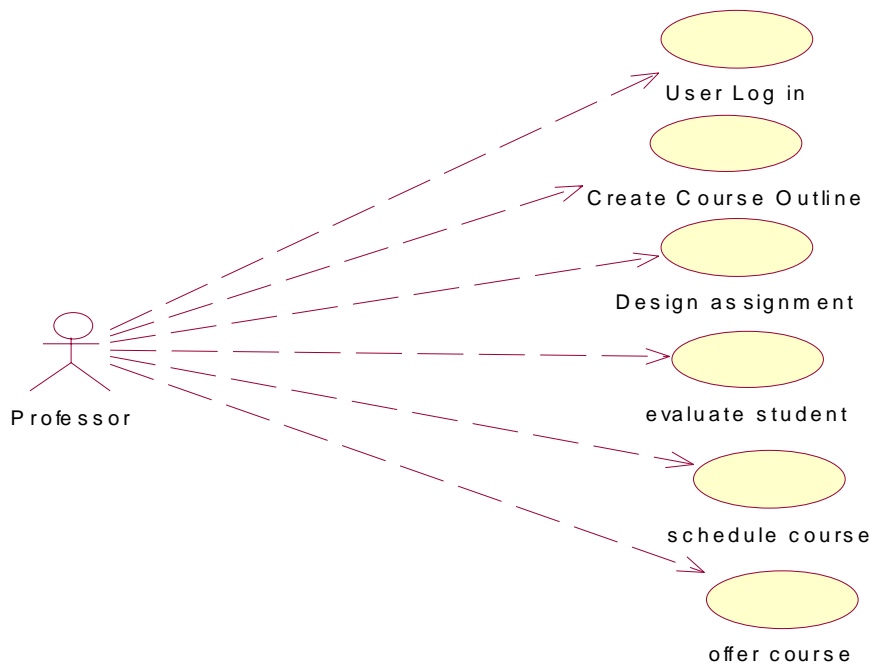


Figure 4. Use cases (partial) for Professor Agent

The following are examples of detailed descriptions for the use cases.

Create Course outline

Brief Description	A professor creates course outline online.
Pre-Condition	A user has already logged in.
Post-condition	N/A
Process steps	
1	A user has to log in.
2	Select course outline option from users' main menu
3	The course outline can be either uploaded from an existing file or create online instantly.
4	Save the outline to the system.

Student Evaluation

Brief Description	A professor evaluates students comprehension.
Pre-Condition	A user has already logged in.
Post-condition	N/A
Process steps	
1	A user has to log in.
2	Select student evaluation option from users' main menu
3	Select an aspect such as: assignments, quiz, mid-term exam, final exam
4	Assign marks to each student based on performance
5	Repeat 3,4 to mark all the aspects (assignments, quiz, mid-term exam, final exam)

4.1.3 Internal architecture for the Professor Agent

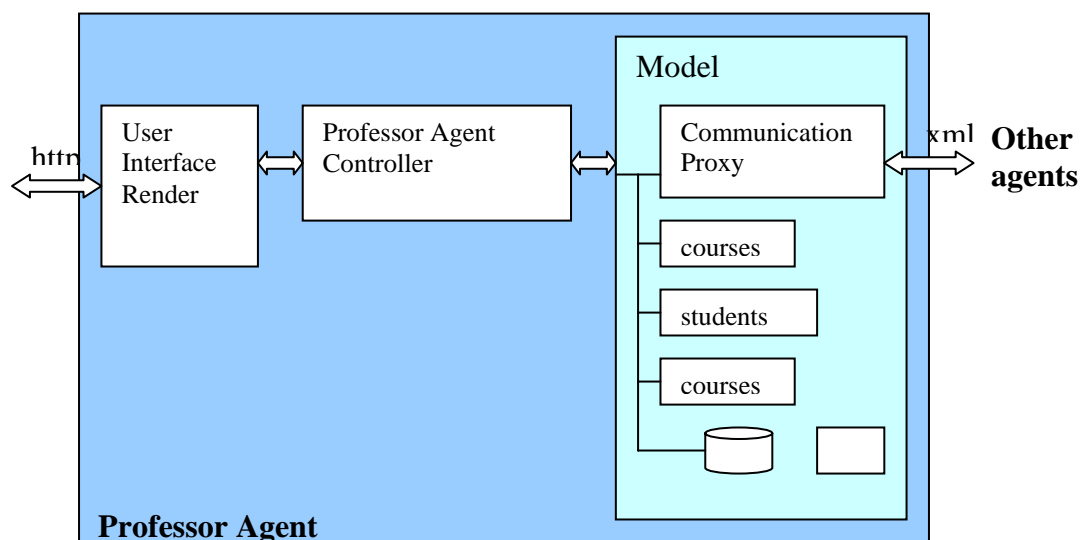


Figure 5, Professor Agent Architecture

4.2. TA Agent

The TA Agent implements requirements of teaching assistant roles. It provides tutoring functions such as lab instructions, marking assignments, question and answer. Although its functionality is quite different from the Professor Agent's, from technical perspective they can be implemented by using the same architecture such as the MVC pattern.

4.2.1 User Interface design

The user interface screens are designed in HTML. The user interface render component is responsible for both static and dynamic THML rendering.

4.2.2 Use cases

Here is a list of some major use case scenarios for TA Agent:

- Create Lab Instruction: coordinate with both Professor Agent and Student Agent
- Mark Assignment: coordinate with both Professor Agent and Student Agent
- Submit assignment marks to Professor Agent
- Online question and answer

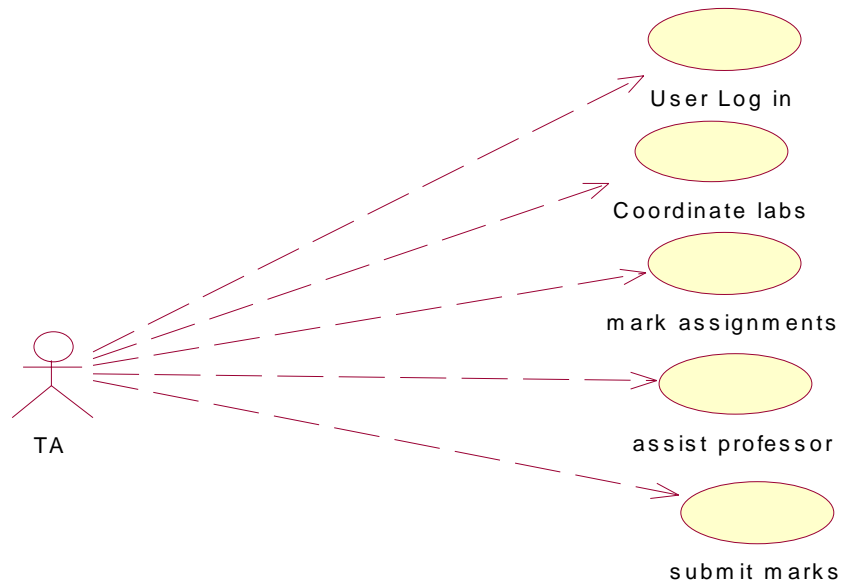


Figure 6. TA Use Case definitions

4.2.3 Internal architecture for TA Agent

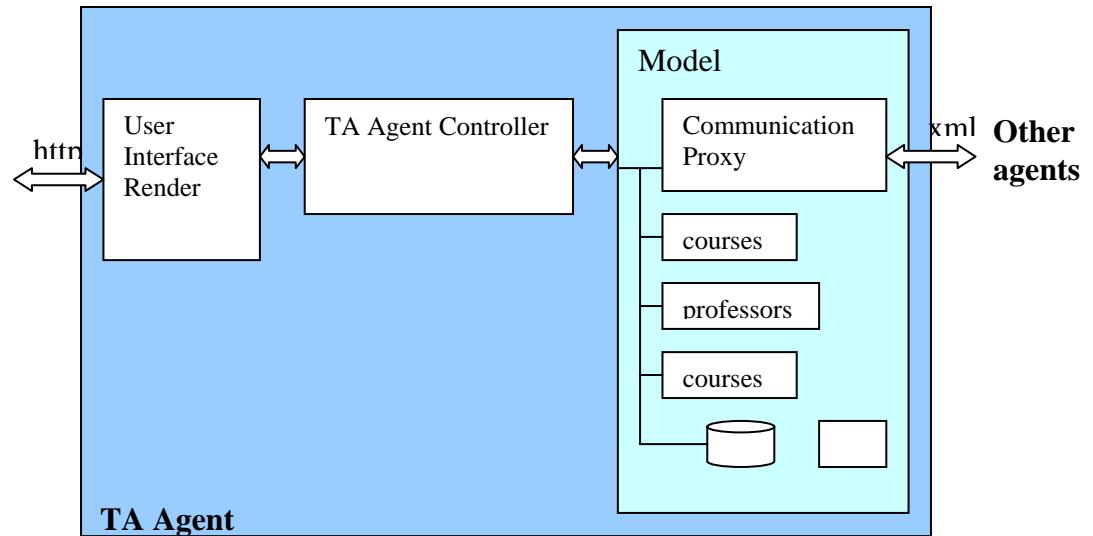


Figure 7. TA Agent internal architecture

4.3 Office Agent

The office Agent is responsible for carrying out all administrative work in the student tutoring and learning system. It receives requests from the facilitator agent from student agent or any other agent. It then responds to the request accordingly. Periodically, it informs the students of their academic standing as well as financial situation. The Office agent registers students on the various courses with the help of the course information system that is internal to it. Though the prime concern of the system is to offer tutoring and learning to students, the office agent may occasionally have to carry out other administrative duties such as adding more e-professors to systems so as to update its current standards.

4.3.1 Use cases

- The office Agent does all administrative work in the student tutoring and learning system.
- The Office agent deals with student registration, keeps a record of all staff i.e. Professors, TAs, students information, course information and financial records.
- The office agent periodically informs students of their academic standings. e.g. academic progress, fees situation

- The office agent is also responsible management of course information
- The Office agent responds to all agents seeking help in administrative matters.

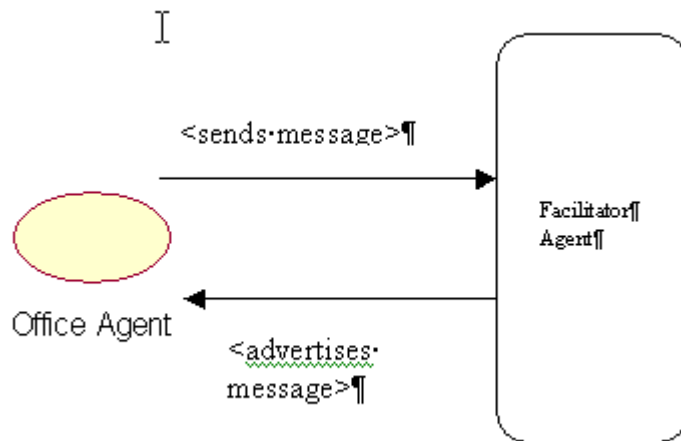


Figure 8. A sample use case for Office Agent

Use Case definitions

Brief Description	Office Agent makes a request through Facilitator to communicate with other agent.
Pre-Condition	An agent is available to communicate with office agent.
Post-condition	Office Agent successfully communicates with target agent
Process steps	
1	Office Agent makes request
2	Facilitator checks request information and agent information to find an appropriate agent for request
3	Facilitator advertises request to target agent
4	Target agent indicates willingness to communicate with office agent.
5	Office agent communicates directly with target agent

4.3.2 Internal Architecture of Office Agent

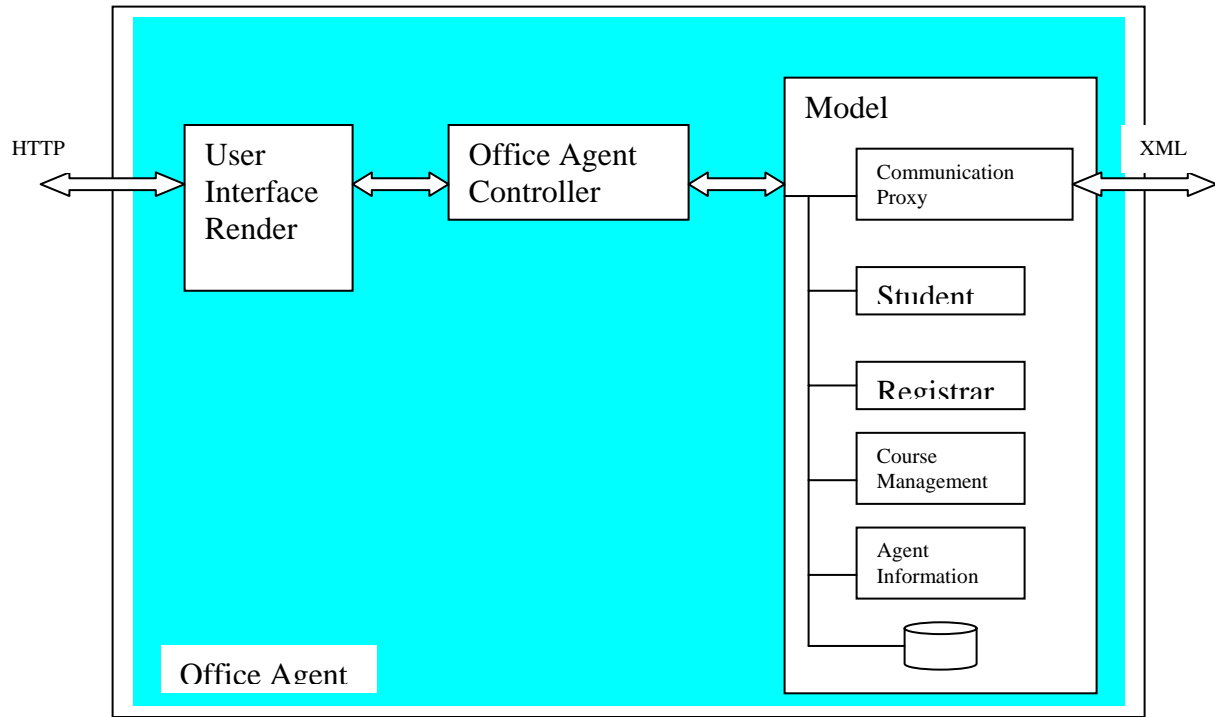


Figure 9. A sample use case for Office Agent

4.4 Library Agent

The library agent is responsible for providing library services to students and staff interested in using the library. It has various items that students can use. These items include, journals, books, periodicals and films. It also offers other services to its users. These services include ordering articles on line, books on line and payment for any of the services on line. It interacts with the facilitator to receive requests from other agents and send responses back to them.

4.4.1 Use cases

- Receives request through the Facilitator from agent
- Prompts for agents details
- Presents a menu of services offered for the agent to choose from.
- Uses library knowledge base to deal with agent's request
- Payment for services. e.g. ordering a book, an article, etc.

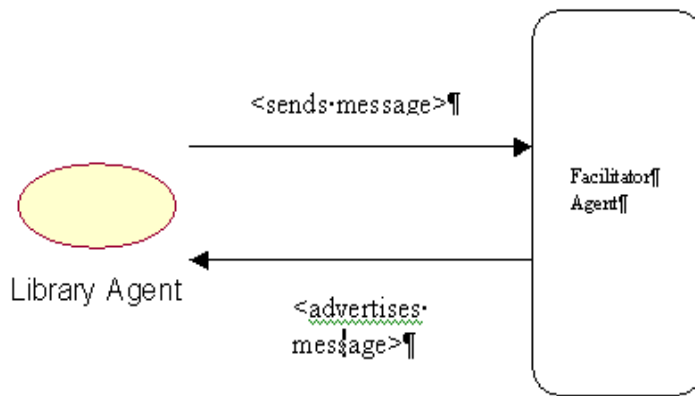


Figure 10. A sample use case for Office Agent

Use Case definitions

Brief Description	Facilitator passes agents request to library agent and once the library agent acknowledges it, it links the agent to the library agent
Pre-Condition	Actor must be a registered library user
Post-condition	Actor successfully uses the library services
Process steps	
1	Actor indicates willingness to use library
2	Facilitator checks actor's profile
3	Facilitator advertises to library agent that there is a request
4	Library agent expresses willingness to accept request
5	Facilitator links up library agent and requesting agent.

4.4.2 Internal architecture of Library Agent

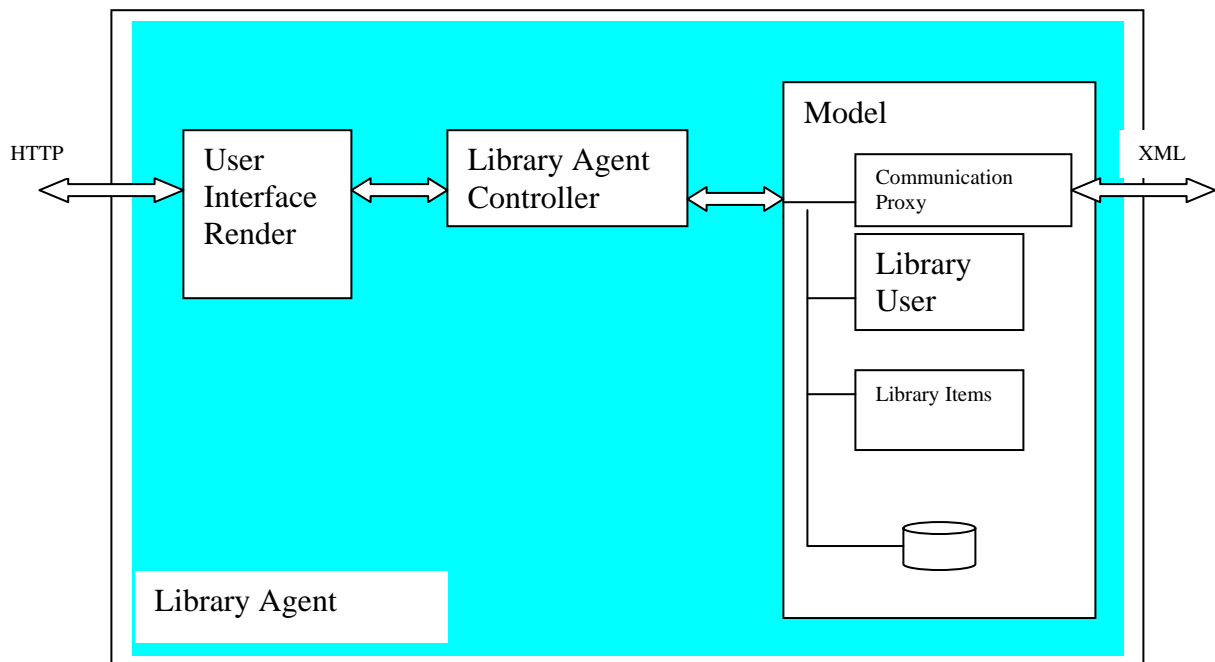


Figure 11. Internal architecture of Library Agent

4.5 Student Agent

The student agent is at the center of this tutoring and learning system. Students use the system to study various courses. They are thought by e-professors who have expertise in the various fields. Students must first register into a course before they can take it. They can be linked up directly by the facilitator to either the professor, TA, library or the office. Once they are linked up, they start conversing with the requested agent. Students can also use MASTL to take exams administered by the professors/TAs.

4.5.1 Use cases

- Student makes requests to use MASTL through the facilitator
- The student waits for a reply from the Facilitator Agent to link up student to appropriate agent
- The student can use the system to register for a course to be taken, to ask a question to a Professor/TA, to enquire about marks and to use the library services.
- The student can use system to verify his/her exam results
- The student must first register and provide his information to the Registrar's office before he/she can use the system

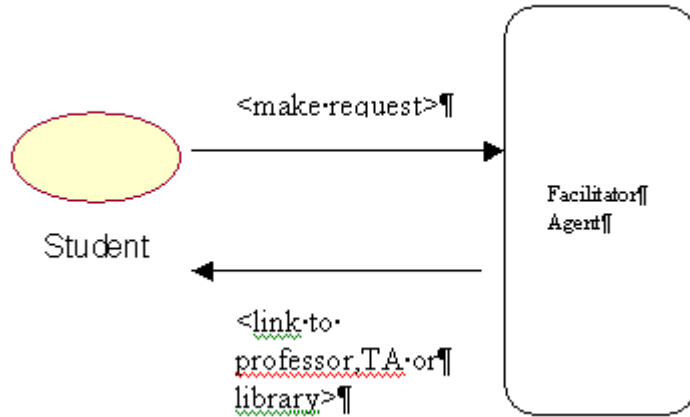


Figure 12. A sample use case for Student Agent

Use Case Definitions

Brief Description	Student Agent makes requests to use system for learning purposes
Pre-Condition	Student must be a eligible to register for courses
Post-condition	If student is allowed to use system he successfully uses it
Process steps	
1	Student makes a request through Facilitator
2	Facilitator checks student's profile
3	Facilitator determines from student's request which agent to link to student
4	Facilitator informs student Agent that target agent is available and links them up
5	Student starts using the target agent e.g Professor, library, office, TA.

4.5.2 Internal architecture of Student Agent

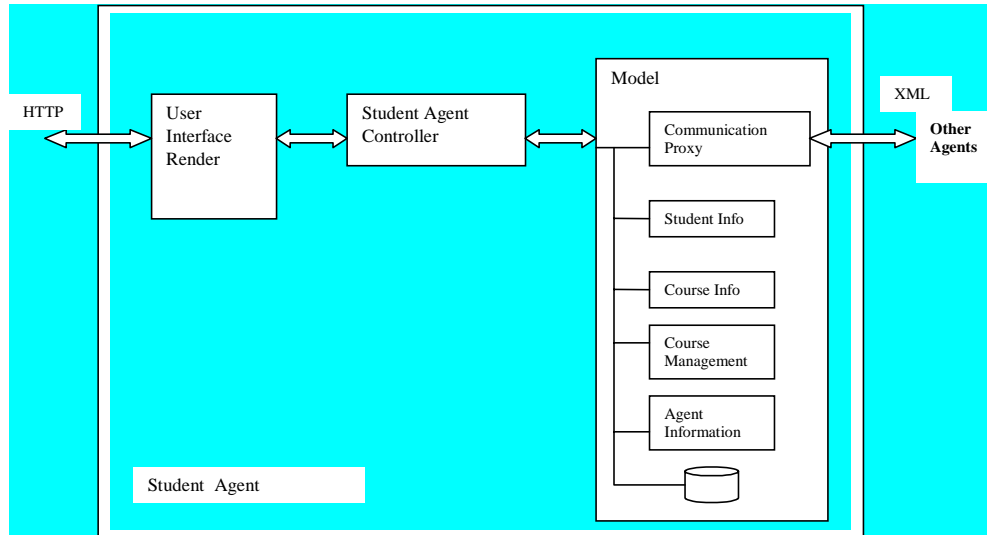


Figure 13. the internal architecture for Student Agent

4.6. Facilitator Agent

The facilitator agent is at the center of MASTL. It receives requests from other agents. It then uses information from the student, professor, TA, library and office databases to determine the most appropriate agent to handle request. It forwards requests to an agent that can handle it, decomposes a request if different parts of the request are to be handled by different agents and combines the different pieces of the request to generate an answer, forwards requests to agents that have expressed interest, and can also translate information between vocabularies to match an agent interface.

4.6.1 Use case

- Receives request from an agent.
- Uses the MASTL DB to check agent's profile
- Works out appropriate agent for request
- Uses the agent information in the knowledge base to link up requesting agent with appropriate agent that can deal with the case
- Keeps a log of all agent's activities

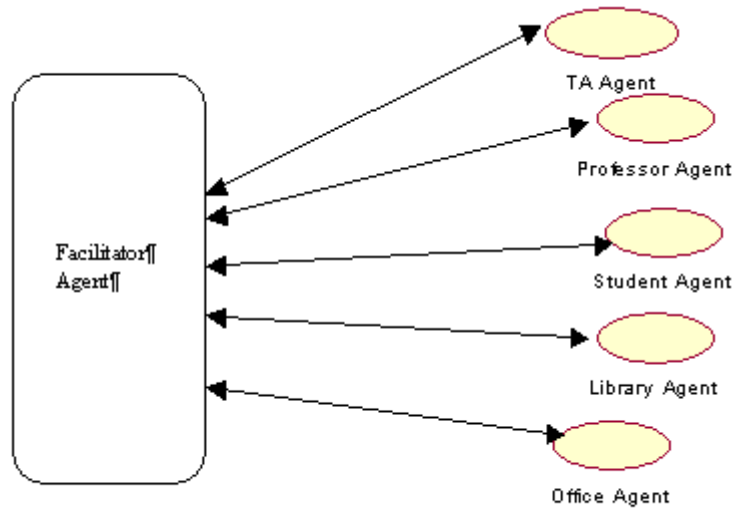


Figure 14. A sample use case for Facilitator Agent

Use Case Definitions

Brief Description	Facilitator coordinates the activities of different agents wishing to use MASTL
Pre-Condition	The agent wishing to use MASTL is eligible to use the system
Post-condition	Facilitator successfully links up agent to appropriate agent willing to work with requesting agent
Process steps	
1	Student or any other makes a request to use system by keying in his ID or registering into the system
2	Agent selects type of service from menu
3	Facilitator examines agent's profile
4	Facilitator determines appropriate agent for to deal with request using knowledge from knowledge base
5	Facilitator links up the agents

4.6.2 Internal architecture of Facilitator Agent

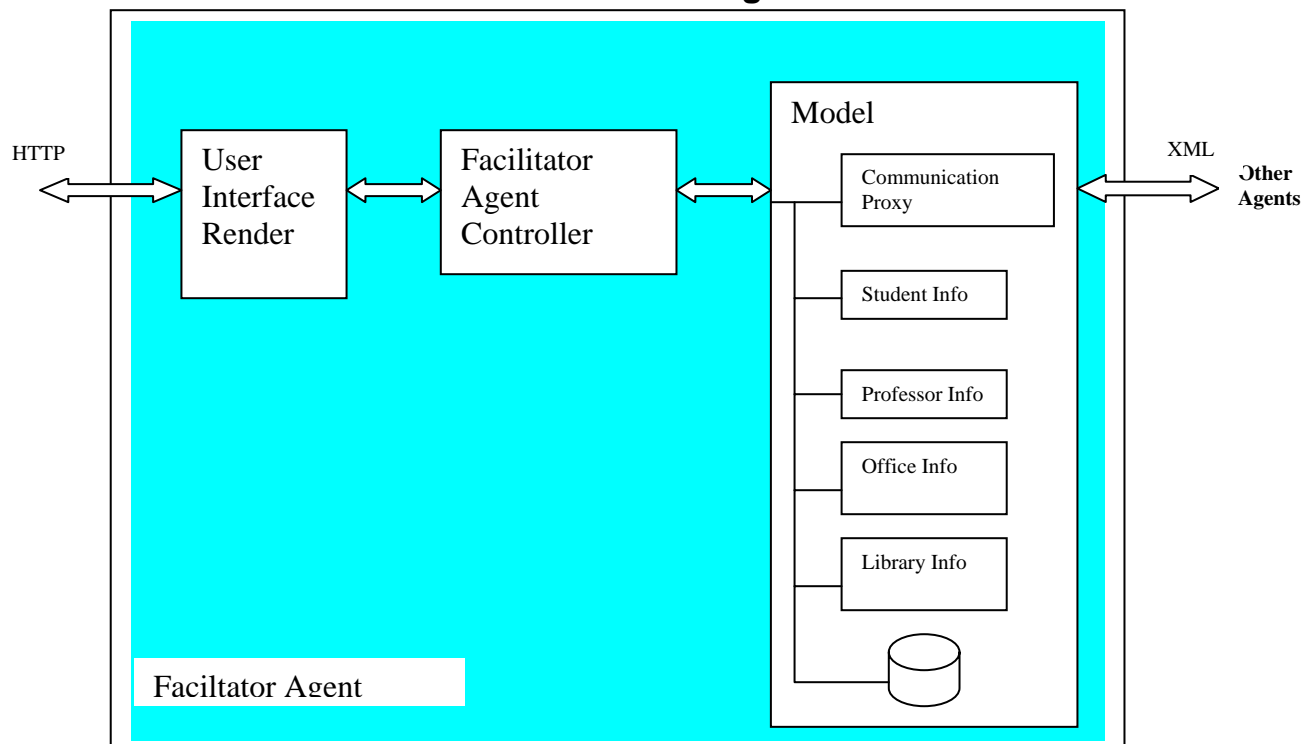


Figure 15. The internal architecture for Facilitator Agent

5 Data Specification

Professor:

Name of the field	Type	Length	Mandatory	Description
Name	String	30 characters		Initials and Surname
Employee ID	Integer	8 digits	C	An 8 digit integer
Office	String	5 characters	C	A letter followed by room numbre
Phone	Integer	9 digits	C	Country code and 7 digit phone number
Cell	Integer	9 digits	O	Country code and 7 digit phone number
Fax	Integer	9 digits	C	Country code and 7 digit phone number
Email	String	At most ten characters	O	Any sequence of letters and digits
Personal Web Page	String	Unlimited	O	Any web page address

Key: C = Compulsory, O = Optional

Student

Field	Description	Type
StudID	Seven Digit number	Integer
StudName	First name and Surname	String (30 characters long)
StudAddress	Street name, Zip Code, City	String (50 characters long)
UserStatus	1= Full Time, 2= Part Time, 3= other	Integer
StudBDate	Date	Yy/mm/dd
StudDegProg	Student's Degree Programme	String (30 characters)
StudDept	Student's Department	String
StudSupervisor	First name and Second Name	String
StudGPA	A four digit real number	Float

DegreeProgramme

Field	Description	Type
DegLevel	MSc, PhD, BSc	Long
DegRequirements	List of Courses required to take	String
DegDept	Enel, SPSC, Maths	String

Course

Field	Description	Type
CourseCode	Six character code	String
Course title	A string of characters	String (30 characters long)
CourseLecturer	First name and second name	String (50 characters long)
CourseTAs	First name and second	String

Office

Field	Description	Type
StaffID	Unique ID	Long
AgentID	First name and Surname	String (30 characters long)
StudID	Street name, Zip Code, City	String (50 characters long)
CourseCode	1= student, 2= staff, 3= other	Integer

RequestProcessor

Field	Description	Type
ReqID	Unique ID	Long
AgentID	Unique ID	Long

AgentInfoClass

Field	Description	Type
AgentID	Unique ID	Long
AgentName	A seven character word	String
AgentType	TA Agent, Professor agent, Office agent, Student agent, Library Agent	String
AgentStatus	Busy, Idle	Integer
AgentExpertise	Administrative services, library services, research/teaching, teaching	String

RequestInfo

Field	Description	Type
ReqID	Unique ID	Long
ReqTime	00:00-24:00	Clock
ReqContent	Message expressed in KIF	String (50 characters)
ReqStatus	1=running, 2= suspended, 3 = completed	Integer
ReqType	1 = Reply 2 = Request	Integer

Library

Field	Description	Type
UserID	Unique ID	Long
LibaryItemID	Unique ID	String (30chars long)

LibraryUser

Field	Description	Type
UserID	Unique ID	Long
UserName	First name and Surname	String (30 characters long)
UserAddress	Street name, Zip Code, City	String (50 characters long)
UserStatus	1= student, 2= staff, 3= other	Integer

LibraryItem

Field	Description	Type
ItemID	Unique Six character code	String
ItemType	1 = Journal, 2 = Books, 3 = Periodicals 4 = Films	Integer
ItemUsageTime	1=short term for students 2= long term for staff	Integer

LibraryService

Field	Description	Type
ServiceType	1=read 2=search 3=download 4=order	Integer
UserID	Unique ID	Long

6. Communication Protocol

6.1. XML over http

Because MASTL will be running on the distributed environment such as Internet, it will use HTTP as low-level communication protocol. Since HTTP is based on request/response, it is a stateless protocol. Therefore all the agents have to consider how to keep session related information. Java is a perfect implementation language not only because it is an object oriented language but also it has built-in networking functions that hides details for the developers. MASTL will use almost all of web application development technologies offered by J2EE. On the other hand, since XML is a platform independent markup language, it is a perfect knowledge description language for Multi-agent system communication. In addition since JAVA provides XML parsing APIs, MASTL will take advantage of platform indecency of both JAVA and XML.

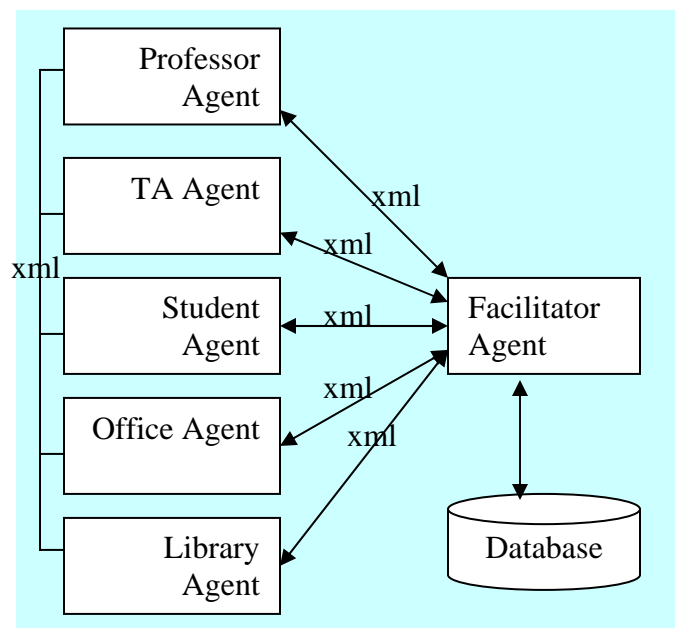


Figure 15. Agent Communication

6.2. Inter-agent messaging

Inter-agent messages are encoded in XML. The XML documents map data formats defined in the data specification section. In this section, some message definitions are described. The definitions for all other messages are to be completed in the next phase of the project.

Facilitator Messages

1. Process Message
2. Find Agent
3. Link Up Agents
4. Advertise Message

1) Process Request

Input

```
<Request>
  <ReqID> 123 </ReqID>
  <ReqTime>Clock</ReqTime>
  <ReqContent>String</ReqContent>
  <ReqStatus>Integer</ReqStatus>
  <ReqType>Integer</ReqType>
</Request>
```

Output

```
<AgentInfo>
  <AgentID>434</AgentID>
  <AgentName>String</AgentName>
  <AgentType>String</AgentType>
  <AgentStatus>Integer</AgentStatus>
  <AgentExpertise>Integer</AgentExpertise>
</AgentInfo>
```

2) Link Up Agents

Input

```
<AgentInfo>
  <AgentID></AgentID>
  <AgentName>String</AgentName>
  <AgentType>String</AgentType>
  <AgentStatus>Integer</AgentStatus>
  <AgentExpertise>Integer</AgentExpertise>
</AgentInfo>
```

Get agent's request

```
<Request>
  <ReqID></ReqID>
  <ReqTime>Clock</ReqTime>
  <ReqContent>String</ReqContent>
  <ReqStatus>Integer</ReqStatus>
  <ReqType>Integer</ReqType>
</Request>
```

Output

```
<AgentInfo>
  <AgentID></AgentID>
  <AgentName>String</AgentName>
  <AgentType>String</AgentType>
  <AgentStatus>Integer</AgentStatus>
  <AgentExpertise>Integer</AgentExpertise>
</AgentInfo>
```

Library Agent

Possible messages

- 1) Accept Message
- 2) Find Agent Identity
- 3) Signal end of use

Student Agent

Possible messages

- 1) Make Request
- 2) Ask Question
- 3) Order a book, article or film

Office Agent

Possible messages

- 1) Accept request
- 2) Ask for details
- 3) Send a response

Professor Agent

Possible messages

- 1) Request for marks from TA
- 2) Send information to student

- 3) Answer question

TA Agent

Possible messages

- 1) Send marks to Professor
- 2) Answer Question to student
- 3) Respond to Facilitators request

7 Future works and Conclusions

Although overall architecture design is done, more details and remaining components need to be completed. A practical implementation is worth to carry on. Since e-learning is very demanding, the potential of Agent based systems such as the one we have been developing is very promising. The future works include:

- Finish detailed design for the remaining components
- Finish all the message definitions
- Start prototyping on the each agent
- Draw a system integration plan
- ...etc.

Through the investigation and design of the MASTL system, we have learned that agent based technologies can be applied to developing intelligent learning and tutoring system that can provide distributed, reliable, and adaptive capability.

8 References

- [1] Nicola, C., M. DeSanto, M., Molinara, S. Salerno, *A Multi-Agent Architecture for Intelligent Tutoring*, CRMPA, Salerno, University of Salerno, Italy.
- [2] Tiffany, Y.T, A. Wu, *The Implementation of a Multi-agent intelligent Tutoring System for the learning of computer programming*, Hong Kong Polytechnic University, Hong Kong.
- [3] C. Petrie, *Agent-Based Software Engineering*,
<http://nrc.stanford.edu/~petrie/agents/abse/abse.html>
- [4] B. H. Far, *Lecture notes of SENG 609.22: Agent-based Software Engineering*,
<http://www.enel.ucalgary.ca/People/far/Lectures/SENG609-22/index.html>
Fall, 2002, The University of Calgary, Calgary, Alberta, Canada
- [5] N. R. Jennings, and M. Wooldridge *Agent-Oriented Software Engineering in Handbook of Agent Technology*, (ed. J. Bradshaw) AAAI/MIT Press, 2000.
- [6] Dignum, F., Greaves, “Issues in agent communication: an introduction”, Springer Verlag, 2000